|  |
| --- |
| 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**

|  |
| --- |
| 09.03.01 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](#_Toc137896789)

[1 Спецификация требований к системе 5](#_Toc137896790)

[1.1 Существующие аналоги 5](#_Toc137896791)

[1.2 Постановка задачи 6](#_Toc137896792)

[1.3 Выводы по главе 8](#_Toc137896793)

[2 Проектирование 10](#_Toc137896794)

[2.1 Выбор инструментов для реализации 10](#_Toc137896795)

[2.1.1 Язык программирования Python 10](#_Toc137896796)

[2.1.2 Среда выполнения 10](#_Toc137896797)

[2.1.3 Выбор среды разработки 11](#_Toc137896798)

[2.2 Разработка прецедентов 11](#_Toc137896799)

[2.3 Разработка диаграмм последовательности 16](#_Toc137896800)

[2.4 Выводы по главе 20](#_Toc137896801)

[3 Реализация и тестирование 21](#_Toc137896802)

[3.1 Создание аккаунта бота в телеграмм 21](#_Toc137896803)

[3.2 Описание использованных библиотек 22](#_Toc137896804)

[3.3 Описание таблиц баз данных и функций 22](#_Toc137896805)

[3.3.1 Основные таблицы базы данных 23](#_Toc137896806)

[3.3.2 Функции 24](#_Toc137896807)

[3.4 Тестирование 29](#_Toc137896808)

[3.5 Выводы по главе 30](#_Toc137896809)

[Заключение 31](#_Toc137896810)

[Список использованных источников 32](#_Toc137896811)

Введение

С развитием цифровых технологий, криптовалюты переходят из ниши специализированных финансовых инструментов в общепринятые цифровые активы. Когда криптовалютные активы начинают занимать существенное место в портфелях инвесторов, и торговля криптовалютами становится основным источником дохода, возникает необходимость в надежных и эффективных инструментах для управления и контроля этими активами.

Современный криптовалютный рынок обладает рядом инструментов для управления цифровыми активами, таких как цифровые кошельки, биржи и портфолио-трекеры. Однако эти инструменты могут быть сложны в использовании для неподготовленных пользователей, а также часто не предоставляют полной информации, которая необходима для эффективного управления активами, особенно при наличии большого количества различных валют.

В этом контексте, выпускная квалификационная работа направлена на разработку инструмента для управления криптовалютным портфелем, который бы был простым в использовании и предоставлял бы всю необходимую информацию для эффективного управления активами.

Целью выпускной квалификационной работы является разработка инструмента, позволяющего пользователям отслеживать состояние своего криптовалютного портфеля. Для достижения этой цели необходимо решить следующие задачи:

* изучение существующих аналогов и подходов к управлению криптовалютными активами;
* анализ требований к функционалу инструмента 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.

Создаваемый в рамках выпускной квалификационной работы инструмент предоставит пользователям возможность эффективного и удобного управления своими криптовалютными активами, что будет способствовать популяризации криптовалют и их широкому использованию.

# Спецификация требований к системе

Для разработки спецификации выпускной квалификационной работы, был проведен анализ 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 репозитория в сообществе github: trentpiercy/trace [2], emmtte/Cryptocurrency-Portfolio [3], huwwp/cryptop [4]. Сравнение их параметров приведено в таблице 1.1.

1. Trentpiercy/trace: 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.
2. Emmtte/Cryptocurrency-Portfolio: 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.
3. Huwwp/cryptop: 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. Данный проект представлен в виде desktop приложения на языке Python.

Таблица 1.1 — Сравнение параметров проектов

|  |  |  |  |
| --- | --- | --- | --- |
| Название аналога | Trentpiercy/trace | Emmtte/Cryptocurrency-Portfolio | Huwwp/cryptop |
| Платформа | Мобильное приложение | Google sheets | Desktop приложение |
| Язык | Dart | JavaScript | Python |
| Функционал | Отслеживание портфеля, истории транзакций, прибыли и убытка | Отслеживание портфеля, разбивка портфеля по валютам | API для отслеживания портфеля |
| Уведомление об изменении цены | нет | нет | нет |

## Постановка задачи

На рынке существует множество сервисов для отслеживания криптовалютного портфеля. Однако, все они предполагают использование отдельных мобильных приложений или desktop-решений, что может быть неудобно для некоторых пользователей. Анализ существующих решений позволяет выделить их основные недостатки:

* Все рассмотренные сервисы требуют установки и использования отдельного ПО;
* Ни одно из рассмотренных решений не имеет функции установления порога изменения цены для уведомления пользователя.

С учётом анализа было принято решение реализовать проект в виде Telegram бота. Такое решение представляется наиболее оптимальным по нескольким причинам. Во-первых, Telegram - это один 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. В-третьих, благодаря возможности мгновенного обмена сообщениями, боты в Telegram могут предоставлять пользователю актуальную информацию о состоянии его портфеля в реальном времени. В общем и целом, реализация проекта в виде Telegram бота позволяет упростить управление криптовалютными активами и сделать его более доступным для широкого круга пользователей.

Таким образом, разрабатываемый проект "Криптовалютный портфель" в виде чат-бота в мессенджере Telegram будет обладать следующими свойствами:

* Удобный и простой доступ к информации о состоянии портфеля прямо в мессенджере без необходимости установки дополнительного ПО;
* Поддержка широкого спектра криптовалют для управления портфелем;
* Интуитивно понятный и удобный интерфейс для управления портфелем;
* Минимальные требования к ресурсам устройства пользователя.

Чат-бот будет предоставлять следующие возможности:

* Добавить монету в портфолио;
* Удалить монету из портфолио;
* Просмотреть портфолио;
* Удалить все монеты из портфолио;
* Установить значение порога изменения цены;
* Просмотреть установленные пороги.

В процессе постановки задач 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 системе. Также для каждого пользователя создается связанная с ним 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.

## Выводы по главе

Данная глава была посвящена изучению существующих сервисов управления криптовалютными портфелями и выявлению их недостатков. Основываясь на проведенном анализе, принято решение о создании крипто-портфеля в виде чат-бота для Telegram, который станет более удобной и доступной альтернативой текущим решениям.

Были выделены основные сущности проекта: TelegramUser, Portfolio, UserCoin и CoinHistory, отражающие ключевые аспекты работы с системой. С их помощью обеспечивается эффективное взаимодействие пользователя с системой и хранение необходимых данных.

Выбор Telegram в качестве платформы обусловлен его широким распространением среди пользователей, функциональностью ботов и возможностью предоставления пользователям актуальной информации о состоянии их портфелей в реальном времени.

Таким образом, в рамках данной главы были определены основные задачи и подходы для дальнейшего разработки и реализации проекта.

# Проектирование

## Выбор инструментов для реализации

Определение инструментов для реализации проекта основывается на ряде факторов, включая особенности проекта, необходимую функциональность, уровень знакомства с предполагаемыми инструментами и другие индивидуальные предпочтения. В данном разделе представлен обзор выбранных инструментов для реализации проекта создания бота для управления криптовалютным портфелем в Telegram.

### Язык программирования Python

Python был выбран в качестве основного языка программирования для данного проекта из-за нескольких причин. Python обладает высокой читаемостью кода и простотой изучения, что облегчает разработку и поддержку проекта. К тому же, Python предлагает богатую экосистему библиотек и фреймворков, позволяющих реализовывать различные функции. В данном случае, для взаимодействия с API Telegram применяется библиотека aiogram.

### Среда выполнения

Средой выполнения для данного проекта служит интерпретатор Python, который встроен в PyCharm и обеспечивает исполнение кода непосредственно в среде разработки. Интерпретатор Python обеспечивает выполнение скриптов на языке Python, включая данного бота. Он также отвечает за обработку операций ввода-вывода и взаимодействия с операционной системой. Использование встроенного интерпретатора Python облегчает отладку кода и позволяет быстро запускать и тестировать разработанный функционал.

### Выбор среды разработки

В качестве среды разработки был выбран PyCharm. PyCharm - это мощная интегрированная среда разработки (IDE), созданная специально для Python. Она предлагает множество удобных функций, таких как подсветка синтаксиса, автодополнение кода, поддержка рефакторинга и отладки, а также интеграция с системами контроля версий. Благодаря этому разработчики могут сосредоточиться на решении задач, минимизируя время на рутинные операции.

## Разработка прецедентов

На рисунке 2.1 представлена диаграмма вариантов использования, отражающая действия пользователя.

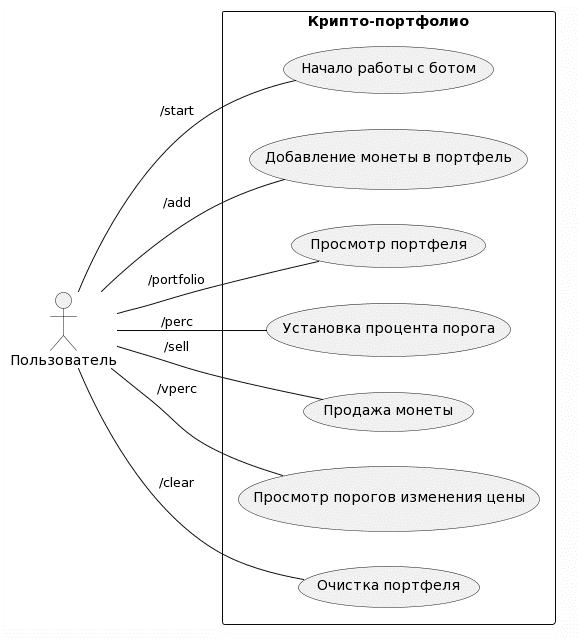


Рисунок 2.1 – Диаграмма прецедентов

**Название прецедента:** Начало работы с ботом.

**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.

## Разработка диаграмм последовательности

Диаграммы последовательности помогают визуализировать и лучше понять процессы, происходящие в системе. В рамках выпускной квалификационной работы были разработаны диаграммы последовательности.

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 2.2.

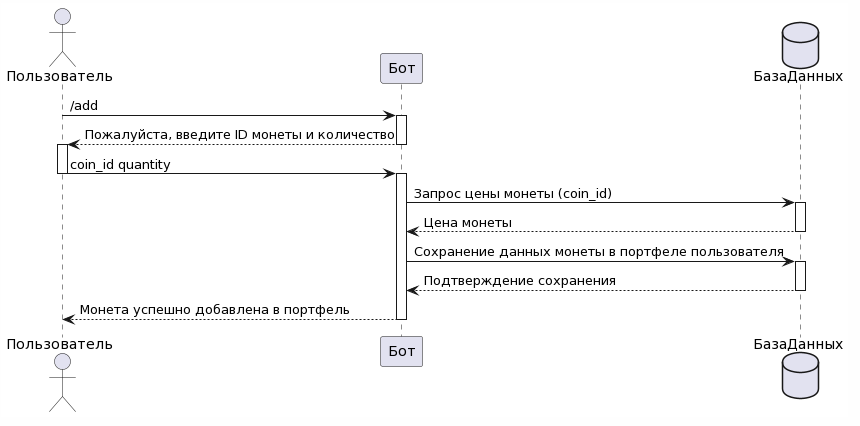


Рисунок 2.2 – Диаграмма последовательности покупки монеты пользователем

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 2.3

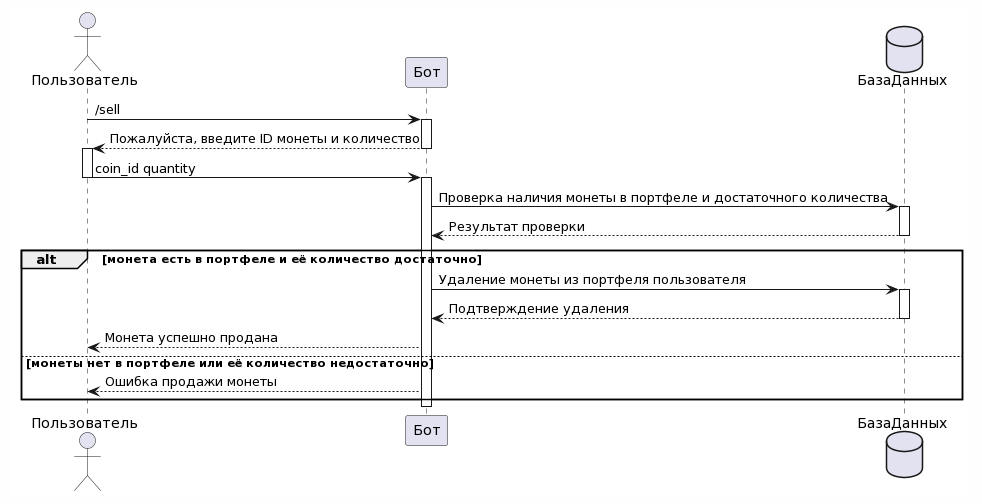


Рисунок 2.3 – Диаграмма последовательности продажи монеты пользователем

В прецеденте "Отображение портфеля" отражены основные шаги, которые происходят при запросе отображения портфеля: пользователь отправляет команду /portfolio, бот получает запрос на отображение портфеля пользователя, обращается к базе данных для извлечения информации о портфеле пользователя, рассчитывает текущую стоимость каждой монеты в портфеле и формирует сообщение с детальной информацией о портфеле пользователя. Затем бот отправляет пользователю сообщение с информацией о его портфеле. Диаграмма представлена на рисунке 2.4



Рисунок 2.4 – Диаграмма последовательности отображения портфолио

В прецеденте "Очистка портфеля" отражены основные шаги, которые происходят при выполнении команды /clear: пользователь отправляет команду /clear боту, бот обрабатывает запрос и обращается к базе данных для удаления всех монет из портфеля пользователя. Диаграмма представлена на рисунке 2.5

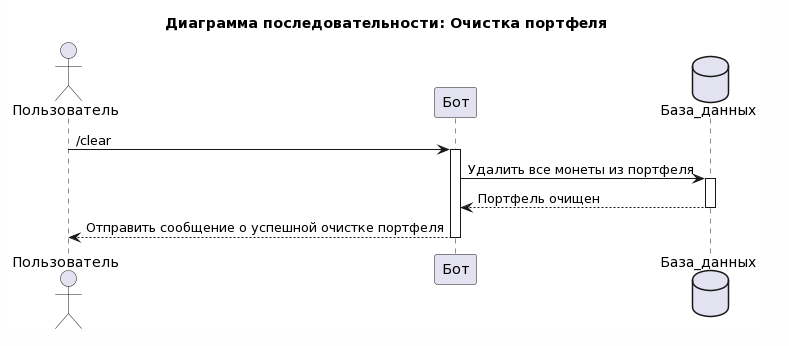


Рисунок 2.5 – Диаграмма последовательности очистка портфоля

В прецеденте "Установка порога процента" отражены основные шаги, которые происходят при выполнении команды /perc: пользователь отправляет команду /perc боту, бот отображает клавиатуру с вариантами промежутков времени, пользователь выбирает промежуток времени и отправляет соответствующую команду, бот обрабатывает выбор пользователя и запрашивает новый процент порога для изменения цены, пользователь отправляет новый процент порога, бот обрабатывает введенный процент и сохраняет его в базе данных. Диаграмма представлена на рисунке 2.6

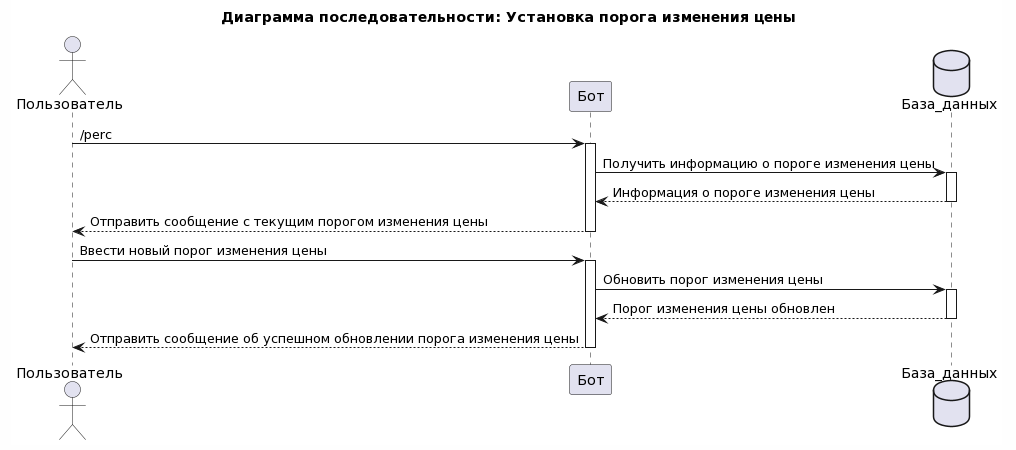


Рисунок 2.6 – Диаграмма последовательности установка порога изменения цены

В прецеденте "Просмотр порогов процента" отражены основные шаги, которые происходят при выполнении команды /vperc: пользователь отправляет команду /vperc боту, бот отображает текущие установленные пороги процента для каждого промежутка времени из базы данных. Диаграмма представлена на рисунке 2.7

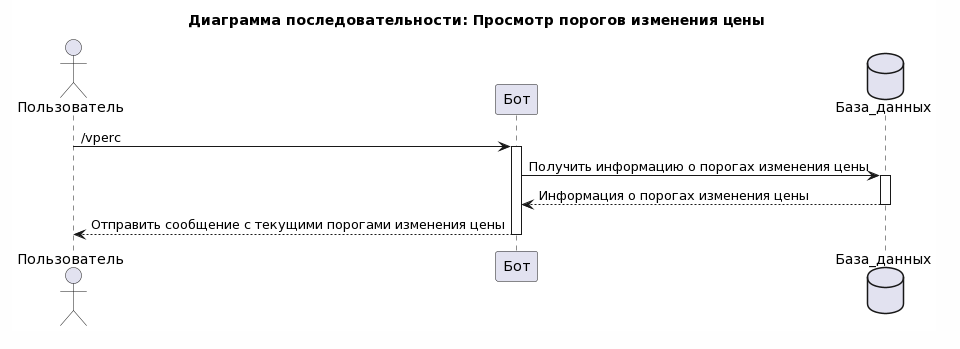


Рисунок 2.7 – Диаграмма последовательности просмотр порогов изменения цены

## Выводы по главе

В данной главе были осуществлены следующие шаги:

* Выбор инструментов и технологий: В этом разделе были выбраны язык программирования, среда выполнения и другие инструменты для реализации крипто-портфолио. Этот выбор основывался на требованиях проекта, и возможностях инструментов.
* Разработка прецедентов: Были определены основные функциональные возможности системы, которые отражаются в прецедентах. Каждый прецедент описывает сценарий использования системы со стороны пользователя. Это позволяет лучше понять потребности пользователей и функциональные требования к системе.
* Разработка диаграмм прецедентов: Были созданы диаграммы прецедентов, которые визуализируют взаимодействие пользователей с системой и основные функциональные возможности, предоставляемые системой. Диаграммы прецедентов помогают лучше понять структуру и функциональность системы.
* Разработка диаграмм последовательностей: Были созданы диаграммы последовательностей, которые показывают взаимодействие различных компонентов системы и последовательность выполнения операций. Диаграммы последовательностей помогают лучше понять, как система обрабатывает запросы и взаимодействует с другими системами или базой данных.

В результате выполнения этих шагов была проведена подготовительная работа для разработки крипто-портфолио. Были определены функциональные требования, описаны сценарии использования системы и проиллюстрированы ключевые аспекты взаимодействия. Это позволяет лучше понять систему, ее функциональность и потребности пользователей перед переходом к следующему этапу разработки.

# Реализация и тестирование

## Создание аккаунта бота в телеграмм

На этом этапе был создан аккаунт бота в Telegram. Для этого использовалась встроенная платформа Telegram для создания ботов, известная как BotFather. BotFather является ботом, разработанным командой Telegram для создания, управления и настройки пользовательских ботов. В процессе создания бота были следованы инструкции, предоставленные BotFather. Это включало выбор имени для бота, создание уникального username и получение уникального токена, который будет использоваться для идентификации бота при взаимодействии с Telegram Bot API. Процесс создания бота представлен на рисунке 3.1.

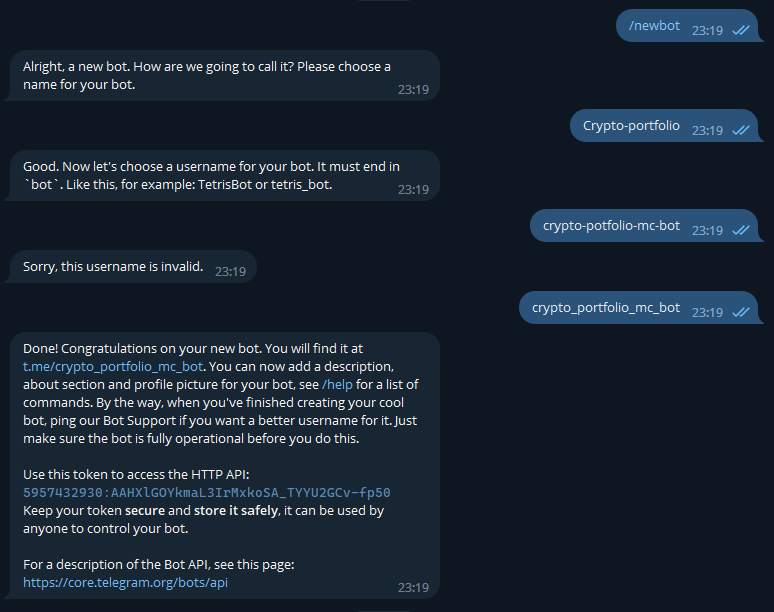


Рисунок 3.1 – Создание бота с помощью инструмента BotFather

## Описание использованных библиотек

На этапе реализации чат-бота было принято решение использовать дополнительные библиотеки для обеспечения различных аспектов функциональности. Основными библиотеками, которые были выбраны для данного проекта, являются:

* aiohttp: Асинхронная HTTP-библиотека, которая была использована для обработки параллельных HTTP-запросов к различным API. Это позволило обеспечить более быструю и эффективную обработку взаимодействий пользователя с ботом.
* aiogram: Современная и мощная библиотека для создания ботов для Telegram на языке Python. Она предоставляет удобные и гибкие инструменты для работы с API Telegram, позволяя создавать сложные сценарии взаимодействия с пользователями.
* asgiref: Библиотека, которая является частью стандарта ASGI (Asynchronous Server Gateway Interface) для асинхронных веб-приложений на Python. Она использовалась для работы с асинхронными вызовами в фреймворке Django, который выбран в качестве основы для проекта.
* requests: Библиотека, используемая для отправки HTTP-запросов. Она была применена для взаимодействия с внешними сервисами и получения информации о криптовалютах.

## Описание таблиц баз данных и функций

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

### Основные таблицы базы данных

Основные таблицы базы данных определены в файле models.py и представляют структуру данных, необходимую для работы приложения:

* Таблица TelegramUser: Она представляет уникального пользователя бота в Telegram. Взаимосвязь с моделью User Django осуществляется через поле OneToOneField, что позволяет каждому пользователю Django иметь одного ассоциированного пользователя Telegram. Одно из ключевых полей в этой таблице - telegram\_id, которое хранит уникальный идентификатор пользователя Telegram. Так же в этой таблице присутствуют поля для установленных порогов изменения цены.
* Таблица Portfolio: Она представляет портфель пользователя, который содержит информацию о его вложениях в различные криптовалюты. Каждая запись в таблице Portfolio связана с конкретным пользователем Django.
* Таблица UserCoin: Она представляет конкретное вложение пользователя в криптовалюту в его портфеле. В таблице UserCoin хранится информация о типе криптовалюты (coin\_id), текущей цене (price), цене покупки (purchase\_price), количестве (quantity) и времени создания (created\_at). Связь с таблицей Portfolio осуществляется через внешний ключ.
* Таблица CoinHistory: Она представляет историю изменения цены криптовалюты, вложенной пользователем. В таблице CoinHistory содержится информация о вложении пользователя (user\_coin), дате и времени изменения цены (date) и самой цене (price). Связь с таблицей UserCoin осуществляется через внешний ключ.

Таблицы представлены на рисунке 3.2.

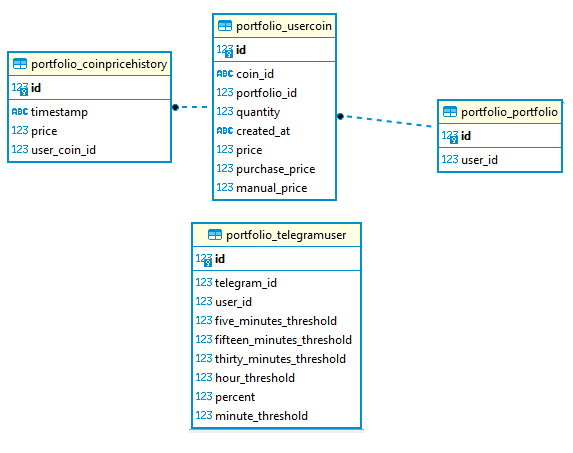


Рисунок 3.2 – Таблицы базы данных

### 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 3.3.

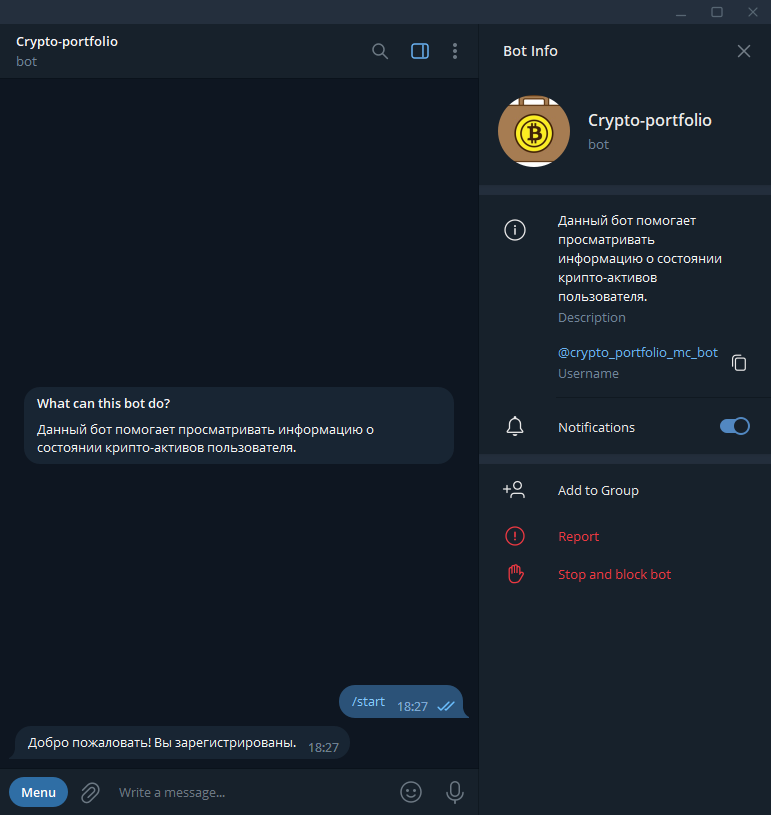


Рисунок 3.3 – команда start

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 3.4.

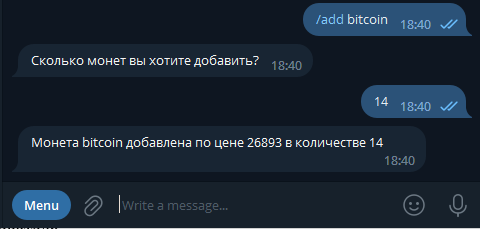


Рисунок 3.4 – команда add

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 3.5.

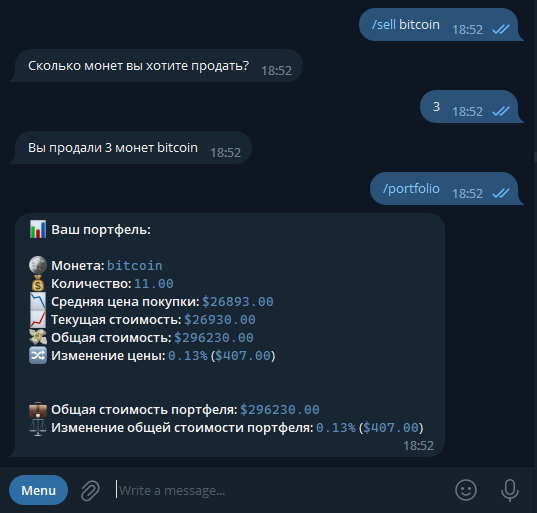


Рисунок 3.5 – команда sell

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 3.6.

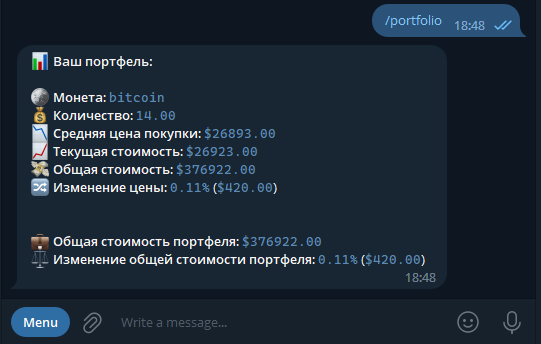


Рисунок 3.6 – команда portfolio

Команда "/clear": эта команда удаляет все монеты из портфеля пользователя и выводит сообщение о том, что все монеты были удалены. Команда представлена на рисунке 3.7.



Рисунок 3.7 – 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 3.8.

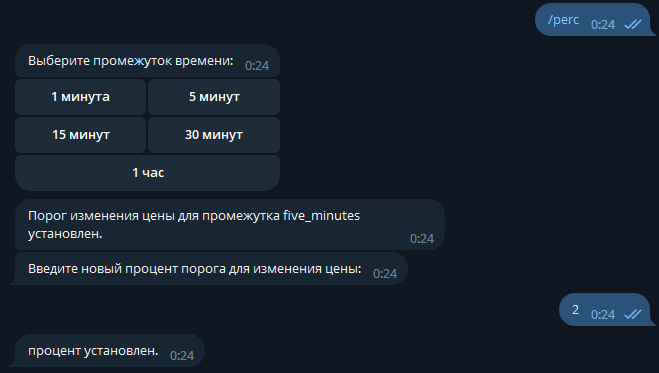


Рисунок 3.8 – команда perc

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 3.9.



Рисунок 3.9 – 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

При тестировании чат-бота использовались встроенные возможности языка программирования Python, включая пакет unittest. Тестовый сценарий был создан для эмуляции взаимодействия пользователя с чат-ботом в Telegram.

Процесс тестирования включал следующие шаги:

* Запуск чат-бота в режиме тестирования.
* Отправка сообщений, имитирующих действия пользователя, включая команды и текстовые сообщения.
* Проверка ответов чат-бота на соответствие ожидаемым результатам.
* Проверка обработки ошибок и исключительных ситуаций.

Для отслеживания действий чат-бота и проверки его взаимодействия с API Telegram использовались механизмы логирования.

В результате тестирования было подтверждено корректное функционирование чат-бота в Telegram и соответствие его ответов ожидаемому поведению. Никаких дефектов или проблем, которые могли бы повлиять на работу бота, не было обнаружено.

## Выводы по главе

В результате данной главы было достигнуто следующее:

* Создан аккаунт бота в Telegram с использованием платформы BotFather. Были выполнены все необходимые шаги, включая выбор имени, создание уникального username и получение токена для идентификации бота при взаимодействии с Telegram Bot API.
* Использованы ключевые библиотеки: aiogram, aiohttp и asgiref. Эти библиотеки обеспечивают удобные инструменты для создания ботов, параллельные HTTP-запросы и асинхронные вызовы веб-приложений.
* Определены таблицы базы данных: TelegramUser, Portfolio, UserCoin и CoinHistory. Каждая таблица отвечает за хранение информации о пользователях, портфелях, вложениях в криптовалюты и истории изменения цены.
* Проведено тестирование функциональности чат-бота с использованием встроенных средств Python, включая пакет unittest. Были проверены различные сценарии взаимодействия пользователя с ботом, а также корректность работы базы данных. Результаты тестирования подтвердили отсутствие дефектов в функционировании бота.

Данная глава работы предоставила основу для успешного использования необходимых библиотек, определения структуры базы данных и проведения тестирования. Это обеспечило надежность и функциональность чат-бота в рамках поставленных задач.

Заключение

В рамках выпускной квалификационной работы был разработан и реализован чат-бот для мессенджера Telegram, целью которого является предоставление удобных функций управления криптовалютным портфелем. Этот инструмент позволяет пользователям добавлять и продавать монеты, просматривать текущее состояние портфеля, полностью его очищать, устанавливать пороги изменения цен и получать уведомления о достижении этих порогов.

Все основные задачи и цели были достигнуты. В результате исследования существующих сервисов управления криптовалютными портфелями, выбора подходящих инструментов, разработки функциональных возможностей и тестирования, созданный чат-бот представляет собой надежное и функциональное решение.

Тем не менее, существуют возможности для дальнейшего развития и улучшения проекта. В будущем можно рассмотреть интеграцию с другими криптовалютными биржами для расширения функциональности или добавление аналитического функционала на основе исторических данных.

Тестирование проекта было выполнено с использованием модульных тестов, которые подтвердили корректность работы инструмента и отсутствие каких-либо дефектов или проблем.

В заключении, созданный в рамках выпускной квалификационной работы чат-бот является полезным и эффективным инструментом для управления криптовалютными портфелями через популярный мессенджер. Проект отвечает всем требованиям и целям работы, но его дальнейшее усовершенствование и расширение функциональности, безусловно, могут повысить его привлекательность и применение в реальных условиях.

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).