Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка платформы для управления задачами команды разработки»

по дисциплине «Проектный практикум»

Заказчик: Макаров Артем Васильевич

Куратор: Макаров Артем Васильевич

Сотрудник компании Альфа-Банк

Студенты команды FireBalls (5)

Кремлев Андрей Викторович

Ковальский Антон Павлович

Васильев Роман Андреевич

Фадеев Антон Павлович

Малов Роман Максимович

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В условиях роста числа проектов, распределённых команд и требований к прозрачности рабочих процессов, системы управления задачами стали неотъемлемой частью современного рабочего пространства. Независимо от отрасли — будь то IT, маркетинг или производство — таск-трекеры позволяют синхронизировать действия сотрудников, планировать ресурсы и отслеживать прогресс. Их использование давно перешло из разряда желаемого в категорию необходимого.

Целью проекта является разработка автономного инструмента управления задачами — AlfaBalls, который позволит командам компании эффективно планировать и контролировать рабочие процессы без зависимости от зарубежных облачных сервисов и подписок. В рамках текущего семестра были сформулированы следующие задачи:

– проанализировать предметную область – изучить существующие решения в сфере управления задачами и определить их сильные и слабые стороны;

– описать функциональные и бизнес-требования – сформулировать перечень обязательных функций системы и ожидаемых бизнес-результатов от её использования;

– создать desktop-приложение – разработать клиентскую часть с графическим интерфейсом для пользователей;

– создать backend-приложение – реализовать серверную часть, обрабатывающую логику приложения, хранение данных и API для взаимодействия с клиентом;

Актуальность объясняется тем, что в текущих условиях доступ к зарубежным трекерам задач, таким как Jira и его аналоги, стал нестабильным или полностью ограничен. Это создает значительные риски для рабочих процессов, снижает надёжность инструментов планирования и ограничивает возможность адаптации систем под внутренние потребности компании. Зависимость от сторонних решений ставит под угрозу устойчивость и безопасность проектного управления, что делает разработку собственного инструмента особенно актуальной и важной.

Программный продукт предназначен для использования внутренними командами компании в целях управления проектами и задачами. Решение ориентировано на поддержку гибких методологий управления проектами (Agile), обеспечивая удобный и интуитивно понятный интерфейс для планирования, контроля и анализа рабочих процессов.

По завершении проекта планируется получить полностью функционирующий инструмент управления задачами, который:

– развёртывается на собственных серверах компании;

– имеет декстопное приложение предоставляющие интерфейс для пользователей;

– обеспечивает поддержку Kanban-досок и других Agile-инструментов.

1 Аналитика

1.1 Требования заказчика.

Команда заказчика выразила потребность в автономной платформе управления задачами, полностью независимой от зарубежных сервисов и платных подписок. Продукт должен работать на внутренних серверах компании, поддерживать Agile-подходы (Kanban, Scrum), быть удобным в использовании и легко расширяемым.

Для выполнения этих требований был составлен backlog проекта, включающий:

1. формализацию бизнес- и функциональных требований, сбор ключевые сценариев использования, требований к интерфейсу, управлению задачами, ролям пользователей и административным возможностям;

2. проектирование API и структуры базы[2] данных, основанное на собранных требованиях, а также разработку схемы взаимодействия между клиентом и сервером;

3. выделение уровней реализации:

3.1. MVP – минимально жизнеспособный продукт);

3.2. базовый уровень – расширения, повышающие удобство и стабильность;

3.3. оптимальный уровень – полный набор возможностей.

1.2 Бизнес- и функциональные требования.

1.2.1 Бизнес-требования

При составлении бизнес-требований был проведен анализ целевой аудитории, а также потенциальных аналогов.

Приложение AlfaBalls ориентировано на:

– малые и средние команды, работающие по Agile-методологиям (Scrum, Kanban);

– разработчиков и проектных менеджеров, которым нужен локальный, простой и функциональный инструмент для управления задачами и координации командной работы;

– компании, стремящиеся к прозрачности процессов и автоматизации сбора прогресса без привязки к облачным решениям.

Среди аналогов можно выделить прямых конкурентов и косвенных.

Основные конкуренты:

– Jira – мощный, но сложный и перегруженный функционалом.

– Kaiten – современный Agile-инструмент, но облачный и платный;

– Github Projects – встроенный в Github инструмент, бесплатный, но облачный и с некоторыми ограничениями

Другие возможные аналоги:

– Trello – прост в использовании, но ограничен (нет спринтов, эпиков, метрик).

– Asana – гибкий, но с ограничениями в бесплатной версии.

После определения конкурентов была составлена сравнительная таблица (таблица 1).

Таблица 1 – Сравнительный анализ

	Task-трекеры			
	AlbaBalls	Jira	Kaiten	Github Projects
Коробочное решение	+	+	-	-
Простота освоения	+	-	+	+
Бесплатное решение	+	+	-	+

1.2.2 Функциональные требования

При составлении функциональных требований были выделены следующие уровни реализации приложения:

1. MVP (минимально жизнеспособный продукт):
    1.1. Авторизация, аутентификация и управление пользователями;
    1.2. создание, редактирование и удаление проектов, досок и задач;
    1.3. поддержка Kanban-досок.
2. базовый уровень (следует иметь):
    2.1. комментарии и вложения к задачам;
    2.2. разграничение прав доступа на проектах;
    2.3. поддержка Scrum-спринтов (создание, управление, завершение).
3. оптимальный уровень (желательно иметь):
    3.1. автоматический сбор метрик продуктивности;
    3.2. визуализация прогресса с помощью диаграммы Ганта и Burndown chart.

2 Организация работы

2.1 Планирование деятельности и распределение задач

В начале разработки проекта AlfaBalls был составлен детализированный план работ на семестр, основанный на функциональных требованиях и уровнях реализации продукта. План был визуализирован в виде диаграммы Ганта, что позволило команде эффективно контролировать сроки и этапы разработки.

Каждая задача в рамках этих уровней была назначена конкретным участникам команды с учетом их опыта и специализации:

– разработка пользовательского интерфейса велась разработчику, имеющему опыт с Python и Qt[7];

– серверная часть и работа с БД была закреплена за разработчиком, владеющим Java[1] и фреймворком Spring[4];

– тестирование каждого из компонентов по отдельности и их взаимодействие выполнял тестировщик.

Такой подход обеспечил прозрачность в управлении задачами, позволил контролировать прогресс на каждом этапе и своевременно адаптировать план в случае изменений или непредвиденных трудностей.

2.2 Описание методологии разработки.

Разработка проекта AlfaBalls велась с использованием Agile-подхода, сочетающего в себе элементы Kanban и Scrum-спринтов. Команда организовала рабочий процесс через GitHub Projects, где использовалась встроенная поддержка досок и трекинга задач. Работа велась в коротких итерациях, каждая из которых завершалась анализом достигнутого результата и планированием следующих задач.

В процессе разработки проводилось регулярное тестирование отдельных модулей, включая авторизацию, управление задачами и досками. На промежуточных этапах проверялись как базовые сценарии работы, так и граничные случаи.

По результатам тестирования выявлялись некритичные баги, связанные, например, с некорректным отображением задач после перетаскивания между столбцами, или же с каскадным удалением сущностей из БД. Все ошибки устранялись в рамках соответствующих задач ближайшего спринта.

3 Разработка

3.1 Информация о работе каждого из участников

3.1.1 Ковальский Антон Павлович

В течение семестра исполнял обязанности тимлида. Организовал рабочие процессы внутри команды, обеспечил распределение задач и контроль их выполнения, а также занимался планированием этапов разработки. Составил диаграмму Ганта, которая служила основой для визуализации и отслеживания прогресса проекта в течение всего семестра. Кроме того, создал и поддерживал в актуальном состоянии все хранилища артефактов, обеспечив команду необходимыми инструментами для совместной работы.

Также проводил встречи, собирал обратную связь и адаптировал план проекта с учетом требований и замечаний заказчика. Занимался подготовкой отчетности по контрольным точкам и составление финального отчета о проделанной работе.

3.1.2 Фадеева Антон Павлович

Исполнял обязанности аналитика. Был ответственен за начальную фазу разработки, связанную с анализом потребностей заказчика и формированием требований к проекту, в результате чего составил бизнес-требования и функциональные требования.

В рамках аналитики также провел анализ аналогичных решений, доступных на рынке, и сформировал преимущества разрабатываемого продукта по сравнению с конкурентами. Кроме того, принимал активное участие в проектировании архитектуры приложения, в частности, участвовал в составлении спецификации API и структуры базы данных. На протяжении

всей разработки следил за тем, чтобы реализуемый функционал соответствовал утвержденным требованиям.

3.1.3 Кремлев Андрей Викторович

В течение семестра отвечал за разработку backend-составляющей проекта. Разработал архитектуру backend на основе фреймворка Spring, обеспечив надежность, масштабируемость и безопасность серверного компонента.

Также участвовал в проектировании спецификации API и структуры базы данных, взаимодействуя с аналитиком и разработчиком клиента. Помимо основной работы над backend, вносил вклад в разработку клиентской части — тестировал взаимодействие компонентов, а также помогал в решении технических задач, возникающих при интеграции.

3.1.4 Малов Роман Максимович

Отвечал за проектирование и реализацию десктопного пользовательского интерфейса с использованием Python и фреймворка Qt[10]. На первом этапе разработал макеты ключевых экранов, проанализировав пользовательские сценарии и удобство взаимодействия. После утверждения структуры интерфейса приступил к реализации.

Кроме визуальной части, также тесно сотрудничал с разработчиком backend, активно участвовал в разработке спецификации API, а также реализовал клиент, обеспечивающий связь между с серверной частью.

3.1.5 Васильев Роман Андреевич

В течение семестра обеспечивал контроль качества программного продукта на всех этапах разработки. Проводил тестирование как серверной,

так и клиентской части проекта. Разработал тест-кейсы, проверяя корректность работы функционала, стабильность взаимодействия компонентов, обработку ошибок и устойчивость к некорректным данным.

Своевременно фиксировал найденные ошибки и помогал разработчикам в их локализации и исправлении, благодаря чему удалось обнаружить и устранить множество проблем еще на промежуточных этапах, что положительно сказалось на финальном качестве приложения.

4 Практические результаты

4.1 Обзор архитектуры

В результате проделанной работы был создан таск-трекер AlfaBalls, представляющий собой приложение для управления проектами и задачами. Оно построено по принципу клиент-серверной архитектуры, обеспечивающей четкое разделение ответственности между компонентами, масштабируемость и независимость пользовательского интерфейса от серверной логики. Приложение состоит из следующих компонентов:

– клиентское приложение, реализованное на Python с использованием фреймворка Qt обеспечивает пользовательский интерфейс и взаимодействие с серверной частью через API.

– серверное приложение, разработанное на Java с использованием Spring Framework, выстроено по принципам чистой[5] архитектуры, что обеспечивает разделение бизнес-логики, доступа к данным и слоя-интерфейса.

Взаимодействие происходит следующим образом: UI-клиент авторизуется по JWT (Json Web Token) и отправляет HTTP-запросы[8] к backend-серверу для получения или отправки данных (например, создание проекта, смена статуса задачи, получение задач с доски), затем backend обрабатывает запросы, выполняет бизнес-логику, взаимодействует с базой данных, и возвращает результаты клиенту.

4.2 Демонстрация работы системы

4.2.1 Демонстрация UI

UI состоит из нескольких окон: авторизации, выбора/создания проекта (рисунок 1) и главного окна (рисунок 2). В главном окне можно перемещаться между проектами пользователя и досками на них. Также в нем содержатся колонки, отображающие статусы, в которых находятся задачи. Задачи можно перемещать между статусами при помощи drag`n`drop[6],. Можно создать новые статуса или задачи. При нажатии на задачу открывается карточка с подробным описанием (рисунок 3).

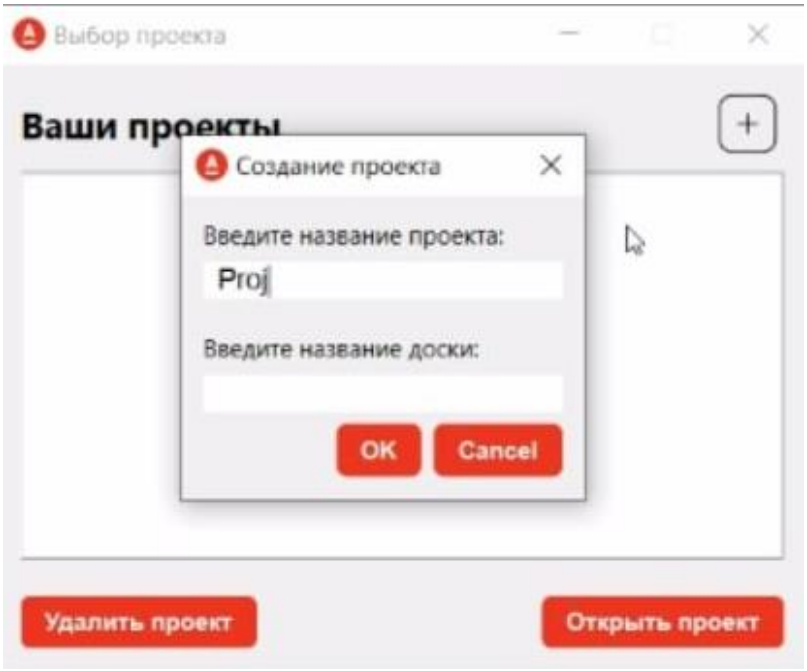

Рисунок 1 – окно выбора/создания проекта

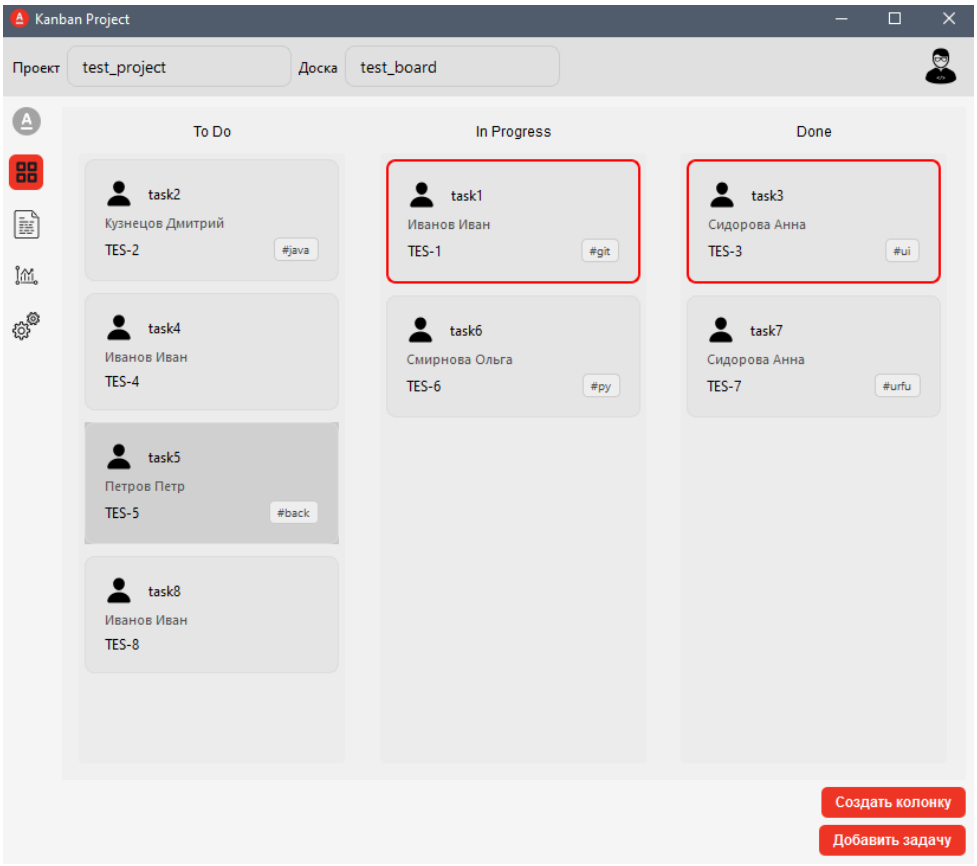

Рисунок 2 – главное окно

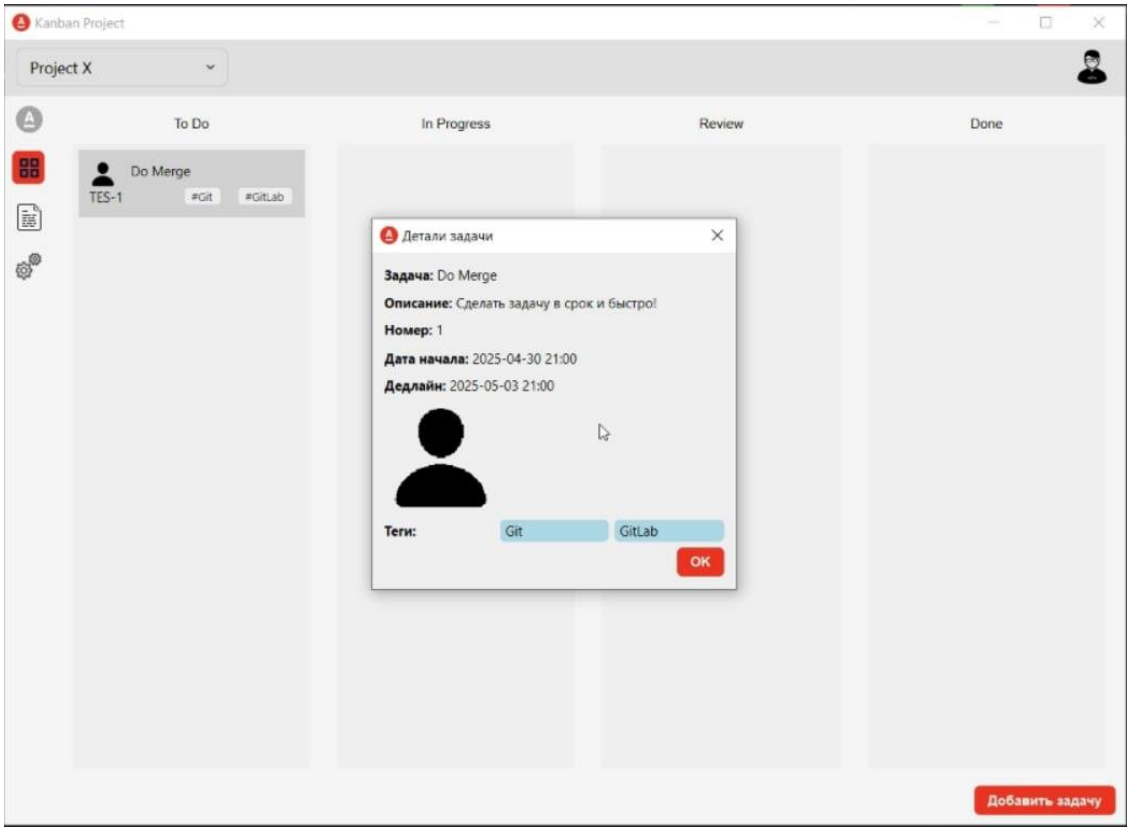

Рисунок 3 – Карточка задачи

4.2.2 Демонстрация backend

Для демонстрации работы backend была написана коллекция запросов в Postman[3], в ходе работы которых происходит логин и получение JWT, а затем создается проект, добавляются пользователи, создается доска, на которую добавляются задачи и новый статус. Результаты запуска коллекции на рисунке 4 (список большой, полностью на рисунок не поместился).

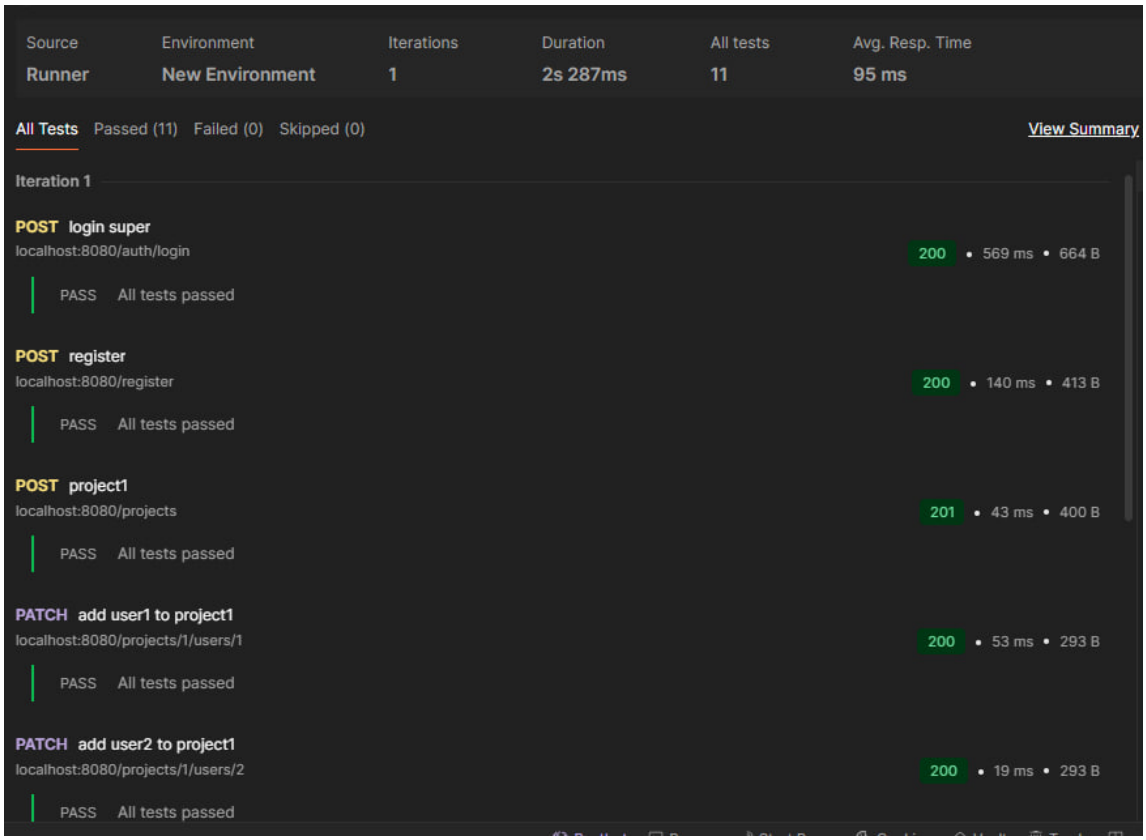

Рисунок 4 – тесты в Postman

ЗАКЛЮЧЕНИЕ

В ходе реализации проекта AlfaBalls удалось разработать минимально MVP, включающую основные функции управления задачами, проектами и досками, а также аутентификацию и авторизацию. Несмотря на ограниченный временной ресурс, удалось реализовать функциональность, соответствующую ключевым требованиям заказчика. По итогу заказчик оказался доволен результатом.

Качество программного продукта оценивалось на основе тестирования, проводимого в рамках каждой итерации. Благодаря регулярному выявлению и устранению ошибок на промежуточных этапах, итоговая версия MVP демонстрирует высокую стабильность и работоспособность. Обнаруженные дефекты не оказали критического влияния на функционирование ключевых сценариев.

Поскольку реализована только базовая версия продукта, существует широкий потенциал для его развития, достижения базового, а затем и оптимального уровней, описанных в функциональных требованиях. В частности, возможны улучшения в части пользовательского интерфейса, внедрение аналитики, расширение системы прав доступа, добавление спринтов, меток, комментариев, вложений, уведомлений, а также интеграция инструментов визуализации и автоматической генерации отчетов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Java and Spring tutorials [Электронный ресурс] – URL: https://www.baeldung.com (дата обращения: 04.04.25).

2. PostgreSQL documentation [Электронный ресурс] – URL: https://www.postgresql.org/docs/17/tutorial-start.html (дата обращения: 25.04.25).

3. Postman official website [Электронный ресурс] – URL: https://www.postman.com/ (дата обращения: 17.04.25).

4. Spring official website [Электронный ресурс] – URL: https://spring.io/ (дата обращения: 10.04.25).

5. The Clean Architecture [Электронный ресурс] – URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (дата обращения: 07.04.25).

6. Перетаскивание (drag & drop) в PyQt5 [Электронный ресурс] – URL: https://pythonworld.ru/gui/pyqt5-dragdrop.html (дата обращения: 01.05.25).

7. PyQt documentation [Электронный ресурс] – URL: https://doc.qt.io/qtforpython-6/gettingstarted.html#getting-started (дата обращения: 04.04.25).

8. Request library [Электронный ресурс] – URL: https://pythonru.com/biblioteki/kratkoe-rukovodstvo-po-biblioteke-python-requests (дата обращения: 27.04.25).

9. Waterfall, Agile, Scrum или Kanban – в чем разница? [Электронный ресурс] – URL: https://kaiten.ru/blog/waterfall-agile-scrum-kanban/ (дата обращения: 01.04.25).

10. PyQt5 Signals, Slots & Events [Электронный ресурс] – URL: https://www.pythonguis.com/tutorials/pyqt-signals-slots-events/ (дата обращения: 15.04.25).