Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт космических и информационных технологий

Кафедра вычислительной техники

**ОТЧЕТ О ПРЕДДИПЛОМНОЙ ПРАКТИКЕ**

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

Красноярск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc135643027)

[1 Спецификация требований к системе 4](#_Toc135643028)

[1.1 Существующие аналоги 4](#_Toc135643029)

[1.2 Разработка прецедентов 5](#_Toc135643030)

[1.3 Выводы по главе 11](#_Toc135643031)

[2 Проектирование 13](#_Toc135643032)

[2.1 Определение сущностей разрабатываемой системы 13](#_Toc135643033)

[2.2 Разработка диаграмм последовательности 14](#_Toc135643034)

[2.3 ER-диаграмма базы данных 15](#_Toc135643035)

[2.4 Выводы по главе 16](#_Toc135643036)

[3 Реализация и тестирование 18](#_Toc135643037)

[3.1 Создание бота в Telegram 18](#_Toc135643038)

[3.2 Инициализация проекта в Django 19](#_Toc135643039)

[3.3 Установка дополнительных библиотек 19](#_Toc135643040)

[3.4 Описание констант и функций 20](#_Toc135643041)

[3.4.1 Константы 20](#_Toc135643042)

[3.4.2 Функции 21](#_Toc135643043)

[3.5 Тестирование бота 22](#_Toc135643044)

[3.6 Выводы по главе 22](#_Toc135643045)

[Заключение 23](#_Toc135643046)

[Список использованных источников 25](#_Toc135643047)

Введение

Криптовалюты продолжают быстро развиваться, оказывая все большее влияние на мировую экономику и изменяя способы ведения финансовых операций. Они предлагают уникальные возможности для инвестирования и торговли, что привлекает все больше индивидуальных и корпоративных инвесторов. В этом контексте возникает необходимость в инструментах, позволяющих управлять криптовалютными активами, отслеживать их текущую стоимость и анализировать их динамику.

**Целью** данной дипломной работы является разработка криптовалютного бота для мессенджера Telegram. Бот позволяет пользователям управлять своим криптовалютным портфелем, добавлять новые монеты, продавать существующие, а также просматривать информацию о состоянии своего портфеля и очищать его.

В ходе выполнения работы были подробно изучены основные принципы функционирования мессенджера Telegram и его API, а также различные инструменты и технологии для создания ботов для этой платформы. Проект основывается на использовании Python и фреймворков aiogram и Django, которые предоставляют мощные возможности для асинхронного программирования и управления базами данных.

В **первой главе** рассматриваются основы работы с Telegram Bot API и библиотекой aiogram, а также архитектура бота и функции, предназначенные для обработки команд пользователя.

**Вторая глава** посвящена использованию Django и его асинхронных возможностей для управления базой данных бота. В ней описываются модели базы данных и функции для работы с данными пользователя.

В **третьей главе** подробно описывается процесс реализации функций бота, включая добавление и продажу монет, просмотр портфеля и его очистка.

В **заключении** подводятся итоги работы, оценивается ее эффективность и обсуждаются возможности для дальнейшего развития и улучшения проекта.

# 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 telegram bot» [1] на GitHub находит лишь 4 упоминания о таких проектах. Это проекты под такими названиями как: «LanyaFolioBot» [2], «crypto\_portfolio\_telegrambot» [3], «telegram\_bot\_portfolio» [4], «crypto-tele-bot» [5]. Был произведен анализ каждого из этих проектов.

«LanyaFolioBot» этот проект требует от пользователя создания аккаунта на криптовалютной бирже «Binance» [6] и импорта оттуда апи ключа для доступа к просмотру состояния кошелька пользователя.

«crypto\_portfolio\_telegrambot» этот проект так же требует от пользователя регистрацию на сторонних платформах, таких как «Binance», «Kucoin» [7], «Blockfrost.io» [8], «Coinmarketcap.com» [9].

«telegram\_bot\_portfolio» этот же проект в свою очередь имеет недостаточно документации чтобы понять что он действительно из себя представляет.

«crypto-tele-bot» а этот проект в принципе не закончен до конца, т.к. в нем есть лишь функции просмотра баланса, добавления монеты и включения уведомлений пользователя.

После анализа этих проектов можно прийти к выводу что у пользователя нет как такового выбора, поэтому было принято решение о разработке своего проекта с более простым обращением для пользователя и добавлением дополнительных функций.

## Разработка прецедентов

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

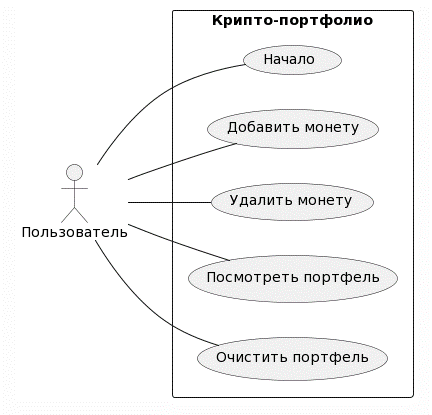


Рисунок 1.1 — Диаграмма вариантов использования

**Название прецедента:** Начало работы с крипто-портфолио.

**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.

Начало работы с крипто-портфолио показано на рисунках 1.2 и 1.3.

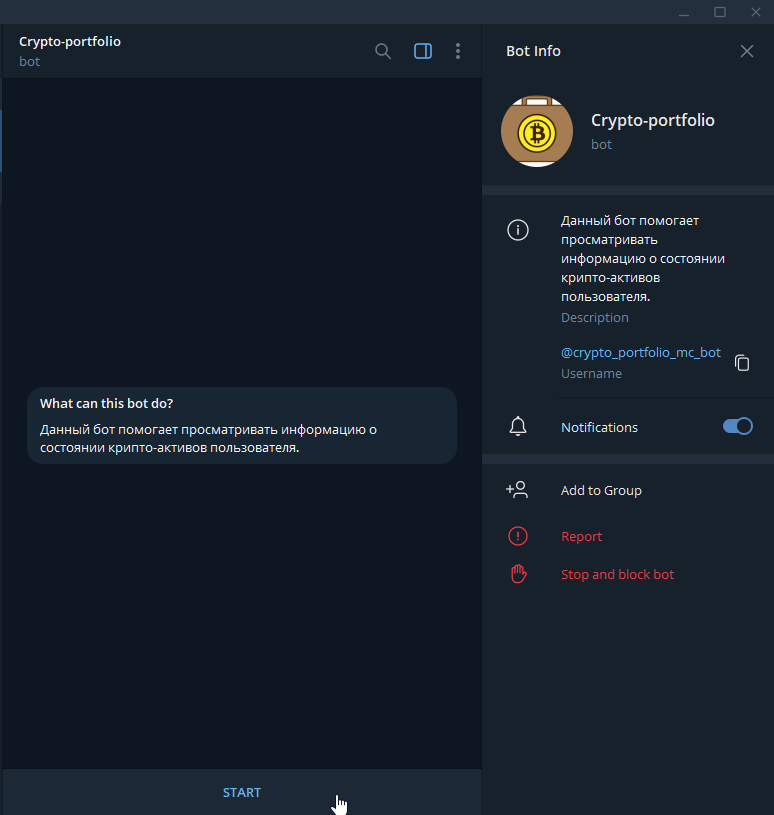


Рисунок 1.2 – начало работы с крипто-портфолио

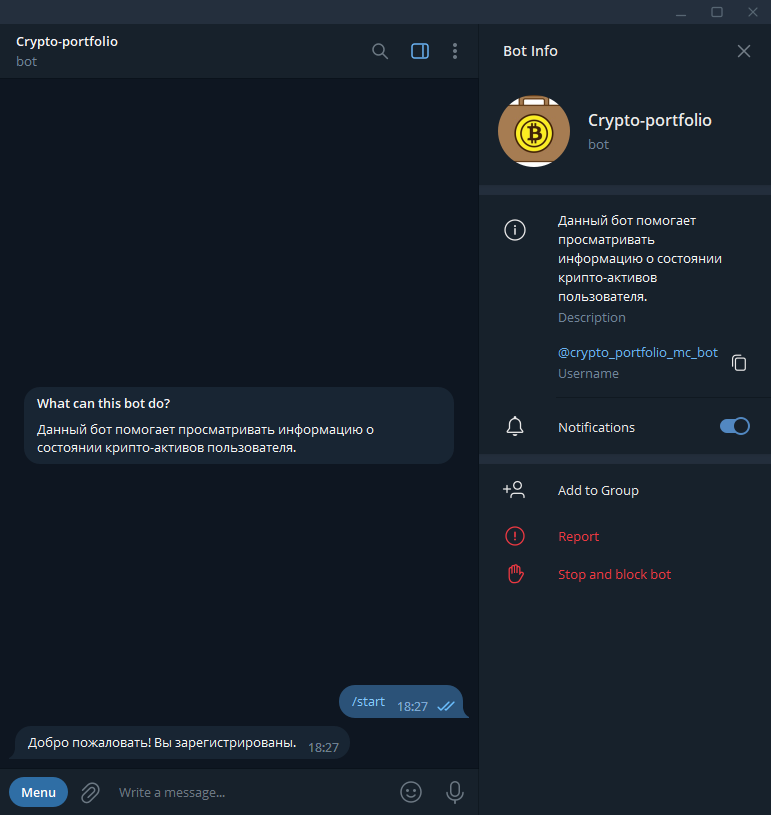


Рисунок 1.3 – начало работы с крипто-портфолио

**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.

Работа функции добавления монеты показана на рисунке 1.4.

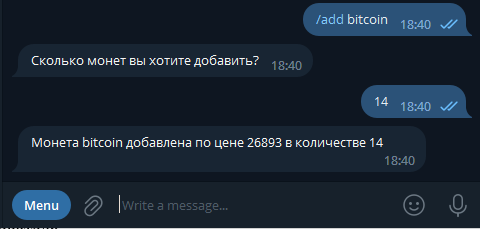


Рисунок 1.4 – добавление монеты в портфель

**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.

Работа функции по отображению портфеля представлена на рисунке 1.5.

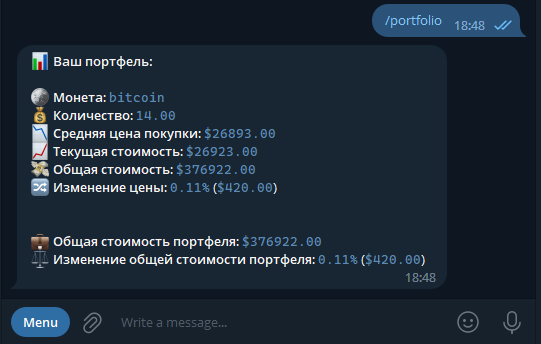


Рисунок 1.5 – отображение портфеля пользователя

**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.

Пример того как это выглядит приведен на рисунке 1.6.

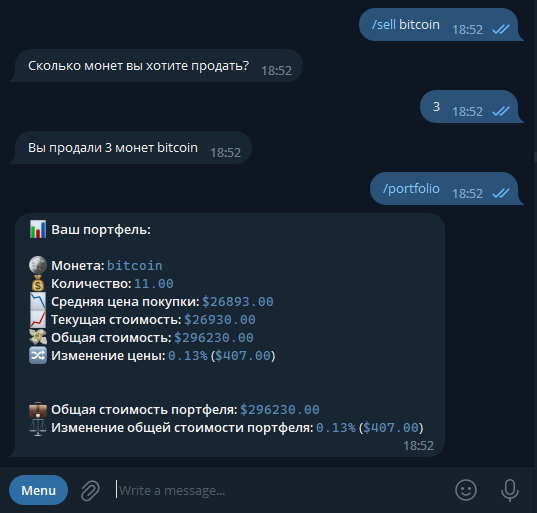


Рисунок 1.6 – Продажа монеты из портфеля

**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.

Пример очистки портфеля пользователя приведен на рисунке 1.7.



Рисунок 1.7 – Удаление всех монет из портфеля пользователя

## Выводы по главе

В первой главе мы провели анализ существующих решений для управления крипто-портфелем и выявили недостаток универсальных инструментов в мессенджере Telegram. Определены ключевые прецеденты использования нашей системы, которые охватывают основные сценарии взаимодействия пользователя с ботом. Этот этап работы предоставил теоретическую основу для дальнейшего проектирования и разработки крипто-портфолио бота.

# 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.

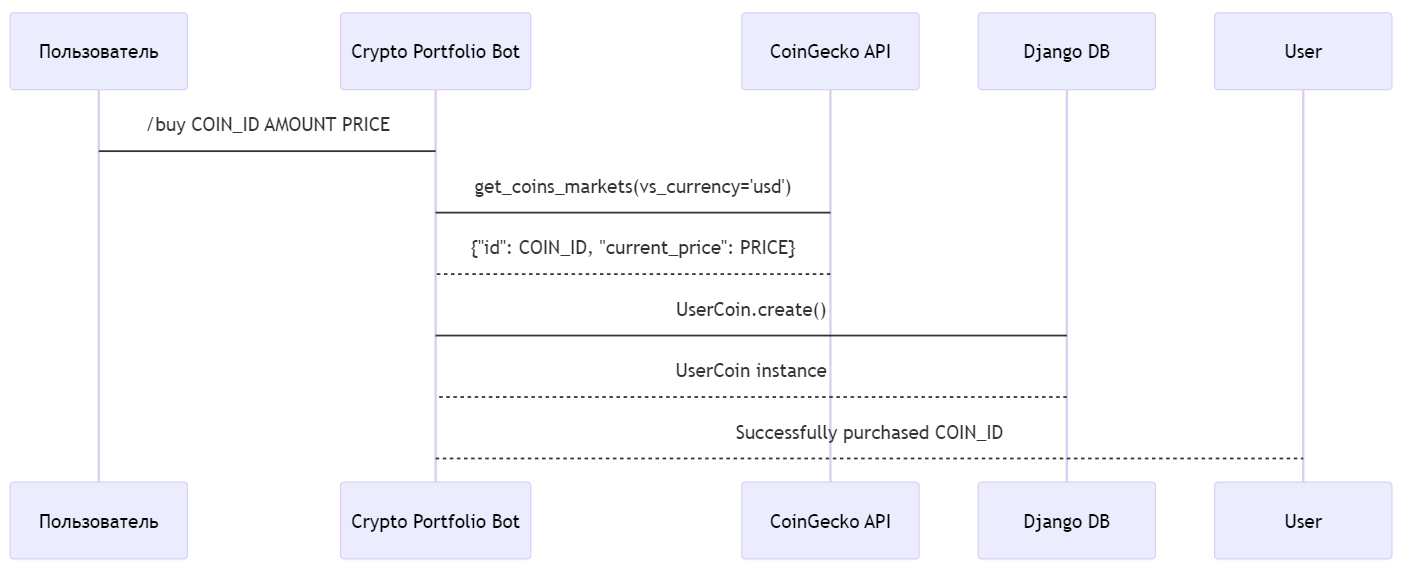


Рисунок 2.1 – Диаграмма последовательности покупки монеты пользователем

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.

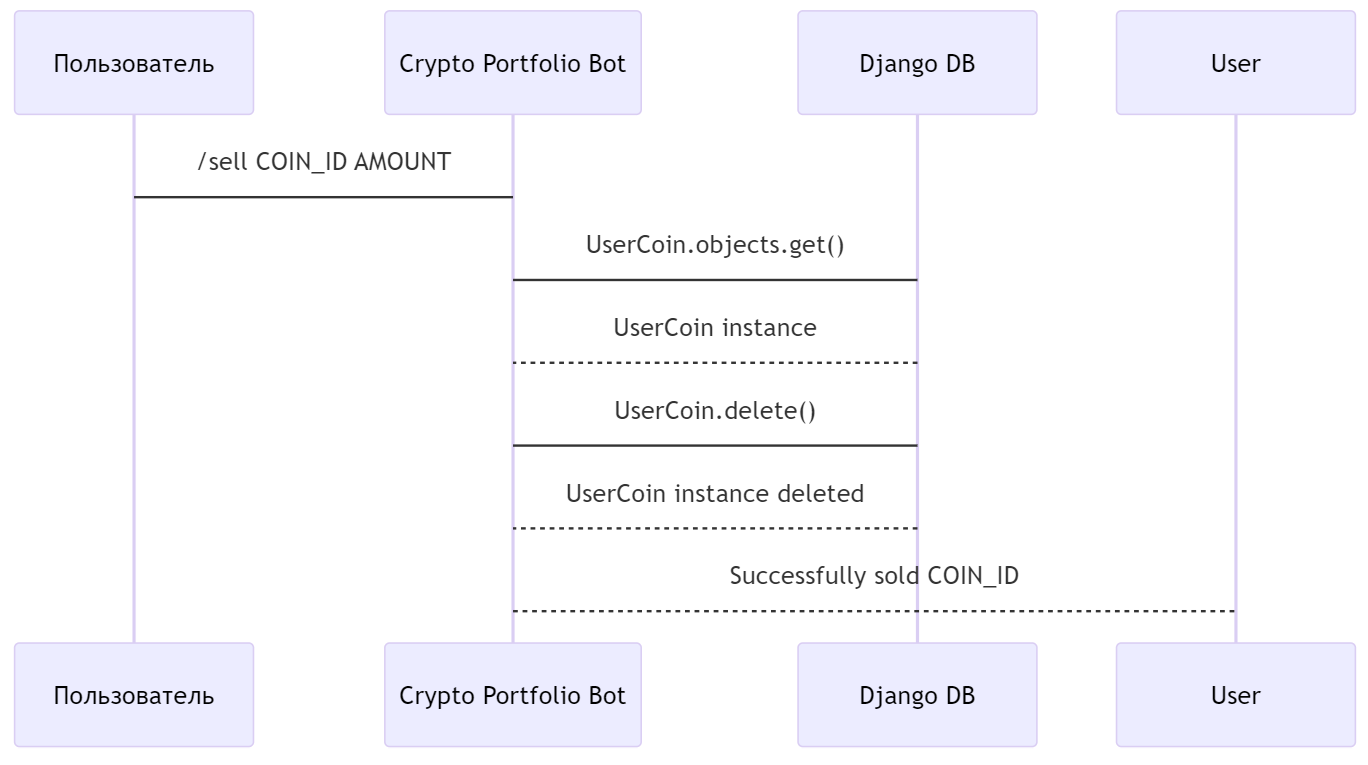


Рисунок 2.2 – Диаграмма последовательности покупки монеты пользователем

## 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.

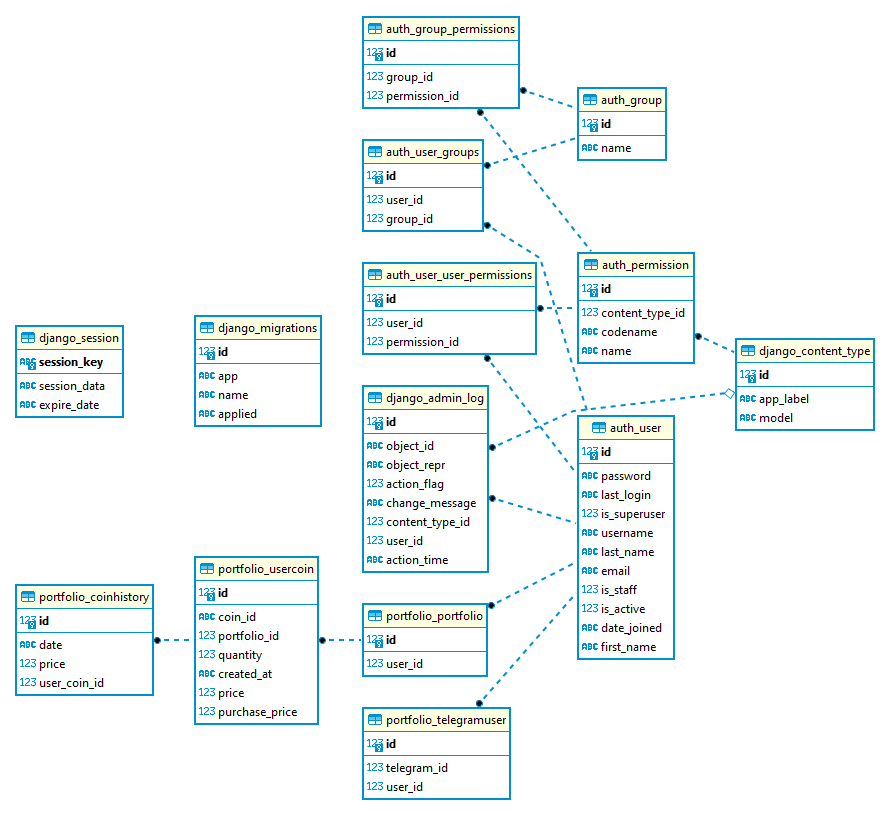


Рисунок 2.3 — ER-диаграмма базы данных

## Выводы по главе

Во второй главе работы были проведены ключевые этапы проектирования нашего проекта - криптовалютного портфеля в виде бота для Telegram. Мы исследовали основные сущности, с которыми будем взаимодействовать, и их связи, что было успешно представлено в виде ER-диаграммы.

Мы выбрали SQLite3 в качестве системы управления базами данных, так как она надежна, быстра и, что особенно удобно, встроена в Django. Использование Django позволило нам упростить работу с базой данных благодаря его ORM, позволяющей нам 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.

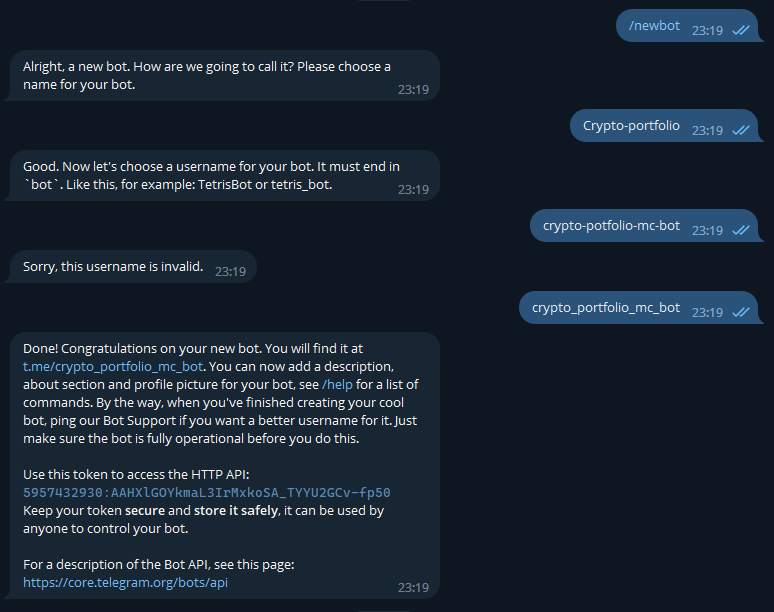


Рисунок 3.1 – Создание бота с помощью инструмента BotFather

## Инициализация проекта в Django

После создания бота, следующим шагом было инициализация нашего проекта в Django. Использование Django было обусловлено множеством причин, из которых ключевыми являются его мощный набор функций и интеграция с базой данных SQLite3.

Django является фреймворком на Python, который позволяет легко и быстро разрабатывать веб-приложения. Он предлагает подход "из коробки" для многих типичных задач, таких как аутентификация пользователей, управление формами и работа с базой данных. Одним из главных преимуществ Django для нашего проекта была его ORM (Object-Relational Mapping), позволяющая нам взаимодействовать с базой данных через высокоуровневые Python модели, что существенно упрощает процесс разработки.

После инициализации проекта с помощью команды django-admin startproject, мы приступили к настройке нашего Django-проекта, что включало в себя создание моделей для нашего приложения.

## 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.

Важно отметить, что все эти библиотеки были установлены с помощью инструмента для управления пакетами Python pip. Они помогли обеспечить эффективное взаимодействие с пользователем и получение актуальной 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.

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

Таким образом, все эти команды вместе обеспечивают полную функциональность бота, позволяя пользователям добавлять, продавать и просматривать монеты в их портфеле, а также очищать их портфель при необходимости.

## Тестирование бота

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

## Выводы по главе

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

Заключение

В ходе выполнения данной дипломной работы был разработан криптовалютный бот для мессенджера Telegram. Разработанный бот позволяет пользователям управлять своим криптовалютным портфелем, добавлять новые монеты, продавать существующие, просматривать информацию о состоянии своего портфеля, а также полностью очищать портфель.

В ходе исследования были подробно изучены основные принципы функционирования мессенджера Telegram и его API, а также технологии и инструменты, которые применяются для разработки ботов для данной платформы.

Проект основывается на использовании Python в связке с фреймворками aiogram и Django, которые предоставляют мощные инструменты для создания асинхронных приложений и управления базой данных. Были подробно рассмотрены различные аспекты работы этих технологий и инструментов.

Проведено тестирование бота, в результате которого были выявлены некоторые ошибки и исправлены. В целом, бот показал хорошую работоспособность и отвечает всем заявленным требованиям.

В результате данной дипломной работы был создан функциональный и полезный инструмент, который может помочь пользователям управлять и отслеживать свои инвестиции в криптовалюты.

Основная цель работы - разработка рабочего криптовалютного бота для Telegram - была достигнута. Тем не менее, есть множество возможностей для дальнейшего развития и улучшения проекта.

Например, можно добавить функционал оповещений, которые будут уведомлять пользователя о значительных изменениях в стоимости монет в их портфеле.

В заключение хочется отметить, что данная работа не только позволила мне применить и углубить свои знания в области программирования и работы с криптовалютами, но и создать реальный продукт, который может быть полезен для многих пользователей.

Список использованных источников

1. Результаты запросов по ключевому словам «crypto-portfolio telegram» / GitHub : сайт. – URL: https://github.com/search?q=crypto-portfolio+telegram (дата обращения: 09.05.2023).
2. Репозиторий «[lanya-dorkin](https://github.com/lanya-dorkin)/[LanyaFolioBot](https://github.com/lanya-dorkin/LanyaFolioBot)» / GitHub : сайт. – URL: https://github.com/lanya-dorkin/LanyaFolioBot (дата обращения: 09.05.2023).
3. Репозиторий «myown-del/crypto\_portfolio\_telegrambot» / GitHub : сайт. – URL: https://github.com/myown-del/crypto\_portfolio\_telegrambot (дата обращения: 09.05.2023).
4. Репозиторий «effectmaks/telegram\_bot\_portfolio» / GitHub : сайт. – URL: https://github.com/effectmaks/telegram\_bot\_portfolio (дата обращения: 09.05.2023).
5. Репозиторий «martin-ri/crypto-tele-bot» / GitHub : сайт. – URL: https://github.com/martin-ri/crypto-tele-bot (дата обращения: 09.05.2023).
6. Binance.com: сайт. – URL: https://www.binance.com/ (дата обращения: 12.05.2023).
7. Kucoin.com: сайт. – URL: https://www.kucoin.com/ (дата обращения: 12.05.2023).
8. Blockfrost.io: сайт. – URL: https://blockfrost.io/ (дата обращения: 12.05.2023).
9. CoinMarketCap.com: сайт. – URL: https://coinmarketcap.com/ (дата обращения: 12.05.2023).
10. Репозиторий «Toxich2012/kursovoi» / GitHub : сайт. – URL: https://github.com/Toxich2012/kursovoi (дата обращения: 15.05.2023).
11. Django documentation: The official documentation for Django. — Текст : электронный // Django : [сайт]. — URL: https://docs.djangoproject.com/ (дата обращения: 16.05.2023).
12. Coingecko API: A web service for retrieving cryptocurrency data. — Сайт. – URL: https://www.coingecko.com/en/api/documentation (дата обращения: 16.05.2023).
13. 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).
14. Git / A free and open source distributed version control system : сайт. – URL: https://git-scm.com/ (дата обращения: 18.05.2023).