

**Федеральное государственное бюджетное образовательное учреждение высшего  
образования  
«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА и  
ГОСУДАРСТВЕННОЙ СЛУЖБЫ  
при Президенте Российской Федерации»**

**КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**Специальность 09.02.07 «Информационные системы и программирование»**

**УТВЕРЖДАЮ**  
Зам. директора КМПО РАНХиГС  
\_\_\_\_\_ С.Ф. Гасанов  
« \_\_\_\_\_ » \_\_\_\_\_ 2025 г.

## **ДИПЛОМНЫЙ ПРОЕКТ**

**на тему: «Разработка программного модуля системы обучения  
тестирования сотрудников железнодорожного транспорта»**

Выполнил студент группы 41ИС-21	_____	/В.С. Егорова/
Руководитель	_____	/ С.П. Киселева /
Нормоконтролер	_____	/ С.П. Киселева /
Консультант по технико-экономическому обоснованию проекта	_____	/ М.М. Трифонова /
Старший консультант	_____	/ О.А. Калашникова /

**МОСКВА  
2025 г.**

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ .....	6
1.1 Актуальность разработки модуля системы в контексте предметной области .....	6
1.2 Анализ существующих решений и технологий для автоматизации задач в предметной области .....	7
1.3 Анализ бизнес-процессов и обоснование необходимости разработки модуля.....	10
1.4 Анализ ИТ-инфраструктуры и интеграционных возможностей.....	11
1.5 Формулировка требований к модулю системы .....	13
1.6 Выбор методологии проектирования и разработки модуля.....	15
Выводы по главе .....	17
2 ПРОЕКТИРОВАНИЕ МОДУЛЯ СИСТЕМЫ .....	18
2.1 Проектирование пользовательского интерфейса модуля.....	18
2.2 Проектирование пользовательского интерфейса модуля.....	19
2.3 Проектирование структуры данных и базы данных модуля.....	19
2.4 Обоснование выбора технологий и платформы для разработки модуля ..	22
2.5 Проектирование функциональных компонентов модуля.....	23
Выводы по главе .....	24
3 РЕАЛИЗАЦИЯ И АТТЕСТАЦИЯ МОДУЛЯ СИСТЕМЫ.....	26
3.1 Реализация функциональности модуля (языки программирования, фреймворки, библиотеки).....	26
3.2 Интеграция модуля с источниками данных (API, SQL-запросы).....	27
3.3 Интеграция и тестирование.....	35
Вывод по главе.....	36
4 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И ВНЕДРЕНИЯ ПРОГРАММНОГО ПРОДУКТА.....	37
4.1 Определение факторов эффективности разработки и внедрения программного продукта .....	37

4.2 Определение трудоемкости разработки отдельных стадий программного продукта и затрат на оплату труда разработчиков .....	38
4.3 Расчет амортизационных отчислений от стоимости основных средств и нематериальных активов, участвующих в процессе разработки программного продукта .....	40
4.4 Расчет текущих затрат на проектирование и внедрение программного продукта .....	42
4.5 Расчет прочих затрат на проектирование и внедрение программного продукта .....	44
4.6 Расчет накладных затрат (коммерческие и управленческие расходы) на проектирование и внедрение программного продукта.....	44
4.7 Определение сметы затрат на разработку программного продукта.....	44
4.8 Расчет экономической эффективности разработки программного продукта .....	45
4.9 Экономические показатели разработки программного продукта .....	47
Вывод по главе.....	47
ЗАКЛЮЧЕНИЕ .....	49
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	51
ПРИЛОЖЕНИЕ А .....	55
ПРИЛОЖЕНИЕ Б.....	59
ПРИЛОЖЕНИЕ В .....	60

## **ВВЕДЕНИЕ**

ОАО «Российские железные дороги» (РЖД) — одна из крупнейших компаний в России, играющая ключевую роль в обеспечении транспортной связности страны. На момент 2024 года в организации работают более 800 тысяч сотрудников[23], каждый из которых вносит свой вклад в бесперебойное функционирование железнодорожного транспорта. Однако масштабы компании и сложность ее инфраструктуры предъявляют высокие требования к уровню подготовки профессиональных кадров. Ошибки сотрудников в этой сфере могут привести к серьезным последствиям, включая аварии, задержки поездов и нарушение логистических процессов. Поэтому обучение и регулярная аттестация работников являются неотъемлемой частью работы компании. Внедрение современных технологий в процесс обучения позволяет не только повысить качество подготовки, но и сделать его более гибким, доступным и удобным для самих сотрудников.

Актуальность работы обусловлена необходимостью снижения рисков, связанных с человеческим фактором в железнодорожной отрасли.

Объект исследования — информационная система для обучения и тестирования сотрудников железнодорожного транспорта.

Предмет исследования — процесс тестирования в рамках данной информационной системы.

Новизна разработки состоит в реализации кроссплатформенного программного модуля, ориентированного на оценку готовности сотрудников к действиям в условиях чрезвычайных ситуаций.

Цель дипломного проекта: разработка программного модуля системы обучения и тестирования сотрудников компании.

Задачи дипломного проекта:

- изучить источники по теме;
- провести анализ предметной области программного продукта;
- составить сравнительный анализ методик оценки профессиональных компетенций и выявить ключевые параметры для оценки;

- спроектировать архитектуру программного решения;
- составить спецификацию программного решения;
- спроектировать и реализовать базу данных;
- спроектировать и реализовать пользовательский интерфейс;
- реализовать основной функционал программного модуля;
- выполнить отладку и провести тестирование функционала программного модуля;
- разработать Инструкцию по установке и настройке и Руководство пользователя;
- рассчитать экономическую эффективность системы.

Практическая значимость работы заключается в том, что разработанный модуль может быть внедрен в учебные центры РЖД, что позволит повысить качество подготовки сотрудников и снизить риски человеческого фактора. Кроме того, предложенное решение может быть адаптировано для других отраслей, где требуется высокий уровень подготовки персонала.

# **1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ**

## **1.1 Актуальность разработки модуля системы в контексте предметной области**

Железнодорожный транспорт — кровеносная система экономики России: на него приходится 83% грузооборота (включая критически важные поставки угля, нефти и зерна) и 28% пассажиропотока страны [30]. В условиях динамичного развития инфраструктуры, внедрения высокоскоростного движения и увеличения грузопотоков особую значимость приобретает повышение уровня безопасности, особенно в части предотвращения и ликвидации аварийных ситуаций. Согласно отчету, в 2023 году на железных дорогах стран СНГ, включая Россию, произошло 1 247 крушений и аварий, причем в 83% случаев ключевым фактором стал человеческий фактор.[29]

Особую тревогу вызывает динамика происшествий на фоне технологической модернизации отрасли. Например, планируется создание высокоскоростных магистралей, таких как Москва – Санкт-Петербург, где скорость движения поездов будет достигать 350 км/ч [28], даже незначительные ошибки персонала могут привести к катастрофическим последствиям.

Увеличение грузопотоков на 37% за последние 5 лет и переход на цифровые системы управления требуют принципиально нового уровня готовности персонала. При этом:

- 42% аварий связаны с неправильными действиями при отказе автоматики;
- каждое пятое происшествие усугубляется недостаточным знанием регламентов;
- в 67% случаев ЧС фиксируются задержки в передаче сигналов. [29]

Аварийные ситуации на железной дороге — от сбоев в работе автоматики до крушений поездов — требуют от сотрудников:

- молниеносной реакции и хладнокровия в условиях стресса, когда каждая секунда влияет на масштаб последствий;

– безупречного знания протоколов - от экстренной остановки поезда до эвакуации пассажиров при ЧС;

– навыков работы с системами аварийного оповещения (например, ЭРА-ГЛОНАСС) и умения анализировать данные телеметрии в реальном времени;

– налаженного межведомственного взаимодействия с МЧС, медицинскими бригадами и правоохранительными органами для локализации угроз.

Разрабатываемый модуль призван повысить эффективность подготовки сотрудников к действиям в аварийных ситуациях за счет системного тестирования и анализа знаний.

Особую актуальность модуль приобретает в свете планируемого обновления инфраструктуры:

1) Строительство вторых/третьих путей на перегруженных участках увеличит плотность движения, что повысит нагрузку на диспетчеров.

2) Внедрение «интеллектуальных поездов» с автоведением потребует от персонала навыков ручного перехвата управления при отказах автоматики.

3) Расширение грузоперевозок опасных веществ (например, химикатов) усилит потенциальные последствия аварий. [28]

Таким образом, разработка модуля для обучения и тестирования сотрудников в условиях аварийных ситуаций отвечает стратегическим приоритетам отрасли: модернизация безопасности и минимизация рисков.

## **1.2 Анализ существующих решений и технологий для автоматизации задач в предметной области**

В последние годы рынок образовательных платформ активно развивается: внедряются новые форматы контента, автоматизация процессов тестирования, цифровизация корпоративного обучения. Однако при рассмотрении решений, применяемых в железнодорожной отрасли, становится очевидно, что они преимущественно ориентированы на общее теоретическое обучение и не охватывают в достаточной степени задачи, связанные с подготовкой к действиям в нештатных и аварийных ситуациях.

Среди наиболее распространенных решений можно выделить системы дистанционного обучения (LMS — Learning Management Systems), такие как Moodle, Canvas, Blackboard и другие. Эти платформы предоставляют базовые возможности для создания курсов, проведения тестирования и отслеживания прогресса обучающихся.

Однако такие решения ориентированы преимущественно на академическое или общее корпоративное обучение и не адаптированы к отраслевой специфике железнодорожного транспорта.

Кроме того, существуют специализированные системы для корпоративного обучения, такие как SAP SuccessFactors, Cornerstone OnDemand и другие. Эти решения предлагают более широкие возможности для управления персоналом, включая планирование обучения, оценку компетенций и интеграцию с другими корпоративными системами. Но их внедрение требует значительных финансовых затрат и длительной адаптации под нужды конкретной отрасли.

В компании РЖД уже внедрены несколько систем, направленных на обучение и оценку персонала. АСПТ РЖД — это платформа, предназначенная для дистанционного обучения и тестирования сотрудников. Она позволяет создавать курсы, проводить тестирование и отслеживать результаты. Однако функциональность системы ограничена базовыми возможностями, такими как создание тестов и управление учебными материалами. Система не учитывает специфику работы с современными технологиями, такими как высокоскоростные поезда или системы автоматизированного управления движением. Корпоративный университет РЖД — это образовательная платформа, которая предоставляет сотрудникам доступ к курсам и программам повышения квалификации. Университет предлагает широкий спектр образовательных материалов, но его функциональность также ограничена. Например, отсутствует возможность адаптивного обучения, которое учитывало бы индивидуальные потребности сотрудников, или интеграция с системами управления персоналом. Для оценки кандидатов при приеме на



работу в РЖД используются специализированные тестовые платформы. Эти системы позволяют проводить тестирование по различным направлениям, таким как технические знания, логическое мышление и психологическая устойчивость. Однако такие платформы ориентированы исключительно на оценку кандидатов и не подходят для постоянного обучения и повышения квалификации действующих сотрудников.

Проведенный анализ демонстрирует ключевые преимущества предлагаемого модуля перед существующими решениями (Таблица 1).

Таблица 1 – Сравнение существующих решений

Характеристика	АСПТ	Корпоративный университет РЖД	Moodle	SAP SuccessFactors	Предлагаемый модуль
Адаптивное обучение	Нет	Нет	Да	Да	Да
Оценка готовности к ЧС	Частично	Нет	Нет	Нет	Да
Кроссплатформенность	Нет	Нет	Да	Да	Да
Автоматическая проверка тестов	Да	Да	Да	Да	Да
Стоимость внедрения	Низкая	Средняя	Низкая	Высокая	Низкая
Поддержка мобильных устройств	Нет	Нет	Да	Да	Да

Таким образом, разрабатываемый модуль не дублирует существующие решения, а закрывает важный функциональный пробел — оценку действий сотрудников в условиях потенциальных инцидентов, с учетом контекста, регламентов и ограниченного времени на принятие решений. Подобный подход позволяет повысить достоверность итогов тестирования, точнее выявлять зоны риска и формировать обоснованные планы повышения квалификации.

### **1.3 Анализ бизнес-процессов и обоснование необходимости разработки модуля**

Железнодорожная отрасль функционирует на основе сложной системы взаимосвязанных процессов, которые можно классифицировать в соответствии с управленческой моделью ОАО «РЖД».

Схема бизнес-процессов компании включает три ключевых блока:

#### **1) Процессы управленческой деятельности**

Обеспечивают стратегическое развитие и контроль. В этот блок входят: анализ со стороны руководства, внутренние проверки, корректирующие и предупреждающие действия, а также мониторинг удовлетворенности потребителя. Например, планирование графика движения поездов корректируется на основе данных о загрузженности линий и обратной связи от клиентов.

#### **2) Основные процессы**

К ним относятся грузовые и пассажирские перевозки (дальние и пригородные), ремонт подвижного состава, услуги инфраструктуры и другие виды деятельности.

#### **3) Процессы управления ресурсами**

Поддерживают эффективность операционной деятельности. В этот блок входят управление материальными, финансовыми, трудовыми и информационными ресурсами.

Систему подготовки персонала можно отнести к ресурсным процессам.

В ОАО «РЖД» уже реализована многоуровневая структура обучения, включающая:

- профессиональную подготовку новых сотрудников;
- техническую учебу и поддержание квалификации;
- повышение квалификации при освоении новых технологий;
- аттестацию персонала, в том числе по вопросам безопасности.

Более детально можно отобразить на схеме (рис 1).



Рисунок 1 - Схема бизнес-процессов обучения и тестирования в РЖД

Для повышения эффективности этих процессов актуально внедрение дополнительных цифровых инструментов, направленных на:

- расширение возможностей оценки готовности персонала к действиям в нестандартных ситуациях;
- повышение точности диагностики уровня знаний;
- адаптацию содержания тестирования под профиль и опыт сотрудника.

Разрабатываемый модуль может быть интегрирован в существующую систему обучения и использоваться в качестве дополнительного средства оценки. Он позволит автоматизировать проведение тематических тестов и повысить точность оценки знаний.

Повышение уровня подготовки сотрудников минимизирует риски и предотвратит аварии, что повысит безопасность на железной дороге.

#### 1.4 Анализ ИТ-инфраструктуры и интеграционных возможностей

Современная ИТ-инфраструктура РЖД представляет собой сложную экосистему, включающую множество систем и платформ, таких как системы

управления движением поездов, базы данных персонала, платформы дистанционного обучения и другие корпоративные решения. Как видно из Схемы «Цифровая трансформация ОАО РЖД» (Рис. 2), компания активно развивает цифровые сервисы, которые условно можно разделить на:

- Бизнес-сервисы (для внутренних и внешних клиентов),
- Цифровые платформы (управление перевозками, логистика, HR),
- Стратегические направления (кибербезопасность, инновации).



Рисунок 2 - Схема «Цифровая трансформация ОАО РЖД»

Схема отражает ключевые направления цифровизации, включая интеграцию модуля обучения в блок «Непроизводственные процессы» (подсистема «Цифровой HR»).

В последние годы РЖД активно развивает свою цифровую экосистему, объединяя разрозненные сервисы в единую платформу. Это открывает новые возможности для интеграции современных решений. Однако текущая ИТ-инфраструктура РЖД имеет свои особенности и ограничения, которые необходимо учитывать при разработке и внедрении новых решений.

Одной из ключевых проблем является фрагментированность систем. Это приводит к тому, что данные о прохождении обучения сотрудниками не автоматически передаются в базу данных персонала, что усложняет процесс

планирования обучения и оценки его эффективности. Также, как показано на схеме, сервисы ДЗО (дочерних предприятий) и корпоративные платформы (например, «Управление перевозочным процессом») часто работают изолированно.

Еще одной проблемой является устаревание части ИТ-решений. Некоторые системы, используемые для управления инфраструктурой, были разработаны более десяти лет назад и не поддерживают современные стандарты интеграции. Это создает дополнительные сложности при попытке подключения новых модулей или обновления существующих.

Проект «Цифровая железная дорога» (аналогичен платформе «Управление перевозочным процессом» на схеме) предполагает использование современных технологий (Kubernetes, микросервисы), что упростит интеграцию.

С учетом текущей стратегии цифровой трансформации ОАО «РЖД», разрабатываемый модуль обучения действиям в аварийных ситуациях изначально проектируется как автономное решение с возможностью дальнейшей интеграции.

### **1.5 Формулировка требований к модулю системы**

Железные дороги — это не только сталь и пар, но и огромный поток данных. Каждый день системы РЖД обрабатывают миллионы операций: от управления движением поездов до учета компетенций сотрудников. Но если данные о локомотивах стекаются в единый центр, то с обучением персонала часто все иначе — разрозненные системы, ручной ввод, задержки.

На основе анализа ИТ-инфраструктуры и ее интеграционных возможностей можно сформулировать ключевые требования к модулю обучения и тестирования.

- 1) требования к функциональным характеристикам:
  - управление тестами;
  - удобная навигация с сортировкой по типам и темам тестов;
  - управление учебными группами;

- поддержка различных типов вопросов;
- создание тем при создании тестов;
- адаптивное тестирование (сложность вопросов изменяется в процессе обучения), в том числе сценарные тесты;

- аналитика по группам и отдельным сотрудникам;
- регистрация и авторизация пользователей;

## 2) требования к надежности:

- адаптивный дизайн (поддержка ПК, планшетов, мобильных устройств);

- удобная навигация;

- для администратора управление пользователями и группами, а так же настройка прав доступа;

## 3) условия эксплуатации:

Текущая инфраструктура:

- Backend: Python (FastAPI);

- Frontend: Kivy;

- База данных: SQLite (текущая версия);

- Сервер: Linux/Windows (минимальные требования: 2 CPU, 4 ГБ RAM);

Планируемая миграция:

- Целевая СУБД: PostgreSQL 15+ ;

## 4) Производительность:

- время отклика не более 2 секунд;

- до 500 одновременных пользователей (в текущей конфигурации).

Планируемая масштабируемость: 5000+ пользователей (после перехода на PostgreSQL);

## 5) Требования к безопасности:

- Ролевая модель доступа;

- Журналирование действий;

- Шифрование хранимых данных;

- Поддержка API для интеграции;

6) Требования к надежности:

- Репликация данных;
- Автоматическое восстановление;
- Мониторинг состояния системы;
- Оптимизация запросов;
- Кэширование данных;

7) Требования к сопровождению:

- Механизм автоматических обновлений;
- Возможность отката версий;
- Документирование изменений;

8) Преимущества интеграции:

- Повышение эффективности управления обучением;
- Автоматизация процессов обмена данными;
- Снижение затрат на поддержку изолированных систем.

Таким образом, разработка модуля с учетом современных стандартов интеграции и требований к безопасности данных позволит успешно внедрить его в существующую ИТ-инфраструктуру РЖД. Это, в свою очередь, будет способствовать повышению эффективности процессов обучения и оценки персонала, а также повышению безопасности и эффективности работы железнодорожного транспорта в целом. Предложенные решения соответствуют стратегии цифровой трансформации РЖД и могут быть реализованы в рамках дальнейшего развития ИТ-инфраструктуры компании.

### **1.6 Выбор методологии проектирования и разработки модуля**

Для проектирования и разработки платформы для обучения и тестирования был выбран гибридный подход, сочетающий элементы Agile и Waterfall. Этот подход позволяет объединить гибкость и итеративность Agile с четким планированием и контролем Waterfall, что особенно важно для проекта, который включает как техническую разработку, так и обеспечение

функциональности для создания и управления учебными материалами и тестами.

Этапы разработки по гибридной методологии:

1) Анализ требований и планирование (Waterfall)

На начальном этапе проводится детальный анализ требований к платформе, включая функциональные и нефункциональные аспекты.

Основные задачи:

- разработка технического задания с описанием архитектуры;
- планирование ресурсов, сроков и бюджета проекта;
- учет требований к безопасности данных и пользовательскому опыту.

2) Проектирование архитектуры и интерфейсов (Waterfall)

На этом этапе разрабатывается архитектура платформы, включая:

- модульную структуру (модуль для тестирования);
- интерфейсы взаимодействия с пользователем (администраторы, преподаватели, студенты);
- требования к безопасности.

3) Итеративная разработка (Agile)

Разработка платформы ведется итеративно, с разбиением на спринты (по 2-3 недели). Каждый спринт включает:

- разработку функциональности (базовый функционал модуля системы тестирования);
- тестирование и обратную связь с заказчиком.

4) Тестирование и доработка (Agile)

Проводится комплексное тестирование платформы, включая:

- Функциональное тестирование (проверка работы всех компонентов);
- Тестирование безопасности и производительности;
- Внесение доработок по результатам тестирования.

5) Внедрение и поддержка (Waterfall)

После завершения разработки платформа готовится к сдаче:

- Подготовка документации;



- Обучение пользователей (если требуется) ;
- Сбор обратной связи для возможных улучшений.

Гибридный подход, сочетающий элементы Agile и Waterfall, является оптимальным выбором для проектирования и разработки платформы для обучения и тестирования. Он обеспечивает гибкость, контроль и высокое качество разработки.

### **Выводы по главе**

Проведенный анализ предметной области подтвердил необходимость разработки специализированного программного модуля для обучения и тестирования сотрудников железнодорожного транспорта. Выявлена актуальная задача — усиление инструментов оценки готовности персонала к действиям в нестандартных и аварийных ситуациях.

Анализ существующих образовательных решений, используемых в ОАО «РЖД», таких как АСПТ РЖД и Корпоративный университет, показал, что данные системы обладают ограниченным функционалом в части адаптивности, сценарного тестирования и гибкой аналитики. Это формирует потребность в дополнении текущей модели подготовки персонала специализированным модулем, ориентированным на оценку поведения в критически важных ситуациях.

Исследование бизнес-процессов компании подтвердило значимость систематической аттестации и повышения квалификации как элементов управления ресурсами. В этом контексте предлагаемое решение может быть интегрировано в существующую модель обучения и использоваться для углубленной оценки знаний в рамках профессиональной деятельности.

Анализ ИТ-инфраструктуры РЖД показал, что, несмотря на определенную фрагментированность систем и наличие устаревших решений, архитектура компании в целом предоставляет возможности для успешной интеграции новых цифровых модулей.

Разработка модуля соответствует задачам цифровой трансформации ОАО «РЖД» и может стать важным элементом в формировании единой

цифровой среды профессиональной подготовки, ориентированной на повышение надежности и безопасности железнодорожного транспорта.

## **2 ПРОЕКТИРОВАНИЕ МОДУЛЯ СИСТЕМЫ**

### **2.1 Проектирование пользовательского интерфейса модуля**

Разрабатываемый модуль системы реализован по принципу клиент-серверной архитектуры. Данное архитектурное решение обусловлено необходимостью обеспечить масштабируемость, гибкость, а также четкое разделение ответственности между различными компонентами системы.

Серверная часть реализована с использованием асинхронного веб-фреймворка FastAPI. Он обеспечивает высокую производительность и позволяет легко строить REST API, совместимые с современными клиентскими приложениями. Все взаимодействие между клиентом и сервером осуществляется по протоколу HTTP с передачей данных в формате JSON. Аутентификация реализована через JWT-токены, что позволяет обеспечить безопасность и разграничение доступа.

Сервер содержит следующие основные модули:

- /auth — отвечает за авторизацию и аутентификацию;
- /users — управление учетными записями пользователей;
- /tests — работа с тестовыми заданиями и прохождение тестов;
- /training — сценарные тесты;
- /groups — управление учебными группами.

Клиентская часть включает в себя мобильное приложение, реализованное с помощью кроссплатформенного фреймворка Kivy. Такое решение обеспечивает совместимость с основными мобильными платформами и снижает затраты на разработку.

Система поддерживает ролевую модель доступа (администратор, преподаватель, студент), что важно для организации процесса обучения

## **2.2 Проектирование пользовательского интерфейса модуля**

Пользовательский интерфейс модуля разработан с ориентацией на удобство, интуитивную понятность и адаптацию под различные устройства. В рамках мобильного приложения на Kivu реализованы следующие экраны:

- экран авторизации и регистрации: позволяет пользователю пройти аутентификацию;
- главное меню: центральная точка входа, содержащая ссылки на основные разделы: учебные материалы, тесты, личный кабинет;
- тестирование: модуль прохождения и создания тестов с отображением прогресса и результатов;
- группы: управление учебными группами, отображение участников;
- пользователи: управление пользователями, назначение ролей.

Интерфейс адаптирован под сенсорное управление и реализует все современные принципы UX-дизайна. Используется гибкая система разметки, анимации и элементы взаимодействия, такие как свайпы и всплывающие уведомления.

## **2.3 Проектирование структуры данных и базы данных модуля**

Структура базы данных построена на реляционной модели и реализована через ORM SQLAlchemy. На этапе разработки применяется SQLite, в дальнейшем запланирован переход на PostgreSQL для обеспечения масштабируемости и высокой надежности.

Основные сущности базы данных:

- 1) учетные данные:
  - логин (уникальный идентификатор),
  - хеш пароля,
  - email (с подтверждением),
  - номер телефона (опционально);
- 2) профиль:
  - ФИО,
  - должность,

- дата регистрации,
- последний вход,
- учет сессий,
- статус (активен/заблокирован);

3) ролевая модель:

- тип учетной записи (студент/преподаватель/админ),
- группы;

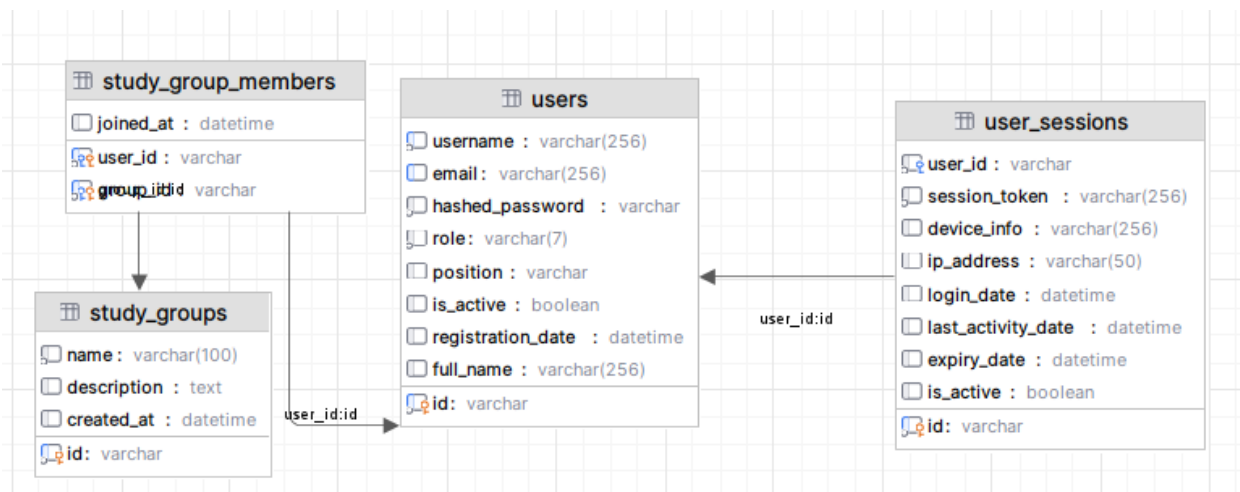


Рисунок 3 - Диаграмма сущностей. Пользователи

4) тесты и вопросы (thrmrs, tests, tasks, group\_assigned\_tests):

- гибкая система создания тестов,
- настройка временных ограничений,
- различные типы вопросов (interaction\_type),
- назначение тестов группам,
- темы тестов
- система оценки сложности заданий;

5) результаты тестирования (test\_answers):

- детальная статистика по каждой попытке,
- время начала и завершения теста,
- набранные баллы и статус прохождения,
- обратная связь по ответам;

6) сценарные тесты (scenario\_tests, scenario\_steps, scenario\_choices):

- многошаговые интерактивные сценарии с ветвлением,

- поддержка таймеров, финальных шагов и критических ошибок,
- гибкая логика переходов между шагами;

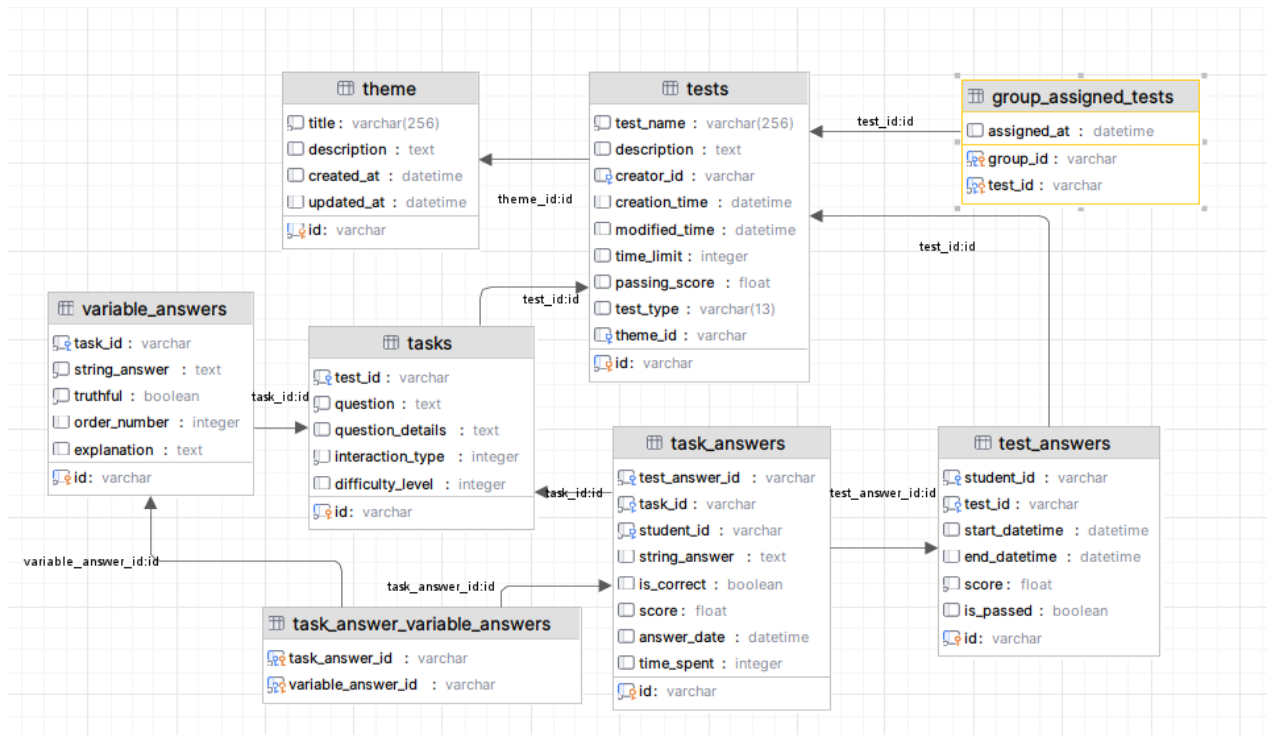
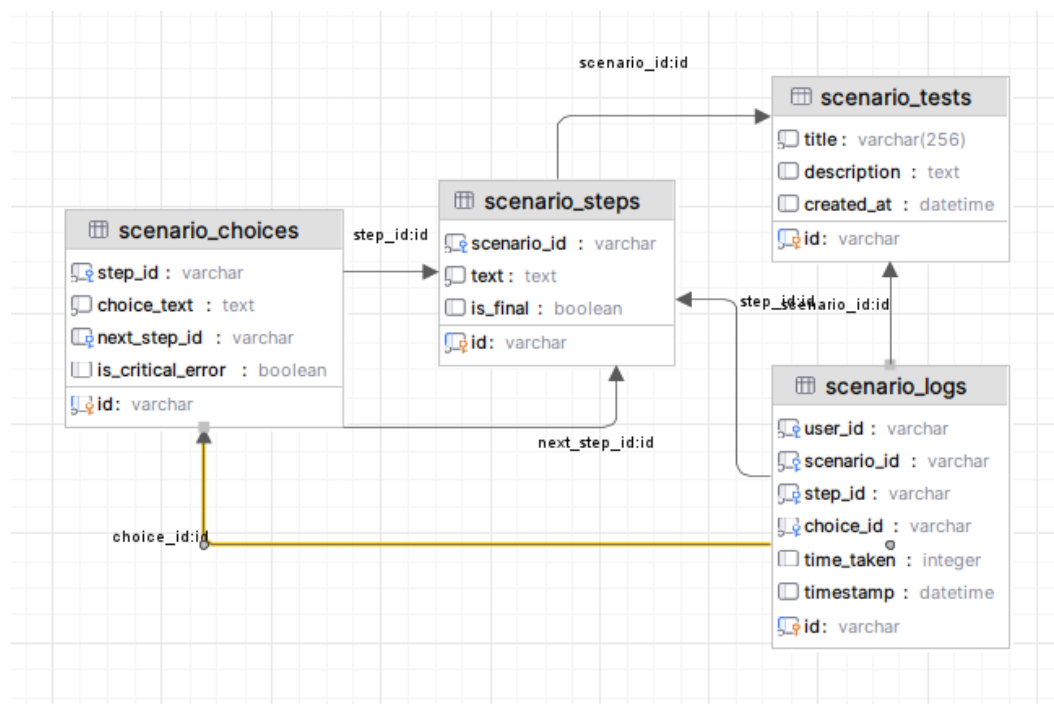


Рисунок 4 - Диаграмма сущностей. Тесты и задания

#### 7) логи сценариев (scenario\_logs):

- запись всех действий пользователя на каждом шаге,
- фиксация времени, ошибок и выбранных вариантов,
- аналитика по прохождению сценариев и принятию решений.



## Рисунок 5 - Диаграмма сущностей. Сценарные тесты

Связи и ограничения обеспечиваются средствами SQLAlchemy. Полная диаграмма представлена в приложении 1. Визуальная работа с БД осуществлялась в DataGrip.

### **2.4 Обоснование выбора технологий и платформы для разработки модуля**

Ключевыми критериями выбора технологий были производительность, поддержка возможности асинхронности, кроссплатформенность, простота в освоении.

Язык программирования: Python был выбран в качестве основного языка разработки благодаря своему лаконичному синтаксису, поддержке асинхронного программирования. Универсальность Python позволила использовать его как для серверной, так и для клиентской части.

Серверный фреймворк: FastAPI.

FastAPI — современный асинхронный фреймворк, подходящий для разработки высокопроизводительных REST API. Среди его преимуществ:

- высокая скорость работы (на уровне Node.js и Go);
- встроенная поддержка OpenAPI/Swagger;
- удобная валидация данных через Pydantic;
- готовность к промышленному развертыванию.

Клиентская часть: Для реализации кроссплатформенного мобильного приложения выбран фреймворк Kivy. Он обеспечивает:

- поддержку Android, Windows, macOS, Linux и iOS;
- адаптацию под сенсорные экраны;
- гибкую систему построения интерфейсов.

База данных: На этапе прототипирования используется SQLite — встраиваемая СУБД без необходимости отдельного сервера. Для промышленной эксплуатации запланирован переход на PostgreSQL, как на более надежную и масштабируемую систему.

ORM: В качестве слоя доступа к базе данных выбрана ORM-библиотека SQLAlchemy. Она позволяет описывать структуру БД декларативно, выполнять миграции, а также эффективно управлять транзакциями.

Средства разработки и инструментальная поддержка:

PyCharm — интегрированная среда для Python, удобная для отладки, навигации и тестирования;

DataGrip — инструмент для работы с базами данных;

GitHub — система контроля версий, используемая для командной работы и отслеживания изменений в проекте.

Таким образом, выбор технологий был обусловлен не только техническими характеристиками, но и возможностью последующего масштабирования.

## **2.5 Проектирование функциональных компонентов модуля**

Функциональные компоненты модуля логически разделены, что упрощает поддержку и расширение системы.

### **1) Аутентификация и авторизация**

Обработывает вход пользователей в систему, регистрацию новых аккаунтов и управление правами доступа. Все пользователи, включая студентов, преподавателей и администраторов, взаимодействуют с этим модулем при начале работы.

### **2) Управление пользователями**

Позволяет администраторам создавать и редактировать учетные записи, назначать роли и управлять доступом. Преподаватели могут просматривать профили студентов своих учебных групп.

### **3) Работа с учебными материалами**

Предоставляет инструменты для создания, редактирования и публикации учебного контента. Преподаватели формируют материалы курсов, а студенты получают к ним доступ согласно учебному плану.

### **4) Тестирование и оценка знаний**

- конструктор тестов для преподавателей,
- интерфейс прохождения тестов для студентов,

- автоматическую проверку ответов,
- систему оценки результатов.

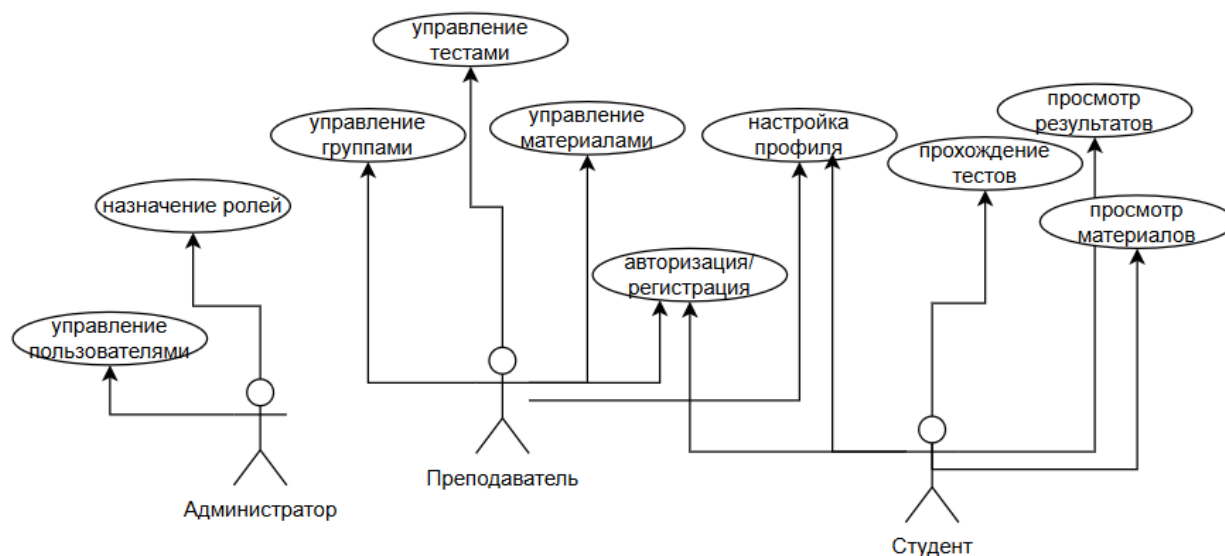


Рисунок 6 - Диаграмма использования

#### 5) Аналитика и отчетность

Генерирует детальные отчеты об успеваемости, активности пользователей. Доступен преподавателям и администраторам для анализа результатов обучения.

#### 6) Управление учебными группами

Обеспечивает создание и администрирование групп, распределение студентов и назначение преподавателей.

#### 7) Принципы взаимодействия:

- все модули связаны через единый API-интерфейс,
- обмен данными осуществляется по четко определенным протоколам,
- модуль может быть обновлен независимо от других.

Каждый компонент взаимодействует с другими через четко определенные API-интерфейсы, что делает систему гибкой и устойчивой к изменениям.

#### Выводы по главе

В ходе проектирования модуля была разработана архитектура, удовлетворяющая всем современным требованиям к системам



дистанционного обучения. Уделено внимание безопасности, масштабируемости, удобству взаимодействия для разных ролей пользователей.

Решения по выбору технологий и структурных компонентов позволяют обеспечить стабильную работу системы в условиях реального использования в железнодорожной отрасли, а также создают задел для дальнейшего расширения функциональности. Благодаря продуманной архитектуре, модуль можно легко адаптировать под новые требования.

## **3 РЕАЛИЗАЦИЯ И АТТЕСТАЦИЯ МОДУЛЯ СИСТЕМЫ**

### **3.1 Реализация функциональности модуля (языки программирования, фреймворки, библиотеки)**

Реализация модуля производилась с применением инструментов, ранее обоснованных на этапе проектирования. На практике были реализованы следующие компоненты:

#### **1) Серверная часть.**

Система реализована в виде REST API с использованием фреймворка FastAPI. Настроены основные маршруты:

- /auth — для авторизации и аутентификации,
- /users — для управления пользователями,
- /tests, /training, /groups — для работы с контентом.

Используется Pydantic для валидации входных и выходных данных, что позволило минимизировать количество ошибок и повысить надежность.

#### **2) Клиентская часть.**

Мобильное приложение создано на Kivy. Реализованы основные экраны:

- авторизация,
- главное меню,
- прохождение тестов,
- просмотр результатов.

Интерфейс адаптирован под сенсорное управление и использует .kv-файлы для описания разметки.

#### **3) Работа с базой данных.**

На этапе разработки применялась SQLite. Структура базы описана через SQLAlchemy. Для управления БД использовалась среда DataGrip (для отладки SQL-запросов и визуализации структуры).

#### **4) Управление проектом**

Код размещен в репозитории на GitHub.

Таким образом, на этом этапе проект был переведен из стадии проектирования в стадию работающего прототипа, в котором реализованы все

основные функции. Реализация симуляторного модуля запланирована на следующий этап разработки в связи с:

- необходимостью дополнительного анализа предметной области для моделирования аварийных ситуаций;
- требованием глубокой интеграции с системами визуализации;
- планируемым привлечением экспертов РЖД для валидации сценариев

### **3.2 Интеграция модуля с источниками данных (API, SQL-запросы)**

Связь между клиентским приложением (Kivy) и сервером (FastAPI) осуществляется по протоколу HTTP с использованием архитектурного стиля REST. Поддерживаются методы GET, POST, PUT, DELETE. Клиент инициирует запросы, сервер обрабатывает и возвращает данные в формате JSON.

Примеры ключевых маршрутов:

- POST /auth/login — авторизация пользователя;
- GET /tests — получение доступных тестов;
- POST /results — отправка результатов прохождения теста;
- GET /results/user/{id} — получение персональной статистики.

Таблица 2 – Примеры ключевых эндпоинтов API

HTTP-метод	Эндпоинт	Описание
POST	/auth/login	Авторизация пользователя
POST	/auth/register	Регистрация нового пользователя
GET	/tests	Получение списка тестов
POST	/results	Сохранение результатов тестирования
GET	/results/user/{id}	Получение результатов конкретного пользователя

Взаимодействие между клиентом и сервером осуществляется через REST API, разработанный на FastAPI. API предоставляет следующие основные эндпоинты:

- /login, /register — аутентификация и регистрация;
- /materials — получение учебных материалов;
- /tests, /task, /results — прохождение и результаты тестирования;

– /progress — отображение индивидуального прогресса.

Работа с базой данных организована через SQLAlchemy, используются как декларативные модели, так и кастомные SQL-запросы при необходимости оптимизации.

API реализует валидацию входных данных, сериализацию/десериализацию, а также возврат статусов и ошибок в стандартизированном формате.

```
17     def load_groups(self):
18         self.ids.groups_box.clear_widgets()
19         headers = {
20             "Authorization": f"Bearer {App.get_running_app().token}",
21             "Content-Type": "application/json"
22         }
23
24         URLRequest(
25             url=f"{App.get_running_app().api_url}/groups/",
26             req_headers=headers,
27             on_success=self.on_success,
28             on_error=self.on_error,
29             method='GET'
30         )
```

Рисунок 7 - Пример запроса на клиенте

Этот код загружает список учебных групп с сервера и отображает их в интерфейсе.

Для хранения и обработки данных используется SQLite, с возможностью миграции на более производительные СУБД (например, PostgreSQL). Все таблицы описаны с использованием SQLAlchemy в декларативном стиле.

```
18 class StudyGroup(Base):
19     __tablename__ = 'study_groups'
20
21     id = Column(String, primary_key=True, default=lambda: str(uuid.uuid4()))
22     name = Column(String(100), nullable=False)
23     description = Column(Text)
24     created_at = Column(DateTime, default=datetime.utcnow)
25
26     members = relationship(argument="StudyGroupMember", back_populates="group")
```

Рисунок 8 - Пример модели

Модель StudyGroup описывает таблицу групп, а relationship связывает ее с таблицей участников через промежуточную таблицу StudyGroupMember.

```
9 @router.get("/groups/{group_id}/members/")
10 def get_group_members(group_id: str, db: Session = Depends(get_db)):
11     group = db.query(StudyGroup).filter(StudyGroup.id == group_id).first()
12     if not group:
13         raise HTTPException(status_code=404, detail="Группа не найдена")
14
15     members = db.query(User).join(StudyGroupMember).filter(StudyGroupMember.group_id == group_id).all()
16     return [{"id": m.id, "username": m.username, "fio": m.fio, "role": m.role} for m in members]
17
```

Рисунок 9 - Получение данных через ORM

Здесь реализуется выборка участников конкретной группы с использованием SQL JOIN через ORM.

```
44 class StudyGroupOut(BaseModel):
45     id: str
46     name: str
47     description: Optional[str] = None
48     created_at: datetime
49
50     Егорова Виктория
51     class Config:
52         orm_mode = True
53
54     Егорова Виктория
55
56 @app.get(path: "/groups/", response_model=List[StudyGroupOut])
57 def get_groups(db: Session = Depends(get_db)):
58     return db.query(StudyGroup).all()
```

Рисунок 10 - Стандартизированная отдача данных

Использование Pydantic-моделей обеспечивает строгую типизацию выходных данных, а FastAPI автоматически документирует структуру ответа через OpenAPI.

- все входные данные валидируются при помощи Pydantic;
- при ошибках клиент получает ответы в формате JSON с описанием и кодами ошибок (422, 404, 401 и др.);
- аутентификация осуществляется через JWT, полученный на этапе логина;
- все чувствительные операции требуют наличия заголовка Authorization.

Для визуализации базы и работы с SQL-запросами использовалась IDE DataGrip:

- отладка ORM-запросов;
- просмотр структуры таблиц;
- ручное редактирование и тестирование SQL;
- импорт/экспорт содержимого таблиц.

Такой подход обеспечивает гибкость, безопасность и удобство сопровождения в процессе дальнейшей разработки и эксплуатации системы.

### 3.3 Реализация клиентской части

Клиентское приложение реализовано с использованием кроссплатформенного Python-фреймворка Kivy, ориентированного на разработку мобильных интерфейсов. Архитектура разделена на визуальную часть (.kv-) и логическую (Python), что обеспечивает удобство поддержки и масштабируемость.

Основные экраны приложения:

- 1) Экран авторизации: форма входа с вводом логина и пароля, выполнение запроса к серверу(рис.11)

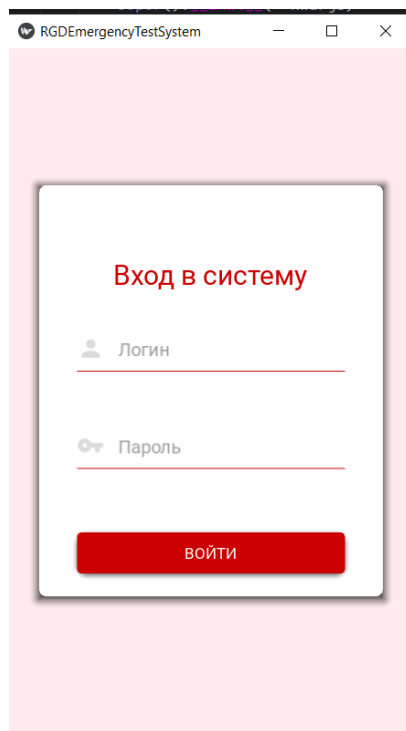


Рисунок 11 - Экран авторизации

Предоставляет пользователю форму ввода логина и пароля. После подтверждения выполняется HTTP-запрос к серверу с передачей введенных данных. В случае успешной авторизации происходит переход на главное меню, в противном случае отображается соответствующее сообщение об ошибке.

```
16     def login(self):
17         username = self.ids.username.text.strip()
18         password = self.ids.password.text.strip()
19         logging.info(f"Попытка входа пользователя: {username}")
20
21         if not username or not password:
22             self.show_error("Введите логин и пароль")
23         return
24
25         self.ids.login_btn.disabled = True
26         self.error_message = "Авторизация..."
27         login_data = {
28             "username": username,
29             "password": password
30         }
31
32         try:
33             URLRequest(
34                 self.api_url,
35                 req_body=json.dumps(login_data),
36                 on_success=self.handle_login_response,
37                 on_error=self.handle_login_error,
38                 req_headers={
39                     'Content-Type': 'application/json',
40                     'Accept': 'application/json'
41                 },
42                 timeout=10,
43                 method='POST'
44             )
45         except Exception as e:
46             self.handle_login_error( req: None, str(e))
```

Рисунок 12 - Функция авторизации

## 2) Главное меню: навигация к основным разделам

Содержит навигационные элементы, предоставляющие доступ к основным разделам приложения: «Тесты», «Профиль», «Группы» и др. Интерфейс адаптирован под мобильные устройства.

```

17 MDTopAppBar:
18     id: menu_button
19     title: root.screen_title
20     left_action_items: [["menu", lambda x: root.open_menu()]]
21     right_action_items: [["logout", lambda x: root.logout()]]
22     elevation: 2
23     md_bg_color: app.rjd_dark_red
24     specific_text_color: app.rjd_white
25     font_style: "H6"
26     pos_hint: {"top": 1}
27     size_hint_y: None
28     height: dp(56)
29

```

Рисунок 13 - Фрагмент кода отображения главного меню

В дальнейшем планируется расширение функционала приложения за счет реализации модуля отображения новостей, уведомлений и других пользовательских сообщений.



Рисунок 14 - Экран главного меню

3) Экран тестов: последовательное отображение существующих тестов, переход на окна создания, удаления, редактирования и прохождения.

Позволяет пользователю:

- просматривать доступные тесты;



- сортировать тесты по темам и типу
- создавать, редактировать и удалять тесты (если роль позволяет);
- запускать тест на выполнение;
- получать результат.

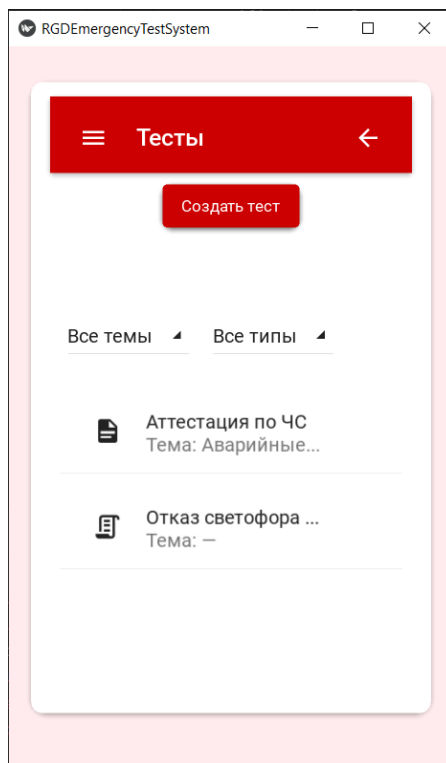


Рисунок 15 - Экран тестов

4) Профиль: отображение и изменение основной информации о пользователе

```

144  ~ 1 usage  👤 Егорова Виктория
145      def _save_success(self, req, result):
146          """Обработка успешного сохранения"""
147          self.show_loading(False)
148          self.show_success("Профиль успешно сохранен")
149          self.original_data = {
150              'username': result.get('username', ''),
151              'full_name': result.get('full_name', ''),
152              'email': result.get('email', '')
153          }
154          if hasattr(self.app, 'user_data'):
155              self.app.user_data.update(self.original_data)
156          self.edit_mode = False

```

Рисунок 16 - Фрагмент кода обновления данных пользователя

Позволяет пользователю просматривать и изменять свою основную информацию: имя, почта, роль и статус. Если пользователь обладает правами администратора, доступна функция управления другими пользователями.

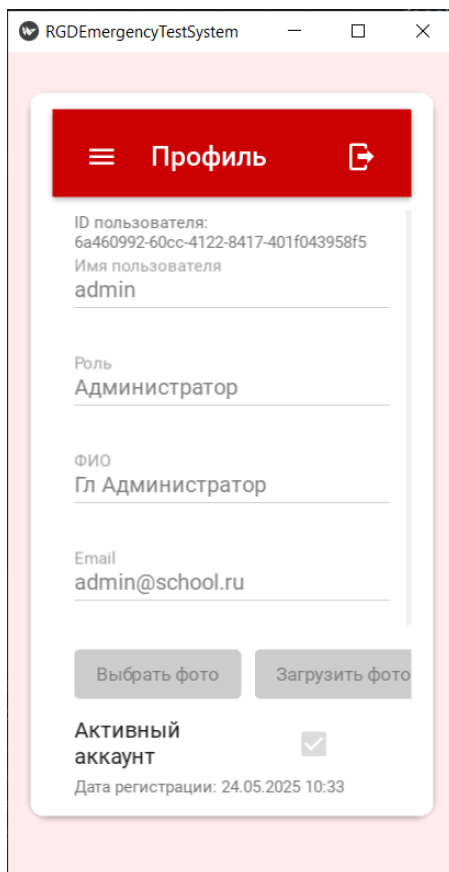


Рисунок 17 - Экран профиля

##### 5) Экран групп: отображение групп, созданных преподавателем

Предназначен для преподавателей, отображает список созданных ими учебных групп. В будущем может быть расширен возможностью добавления, удаления и редактирования групп.

Таким образом, клиентская часть приложения предоставляет удобный и расширяемый интерфейс, охватывающий ключевые функции системы. За счет использования Kivy/KivyMD удалось реализовать адаптивный и современный UI. В перспективе предусмотрено добавление новых разделов, улучшение визуального оформления и внедрение интерактивных элементов.

### 3.3 Интеграция и тестирование

По завершении этапа разработки были интегрированы основные модули клиентской части приложения, а также проведено начальное тестирование на предмет корректности взаимодействия между клиентом и сервером. Реализованный функционал позволил подтвердить работоспособность ключевых компонентов системы, а именно:

- реализован вход пользователей с различными ролями (администратор, преподаватель, студент);
- реализован механизм выхода из системы, предполагающий как удаление данных на клиентской стороне, так и отзыв токена на сервере;
- обеспечена возможность получения и отправки данных через REST API, соблюдением прав доступа, соответствующих роли пользователя;
- протестированы базовые сценарии работы с тестами: просмотр доступных тестов, создание, удаление, право доступа к манипуляциям.
- реализован переход к прохождению теста;
- протестирована работа с профилем пользователя;
- проверено отображение учебных групп преподавателей.

Часть тестирования отражена в приложении 1.

Тестирование проводилось в среде разработки, с использованием встроенных инструментов логирования и визуального контроля через интерфейс. Были проверены как корректные, так и граничные сценарии (ошибочные входы, отсутствие соединения с сервером, неправильный формат данных). Особое внимание уделялось стабильности обмена данными с сервером и обработке исключительных ситуаций.

В ходе тестирования приложение показало устойчивость к отказам соединения, корректную реакцию на ошибочные действия пользователя, а также соответствие ролей предоставленным возможностям.

Перспективным направлением развития системы является интеграция с разрабатываемым симулятором аварийных ситуаций. Реализация данного компонента потребует дополнительных работ по:

- разработке игровой логики и физики процессов;
- созданию базы типовых сценариев;
- оптимизации производительности для мобильных устройств.

Реализация симулятора является ключевым этапом. После завершения разработки и тестирования взаимодействия с симулятором можно будет переходить к интеграции с другими сервисами и модулями, а также расширению функциональности системы.

### **Вывод по главе**

В рамках третьей главы была осуществлена практическая реализация и первичная аттестация модуля информационной системы. Разработан и протестирован рабочий прототип, включающий серверную часть на FastAPI и клиентское мобильное приложение на базе Kivy. Реализована базовая функциональность: авторизация пользователей, работа с тестами, профилем и учебными группами, обеспечена безопасная передача данных через REST API и интеграция с базой данных на SQLite.

Тестирование подтвердило корректность функционирования основных сценариев взаимодействия пользователя с системой, а также выявило направления для дальнейшей доработки.

Таким образом, глава завершает этап перевода проекта из стадии проектирования в стадию работающего прототипа.

## **4 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И ВНЕДРЕНИЯ ПРОГРАММНОГО ПРОДУКТА**

### **4.1 Определение факторов эффективности разработки и внедрения программного продукта**

Изменение основных параметров объектов в результате разработки и внедрения программного продукта приводит к изменению эффективности деятельности предприятия. При этом могут наблюдаться как социальные, так и экономические эффекты.

В качестве факторов экономической эффективности разработанного продукта выступают:

- сокращение времени и затрат на техническую подготовку производства;

Во многих проектах, разрабатываемых компанией, присутствует дублирование функционала авторизации с минимальными отличиями. Реализация системы позволит уменьшить затраты на проектирование и внедрение компонента в новые системы.

- улучшение функционирования согласно основному назначению;

Исходя из написанного выше можно утверждать, что продукт можно использовать как централизованный модуль и минимизировать возможные ошибки при повторных реализациях.

- экономия фонда заработной платы;

Использование системы позволит сконцентрировать трудовые ресурсы на предметной области в новых проектах, что влечет за собой снижение трудозатрат и общего времени разработки.

- повышение производительности труда работников и др.

При написании новой логики часть функционала уже будет реализована, что позволяет получать данные быстрее и проще интегрировать их в новые процессы.

Разрабатываемый программный продукт предназначен для оптимизации внедрения авторизации в иные продукты, а также предоставление информации.

Для написания дальнейшего расчета необходимо учитывать количество написанного кода, который необходим для запуска проекта, в МБ.

В среднем для запуска одного проекта требуется около 4-7 МБ написанного кода. Если проект использует клиент-серверное взаимодействие и идентификацию пользователя для обработки, то примерно 20-30% это функции авторизации, для реализации которых необходимо около 0,8-1,4 МБ.

При использовании разрабатываемого продукта возможно сократить этот объем до 0,2МБ на один проект.

Также следует рассмотреть временные затраты для внедрения функционала. Как правило написание авторизации в проекте требует около 12 часов вместе с отладкой. При использовании разрабатываемого продукта время сократится до 4 часов.

#### **4.2 Определение трудоемкости разработки отдельных стадий программного продукта и затрат на оплату труда разработчиков**

Для расчета затрат на этапе проектирования необходимо определить продолжительность каждой работы, начиная с составления технического задания и заканчивая оформлением документов.

Трудоемкость относится к ключевым экономическим показателям и позволяет дать оценку эффективности использования рабочего времени при производстве товаров или услуг, а также при выполнении каких-либо работ. Этот коэффициент говорит о том, сколько труда (часов, дней или месяцев) нужно затратить для изготовления одной единицы продукции.

Трудоемкость разработки программного продукта определяется на основании времени реального времени, затраченного на его разработку.

Результаты длительности работ сведены в таблицу 3:

Таблица 3 – Длительность работ при разработке программного продукта

Наименование работ	Длительность работ, дней	Длительность работ, мес.
Анализ требований	2	0,1
Определение спецификаций	3	0,1
Проектирование	4	0,2
Кодирование	22	1,0
Тестирование	3	0,1
Сдача темы	6	0,3
<b>Итого:</b>	40	1,8

Длительность работ при разработке программного продукта в месяцах рассчитывается путем деления длительности работ в днях (Дрд) на количество рабочих дней в месяце (Кд = 22 дня):

$$\text{Дрм} = \text{Дрд} / \text{Кд}, [\text{мес.}] \quad (1)$$

Распределим выполняемые виды работ по исполнителям и определим их заработную плату за выполняемый объем работ.

Затраты на заработную плату по видам работ определяются путем умножения длительности работы в месяцах (Дрм) на оклад (Ок).

$$\text{ЗП} = \text{Дрм} * \text{Ок}, [\text{руб.}] \quad (2)$$

Результаты проведенных расчетов сведем в таблицу 4.

Таблица 4 – Расчет основной заработной платы разработчиков программного продукта

Наименование работ	Исполнитель	Длительность работ, мес.	Оклад специалиста, руб.	Затраты на ЗП, руб.
Анализ требований	Специалист по информационному обеспечению	0,1	62400	6240
Определение спецификации	Специалист по информационному обеспечению	0,1	62400	6240
Проектирование	Программист	0,2	48000	9600
Кодирование	Программист	1,0	48000	48000
Тестирование	Специалист по тестированию	0,1	33600	3360
Сдача темы	Программист	0,3	48000	14400
<b>Итого (ЗПобщ):</b>				87840

Отчисления от общего фонда оплаты труда рассчитываются по формуле:

$$\text{Озп} = \text{ЗПобщ} * 0,30, [\text{руб.}] \quad (3)$$

где ЗПобщ – общие затраты на заработную плату за выполненный объем работ, руб.

$$\text{Озп} = 87840 * 0,3 = 26352 \text{ руб.}$$

Фонд оплаты труда работников за разработку программного продукта определяется по формуле:

$$\text{ФОТ} = \text{ЗПобщ} + \text{Озп}, [\text{руб.}] \quad (4)$$

где ЗП общ – общие затраты на заработную плату за выполненный объем работ, руб.

Озп – сумма отчислений от общего фонда заработной платы, руб.

$$\text{ФОТ} = 87840 + 26352 = 114192 \text{ руб.}$$

#### **4.3 Расчет амортизационных отчислений от стоимости основных средств и нематериальных активов, участвующих в процессе разработки программного продукта**

Для разработки программного продукта используются основные фонды и нематериальные активы.

Основные фонды – средства труда, которые участвуют в производственном процессе многократно, сохраняя при этом свою натуральную форму, но переносят часть стоимости на производимую продукцию по мере износа. К основным фондам относятся здания, сооружения, машины и оборудование, измерительные и регулирующие приборы и устройства, вычислительная техника, транспортные средства, инструмент и прочие основные средства.

Перечень основных средств, используемых при разработке и внедрении программного продукта, и их стоимость приведены в таблице 5.



Таблица 5 – Перечень основных средств, используемых при разработке и внедрении программного продукта

Наименование используемых основных средств	Модель основных средств	Стоимость основных средств (руб.)
Ноутбук	HONOR MagicBook X15 i5	56000
Набор периферии (мышь)	Smartbuy	1700
Итого (Coc):		57700

Стоимость основных средств берется в соответствии со средней сложившейся на рынке ценой.

Расчет амортизационных отчислений от стоимости основных средств рассчитывается по формуле:

$$A_{oc} = (C_{oc} * Na_{oc}) / 100, [\text{руб.}] \quad (5)$$

где  $C_{oc}$  – стоимость основных средств, руб.

$Na_{oc}$  – норма амортизационных отчислений, % ( $Na = 6 \%$ )

$$A_{oc} = 57700 * 0,06 = 3462 \text{ руб.}$$

Нематериальные активы – это ценности, имеющие стоимостное выражение и не являющиеся физическими объектами, стоимость которых также, как и основных средств, включается в затраты частично, в соответствии с начисленным износом. Перечень нематериальных активов, используемых при разработке и внедрении программного продукта, и их стоимость приведены в Таблице 6.

Таблица 6 – Перечень нематериальных активов, используемых при разработке и внедрении программного продукта

Наименование используемых нематериальных активов	Характеристика нематериальных активов	Стоимость нематериальных активов (руб.)
Операционная система	Windows 10	4000
Интегрированная среда разработки	JetBrains Pycharm Professional 2024	19900
Итого (Cна):		23900

Наиболее часто к нематериальным активам относят используемое для разработки программное обеспечение, стоимость которого определяется на основе сложившейся на рынке цены.

Расчет амортизационных отчислений от стоимости нематериальных активов рассчитывается по формуле по формуле:

$$A_{на} = (C_{на} * H_{а.на.}) / 100, [\text{руб.}] \quad (6)$$

где  $C_{ос}$  – стоимость нематериальных активов, руб.

$H_{а.на.}$  – норма амортизационных отчислений, % ( $H_{а} = 3 \%$ )

$$A_{на} = 23900 * 0,03 = 717 \text{ руб.}$$

Сумма амортизационных отчислений, приходящаяся на программный продукт равна:

$$A_{о} = ((A_{ос} + A_{на}) / 12) * D_{рм}, [\text{руб.}] \quad (7)$$

где  $A_{ос}$  - амортизационные отчисления от стоимости основных средств, руб.

$A_{на}$  - амортизационные отчисления от стоимости нематериальных активов, руб.

$D_{рм}$  - длительности работы в месяцах;

$$A_{о} = ((3462 + 717) / 12) * 1,8 = 627 \text{ руб.}$$

#### **4.4 Расчет текущих затрат на проектирование и внедрение программного продукта**

Перечень текущих затрат зависит от расходов, связанных с разработкой программного обеспечения и области его применения. Наиболее часто встречаются затраты на:

- услуги хостинга;
- оплата Интернет-соединения;
- затраты на электроэнергию, потребляемую компьютером;

В среднем офисный компьютер потребляет в час 0,4 кВт/час электроэнергии, соответственно за восемь часов работы потребление составит 3,2 кВт/часа. Количество электроэнергии, потребленное компьютером в процессе разработки:

$$С_{эл.} = (3,2 * Д_{рд}) * Ц_{эл}, [\text{руб.}] \quad (8)$$

где  $Д_{рд}$  – длительность работы компьютера в днях;

$Ц_{эл.}$  – стоимость 1 кВт/часа электроэнергии, руб.

$$С_{эл.} = 3,2 * 40 * 6,5 = 832 \text{ руб.}$$

– коммунальные услуги, которые рассчитываются по формуле:

$$С_{ку} = Д_{рм} * К_{п}, [\text{руб.}] \quad (9)$$

где  $Д_{рм}$  - длительности работы в месяцах;

$К_{п}$  – стоимость коммунальных платежей за месяц, руб.

$$С_{ку} = 1,8 * 7800 = 14040 \text{ руб.}$$

– интернет, который рассчитывается по формуле:

$$С_{и} = Д_{рм} * И_{с}, (\text{руб.}) \quad (10)$$

где  $Д_{рм}$  – длительности работы в месяцах;

$И_{с}$  – стоимость интернет-соединения за месяц.

$$С_{и} = 1,8 * 1300 = 2340 \text{ руб}$$

Текущие затраты определяются за период времени, затраченный на разработку программного продукта и сведены в таблицу 7.

Таблица 7 – Текущие затраты на разработку и внедрение программного продукта

Вид текущих затрат	Единица измерения	Тариф, руб.	Стоимость за период разработки, руб.
Электроэнергия	Кв/ч	6,5	832
Коммунальные	-	7800	14040
Интернет	Мб/с	1300	2340
Итого (Тз):			17212

#### **4.5 Расчет прочих затрат на проектирование и внедрение программного продукта**

Прочие затраты принимаются в размере 5 – 10 % от суммы общей заработной платы.

$$\text{Пр.з} = \text{ЗПобщ} * \% \text{Пр.з, [руб.]} \quad (11)$$

где ЗПобщ – сумма общей заработной платы, руб.

$$\text{Пр.з} = 87840 * 0,06 = 5270 \text{ руб.}$$

#### **4.6 Расчет накладных затрат (коммерческие и управленческие расходы) на проектирование и внедрение программного продукта**

Накладные расходы принимаются в размере 3 – 5 % от суммы общей заработной платы.

$$\text{Нр} = \text{ЗПобщ} * \% \text{Нр, [руб.]} \quad (12)$$

где ЗПобщ – сумма общей заработной платы, руб.

$$\text{Нр} = 87840 * 0,04 = 3514 \text{ руб.}$$

#### **4.7 Определение сметы затрат на разработку программного продукта**

Себестоимость продукции - общая величина затрат на производство и реализацию продукции в денежном выражении. Фактические затраты, понесенные предприятием в связи с производством и реализацией продукции, группируются в разрезе калькуляционных статей.

Калькуляционная статья – определенный вид затрат, образующих себестоимость продукции в целом или отдельного ее вида.

Калькуляция полной себестоимости разработки программного продукта представлена в форме таблицы 6.

Таблица 8 – Калькуляция себестоимости разработки программного продукта

Номер п/п	Наименование статей расходов	Сумма (руб.)
1.	Фонд оплаты труда (ФОТ)	114192
2.	Сумма амортизационных отчислений (Ао)	627
<b>Цеховая себестоимость (Сц)</b>		114819
5.	Текущие затраты (Тз):	17212
6.	Прочие затраты (Пр.з)	5270
<b>Производственная себестоимость (Спр)</b>		137301
7.	Накладные расходы (Нр)	3514
<b>Полная себестоимость (Сп)</b>		140815

#### 4.8 Расчет экономической эффективности разработки программного продукта

Ниже будет рассчитана эффективность внедрения функционала авторизации в проект относительно обычного продукта (ручная обработка) и с использованием разрабатываемого продукта (как автоматики).

Экономия от замены ручной обработки информации на автоматизированную:

– затраты на ручную обработку информации:

$$З_p = O_{и} * Cч * \Gamma_d / H_v, [\text{руб.}] \quad (13)$$

где  $O_{и}$  – объем информации, обрабатываемой вручную, Мбайт; ( $O_{и} = 1,4$  (исходя из пункта 4.1))

$\Gamma_d$  – коэффициент, учитывающий дополнительные затраты времени на логические операции при ручной обработке информации ( $\Gamma_d = 3,4$  (установлен экспериментально));

$H_v$  – норма выработки, Мбайт/час. ( $H_v = 0,05$  проектов/час)

$Cч$  – стоимость одного часа работы, руб./час, которая рассчитывается по формуле:

$$Cч = \text{ФОТ} / (\text{Дрм} * \text{Кд.м.} * \text{Др}), [\text{руб.}] \quad (14)$$

где ФОТ – фонд оплаты труда, руб.

Дрм - длительности работы в месяцах;

Кд.м – количество рабочих дней в месяце, дни ( Кд.м = 22 дня)

Др – длительность рабочего дня, час. (Др = 8 часов)

$$Сч = 114192 / (1,8 * 22 * 8) = 360 \text{ руб.}$$

$$З_p = 1,4 * 360 * 3,4 / 0,05 = 34272 \text{ руб.}$$

– затраты на автоматизированную обработку информации:

$$З_a = O_a * Сч * \Gamma_d / H_v, [\text{руб.}] \quad (15)$$

где  $O_a$  – объем информации, обрабатываемой вручную, с учетом автоматики, Мбайт; ( $O_a = 0,2$  (исходя из пункта 4.1))

$\Gamma_d$  – коэффициент, учитывающий дополнительные затраты времени на логические операции при ручной обработке информации ( $\Gamma_d = 3,4$  (установлен экспериментально));

$H_v$  – норма выработки, Мбайт/час. ( $H_v = 0,05$  проектов/час)

$Сч$  – стоимость одного часа работы, руб./час

$$З_a = 0,2 * 360 * 3,4 / 0,05 = 4896 \text{ руб.}$$

– экономия от замены ручной обработки информации на автоматизированную:

$$\mathcal{E}_y = З_p - З_a, [\text{руб.}] \quad (16)$$

где  $З_p$  – затраты на ручную обработку информации, руб.;

$З_a$  – затраты на автоматизированную обработку информации, руб.

$$\mathcal{E}_y = 34272 - 4896 = 29376 \text{ руб.}$$

Экономический эффект (экономия) от использования программного продукта за год:

$$\mathcal{E}_r = \mathcal{E}_y - 0,135 * C_{п}, [\text{руб.}] \quad (17)$$

где  $C_p$  – полная себестоимость программного продукта, руб.

$$\mathcal{E}_r = 29376 - 0,135 * 140815 \text{ руб.} = 10366 \text{ руб.}$$

Срок окупаемости программного продукта

$$T = C_p / \mathcal{E}_r, \text{ год. (мес.)} \quad (18)$$

где  $C_p$  – полная себестоимость программного продукта, руб.

$\mathcal{E}_r$  - экономический эффект (экономия) от использования программного продукта, руб.

$$T = 140815 / 10366 = 13,6 \text{ мес.} \approx 1,1 \text{ г.}$$

#### 4.9 Экономические показатели разработки программного продукта

Таблица 9 – Экономические показатели разработки программного продукта

Наименование	Единица измерения	Значения показателя
Длительность работ при разработке программного продукта	дни	40
Полная себестоимость программного продукта	руб.	140815
Экономия от замены ручной обработки информации на автоматизированную	руб.	29376
Экономический эффект (экономия) от использования программного продукта	руб.	10366
Срок окупаемости программного продукта	г.	1.1

#### Вывод по главе

В ходе экономического обоснования была проанализирована эффективность разработки и внедрения программного модуля.

Выявлены ключевые факторы экономической эффективности:

- сокращение времени и трудозатрат на разработку и отладку повторяющихся компонентов;
- уменьшение фонда заработной платы за счёт перераспределения ресурсов;
- повышение производительности труда за счёт повторного использования функционала;

Были рассчитаны основные экономические показатели:

- полная себестоимость разработки: 140 815 руб.;
- годовой экономический эффект от внедрения: 10 366 руб.;
- срок окупаемости продукта: 1,1 года.

Полученные результаты подтверждают, что реализация программного модуля является экономически обоснованной и обеспечивает возврат вложений в приемлемые сроки. Разработанное решение демонстрирует высокий уровень экономической эффективности и может быть рекомендовано к практическому применению.



## ЗАКЛЮЧЕНИЕ

В процессе разработки была создана модульная система тестирования и обучения, включающая клиентскую часть на базе кроссплатформенного фреймворка Kivu и серверную часть, реализованную с использованием современного веб-фреймворка FastAPI. Модуль реализует базовую функциональность, необходимую для взаимодействия пользователей с различными ролями: авторизацию, управление тестами и группами, а также просмотр профиля.

Разработка была ориентирована на соответствие требованиям масштабируемости, безопасности и возможности дальнейшей интеграции в существующую корпоративную ИТ-инфраструктуру. Для этого архитектура модуля изначально строилась по принципу модульности, с четким разделением бизнес-логики, пользовательского интерфейса и интерфейсов взаимодействия с внешними системами. Применение REST API обеспечило совместимость с корпоративными платформами и заложило основу для последующего расширения функциональности.

На текущем этапе реализован базовый набор возможностей, включая:

- авторизацию и разграничение прав пользователей по ролям;
- управление тестами и пользовательскими группами;
- просмотр результатов и базовую аналитику;
- клиент-серверную архитектуру с потенциальной поддержкой мобильных устройств.

В процессе разработки был сформирован перечень направлений для дальнейшего развития системы, включая:

- внедрение механизма восстановления доступа через подтверждение личности;
- добавление многоязычного интерфейса;
- переход к автоматизированному тестированию;
- расширение аналитики и отчетности;
- внедрения модуля симуляции.

Сценарный подход, реализованный в модуле, позволяет пользователям пошагово проходить ветвящиеся ситуации, основанные на действиях в условиях ЧС. Это функционально выводит систему за рамки типичных LMS-решений и делает её применимой к реальным критическим ситуациям на транспорте.

Экономическая часть проекта подтвердила его финансовую целесообразность. Внедрение разработанного модуля позволяет снизить трудозатраты, оптимизировать фонд оплаты труда за счёт повторного использования кода и перераспределения ресурсов, а также повысить производительность разработки. Расчёты показали, что срок окупаемости составляет 1,1 года, а годовой экономический эффект — 10 366 рублей, что делает проект выгодным для внедрения в масштабах предприятия.

Проект полностью реализует поставленные цели: разработан, протестирован и документирован программный модуль, ориентированный на обучение и проверку действий сотрудников в условиях аварийных ситуаций. Полученные результаты демонстрируют как практическую применимость решения, так и потенциал его масштабирования и интеграции в цифровую инфраструктуру РЖД.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

### Нормативно-правовые источники

1. ГОСТ 19.101–77. Единая система программной документации. Виды программ и программных документов.
2. ГОСТ 19.201–78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.
3. ГОСТ 34.602–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
4. ГОСТ 28195–89. Оценка качества программных средств. Общие положения.
5. ГОСТ Р ИСО/МЭК 8631–94. Информационная технология. Программные конструктивы и условные обозначения для их представления.
6. ГОСТ Р ИСО/МЭК 9126–93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.
7. ГОСТ Р ИСО/МЭК 15288–2005. Информационная технология. Системная инженерия. Процессы жизненного цикла систем.
8. ГОСТ Р ИСО/МЭК 15910–2002. Информационная технология. Процесс создания документации пользователя программного средства.
9. ГОСТ Р ИСО 9241-210–2012. Эргономика взаимодействия человек-система. Ориентированный на пользователя процесс проектирования интерактивных систем.
10. ГОСТ 7.32–2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления
11. ГОСТ 7.1–2003. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления

12. ГОСТ 7.9—95 Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования

13. ГОСТ 7.11—2004 (ИСО 832:1994) Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках

14. ГОСТ 7.12—93 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила

15. ГОСТ 7.80—2000 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления

16. ГОСТ 7.82—2001 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления

17. ГОСТ Р 7.0.97–2016 Национальный стандарт Российской Федерации. Система стандартов по информации, библиотечному и издательскому делу. Организационно-распорядительная документация. Требования к оформлению документов

18. ГОСТ Р 7.0.100–2018 Библиографическая запись. Общие требования и правила составления

#### Учебные пособия

19. Бейдер, Д. Чистый Python. Тонкости программирования для профи / Д. Бейдер. — СПб. : Питер, 2022. — 288 с. — (Библиотека программиста).

20. Методические рекомендации по организации выполнения и защиты выпускной квалификационной работы (дипломный проект) в КМПО РАНХиГС. — М. : КМПО РАНХиГС, 2023. — 50 с.

21. Мэтиз, Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения 3-е изд., дополненное и переработанное / Э. Мэтиз. — СПб. : Питер, 2025. — 496 с. — (Библиотека программиста).

22. Riggs, S., Ciolli, G. PostgreSQL 16 Administration Cookbook / S. Riggs, G. Ciolli. — Packt Publishing, 2023. — 432 p.

23. Smith, J. P. Entity Framework Core in Action / J. P. Smith. — 2nd ed. — Shelter Island : Manning Publications, 2023. — 576 p. — ISBN 978-1-61729-836-8.

#### Интернет-источники

24. Академия РЖД. Предупреждение и ликвидация ЧС [Электронный ресурс]. — URL: <https://rzda.ru/services/akademiya/preduprezhdenie-i-likvidatsiya-chrezvychaynykh-situatsiy-na-zheleznodorozhnom-tra/> (дата обращения: 11.03.2025).

25. АСПТ РЖД [Электронный ресурс]. — URL: <https://sdo-kaskor-aspt.ru/aspt> (дата обращения: 01.03.2025).

26. Железнодорожная отрасль: цифры и факты [Электронный ресурс] // ТАСС. — URL: <https://tass.ru/ekonomika/23115697> (дата обращения: 11.03.2025).

27. Информационные технологии в РЖД [Электронный ресурс] // TAdviser. — URL: <https://www.tadviser.ru/index.php/> (дата обращения: 11.03.2025).

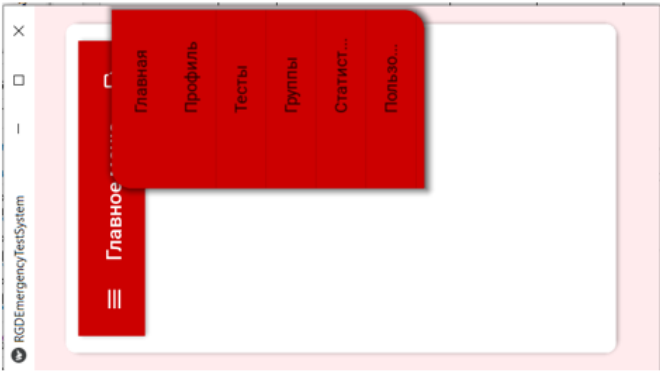
28. Концепция развития цифровой модели бизнеса в Холдинге РЖД [Электронный ресурс]. — URL: <https://www.irgups.ru/sites/default/files/irgups/obrazovanie/Innovation%20and%20Technology%20Center%20for%20the%20Development%20of%20the%20Eastern%20Test%20Site/Концепция%20развития%20цифровой%20модели%20бизнеса%20в%20холдинге%20РЖД.pdf> (дата обращения: 11.03.2025).

29. Корпоративный университет РЖД [Электронный ресурс]. — URL: <https://universitetrzd.ru/> (дата обращения: 11.03.2025)

30. Отчет МСКЖД: статистика происшествий на железных дорогах [Электронный ресурс] // ZDMир. — URL: <https://zdmira.com/articles/otchet-mszhd-2024-statistika-proisshествij-na-zheleznikh-dorogakh> (дата обращения: 11.03.2025).
31. Работа в РЖД [Электронный ресурс]. — URL: <https://social.rzd.ru/> (дата обращения: 22.03.2025).
32. Сервисы РЖД объединятся в экосистему [Электронный ресурс] // RZDDigital. — URL: <https://rzddigital.ru/opinions/servisy-rzhd-obedinyatsya-v-ekosistemu/> (дата обращения: 11.03.2025).
33. Создание собственного API на Python (FastAPI): Знакомство и первые функции [Электронный ресурс] // Хабр. — URL: <https://habr.com/ru/companies/amvera/articles/826196/> (дата обращения: 11.03.2025).
34. Стратегия развития железнодорожного транспорта в Российской Федерации до 2030 года [Электронный ресурс]. — URL: <https://mintrans.gov.ru/documents/1/1010> (дата обращения: 11.03.2025).
35. Тесты при приеме на работу в РЖД [Электронный ресурс]. — URL: <https://testlearn.ru/blog/test-v-rzhd> (дата обращения: 11.03.2025).
36. Kivy — фреймворк для кроссплатформенной разработки №1 [Электронный ресурс] // Хабр. — URL: <https://habr.com/ru/articles/418839/> (дата обращения: 11.03.2025).
37. Kivy · PyPI [Электронный ресурс]. — URL: <https://pypi.org/project/Kivy/> (дата обращения: 11.03.2025).
38. KivyMD Documentation [Электронный ресурс]. — URL: <https://kivymd.readthedocs.io/en/latest/index.html> (дата обращения: 11.03.2025).
39. PostgreSQL Documentation / PostgreSQL Global Development Group [Электронный ресурс]. — URL: <https://www.postgresql.org/docs/> (дата обращения: 11.03.2025).
40. Welcome to Python.org [Электронный ресурс]. — URL: <https://www.python.org/> (дата обращения: 11.03.2025).

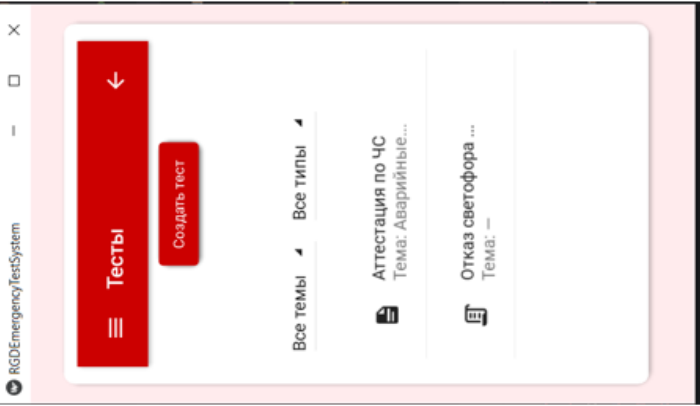
## ПРИЛОЖЕНИЕ А


### Тест-кейс

ID	Описание	Шаги	Входные данные	Ожидаемые результаты	Фактические результаты	Статус
ТС-01	Авторизация пользователь (админа)	<ol style="list-style-type: none"> <li>1. Открыть приложение</li> <li>2. Ввести логин и пароль</li> <li>3. Нажать «Войти»</li> <li>4. Открыть меню и убедиться, что есть расширенное меню админа.</li> </ol>	<p>Логин: admin</p> <p>Пароль: admin</p>	<p>Переход в главное меню под ролью администратора.</p> <p>От сервера: 200 ОК.</p>	<p>Переход выполнен, ошибок нет.</p>  <pre> sqlalchemy.engine.Engine [cached "POST /auth/login HTTP/1.1" 200 OK sqlalchemy.engine.Engine COMMIT                     </pre>	Пройден

ID	Описание	Шаги	Входные данные	Ожидаемые результаты	Фактические результаты	Статус
ТС-02	Ошибка при вводе неверного пароля	1. Ввести логин 2. Ввести неверный пароль 3. Нажать «Войти»	Логин: user1 Пароль: 111	Появляется сообщение об ошибке	Сообщение: «Неверные учетные данные» <pre>0 sqlalchemy.engine.Engine [cached] 0 sqlalchemy.engine.Engine ROLLBACK "POST /auth/login HTTP/1.1" 401</pre>	Пройден
ТС-03	Просмотр профиля пользователя	1. Авторизоваться 2. Открыть раздел «Профиль»	токен пользователя	Отображаются имя, роль, e-mail	 <p>Информация отображается корректно, кроме ф.и. В процессе разработки утеряно.</p>	Не пройден. Требуется доработка.



ID	Описание	Шаги	Входные данные	Ожидаемые результаты	Фактические результаты	Статус
ТС-04	Получение списка тестов преподавателем	1. Авторизоваться под ролью преподавателя 2. Открыть через меню страничку тестов. 3. Попробовать сортировку по темам и типам. 4. Нажать кнопку создания тестов.	токен пользователя	Список доступных тестов. Возможность работать с сортировкой по темам и типу теста(сценарный или обычный). Для преподавателя возможность создавать тесты	 <p>Тесты показываются корректно. Сортировка работает. Кнопка создания есть и работает переход.</p>	Пройден

ID	Описание	Шаги	Входные данные	Ожидаемые результаты	Фактические результаты	Статус
	Создание теста	<ol style="list-style-type: none"> <li>1. Авторизоваться под ролью преподавателя</li> <li>2. Открыть через меню страничку тестов.</li> <li>3. Нажать кнопку создания тестов.</li> <li>4. Заполнить форму теста, выбрать тему и количество заданий.</li> <li>5. Сохранить тест.</li> <li>6. Создать задания и сохранить.</li> </ol>	Форма заполнена	Тест сохранен в БД.		Пройден

## ПРИЛОЖЕНИЕ Б

### Руководство пользователя

#### 1. Введение

Программный модуль предназначен для проведения тестирования профессиональных компетенций в аварийных ситуациях.

Целевая аудитория:

- сотрудники РЖД (студенты);
- преподаватели/администраторы (создание тестов, управление группами).

#### 2. Интерфейс и функции

Экраны приложения:

- Авторизация и регистрация:

Ввод логина/пароля (роли: студент, преподаватель, администратор).

Создание нового пользователя (студента)

- Главное меню:

Доступ к тестам, группам, профилю.

- Тестирование:

Прохождение тестов с таймером и автоматической проверкой.

- Администрирование (только для преподавателей):

Создание и редактирование тестов;

Управление учебными группами;

Просмотр статистики.

#### 3. Частые проблемы и решения

Проблема	Решение
"Ошибка 401"	Проверьте логин/пароль.
Нет доступа к управлению тестами или группами	Убедитесь, что ваша роль имеет соответствующие права.
Медленная работа	Проверьте подключение.

Техподдержка: support@rzd-learning.ru.

## ПРИЛОЖЕНИЕ В

### Инструкция по установке и настройке

#### 1. Системные требования

ОС: Windows 10/11, Linux (Ubuntu 22.04+), Android/iOS

Память: 4 ГБ ОЗУ (рекомендуется 8 ГБ)

Зависимости: Python 3.10+, FastAPI, Kivy, SQLite

#### 2. Развертывание серверной части (FastAPI)

Установите Python 3.10+ и зависимости: *pip install -r requirements.txt*

Запустите сервер: *uvicorn main:app --reload*

API будет доступен по адресу: <http://localhost:8000/docs> (Swagger UI)

#### 3. Развертывание клиентской части (Kivy)

Запустите клиентское приложение: *python main.py*

#### 4. Настройка проекта

1) Клонирование репозитория:

*git clone https://github.com/pelmesheck007/RGDEmergencyTestSystem.git*

2) Настройка БД: измените параметры в `api/run.py`

3) Настройка клиента: измените URL сервера в `client/config.py`

#### 5. Интеграция с ИТ-инфраструктурой РЖД

Используйте REST API, доступный через Swagger-документацию. При необходимости подключайте модули авторизации и логирования.

#### Приложения

Скриншоты интерфейса: см. раздел 3.3 ВКР

Диаграммы БД: Рисунки 3-5

Примеры API-запросов: Таблица 2

Данное руководство охватывает все этапы работы с модулем: от установки до анализа результатов.

Дальнейшее развитие:

- интеграция с симулятором аварийных ситуаций;
- расширение модуля аналитики;
- улучшение пользовательского интерфейса.