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 научно-исследовательской работе (производственной) практике

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

(ФИО студента) (Группа) (Подпись)

Научный руководитель: доцент кафедры САИТ,

канд. физ.-мат. наук Фамилия И.О.

(Подпись)

Руководитель практики от кафедры:

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

(Подпись)

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

(Подпись)

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

Казань – 2025

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc197950488)

[1. Сравнительный анализ разработанной системы с системами-аналогами 4](#_Toc197950489)

[2. Анализ пользовательского взаимодействия с системой 7](#_Toc197950490)

[3. Анализ архитектуры разработанной системы 11](#_Toc197950491)

[ЗАКЛЮЧЕНИЕ 15](#_Toc197950492)

[СПИСОК ЛИТЕРАТУРЫ 18](#_Toc197950493)

ВВЕДЕНИЕ

Производственная практика проходила на кафедре системного анализа и информационных технологий Института вычислительной математики и информационных технологий КФУ с 06 мая 2025 года по 27 мая 2025 года.

В ходе предыдущих практик была разработана и протестирована система, направленная на автоматизацию и оптимизацию процессов взаимодействия пользователя с финансовыми инструментами. Данная система представляет собой программный комплекс, обеспечивающий функции анализа инвестиционного портфеля, генерации торговых сигналов, получения уведомлений о дивидендах и управления торговыми роботами. Она предоставляет пользователю удобный интерфейс в виде Telegram-бота и возможность гибкой настройки стратегий и сигналов.

В контексте данной работы возникла необходимость в проведении анализа созданной системы. Это обусловлено важностью оценки проделанной работы и конечной разработанной системы, выявления сильных и слабых сторон, а также определения направлений для дальнейшего улучшения.

Цель данной практики заключается в проведении комплексного анализа разработанной программной системы, который включает оценку ее архитектуры, функциональности, пользовательского взаимодействия, а также выявление достоинств и недостатков.

Для достижения поставленной цели были определены следующие задачи:

1. изучить и проанализировать системы-аналоги в сравнении с разработанной системой,
2. провести анализ системы с точки зрения пользователя,
3. провести анализ архитектуры разработанной системы,
4. выявить преимущества и недостатки системы,
5. сформулировать предложения по улучшению и возможному расширению функциональности.

1. Сравнительный анализ разработанной системы с системами-аналогами

На текущий момент существуют решения в области автоматизации торговли на финансовых рынках, в том числе и с открытым исходным кодом. Однако при более глубоком изучении можно заметить, что такие решения имеют ряд недостатков, которые ограничивают их практическое применение. Главной проблемой подобных систем является отсутствие удобного и интуитивно понятного пользовательского интерфейса. В большинстве случаев взаимодействие с программой осуществляется напрямую через исходный код, что требует от пользователя определённых умений программирования. Например, настройку конфигурации для торгового робота в аналогичной программе можно увидеть на рисунке 1.

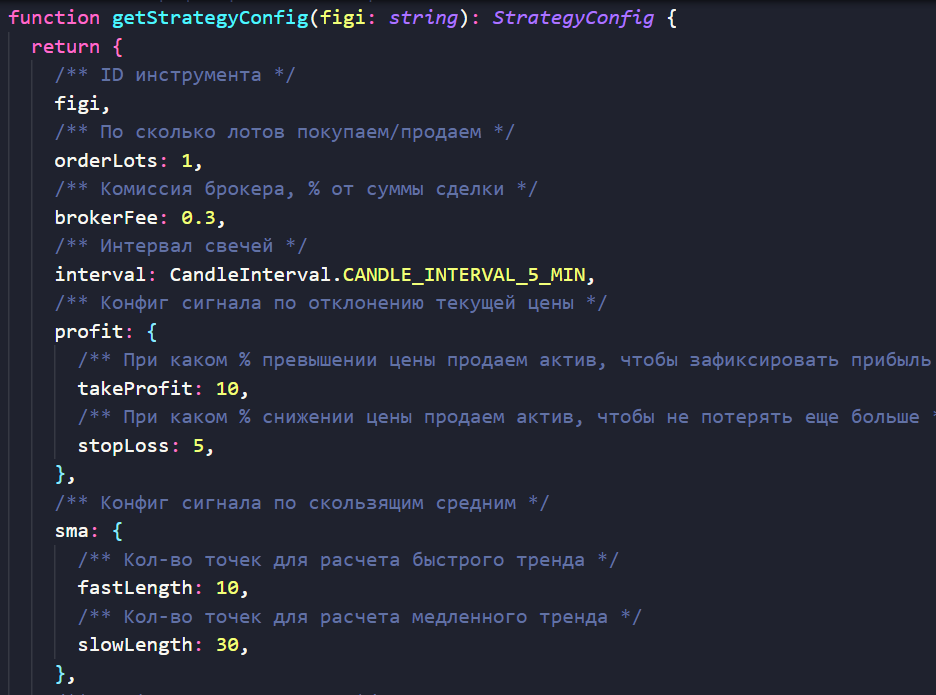


Рисунок - Функция настройки торгового робота

Это создает некоторую преграду для менее технически обученных пользователей, желающих воспользоваться системой, но не обладающих достаточными знаниями в области разработки программного обеспечения.

Кроме того, даже если пользователь обладает необходимыми навыками, структура подобных проектов может оказаться неудобной для модификации. Отсутствие визуального отображения процессов, а также невозможность оперативной настройки параметров без изменения исходного кода делает использование таких систем тяжелым в понимании и подверженным ошибкам. Даже незначительные изменения могут повлечь некорректную работу программы, так как в данном случае не предусматриваются средства валидации вводимых данных, подсказки или защита от пользовательских ошибок.

Многие из существующих решений можно разбить на две категории. Такие решения либо предоставляют чрезмерно ограниченный набор функций, либо, наоборот, включают в себя излишне сложные и перегруженные модули. В первом случае такие системы не способны удовлетворить базовые потребности трейдера, так как выполняют лишь узкий спектр задач, чаще всего — запуск и остановку торгового робота. Во втором случае речь идёт о программных продуктах, в которых реализовано множество различных функций, однако их интерфейс обычно перегружен, плохо структурирован и ориентирован на профессиональных пользователей. Такие приложения дополнительно требуют значительного времени для изучения и настройки. Пример реализации интерфейса такого приложения можно увидеть на рисунке 2.

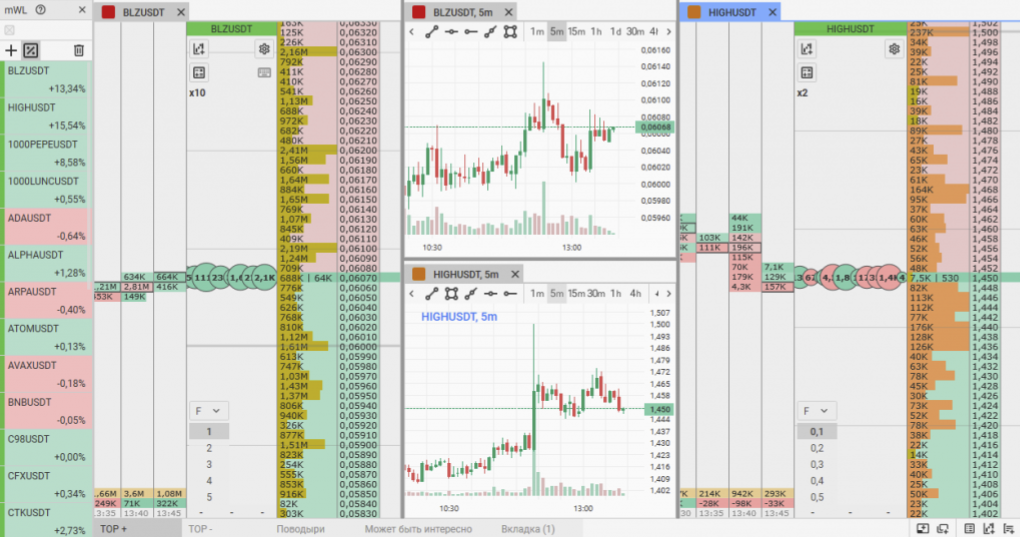


Рисунок - Пример интерфейса программы для торговли на бирже

Особого внимания заслуживают коммерческие системы, которые обладают широкими возможностями, имеют проработанный интерфейс и обеспечивают постоянную техническую поддержку. Однако в таких решениях отсутствует доступ к исходному коду. Пользователь лишен возможности адаптировать программу под свои индивидуальные потребности, проследить алгоритмы работы системы или удостовериться в корректности ее функционирования. Это создает ситуацию зависимости от разработчиков, а также снижает уровень доверия к таким продуктам.

На фоне вышеперечисленных недостатков разработанная система занимает промежуточное положение между простыми скриптами для запуска торговых ботов и крупными профессиональными платформами. Она включает в себя открытость архитектуры, удобство применения и расширенную функциональность. Приложение может рассматриваться как универсальный помощник трейдера, предоставляющий доступ как к торговым алгоритмам, так и ко всем необходимым вспомогательным сопутствующим функциям.

Основной недостаток и в то же время преимущество данной системы – это ее промежуточное положение. Данная система может подойти далеко не каждому пользователю, так как она не может в полной мере конкурировать с полноценными платформами, интегрированными с большим числом брокеров, обладающими развитой системой аналитики и многопоточной архитектурой. В то же время она превосходит простые решения в виде скриптов, но это тоже необходимо далеко не каждому пользователю.

1. Анализ пользовательского взаимодействия с системой

Разработанная система ориентирована в первую очередь на конечного пользователя. Важным аспектом её оценки является анализ с точки зрения удобства, понятности и доступности взаимодействия со стороны потребителя. Как и любое программное обеспечение, данная система обладает как положительными, так и отрицательными сторонами, влияющими на пользовательский опыт.

Одним из ключевых преимуществ системы является возможность гибкой настройки параметров, что можно увидеть на рисунке 3.

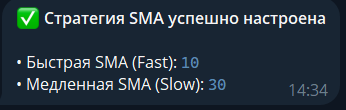


Рисунок - Конечный этап настройки сигнала SMA

Такая настройка позволяет адаптировать поведение системы под индивидуальные предпочтения пользователя, что очень удобно в сфере автоматизированной торговли, где каждая стратегия может требовать уникального набора параметров. Пользователю доступен полный контроль над конфигурацией сигналов и условий исполнения торговых алгоритмов, что в совокупности создаёт пространство для экспериментов и поиска оптимальных решений, не ограничиваясь жёсткими рамками заранее заданной логики.

Из преимуществ системы также можно выделить наличие в ней «песочницы» — специального режима работы системы, предназначенного для тестирования различных сценариев без риска совершения реальных операций на бирже. Благодаря этой функции пользователь может проверить корректность работы настроенных им сигналов и провести полноценный анализ стратегий, оценить их эффективность и внести необходимые корректировки до перехода к реальной торговле. Варианты выбора счетов представлены на рисунке 4.

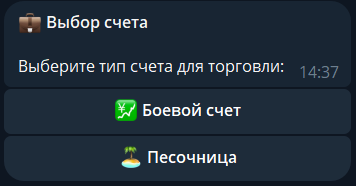


Рисунок – Выбор пользовательского счета

Интерфейс системы построен таким образом, чтобы все действия пользователя были сосредоточены в одном месте. Все пользовательские команды находятся в одной клавиатуре, разделенной на функциональные модули приложения. При взаимодействии с системой большого нагромождения сообщений не будет, так как работа с одним конкретным функциональным модулем происходит в рамках одного сообщения, которое меняет свой внешний вид и наполнение в зависимости от действий пользователя. Пример последовательного изменения сообщения можно увидеть на рисунке 5 и на рисунке 6.

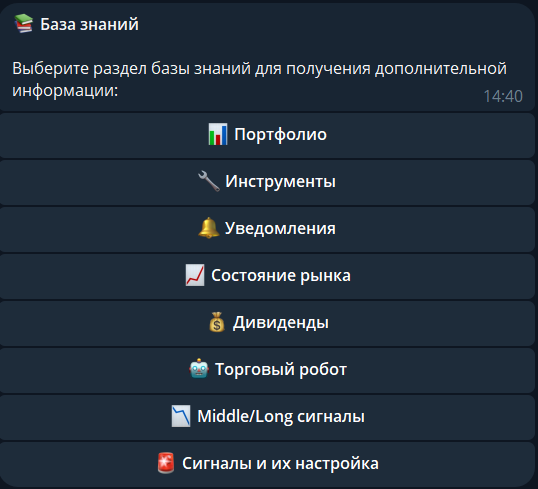


Рисунок - Начальное состояние текущего сообщения

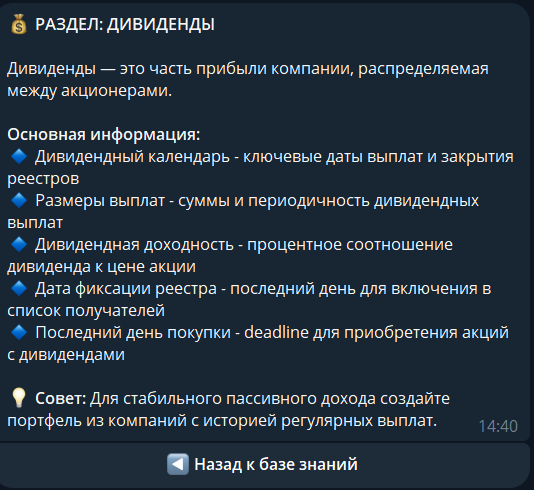


Рисунок - Конечное состояние текущего сообщения

Не менее важной частью пользовательского опыта является информирование. На каждую операцию у системы есть определенный ответ, который будет отправляться пользователю. Если торговый робот совершил какую-либо операцию на бирже, то пользователь получит об этом уведомление, что можно увидеть на рисунке 7. Если произошла какая-либо ошибка, пользователь будет об этом проинформирован.

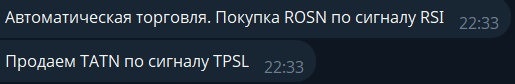


Рисунок - Уведомления об операциях на бирже

Наличие в системе базы данных обеспечивает сохранность всех пользовательских настроек и результатов торговли. Это делает возможным восстановление конфигурации при повторных запусках, а также дает пользователю возможности для построения анализа торговли, выявляя закономерности и оценивая эффективность принятых решений.

Несмотря на ряд безусловных достоинств, система также имеет и некоторые недостатки. Одной из самых ощутимых проблем является зависимость от стабильного интернет-соединения. При потере связи система может перестать выполнять запросы, а при запуске в локальном режиме без использования Docker-контейнера не предусмотрено автоматическое восстановление работы, что может стать критичным фактором в моменты активной торговли.

Стабильность работы интерфейса напрямую зависит от Telegram API, через который осуществляется взаимодействие. В отдельных случаях задержки между действиями пользователя и ответом системы могут быть заметны, что способно вызывать ощущение «подвисания» системы и снижать уровень удовлетворённости от использования. В дальнейшем планируется перенос системы с библиотеки telebot на более продвинутую библиотеку aiogram, поддерживающую асинхронное взаимодействие [1].

Как отдельный недостаток можно выделить ограниченность интерфейса. В виду ограничений интерфейса приложения telegram, не является возможным внедрить в приложение различные всплывающие подсказки, удобное навигационное меню или любые другие различные элементы интерфейса, облегчающие работу пользователей.

Дополнительной сложностью можно считать информационную перегруженность отдельных модулей. Например, при получении информации о портфеле пользователю выводятся все доступные данные, включая такие параметры, как средняя цена покупки или статус актива, что можно увидеть на рисунке 8. Для опытных пользователей такие сведения могут быть полезны, при этом начинающие трейдеры могут испытывать затруднения при интерпретации большого объёма информации.

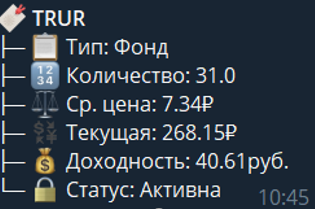


Рисунок - Пример позиции в портфолио

1. Анализ архитектуры разработанной системы

Архитектура разработанной системы представляет собой важную составляющую, которая определяет как внутреннюю организацию компонентов системы, так и ее дальнейшие перспективы масштабирования, сопровождения и адаптации под изменяющиеся требования.

В первую очередь необходимо сказать, что система построена на принципах модульности [2]. Структура приложения представлена на рисунке 9.

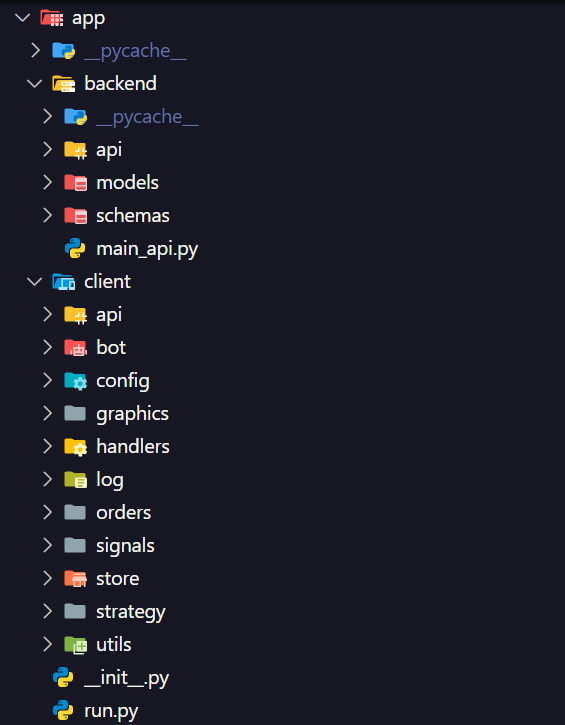


Рисунок - Структура приложения

Каждый функциональный компонент приложения выделен в отдельную папку, а логика внутри файлов структурирована так, чтобы обеспечивать изоляцию конкретных задач. Каждый модуль системы отвечает строго за свою область ответственности, что делает такой подход подходящим для лучшего пониманию структуры кода, упрощает масштабирование в будущем, а также облегчает тестирование и внедрение новых функций.

Несмотря на очевидные преимущества модульной структуры, данный подход имеет свои недостатки. Одним из них является избыточное количество модулей и хранилищ для различных компонентов, что приводит к увеличению объема кода и усложнению навигации в проекте. Таким подходом с разделением ответственности под каждый модуль возможно перегрузить систему избыточной детализацией, особенно если структура недостаточно стандартизирована или дублирует однотипные подходы в разных модулях. Стоит отметить, что модульность в текущей реализации не исключает наличие монолитности [3] на уровне всей архитектуры. Несмотря на логическую изоляцию компонентов и разделение ответственности, система по-прежнему функционирует как единое целое, где все модули запускаются и взаимодействуют внутри одного процесса.

Подобная архитектура в долгосрочной перспективе может стать ограничением. С внедрением многопользовательской архитектуры и ростом числа функциональных возможностей возникает необходимость в более гибком и масштабируемом подходе. Наиболее подходящим в таком контексте представляется переход к микросервисной архитектуре [4]. В рамках микросервисного подхода каждый функциональный модуль может быть преобразован в отдельный сервис, взаимодействующий с другими через API, что позволит упростить развертывание обновлений и ускорить масштабирование отдельных компонентов в зависимости от нагрузки.

Отдельной позицией можно рассмотреть реализацию пользовательского интерфейса. Основной средой взаимодействия пользователя с системой выступает Telegram, что делает приложение доступным и удобным в использовании на любом устройстве. Однако Telegram-интерфейс обладает как своими преимуществами, так и ограничениями. Он удобен для быстрой работы и обмена сообщениями, но ограничен в визуальных возможностях, таких как всплывающие окна, интерактивные графические элементы и сложные навигационные компоненты. Сознательно учитывая эти ограничения, при проектировании архитектуры было принято решение разделить систему на две ключевые части: backend и интерфейсную. Backend отвечает за выполнение бизнес-логики, обработку запросов, хранение данных и генерацию ответов. Интерфейсная часть, основанная на Telegram API, реализует пользовательские сценарии, принимая команды и отображая ответы в виде сообщений, кнопок и интерактивных элементов Telegram-бота.

Такой подход обеспечивает гибкость в дальнейшем развитии системы. Пользователь, обладающий необходимыми навыками, может заменить Telegram-интерфейс на любой другой — например, реализованный на базе фреймворков React, Vue или даже в виде мобильного приложения [5]. Весь набор бизнес-операций уже вынесен в виде API, и при необходимости может быть повторно использован или расширен.

Если обобщить результаты анализа по ключевым критериям, то преимущества и недостатки разработанной системы можно наглядно представить в виде следующей таблицы (таблица 1):

Таблица - Сравнительный анализ разработанной системы

|  |  |  |
| --- | --- | --- |
| Критерий | Преимущества | Недостатки |
| Независимость от разработчиков | Пользователь контролирует всё — нет зависимости от внешней технической поддержки, можно работать автономно | Необходимы базовые технические знания для устранения проблем |
| Архитектура | Модульная архитектура с разбиением на зоны ответственности | При дальнейшем расширении может возникнуть чрезмерная избыточность. Основа всей системы – монолит, при масштабировании необходимо рассмотреть вариант микросервисной архитектуры |
| Интерфейс пользователя | Удобный, интуитивно понятный Telegram-интерфейс, использующий изменяемые сообщения | Отсутствие кастомизации интерфейса, внедрения подсказок и удобных пользовательских компонентов, нестабильность в некоторых случаях |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Критерий | Преимущества | Недостатки |
| Гибкость настройки | Расширенные возможности конфигурации стратегий и сигналов. Пользователь может адаптировать систему под свои нужды | Может быть нужно не каждому пользователю, нет уже готовых настроек |
| Открытость | Открытый исходный код позволяет пользователям вносить изменения, модифицировать поведение и анализировать алгоритмы | Отсутствие технической поддержки, как у коммерческих решений. Требует самостоятельного изучения кода при глубокой кастомизации |
| Функциональность | Содержит в себе все необходимые трейдеру функции для работы с личным портфелем, избранными ценными бумагами и торговым роботом | Для некоторых пользователей может быть избыточной, а для продвинутых пользователей наоборот, недостаточной |

ЗАКЛЮЧЕНИЕ

В ходе выполнения текущей практики был проведён комплексный анализ ранее разработанной информационной системы, предназначенной для взаимодействия с биржей и управления пользовательским инвестиционным портфелем.

На основе архитектурного, функционального и пользовательского анализа были выявлены ключевые достоинства и недостатки системы. Особое внимание было уделено модульной структуре приложения, возможности масштабирования, а также особенностям пользовательского взаимодействия через Telegram-интерфейс. Установлено, что несмотря на модульную организацию, система имеет черты монолита, что затрудняет её дальнейшее масштабирование. В качестве одного из направлений развития предложен переход на микросервисную архитектуру.

Проведённый пользовательский анализ позволил выявить такие преимущества системы, как гибкая настройка параметров, наличие песочницы для тестирования стратегий, централизованный интерфейс управления и система оповещений. Вместе с тем, зафиксированы такие ограничения, как зависимость от стабильного интернет-соединения, ограниченность интерфейса Telegram и возможные задержки в обработке команд.

За период выполнения текущей практики были приобретены следующие компетенции (таблица 2):

Таблица - Приобретенные компетенции

|  |  |  |
| --- | --- | --- |
| 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 | Сформулировал задачи анализа программной системы, подобрал методы исследования, выявил критические точки архитектуры и предложил пути их оптимизации |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| Компетенция | Расшифровка компетенции | Описание приобретенных знаний, умений и навыков |
| ОПК-1 | Способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной деятельности | Использовал знания в области теории систем для анализа архитектуры и обработки данных в программном комплексе |
| ОПК-2 | Способен применять компьютерные/суперкомпьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности | Применил инструменты анализа программного обеспечения и визуализации данных для оценки функциональности системы |
| ОПК-3 | Способен к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям | Разработал тестовые сценарии и провёл моделирование пользовательского поведения для оценки отклика и устойчивости системы |
| ОПК-4 | Способен участвовать в разработке технической документации программных продуктов и комплексов с использованием стандартов, норм и правил, а также в управлении проектами создания информационных систем на стадиях жизненного цикла | Сформировал аналитическую часть отчётной документации по результатам анализа, дал рекомендации по улучшению архитектуры и повышению устойчивости к ошибкам |

Практика позволила проанализировать общую составляющую системы, а также ее архитектуру и функционал взаимодействия с пользователем, при этом выявить направления для улучшения, а также выработать рекомендации по повышению надёжности и масштабируемости программного обеспечения. Полученные результаты подтверждают целесообразность применения данной системы в задачах автоматизации инвестиционной деятельности и выделяют её потенциал для дальнейшего развития.

СПИСОК ЛИТЕРАТУРЫ

1. Aiogram documentation [Электронный ресурс]. – 2025. — URL: https://docs.aiogram.dev/ (дата обращения 10.05.2025).
2. Правильный подход к модульной архитектуре [Электронный ресурс]. – 2024. — URL: https://habr.com/ru/articles/799169/ (дата обращения 15.05.2025).
3. Хорошие монолиты. Простая архитектура лучше всего [Электронный ресурс]. – 2024. — URL: https://habr.com/ru/companies/ /articles/676780/ (дата обращения 19.05.2025).
4. Микросервисная архитектура [Электронный ресурс]. – 2024. — URL: https://www.atlassian.com/ru/microservices/microservices-architecture (дата обращения 22.05.2025).
5. Vue 3 documentation [Электронный ресурс]. – 2025. — URL: https://vuejs.org/guide/introduction (дата обращения 24.05.2025).