Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт Вычислительной математики и информационных технологий

ОТЧЕТ

по проектно-технологической (производственной) практике

Обучающийся Фамилия Имя Отчество \_гр.09-132\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО студента) (Группа) (Подпись)

Научный руководитель: доцент кафедры САИТ,

канд. физ.-мат. наук Фамилия И.О.

(Подпись)

Руководитель практики от кафедры:

Ст.преподаватель КСАИТ Фамилия И.О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Оценка за практику \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Дата сдачи отчета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Казань – 2025

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc193797071)

[1. Общая схема системы 5](#_Toc193797072)

[2. Используемые технологии 7](#_Toc193797073)

[3. Пользовательские требования 9](#_Toc193797074)

[4. Архитектура системы 11](#_Toc193797075)

[5. Функциональные компоненты 14](#_Toc193797076)

[5.1. Портфолио и инструменты 14](#_Toc193797077)

[5.2. Уведомления и состояние рынка 15](#_Toc193797078)

[5.3. Статистика и графики 16](#_Toc193797079)

[5.4. Настройка сигналов и торговый робот 17](#_Toc193797080)

[5.5. Дивиденды и база знаний 18](#_Toc193797081)

[6. Начальная демонстрация работы системы 20](#_Toc193797082)

[ЗАКЛЮЧЕНИЕ 22](#_Toc193797083)

[СПИСОК ЛИТЕРАТУРЫ 24](#_Toc193797084)

[ПРИЛОЖЕНИЯ 25](#_Toc193797085)

[Приложение 1. Код создания торгового поручения на покупку 25](#_Toc193797086)

[Приложение 2. Функция вычисления изменения цены 26](#_Toc193797087)

ВВЕДЕНИЕ

Производственная практика проходила на кафедре системного анализа и информационных технологий Института вычислительной математики и информационных технологий КФУ с 10 марта 2025 года по 12 апреля 2025 года.

Инвестирование представляет собой процесс вложения капитала в различные активы с целью получения прибыли. В современном мире финансовые рынки характеризуются высокой волатильностью, что делает процесс управления активами сложным и требующим значительных временных затрат. Для успешного ведения инвестиционной деятельности необходимо анализировать рыночные данные, оценивать риски, строить прогнозы и своевременно принимать торговые решения.

Одним из способов автоматизации этих процессов является алгоритмический трейдинг, который предполагает использование специальных программных решений для анализа рынка и выполнения сделок без участия человека. Такие системы позволяют оперативно реагировать на изменения рыночной конъюнктуры, снижая влияние человеческого фактора и повышая эффективность управления инвестициями.

Однако управление активами включает в себя не только торговые операции. Важно не только своевременно открывать и закрывать позиции, но и анализировать эффективность инвестиционного портфеля, отслеживать дивиденды, учитывать статистику прошлых сделок, а также получать оперативные данные о состоянии рынка. Без автоматизированных инструментов такие задачи требуют значительных временных затрат и высокой концентрации внимания, что усложняет процесс принятия решений.

Торговая стратегия — это совокупность правил и алгоритмов, определяющих моменты входа и выхода из сделок. Эффективные стратегии базируются на анализе рыночных данных, индикаторов и сигналов, помогающих инвесторам принимать обоснованные решения. Однако самостоятельный анализ данных и торговля вручную требуют постоянного мониторинга и глубоких знаний в области финансовых рынков. Поэтому интеграция алгоритмов обработки данных и автоматизированных систем уведомлений позволяет снизить нагрузку на пользователя и предоставить ему удобные инструменты для управления капиталом.

В связи с этим целью практики является разработка системы, которая обеспечит выполнение торговых операций и комплексное управление инвестиционным портфелем, предоставляя пользователю необходимые инструменты для эффективного анализа и принятия решений. Такая система должна предоставлять пользователю актуальную информацию о состоянии его активов, анализировать рыночные данные, поддерживать индивидуальные стратегии, а также уведомлять о значимых изменениях на рынке. Гибкость системы должна позволять пользователю настраивать параметры сигналов, устанавливать уровни риска и получать отчеты о финансовых результатах.

Для достижения указанной цели были определены следующие задачи:

1. определение ключевых требований к системе на основе анализа потребностей пользователей,
2. проектирование архитектуры системы,
3. разработка функционала для каждого отдельного модуля,
4. интеграция с внешними источниками данных для получения актуальной информации о рынке,
5. реализация алгоритмов для прогнозирования и оптимизации торговых стратегий,
6. создание пользовательского интерфейса для удобного взаимодействия с системой,
7. тестирование системы на корректность работы.
8. Общая схема системы

При проектировании системы необходимо учитывать общую структуру взаимодействия ее компонентов, поскольку это напрямую влияет на выбор используемых технологий и инструментов. На рисунке 1 представлена схема, которая отражает ключевые этапы обработки запросов пользователя и взаимодействия внутренних модулей системы.

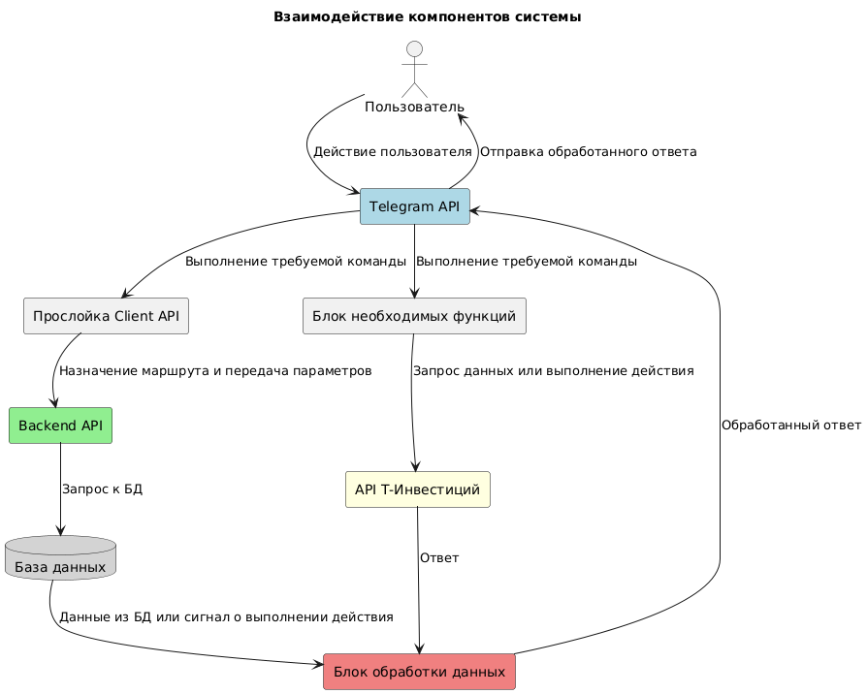


Рисунок - Схема взаимодействия компонентов системы

Как можно увидеть из схемы, процесс работы системы включает несколько важных этапов:

* инициирование команды пользователем. Пользователь взаимодействует с Telegram-ботом, отправляя запрос или команду. Эти данные передаются через Telegram API с использованием библиотеки telebot. Запросы могут направляться либо к backend-части приложения через клиентскую прослойку, либо напрямую к методам API системы Т-Инвестиций через заранее разработанные функции;
* обработка backend-частью. Если запрос передается на backend, то выполняется обращение к базе данных и производится выполнение требуемой операции с возможным возвратом результата либо подтверждением завершения задачи;
* обращение к Т-Инвестициям. Если запрос направлен к API Т-Инвестиций, результатом может быть получение требуемых данных или выполнение операции. В пример можно привести возврат ценовых свечей или размещение торгового поручения;
* подготовка ответа пользователю. Независимо от маршрута обработки данных собранные данные приводятся в удобочитаемый вид и отправляются пользователю через Telegram-бот.

Данная схема детализирует взаимодействие между компонентами системы и показывает зависимости, которые необходимо учитывать при выборе технологий. Реализация такого взаимодействия требует использования инструментов, обеспечивающих стабильную работу различного рода подключений, а также соответствующих библиотек для обработки запросов к Telegram и внешним сервисам.

1. Используемые технологии

Выбор каждой технологии обоснован с точки зрения функциональности, удобства использования, масштабируемости и производительности. Основные инструменты включают язык программирования Python, систему управления базами данных SQLite, контейнеризацию с Docker, использование web-фреймворка FastAPI, взаимодействие с Telegram API с помощью библиотеки telebot, а также работу с API брокера Т-Инвестиции.

Python был выбран в качестве основного языка программирования, поскольку он может предоставить именно те наборы инструментов и библиотеки, которые необходимы для разработки такого рода приложений.

В качестве базы данных была выбрана SQLite, поскольку она встроена в Python и не требует отдельной установки и настройки [1]. Это позволяет пользователю быстро развернуть и запустить приложение без дополнительных манипуляций. Несмотря на свою легковесность, SQLite поддерживает все основные функции реляционной базы данных. Поскольку при разработке системы будет использоваться фреймворк FastAPI, если пользователь по какой-то причине захочет перейти на другое решение для управления базой данных, такое как PostgeSQL или MySQL, то это не составит для него труда, поскольку данный фреймворк поддерживает работу с различными базами данных через ORM-библиотеки, такие как SQLAlchemy и Tortoise-ORM [2]. Это позволяет легко изменить тип используемой базы данных, просто обновив параметры подключения и, при необходимости, адаптировав модели данных.

Для обеспечения удобного развертывания приложения на различных платформах используется контейнеризация с Docker. Этот инструмент позволяет создать изолированную среду, в которой приложение работает независимо от операционной системы и её конфигурации. Благодаря контейнерам можно избежать проблем, связанных с несовместимостью библиотек и зависимостей, а также упростить процесс развертывания и обновления системы. Docker также обеспечивает отказоустойчивость приложения, предоставляя настраиваемые параметры автоматического перезапуска приложения в случае возникновения каких-либо ошибок [3].

Backend-система разработана на основе web-фреймворка FastAPI, который был выбран из-за своей высокой производительности, отсутствия лишних деталей и удобства разработки. Преимуществами данного фреймворка являются встроенная поддержка OpenAPI и автоматическая генерация документации, что упрощает тестирование и интеграцию с другими сервисами. Также FastAPI предоставляет удобные инструменты для валидации данных, снижая вероятность ошибок при их передаче [4].

Для взаимодействия с пользователем был выбран Telegram и библиотека telebot, поскольку Telegram является удобной платформой для автоматизации взаимодействия с пользователями, а библиотека telebot упрощает работу с API, предоставляя готовые методы для отправки сообщений и обработки различных команд.

API брокера Т-Инвестиции используется для получения рыночных данных, данных пользовательского портфеля и исполнения торговых операций. Данный брокер был выбран потому, что предоставляет открытое API с подробной документацией и готовые SDK для различных языков программирования, в частности для Python [5].

1. Пользовательские требования

Для построения эффективной системы первостепенной задачей является определение пользовательских требований. Эти требования определяют ожидаемый функционал, который система должна предоставлять в процессе работы. Основная цель — сделать использование системы максимально удобным, интуитивным и полезным для пользователя. Далее будут перечислены основные требования к функционалу системы:

* система должна предоставлять пользователю возможность получать состояние его инвестиционного портфеля, включая данные по каждому инструменту;
* необходимо, чтобы пользователь мог добавлять инструменты в персональную базу данных, получать список добавленных инструментов и удалять их при необходимости;
* система должна поддерживать функционал уведомлений, позволяя пользователю подписываться на обновления состояния рынка или изменения цен и отписываться от этих уведомлений при необходимости;
* пользователь должен иметь возможность получать информацию о состоянии рынка, включая обвалы или рост цен активов за определенный промежуток времени;
* система должна предоставить инструмент для настройки сигналов, которые отправляются пользователю на основании коэффициентов и параметров, выбранных вручную;
* система должна включать функционал торгового робота, который позволяет автоматизировать торговлю на основе настроенной стратегии. Пользователь должен иметь возможность гибко настраивать стратегию, выбирать временные промежутки для срабатывания сигналов, а также работать с реальным или тестовым счетом;
* система должна предоставлять отчеты по дивидендам за указанный период;
* пользователь должен иметь возможность получать построенные графики, которые будут отражать изменение цены и параметры сигналов;
* система должна включать базу знаний с краткими справками и инструкциями по настройке отдельных возможностей системы;
* пользователь должен иметь доступ к статистике операций, включая количество сделок, положительную и отрицательную маржу, а также аналитику совершения операций по сигналам, тикерам и времени.

Эти пользовательские требования составляют основу проектируемой системы и определяют набор функциональных компонентов, которые будут реализованы.

1. Архитектура системы

Структура проекта построена на принципах модульности, что позволяет разделить различные аспекты функциональности на независимые компоненты. Это упрощает управление проектом, его дополнение новыми функциями и интеграцию с внешними системами. Основные папки и файлы организованы по их функциональному назначению, что можно увидеть в структуре самого проекта:

* models. В данной папке находятся файлы, которые отвечают за создание сущностей базы данных;
* schemas. В данной папке находятся файлы, отвечающие за схемы данных, которые обеспечивают валидацию и сериализацию данных;
* api. В данной папке находятся файлы, которые содержат в себе функции для взаимодействия с базой данных;
* bot. Данная папка содержит файл с экземпляром класса бота;
* config. Данная папка содержит в себе файлы, отвечающие за конфигурацию базы данных и планировщиков задач;
* graphics. Данная папка содержит в себе файлы, отвечающие за создание графиков;
* handlers. Данная папка содержит в себе обработчики команд каждого отдельного модуля;
* log. Данная папка содержит в себе файл с логами системы;
* orders. Данная папка содержит в себе файлы, отвечающие за торговые поручения. Пример программного кода для создания торгового поручения на покупку представлен в [приложении 1](#Приложение_1);
* signals. Данная папка содержит в себе файлы, отвечающие за различные сигналы;
* store. Данная папка содержит файлы для временного хранения данных;
* strategy. Данная папка содержит в себе файлы, отвечающие за управление торговой стратегией;
* utils. Данная папка содержит в себе файлы, которые предоставляют вспомогательные функции;
* Dockerfile и docker-compose.yml. Файлы для создания и запуска docker-образа.

Для хранения информации была разработана база данных. Структура сущностей в базе данных показана на рисунке 2.

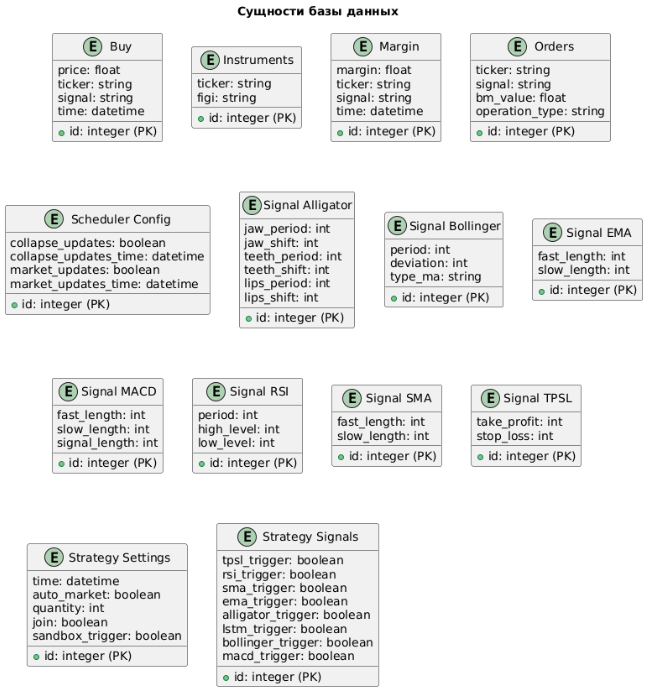


Рисунок 2 - Структура сущностей в базе данных

Каждая сущность, которая начинается с Signal, отвечает за хранение и изменение параметров определенного сигнала, который используется при построении графиков или в торговой стратегии. Buy и Margin отвечают за хранение данных о покупках и продажах ценных бумаг. Orders хранит в себе промежуточные данные о выставленных торговых заявках. Instruments хранит в себе тикер ценной бумаги и ее FIGI-идентификатор. Scheduler Config нужен для хранения данных об установленных настройках уведомлений для планировщиков задач. Strategy Settings хранит данные об установленных настройках для стратегии, а Strategy Signals о сигналах, используемых в стратегии.

1. Функциональные компоненты

Функциональные компоненты системы предназначены для автоматизации процессов управления инвестициями и упрощения взаимодействия пользователя с необходимыми данными. Каждая компонента реализует определенную задачу, обеспечивая эффективное взаимодействие с API и базой данных, а также предоставляя информацию в удобном и понятном формате. В их разработке использовались обработчики команд, методы интеграции с внешним API и таблицы базы данных, структурированные в соответствии с требованиями системы.

5.1. Портфолио и инструменты

Для просмотра портфолио и управления инструментами пользователей были созданы обработчики portfolio и instruments, каждый из которых выполняет свои определенные функции в системе.

Обработчик portfolio предоставляет пользователям возможность отслеживать текущее состояние их портфеля. Основной задачей этого обработчика является получение данных о портфеле пользователя через API Т-Инвестиций, что осуществляется в зависимости от типа счета, выбранного пользователем — боевого или песочницы. Важно, что для каждого типа счета используются различные API-методы, что позволяет пользователю получать актуальную информацию о стоимости активов как для реальных, так и для виртуальных счетов.

При обработке данных портфеля, система формирует отчет по каждому активу. Это позволяет пользователю оценить эффективность его инвестиций и принять решения о возможных корректировках в портфеле.

Обработчик instruments отвечает за управление самими инструментами, с которыми работает пользователь. Данная компонента включает в себя четыре основных действия:

* добавить инструмент. Данная команда реализует добавление инструмента в базу данных. Пользователь вводит тикер инструмента, после чего ищется его FIGI-идентификатор. Если он найдет и данный тикер существует, то инструмент записывается в базу данных;
* получить мои инструменты. Команда, которая вызывает метод для получения всех инструментов из базы данных и отправляет их пользователю;
* удалить мои инструменты. Команда, которая вызывает метод удаления всех инструментов из базы данных;
* удалить инструмент. При выборе данной команды пользователю сначала присылается интерактивный список его инструментов. При нажатии на конкретный инструмент он будет удален из базы данных посредством вызова нужного метода.

5.2. Уведомления и состояние рынка

Для реализации компоненты уведомлений компоненты был создан обработчик notification и соответствующая таблица Scheduler Config в базе данных. При проверке изменения цены вызывается специальный метод, который получает ценовые свечи инструмента на данный момент и введенное пользователем время назад, после чего сравнивает их значения. Программный код функции изменения цены представлен в [приложении 2](#Приложение_2). У пользователя есть четыре доступных команды:

* подписка на падения рынка. При выборе данной команды пользователю предложат записать временной промежуток, через который ему будут отправляться уведомления. После указания промежутка нужные данные добавляются в соответствующую таблицу. При перезапуске приложения конфигуратор автоматически настроит планировщики задач по указанным пользователям параметрам. Данные уведомления будут присылать данные по инструментам пользователя, которые за указанный промежуток времени потеряли в стоимости какой-то процент;
* отписка от падений рынка. При выборе данной команды пользователь отпишется от соответствующих уведомлений, триггер в базе данных изменит свое значение на нулевое, а время удалится;
* подписка на обновления рынка. При выборе данной команды пользователю предложат записать временной промежуток, через который ему будут отправляться уведомления. После указания промежутка нужные данные добавляются в соответствующую таблицу. При перезапуске приложения конфигуратор автоматически настроит планировщики задач по указанным пользователям параметрам. Данные уведомления будут присылать данные по инструментам пользователя, которые обновили свои показатели в какую-либо сторону;
* отписка от обновлений рынка. При выборе данной команды пользователь отпишется от соответствующих уведомлений, триггер в базе данных изменит свое значение на нулевое, а время удалится.

Для реализации компоненты получения состояния рынка был создан обработчик market. Данный обработчик содержит в себе три доступные команды:

* получить обвал рынка. При выборе пользователем данной команды ему предлагается сначала выбрать промежуток времени, за который необходимо получить свечи, а затем процент падения, на который должна была обвалиться ценная бумага. После этого пользователю отправятся обработанные и отформатированные данные в удобном для восприятия формате;
* получить рост рынка. Данная команда имеет такую же концепцию, как и обвал рынка, только указывается процент роста, а не падения;
* получить изменение состояния рынка. При выборе данной команды пользователь получает данные о всех движениях цен инструментов из его базы данных за введенный период.

5.3. Статистика и графики

Для компоненты статистики был создан обработчик statistics и таблицы Buy и Margin. При выборе пользователем данной команды ему будет предложено либо ввести период, за который он желает получить статистику, либо получить общую статистику по покупкам и продажам. Если данных для отображения статистики нет, пользователю отправится соответствующее уведомление. Если же статистику рассчитать возможно, то пользователю отправятся графики количества покупок и продаж по определенным тикерам, по сигналам, по времени и соотношение положительной и отрицательной маржи, а также общая сумма покупок и маржи.

Для компоненты графиков был реализован обработчик graphics. Если пользователь выберет данную команду, ему будет предложено выбрать сигнал и промежуток времени для получения свечей. После получения ценовых свечей будет построен специальный график, на котором будут изображены ценовые свечи и рассчитанные в модуле signals значения сигнала, выбранного пользователем.

5.4. Настройка сигналов и торговый робот

Для компоненты настройки сигналов был реализован обработчик signals и таблицы, соответствующие каждому определенному сигналу. При выборе данной команды пользователю присылается список всех доступных сигналов. После того, как пользователь выбрал определенный сигнал, ему будут присылаться сообщения с просьбой ввести значения для каждого необходимого поля сигнала. Какие значения вводить – пользователь выбирает сам. От введенных значений будет зависеть обработка сигналов и торговая стратегия. Для каждого типа значений были реализованы специальные валидаторы, не дающие ввести некорректные данные.

Для реализации компоненты торгового робота был создан обработчик bot и таблицы Strategy Signals и Strategy Settings. Стратегия срабатывает для каждого инструмента из базы данных пользователя через определенный промежуток времени, который так же указывается пользователем. Для каждого инструмента проверяется каждый включенный в стратегию сигнал и проверяется возвращаемое значение сигнала. После проверки всех условий определяется требуемая операция, то есть покупка, продажа или удержание. Всего пользователю доступны четыре команды:

* настроить стратегию. При выборе данной команды пользователю отправится список всех доступных сигналов. Пользователь может включить в стратегию любой сигнал, выбрав его, при условии, что данный сигнал был настроен. После выбора сигналов пользователь получит список временных промежутков, через которые необходимо, чтобы стратегия срабатывала. После этого пользователь должен выбрать, включить ли автоматическую торговлю или только присылать рекомендации по покупке, продаже или удержанию ценной бумаги. Далее пользователь должен будет указать количество ценных бумаг, которые будут продаваться и покупаться, а также выбрать, совершать торговую операцию при срабатывании хотя бы одного сигнала или всех сигналов сразу;
* отключить стратегию. После выбора пользователем данной команды торговая стратегия будет отключена;
* выбор счета. При выборе пользователем данной команды он может поменять счет, на котором будут совершаться расчеты и торговые операции. Если пользователь выберет «боевой» счет, то все торговые поручения будут выставляться на его основном счете с его действительным балансом, если же будет выбрана «песочница», то торговые операции будут производиться на счете с ненастоящими денежными средствами, но по реальным рыночным данным. Это может быть полезно, если пользователь хочет проверить свою стратегию в действии;
* информация о «песочнице». При выборе данной команды пользователь может получить портфолио счета своей «песочницы», либо пополнить баланс.

5.5. Дивиденды и база знаний

Для компоненты дивидендов был реализован обработчик dividends. После выбора пользователем данной команды ему будет предложено указать период, за который он хочет посмотреть дивиденды по ценным бумагам, находящимся у него в базе данных. После введения периода специально разработанный метод из модуля utils вызовет для каждого инструмента функцию получения данных о дивидендах, сформирует отчет и отправит его пользователю.

Для компоненты базы знаний был создан обработчик knowledge\_base. При выборе данной команды пользователю будет предложен список всех функциональных компонент. При выборе определенной функциональной компоненты в базе знаний пользователю будет присылаться информация по данной компоненте. Например, пользователь может узнать, как настроить стратегию или определенный сигнал, получить рекомендации и базовые значения по их настройке или получить информацию по взаимодействию с определенным компонентом системы.

1. Начальная демонстрация работы системы

После разработки системы попробуем провести запуск и выполнить несколько команд. Состояние терминала после успешного запуска системы с настроенными планировщиками уведомлений и неактивной стратегией можно увидеть на рисунке 3.

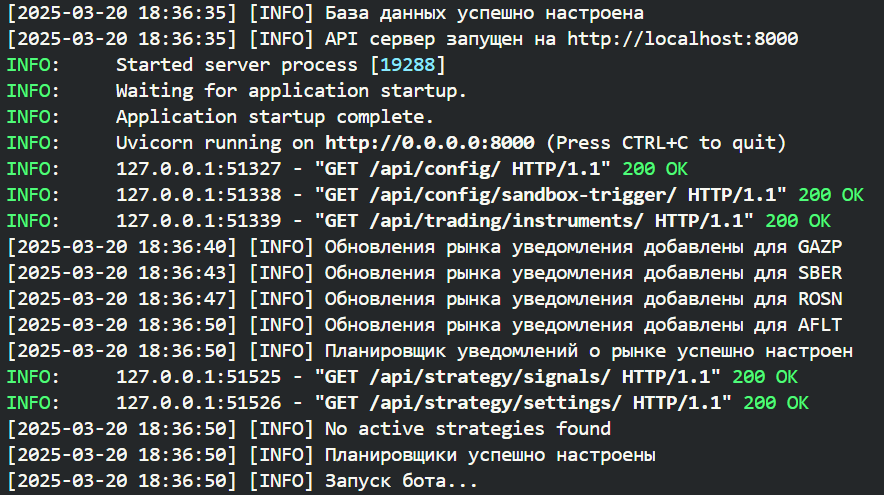


Рисунок - Состояние терминала после успешного запуска системы

Как можно увидеть, успешно происходит настройка базы данных, после чего запускается backend-сервер, настраиваются все планировщики и бот переходит в режим polling, непрерывно ожидая сообщения от пользователя.

После успешного запуска системы выполним команду по получению портфолио. После нажатия пользователем на кнопку «Получить портфолио» ему в ответ приходит сообщение, часть которого представлена на рисунке 4.

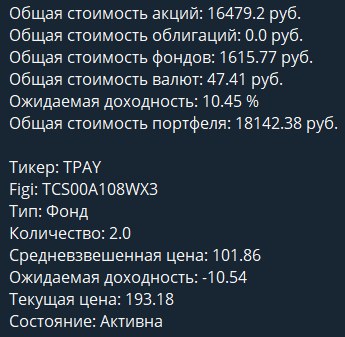


Рисунок - Портфолио пользователя

Как можно заметить, пользователю приходит отчет по его портфелю, включая общую стоимость различных ценных бумаг, ожидаемую доходность, а также информация по каждому отдельному инструменту.

При запуске приложения были настроены планировщики задач, которые отправляют уведомления через указанные пользователем промежутки времени. По прошествии десяти минут, система прислала уведомления с изменениями цен инструментов пользователя. Примеры уведомлений можно увидеть на рисунке 5.

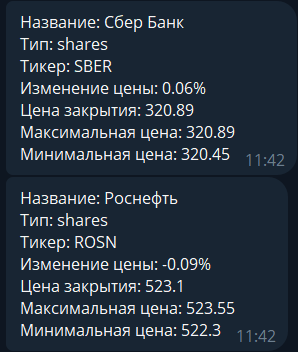


Рисунок - Уведомления об изменении цен инструментов

Таким образом, можно сказать, что система успешно выполняет поставленные в примерах задачи, включая обработку некоторых пользовательских команд, получение и отображение данных по портфелю, а также автоматическую отправку уведомлений о рыночных изменениях.

ЗАКЛЮЧЕНИЕ

В ходе практики была разработана интерактивная система для автоматизированного трейдинга и управления финансовыми активами. Работа включала в себя проектирование архитектуры системы, разработку backend-сервера на FastAPI, создание Telegram-бота на основе библиотеки Telebot, интеграцию с API брокера Т-Инвестиции, а также реализацию взаимодействия с базой данных SQLite.

В процессе разработки была реализована система хранения и обработки рыночных данных, а также механизмы анализа и принятия торговых решений на основе сигналов. Были созданы обработчики команд, позволяющие пользователю получать актуальную информацию о состоянии портфеля, рыночных инструментах, дивидендах и статистике сделок. Внедрен функционал настройки уведомлений, позволяющий пользователю получать автоматические оповещения о рыночных изменениях.

Система поддерживает выполнение торговых стратегий с возможностью автоматического размещения заявок. Была реализована структура базы данных, обеспечивающая хранение информации о сигналах, инструментах, стратегиях и заявках. Для удобства развертывания системы был подготовлен Docker-контейнер с конфигурацией в Dockerfile и docker-compose.yml, что позволяет запускать приложение в изолированной среде.

За период практики были приобретены следующие компетенции (таблица 1):

Таблица - Приобретенные компетенции

|  |  |  |
| --- | --- | --- |
| Компетенция | Расшифровка компетенции | Описание приобретенных знаний, умений и навыков |
| УК-1 | Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач | Применял системный подход для разработки и интеграции функционала системы, а также анализа и выбора оптимальных решений. |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Компетенция | Расшифровка компетенции | Описание приобретенных знаний, умений и навыков |
| УК-2 | Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений | Определил ключевые задачи разработки системы, выбрал оптимальные методы реализации с учетом ограничений и доступных ресурсов. |
| УК-6 | Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни | Организовывал процесс разработки с четким планированием задач и соблюдением сроков, улучшая свои навыки управления временем. |
| ПК-8 | Выполнение работ по созданию (модификации) и сопровождению информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы | Разработал систему для автоматизации задач, интегрируя внешние сервисы и создавая алгоритмы для обработки данных. |
| ПК-9 | Создание и сопровождение требований и технических заданий на разработку и модернизацию систем и подсистем малого и среднего масштаба и сложности | Сформулировал требования и написал технические задания для разработки и настройки системы с учетом всех функциональных потребностей. |

В ходе практики были приобретены знания и навыки, связанные с разработкой и развертыванием автоматизированных систем для финансового анализа и трейдинга. Проект позволил на практике освоить современные подходы к созданию web-сервисов, обработке данных, интеграции с внешними API и организации удобного развертывания приложения.

В процессе работы была изучена архитектура построения backend-систем, взаимодействие между различными компонентами приложения и способы оптимизации обработки данных в режиме реального времени. Разработка Telegram-бота помогла освоить работу с внешними API и реализацию интерактивного интерфейса для пользователей.

Кроме того, практика способствовала развитию навыков планирования и организации рабочего процесса, поскольку проект включал в себя несколько взаимосвязанных компонентов, требующих последовательной и согласованной разработки.

СПИСОК ЛИТЕРАТУРЫ

1. SQLite Documentation [Электронный ресурс]. – 2025. — URL: https://www.sqlite.org/docs.html (дата обращения 10.03.2025).
2. SQLAlchemy – The Database Toolkit for Python [Электронный ресурс]. – 2023. — URL: https://www.sqlalchemy.org/ (дата обращения 31.03.2025).
3. Docker: как развернуть фуллстек-приложение [Электронный ресурс]. – 2019. — URL: https://habr.com/ru/articles/448094/ (дата обращения 05.04.2025).
4. FastAPI Documentation [Электронный ресурс]. – 2024. — URL: https://fastapi.tiangolo.com/ (дата обращения 24.03.2025).
5. T-Bank Invest API [Электронный ресурс]. – 2023. — URL: https://tinkoff.github.io/investAPI/ (дата обращения 12.03.2025).

ПРИЛОЖЕНИЯ

Приложение 1. Код создания торгового поручения на покупку

avaliable\_lots = calc\_avaliable\_lots(token, figi, client, sandbox\_method)

# Проверка на наличие позиции в портфеле

if (avaliable\_lots > 0):

print(f"Позиция {figi} уже в портфеле, ждем сигнала к продаже...")

return

# best or fast

price\_sell, price\_buy = get\_current\_price(figi, client, 'fast')

# Проверка баланса

if check\_enough\_currency(token, figi, client, price\_buy, quantity, sandbox\_method):

# Генерация id

order\_id = str(uuid.uuid4())

r = sb.post\_sandbox\_order(

figi=figi,

quantity=quantity,

price=price\_buy,

account\_id=account\_id,

order\_id=order\_id,

direction=OrderDirection.ORDER\_DIRECTION\_BUY,

order\_type=OrderType.ORDER\_TYPE\_LIMIT,

)

# Создаем новый ордер через API клиент

api\_client.add\_order({

"order\_id": order\_id,

"ticker": ticker,

"signal": signal,

"bm\_value": cast\_money(price\_buy),

"operation\_type": operation

})

logger.info(r)

logger.info(f"Создаем заявку на покупку по цене {cast\_money(price\_buy)}")

except RequestError as e:

logger.error(str(e))

Приложение 2. Функция вычисления изменения цены

def get\_price\_change\_in\_current\_interval(figi, start\_time, end\_time, candle\_interval):

    """

    Вычисляет изменение цены для заданного интервала времени.

    :param figi: Строка, идентификатор финансового инструмента.

    :param start\_time: Начальное время интервала.

    :param end\_time: Конечное время интервала.

    :param candle\_interval: Интервал свечи для получения данных.

    """

    try:

        data = get\_historic\_candles(figi, start\_time, end\_time, candle\_interval)

        df = create\_df(data.candles)

        # Проверяем, есть ли данные в DataFrame

        if df.empty:

            print("Нет данных за указанный период")

            return None

        # Получаем цену открытия и цену закрытия

        open\_price = df['open'].iloc[0]

        close\_price = df['close'].iloc[-1]

        max\_price = df['high'].max()

        min\_price = df['low'].min()

        # Рассчитываем изменение цены

        price\_change = close\_price - open\_price

        # Рассчитываем процентное изменение цены

        price\_change\_percent = (price\_change / open\_price) \* 100

        # Выводим результат

        logger.info(f"Изменение цены за период: {price\_change:.2f} ({price\_change\_percent:.2f}%)")

        logger.info(f"Максимальная цена: {max\_price:.2f}\n Минимальная цена: {min\_price:.2f}")

        # Возвращаем результат

        return price\_change, price\_change\_percent, max\_price, min\_price, close\_price

    except RequestError as e:

        logger.error(f"Ошибка запроса: {e}")

        return None