

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

**Разработка системы информационной поддержки иностранных
студентов**

Студент

(подпись, дата)

Макбол Али Али Омар

(ФИО полностью)

Руководитель

(подпись, дата)

Кольчугина Е.А.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 9.06.2026 г. №12)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Кафедра

Вычислительной техники

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«20» декабря 2025 г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Макбол Али Али Омар гр. 22ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Разработка системы информационной поддержки иностранных студентов

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

2. Руководитель работы д.т.н., профессор кафедры «МОиПЭВМ» Кольчугина Е.А.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

- 1) Разработать приложение-личный кабинет для студентов RagEdu
- 2) Среда функционирования: web-браузер под управлением ОС Windows 11
- 3) Среда разработки: ОС Windows 11

Языковые средства разработки: PHP, [Vue.js](#), JavaScript, HTML, CSS

Инструментальные средства: JetBrains WebStorm, PHP Storm, SQL Server,
FigJam

Браузеры для тестирования: Google Chrome, Mozilla Firefox, Yandex

- 4) Основные функции приложения:

- 1) Анализ предметной области и постановка задачи;
- 2) Проектирование;
- 3) Реализация;
- 4) Отладка и тестирование;
- 5) Контроль качества приложения;
- 6) Оформление пояснительной записки.

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2026 – 14.04.2026	
2	Проектирование программных средств	20%	15.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	26.04.2026 – 12.05.2026	
4	Тестирование программных средств	20%	13.05.2026 – 22.05.2026	

5	Оформление пояснительной записки	20%	23.05.2026 – 03.06.2026	
---	----------------------------------	-----	----------------------------	--

Дата выдачи задания «20» декабря 2025г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 140 листов, 30 рисунков, 5 таблиц, 34 использованных источника и 7 приложений.

ЛИЧНЫЙ КАБИНЕТ СТУДЕНТА, WEB-ПРИЛОЖЕНИЕ, REST API,
PHP, VUE.JS, JavaScript, MYSQL.

Объект исследования – процесс информационного сопровождения иностранных студентов, поступающих в образовательные организации Российской Федерации.

Цель выпускной квалификационной работы – проектирование и реализация системы информационной поддержки иностранных студентов для компании Russian Arabian Gate с интеграцией в CRM-систему Bitrix24 и обеспечением централизованного хранения данных обучающихся.

Технологии разработки – в среде разработки Visual Studio code была разработана клиентская часть приложения на языке программирования Vue.js и Java Script. В среде разработки PhpStorm на языке программирования PHP была реализована серверная часть приложения. В качестве системы управления базами данных была выбрана phpmyadmin. Взаимодействие между клиентской и серверной частями осуществляется через API.

Результатом работы является функционирующее веб-приложение, внедрённое в деятельность компании Russian Arabian Gate и обеспечивающее автоматизацию процессов сопровождения иностранных студентов.

					ПГУ 09.03.02– 8BP211.21 ПЗ			
Изм.	Лист	№ докум	Подпись	Дата				
Разр	Макбол А.А.О.				Разработка системы информационной поддержки иностраннх студентов. Пояснительная записка	Лит.	Лист	Листов
Про	Кольчугина Е.А.						4	140
Рец						Группа 22ВИ1		
Н.	Такташкин Д.В.							
Утв.	Козлов А.Ю.							

Содержание

Введение.....	7
1 Анализ требований к поставленной задаче и предметной области.....	10
1.1 Постановка задачи.....	10
1.2 Анализ требований.....	12
1.3 Выбор методов и средств разработки.....	13
2 Проектирование системы.....	14
2.1 Проектирование архитектуры системы и программного обеспечения.....	14
2.2 Проектирование структур данных.....	20
2.2.1 Концептуальная модель базы данных.....	20
2.2.2 Логическая модель базы данных.....	21
2.2.3 Физическая модель базы данных.....	23
2.3 Проектирование интерфейса.....	23
3 Реализация приложения-личного кабинета для RagEdu.....	26
3.1 Общий состав разработанного приложения.....	26
3.2 Реализация серверной части приложения.....	27
3.3 Реализация клиентской части приложения.....	29
3.4 Реализация интерфейса приложения.....	31
4 Отладка и тестирование.....	45
4.1 Обоснование выбора режима тестирования.....	45
4.2 Функциональное тестирование веб-приложения.....	46
5 Внедрение и опыт эксплуатации программного обеспечения.....	49
6 Расчет бюджета разработки. Планирование разработки и оценка бюджета.....	50
Заключение.....	51
Список использованных источников.....	52
Приложение А (справочное) Глоссарий.....	55
Приложение Б (обязательное) Атрибуты сущностей базы данных.....	57
Приложение В (обязательное) Листинг серверной части приложения.....	60
Приложение Г (обязательное) Листинг клиентской части.....	88
Приложение Д (обязательное) Результаты тестирования клиентской части.....	132
Приложение Е (обязательное) Листинг модульных тестов.....	136
Приложение Ж (обязательное) Копия акта внедрения программного обеспечения.....	139

Введение

В условиях активной цифровой трансформации образовательной сферы информационные технологии становятся неотъемлемой частью процессов сопровождения обучающихся. Особенно актуальной данная тенденция является в сфере международного образования, где взаимодействие между организациями и иностранными студентами требует высокой степени координации, прозрачности и оперативности обмена информацией.

Российская Федерация остаётся одним из востребованных направлений получения высшего образования для иностранных граждан, в том числе студентов из арабских стран. Процесс поступления в российские образовательные организации включает в себя подачу и проверку значительного объёма документов, отслеживание статуса зачисления, взаимодействие с представителями принимающей стороны и решение организационных вопросов. Эффективность этих процессов напрямую влияет на качество предоставляемых услуг и уровень доверия со стороны студентов.

Компания Russian Arabian Gate с 2016 года осуществляет комплексную поддержку арабских студентов, желающих обучаться в Российской Федерации. Организация сопровождает обучающихся на всех этапах – от выбора образовательной программы и подготовки документов до завершения обучения. Однако в процессе развития компании и увеличения количества сопровождаемых студентов возникла необходимость создания единой цифровой платформы, обеспечивающей централизованное хранение информации и удобный доступ к данным.

До начала описываемой далее разработки у студента отсутствовал личный кабинет, который выполнял бы функцию единого хранилища документов и позволял бы отслеживать статус зачисления. Информация хранилась разрозненно, что усложняло процессы взаимодействия и увеличивало временные затраты на обработку данных. Кроме того,

существовала необходимость интеграции с корпоративной CRM-системой Bitrix24 для автоматизации создания и управления контактами студентов.

Актуальность темы выпускной квалификационной работы обусловлена потребностью в разработке современного веб-приложения, обеспечивающего автоматизацию процессов сопровождения иностранных студентов, повышение прозрачности работы компании и улучшение качества сервиса. Создание личного кабинета студента позволяет структурировать данные, минимизировать риски потери информации, сократить временные затраты сотрудников и обеспечить удобный доступ к сведениям для пользователей.

Объектом исследования является процесс информационного сопровождения иностранных студентов, поступающих в образовательные организации Российской Федерации.

Предметом исследования является разработка веб-приложения «Личный кабинет студента» с интеграцией в CRM-систему Bitrix24.

Объектом исследования является процесс информационного сопровождения иностранных студентов, поступающих в образовательные организации Российской Федерации.

Предметом исследования является разработка веб-приложения «Личный кабинет студента» с интеграцией в CRM-систему Bitrix24.

Целью выпускной квалификационной работы является проектирование и реализация адаптивного веб-приложения для компании Russian Arabian Gate, обеспечивающего функционирование личного кабинета студента с возможностью хранения документов, отображения статуса зачисления и автоматической синхронизацией данных с CRM-системой Bitrix24 посредством REST API.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ предметной области и существующих бизнес-процессов компании;

- определить функциональные и нефункциональные требования к системе;
- разработать архитектуру веб-приложения и структуру базы данных;
- реализовать серверную часть приложения с использованием PHP и СУБД MySQL;
- реализовать клиентскую часть адаптивного веб-приложения на языке VUE и JS;
- обеспечить интеграцию с CRM-системой Bitrix24 через REST API с автоматическим созданием контактов после регистрации пользователя;
- реализовать механизм регистрации и авторизации пользователей;
- внедрить многоязычную поддержку (русский, английский и арабский языки);
- провести тестирование разработанной системы;
- выполнить оценку качества программного кода.

Практическая значимость работы заключается в создании и внедрении функционирующего веб-приложения в деятельность компании Russian Arabian Gate (RagEdu). Разработанная система позволяет централизовать хранение документов студентов, повысить прозрачность статуса обработки заявок и автоматизировать взаимодействие с CRM-системой, что способствует повышению эффективности работы компании и улучшению качества предоставляемых услуг.

Таким образом, разработка личного кабинета студента является важным этапом цифровой модернизации компании и соответствует современным тенденциям автоматизации образовательных процессов.

1 Анализ требований к поставленной задаче и предметной области

1.1 Постановка задачи

Основной задачей выпускной квалификационной работы является разработка системы информационной поддержки иностранных студентов для компании Russian Arabian Gate, предназначенного для автоматизации процесса информационного сопровождения иностранных студентов, поступающих в образовательные организации Российской Федерации. При разработке следует учесть современный уровень развития информационных технологий и систем, методов и средств их разработки [1-34].

Разрабатываемая система должна обеспечить создание единого цифрового пространства для хранения и отображения информации о студенте, его документах и статусе зачисления. Пользовательской ролью в системе является студент. Администрирование данных, изменение статусов и работа с документами осуществляются сотрудниками компании через CRM-систему Bitrix24 [27], с которой разрабатываемое приложение должно быть интегрировано.

Веб-приложение должно функционировать в среде веб-браузера и обеспечивать доступ к информации с различных устройств. Система должна быть адаптивной и поддерживать многоязычный интерфейс, включая русский, английский и арабский языки. Для корректного отображения арабского языка должна быть реализована поддержка направления текста справа налево (RTL).

В рамках поставленной задачи необходимо реализовать следующие функциональные возможности:

- регистрация нового пользователя с проверкой уникальности адреса электронной почты;
- авторизация пользователя с использованием безопасного механизма аутентификации;
- отправка письма подтверждения при регистрации;
- просмотр статуса зачисления;

- просмотр списка документов;
- просмотр загруженных документов;
- отображение контактной информации назначенного куратора.

При регистрации пользователя должна выполняться запись данных в локальную базу данных, а также автоматическое создание контакта в CRM-системе Bitrix24 посредством REST API [24]. В процессе дальнейшей работы приложения должна обеспечиваться синхронизация данных между системой и Bitrix24, включая контактные данные, перечень документов и поля профиля студента.

Документы студентов хранятся в системе Bitrix24, а веб-приложение выполняет функцию отображения актуальной информации, полученной через API.

К системе предъявляются следующие нефункциональные требования:

- обеспечение безопасности передачи данных посредством протокола HTTPS;
- реализация механизма аутентификации на основе JWT (JSON Web Token);
- защита от несанкционированного доступа;
- обеспечение корректной работы на различных устройствах (адаптивность интерфейса);
- поддержка многоязычности и RTL-отображения.

Таким образом, в результате выполнения работы должна быть разработана интегрированная информационная система, обеспечивающая централизованный доступ студентов к данным о процессе поступления и взаимодействующая с корпоративной CRM-платформой Bitrix24.

1.2 Анализ требований

Результаты анализа функциональных требований к системе представлены на рисунке 1.1 в виде диаграммы вариантов использования. Главными действующими лицами, взаимодействующим с системой, являются студент и сотрудник.

Студент выполняет 7 сценариев: регистрацию, авторизацию, просмотр профиля, просмотр и поиск документов, скачивание файлов и восстановление пароля.

Сотрудник RagEdu управляет данными через Bitrix24.

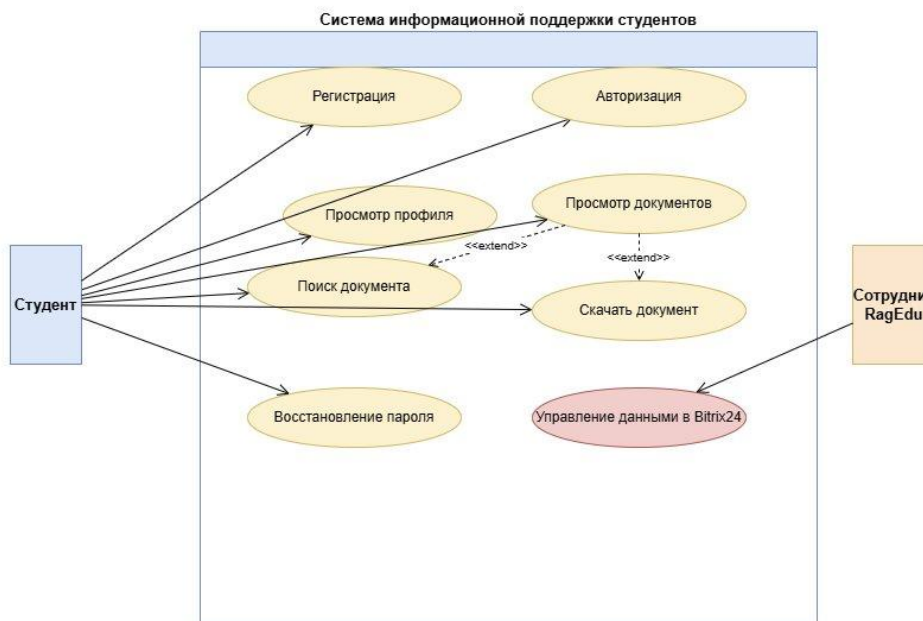


Рисунок 1.1 – Диаграмма вариантов использования системы

В ходе анализа предметной области и требований к разрабатываемой системе был составлен терминологический словарь, представленный в приложении А.

1.3 Выбор методов и средств разработки

Выбор методов и средств разработки опирался на анализ источников [1-31], посвященных современным технологиям программирования и проектирования.

Серверная часть приложения будет представлять собой многослойную монолитную архитектуру и будет реализована на языке PHP в среде программирования PHP Storm. Как СУБД была выбрана MySQL.

Клиентская часть будет написана на языке JavaScript с использованием HTML и CSS, а также фреймворка Vue.js в среде программирования WebStorm. Серверная и клиентская части будут взаимодействовать через REST API, что обеспечит надежность и производительность системы, а также позволит масштабировать ее при необходимости.

2 Проектирование системы

2.1 Проектирование архитектуры системы и программного обеспечения

Общая концепция разрабатываемой системы представлена на рисунке 2.1 в виде диаграммы коммуникаций. Данная диаграмма показывает основные функциональные блоки, из которых строится программная система, и взаимодействия между ними.

Диаграмма коммуникаций описывает обмен сообщениями между пятью узлами: браузером Vue.js, PHP API, MySQL, Bitrix24 и SMTP. Пронумерованные сообщения отражают полный цикл авторизации и регистрации.

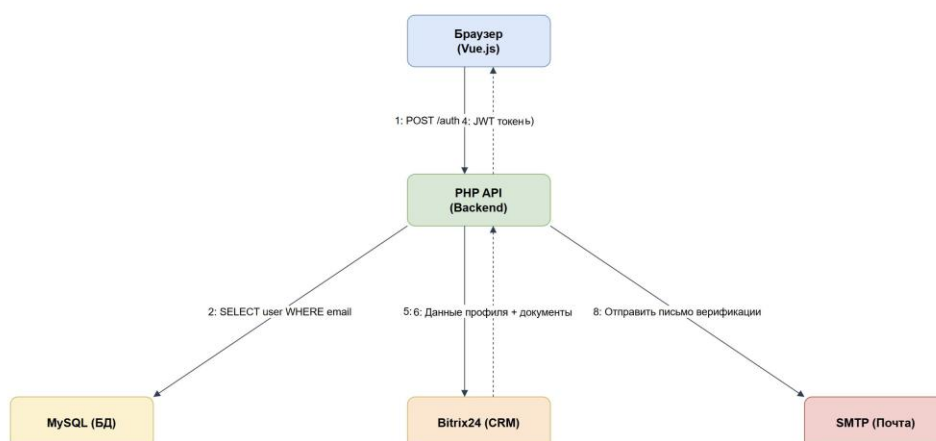


Рисунок 2.1 – Диаграмма коммуникаций системы

Для реализации функциональных возможностей разрабатываемой программной системы потребовалось разработать собственную библиотеку классов. Иерархия классов серверной части приведена на рисунке 2.2.

Диаграмма классов отображает структуру серверной части. Класс User содержит атрибуты БД и методы. Классы JWT, Database, BitrixService и MailService обеспечивают аутентификацию, работу с MySQL, интеграцию с Bitrix24 и отправку почты.

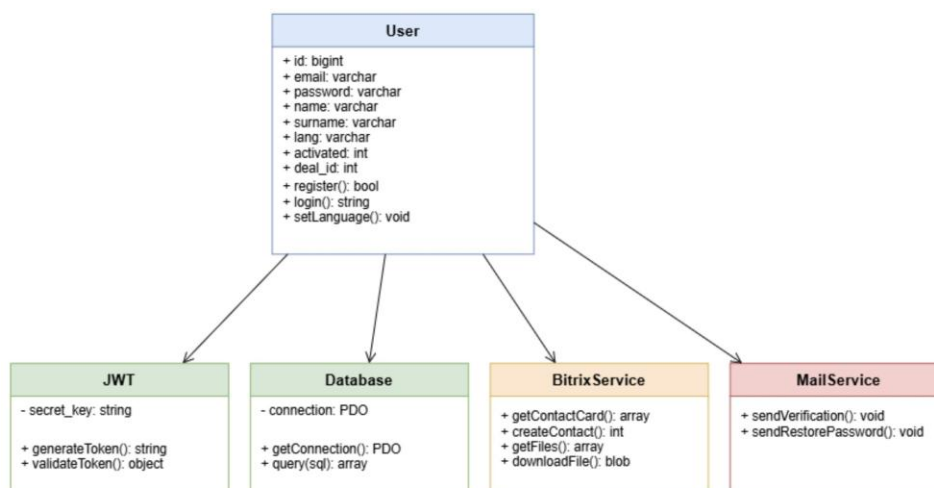


Рисунок 2.2 – Диаграмма классов серверной части

Одной из основных целей создания описываемой программной системы было обеспечение пользователя набором удобных средств, упрощающих решение его повседневных задач связанных с организацией обучения. Алгоритм работы пользователя с системой приведен на рисунке 2.3 в виде диаграммы деятельности.

Диаграмма деятельности отображает полный процесс взаимодействия студента с приложением: от открытия страницы до скачивания документа. Включает ветвление по авторизации, запрос Bitrix24, поиск и скачивание файлов.

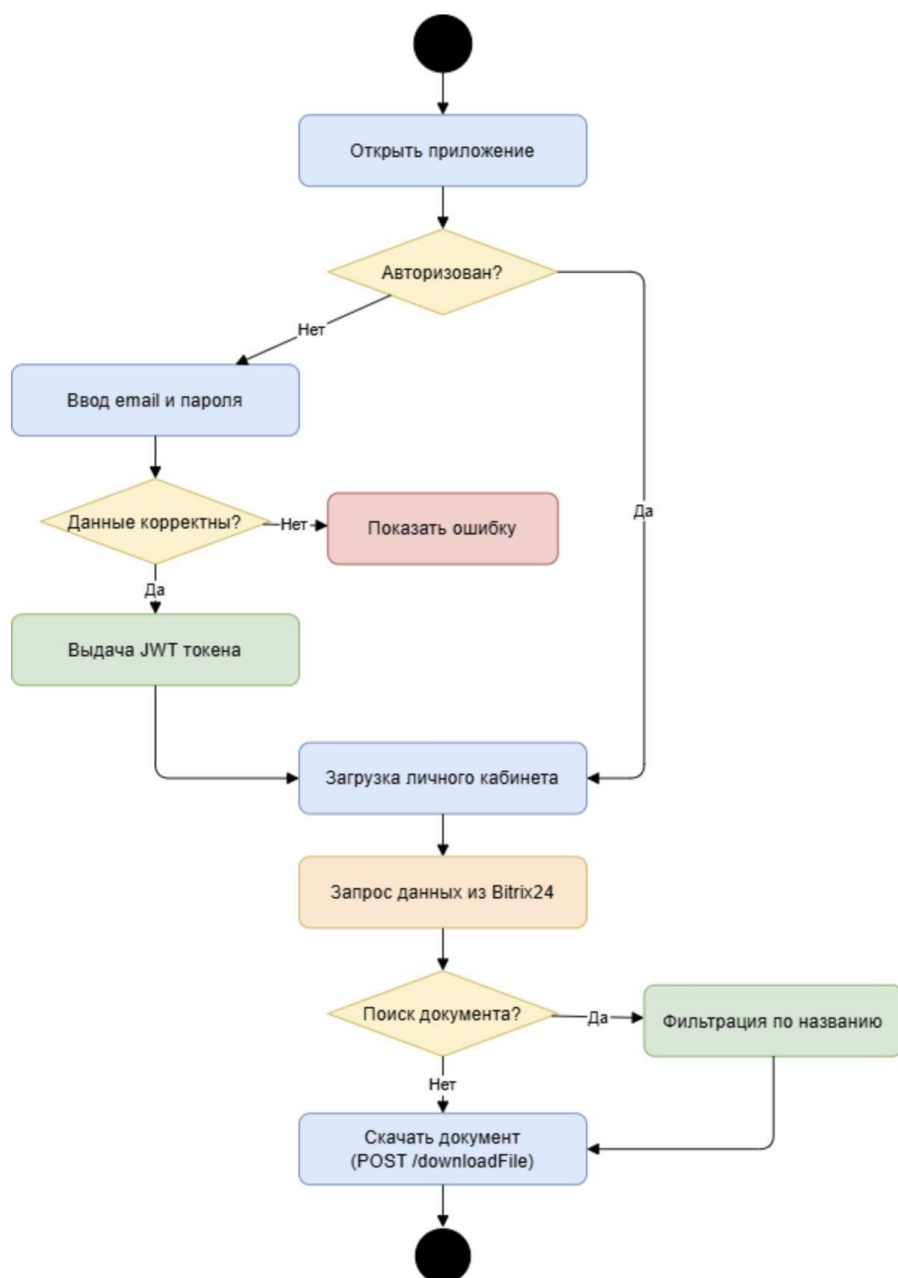


Рисунок 2.3 – Диаграмма деятельности пользователя

На рисунке 2.4 приведена диаграмма последовательности, которая описывает порядок взаимодействий при регистрации: 14 шагов от заполнения формы до перенаправления на авторизацию. Задействованы 6 участников: студент, Vue.js, PHP API, MySQL, Bitrix24 и SMTP.

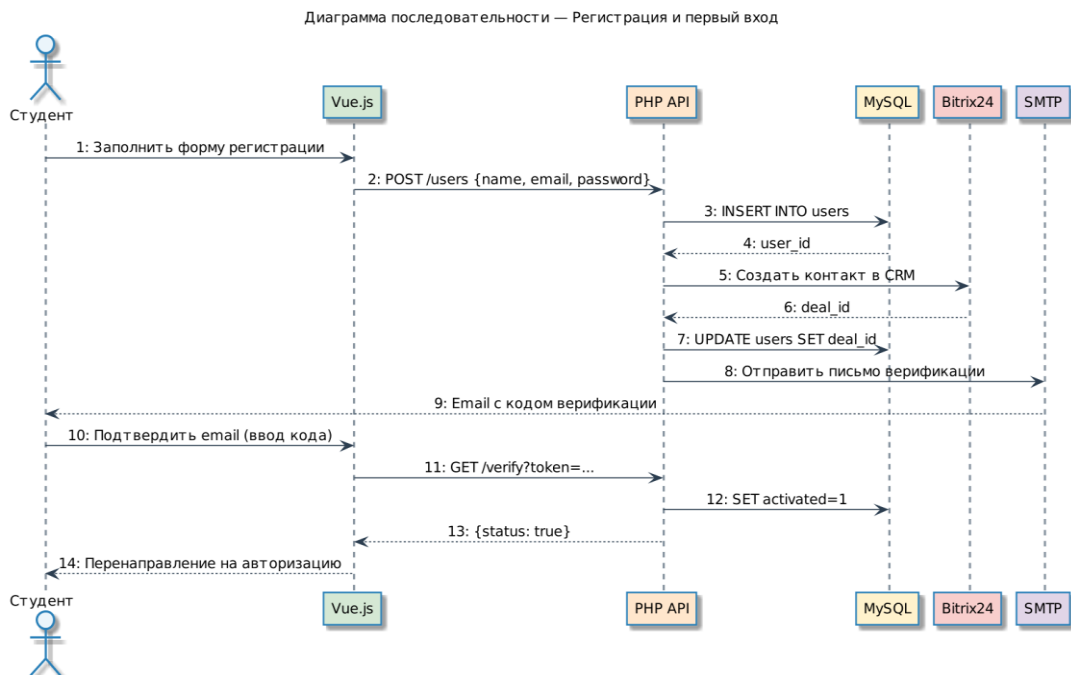


Рисунок 2.4 – Диаграмма последовательности регистрации

На рисунке 2.5 приведена диаграмма состояний, которая описывает жизненный цикл аккаунта: незарегистрирован, ожидание подтверждения, активен, авторизован и сброс пароля. Переходы между состояниями вызываются HTTP-запросами к API.

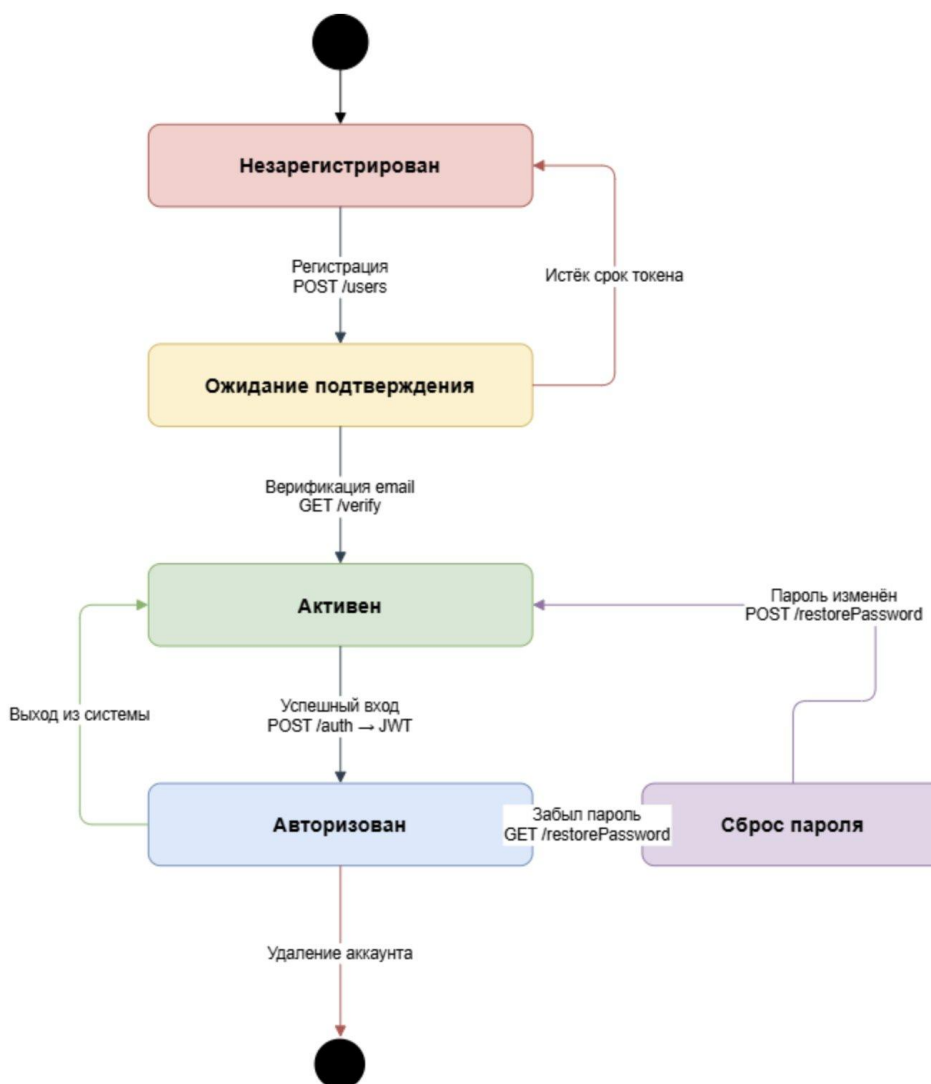


Рисунок 2.5 – Диаграмма состояний аккаунта

Для приложения будет использоваться монолитный многослойный архитектурный подход. Архитектура будет состоять из 4 слоев:

- UI (пользовательский интерфейс);
- бизнес-логика;
- средства взаимодействия с БД;
- собственно сама БД.

Каждый слой будет выполнять определенные задачи в рамках своей области. Они будут оставаться независимыми, что позволит изменениям одного слоя не влиять на работу остальных слоев. Физическое представление архитектуры показано на рисунке 2.6.

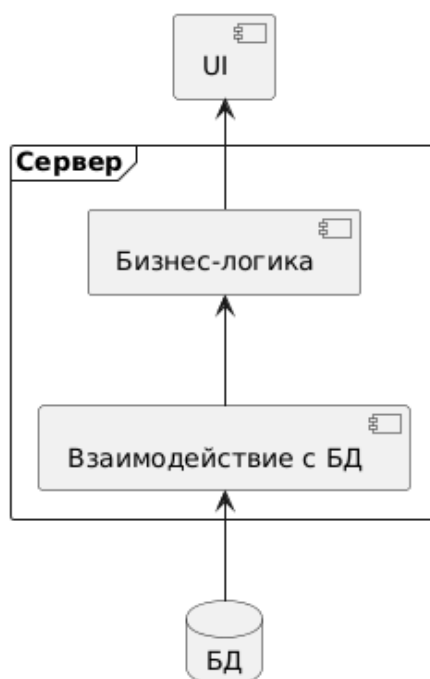


Рисунок 2.6 – Физическое представление архитектуры

Слой UI (пользовательский интерфейс) обеспечивает взаимодействие с пользователем. В нем обрабатываются клиентские запросы, предоставляются хранимые данные, а также передаются в дальнейший слой API. Слой содержит функции для авторизации и регистрации пользователей, а также для обработки запросов, связанных с получением данных.

Слой API обеспечивает выполнение процессов и различных операций операций с данными.

Внутри следующего слоя происходит взаимодействие с базой данных. Функции этого слоя инициализируют операции с базой данных.

Слой БД хранит все данные о пользователях. Это слой внешней базы данных.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

Проектирование базы данных [19-21] следует начать с построения концептуальной модели. Концептуальная модель базы данных не зависит от физической реализации. На стадии концептуального проектирования необходимо определить типы сущностей, определить связи между ними, выявить атрибуты каждой сущности.

Описание основных сущностей предметной области приложения для системы информационной поддержки иностранных студентов представлено в таблице 1.

Таблица 1 – Сущности предметной области

Название сущности	Идентификатор сущности	Описание сущности
Пользователи	users	Содержит информацию о пользователях приложения

Разработанная диаграмма концептуальной модели базы данных представлена на рисунке 2.7.

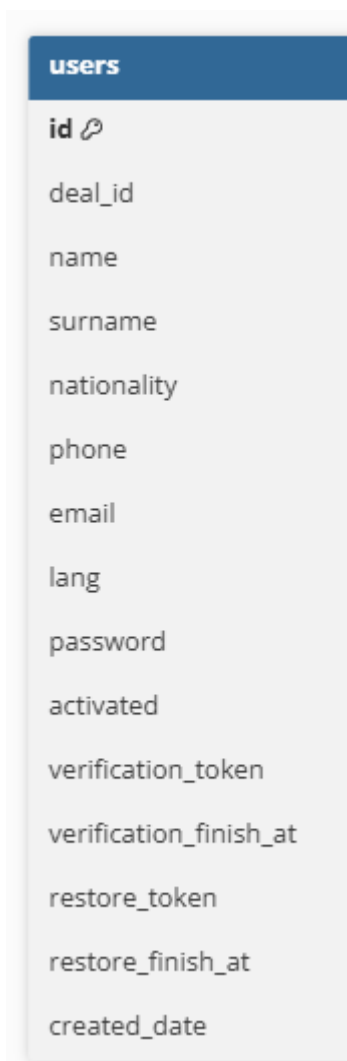


Рисунок 2.7 – Диаграмма концептуальной модели

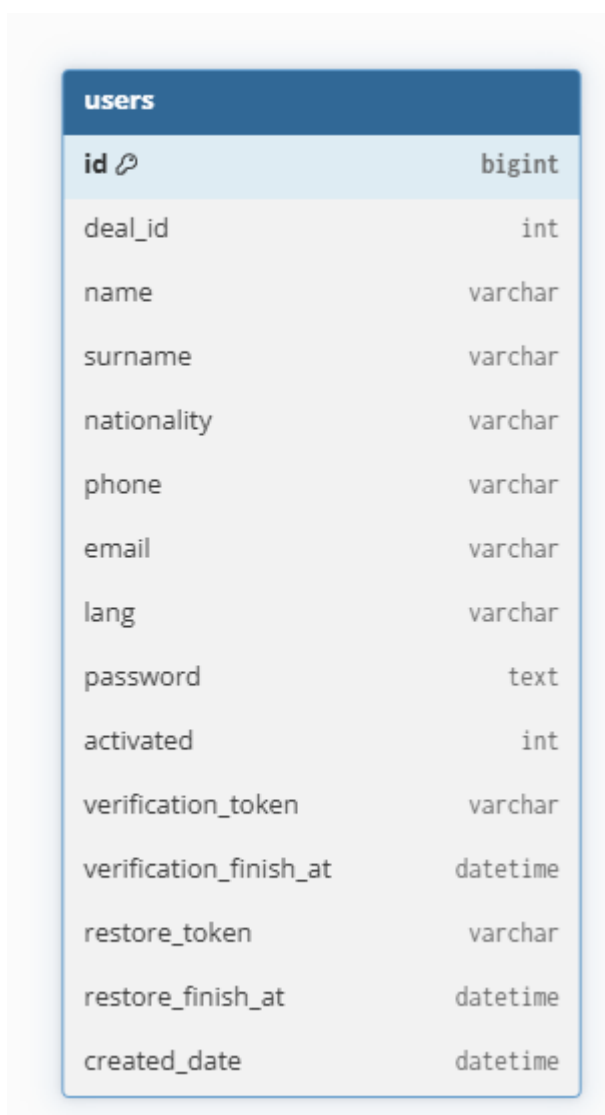
Данная концептуальная модель является основой для выполнения следующих этапов разработки: логического и физического проектирования.

2.2.2 Логическая модель базы данных

Логическая модель является развитием концептуальной модели базы данных. На этом этапе абстрактные атрибуты сущностей преобразуются в конкретные типы данных, а также добавляются ключевые элементы структуры: первичные и внешние ключи. Таким образом, логическая модель определяет что должно быть реализовано, служа непосредственной основой для создания физической модели.

Отношения между таблицами отсутствуют по причине наличия всего одной таблицы.

Логическая модель базы данных была построена на основе концептуальной модели. Логическая модель показана на рисунке 2.8.



users	
id	bigint
deal_id	int
name	varchar
surname	varchar
nationality	varchar
phone	varchar
email	varchar
lang	varchar
password	text
activated	int
verification_token	varchar
verification_finish_at	datetime
restore_token	varchar
restore_finish_at	datetime
created_date	datetime

Рисунок 2.8 – Логическая модель

Анализ функциональных связей между атрибутами данной таблицы позволяет выделить главный ключ отношения: атрибут `id`. Однако, помимо главного ключа, имеется еще ряд атрибутов, которые в совокупности могут выполнять роль ключа: `name`, `surname`, `nationality`, `phone`, `email`. Таким образом, в отношении присутствуют транзитивные зависимости, и оно находится в второй нормальной форме (2НФ) [19-21]. Однако, поскольку база данных состоит лишь из одной таблицы, дальнейшая нормализация может не производиться.

2.2.3 Физическая модель базы данных

Этап физического проектирования предполагает выбор типов данных для представления атрибутов с учетом возможностей СУБД, определение областей допустимых значений атрибутов. Результат проектирования представления таблицы базы данных на этом уровне приведен в приложении Б.

2.3 Проектирование интерфейса

Разрабатываемое приложение будет функционировать как одностраничное веб-приложение. В этом формате обновляется только та часть интерфейса, в которой происходят изменения, без полной перезагрузки страницы. Такой подход обеспечивает более высокую скорость отклика, плавность взаимодействия и снижает нагрузку на сеть. Благодаря этому пользователь получает более удобный, интуитивный и непрерывный опыт работы с приложением.

Интерфейс приложения [3] будет разделён на несколько ключевых страниц: регистрацию с подтверждением почты, авторизацию, страницу личного кабинета, а также страницу сброса пароля и установки нового. Особое внимание уделяется ясности форм, понятному расположению элементов и единообразию визуальных паттернов, чтобы пользователь мог быстро ориентироваться и уверенно завершать каждое действие.

Так как мы проектируем мультиязычное приложение, при макетировании важно учитывать возможность переключения направления письма и чтения с привычного Left-to-Right (далее ltr) на Right-to-Left (далее rtl). Интерфейс должен корректно адаптироваться под оба варианта: элементы навигации, выравнивание текста, порядок блоков и иконок должны автоматически подстраиваться под выбранный язык. Это обеспечивает единообразный и

удобный опыт для пользователей из разных регионов, независимо от того, на каком языке они работают.

Структурная схема интерфейса приложения, его страниц и взаимосвязей между ними показана на рисунке 2.9.

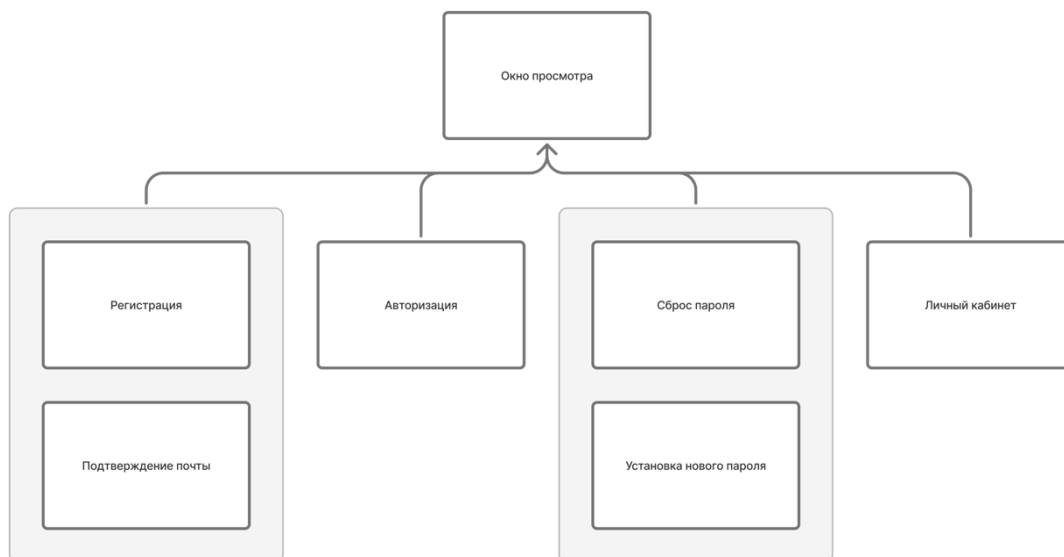


Рисунок 2.9 – Структурная схема интерфейса приложения

Страницы регистрации, подтверждения почты, авторизации и действий с паролем будут представлять собой формы, позволяющие пользователю выполнить необходимые шаги для доступа к системе. Каждая форма будет максимально простой и понятной, чтобы снизить вероятность ошибок и ускорить процесс взаимодействия.

Страница личного кабинета будет состоять из нескольких функциональных блоков:

- первый блок отображает информацию, связанную с местом обучения: университет, специальность и текущий статус обучающегося;
- второй блок представляет собой карточку пользователя с указанием имени и фамилии;
- третий блок содержит данные о кураторе: его имя, фамилию и доступные контактные данные;

– четвертый блок является ключевым элементом приложения – он включает в себя набор документов, которые могут понадобиться обучающемуся.

Чтобы упростить работу с потенциально большим списком файлов, будет реализован переключатель вида отображения: «список» или «плитка». Дополнительно документы будут визуально сгруппированы по форматам с помощью цветового различия иконок, что облегчит их восприятие и поиск. Также в этом блоке будет доступно поле поиска, позволяющее быстро найти документ по названию или формату. Это также упростит взаимодействие при большом количестве материалов, когда прокручивать весь список становится неэффективно.

3 Реализация приложения-личного кабинета для RagEdu

3.1 Общий состав разработанного приложения

Состав и структура разработанного приложения приведены на диаграмме компонентов, представленной на рисунке 3.1. На диаграмме отображены клиентская часть Vue.js 3 (Router, Pinia, i18n, AuthPages, CabinetPage) и серверная часть PHP (AuthController, UserController, FileController, JWT). Также представлены внешние сервисы: MySQL, Bitrix24 REST API, SMTP.

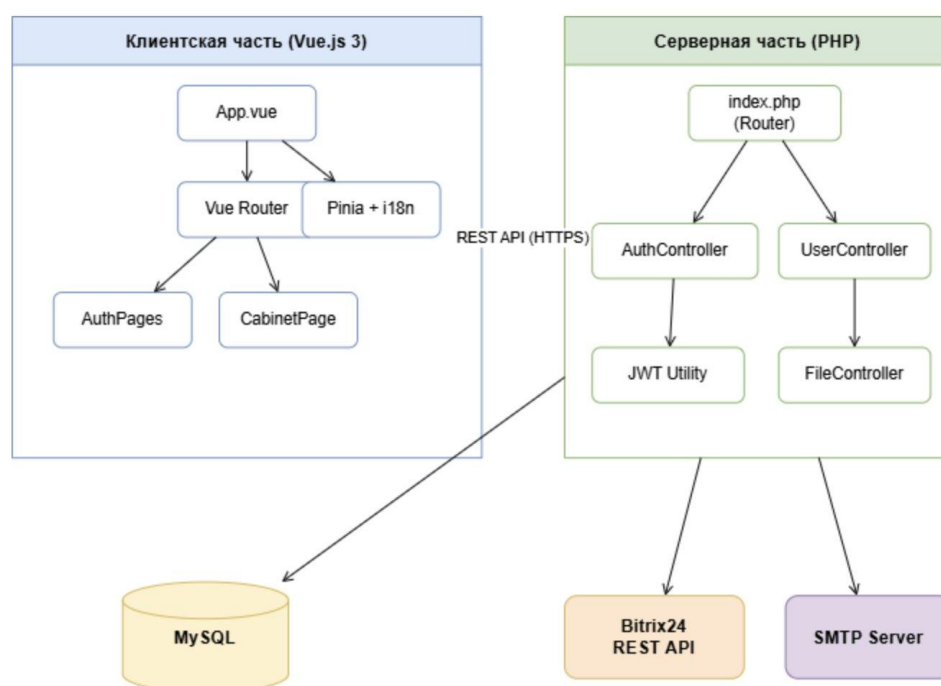


Рисунок 3.1 – Диаграмма компонентов системы

На рисунке 3.2 приведена диаграмма развёртывания, которая отображает физическое размещение компонентов приложения: веб-сервер Apache/Nginx поддерживает Vue.js SPA и PHP API, MySQL на отдельном сервере. При этом используются следующие протоколы: HTTPS, TCP 3306, HTTPS REST, SMTP 465.

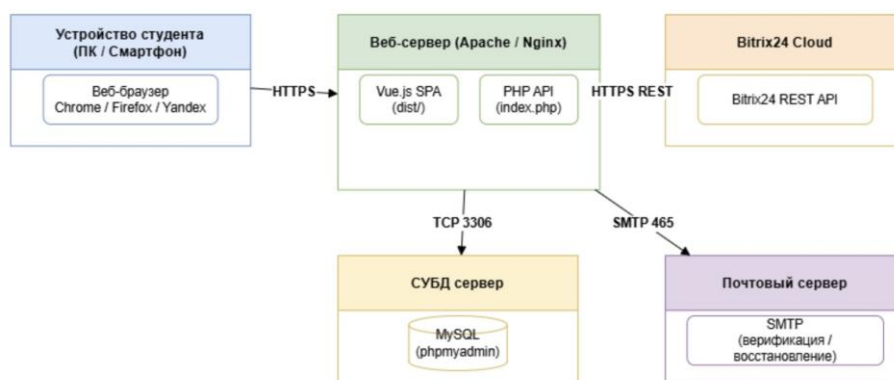


Рисунок 3.2 – Диаграмма развёртывания системы

Полные листинги разработанного программного обеспечения приведены в приложениях В и Г.

3.2 Реализация серверной части приложения

Серверная часть веб-приложения представляет собою API, получающее запросы от клиентской части. Получая запросы, API обрабатывает их и выполняет определенные действия, в зависимости от маршрута вызова.

Серверная часть приложения была реализована в среде программирования PHPStorm 2023.2.5. Код серверной части представлен в приложении В. Серверная часть содержит маршруты для обработки HTTP-запросов. Сами маршруты описаны в таблице 2.

Регистрация в системе подтверждается с помощью кода, который отправляется на почту пользователю. Для поддержания сессии используется JWT-токен, который позволяет API части понимать, кто делает запросы. Основная часть данных на сайте берётся из системы Bitrix24 из карточки пользователя.

Таблица 2 – Маршруты для обработки HTTP-запросов

URL-адрес	HTTP-метод	Описание
/users	GET	Получение данных о пользователе из базы
/bitrix	GET	Получение данных карточки пользователя в системе Bitrix24
/verify	GET	Проверка кода, отправленного на почту
/checkRestoreToken	GET	Проверка токена восстановления
/restorePassword	GET	Восстановление пароля (отправка письма)
/auth	POST	Авторизация
/users	POST	Регистрация
/repeatVerify	POST	Повторная отправка письма с кодом
/restorePssword	POST	Восстановление пароля по коду
/downloadFile	POST	Скачать файл, прикрепленный в Bitrix24
/setLang	PUT	Сменить язык по-умолчанию

Из таблицы 2 видно, что при реализации системы использовались основные HTTP-методы GET, POST и PUT, позволяющие

3.3 Реализация клиентской части приложения

Клиентская часть приложения была разработана в среде программирования JetBrains WebStorm с использованием фреймворка Vue.js 3 и языка JavaScript. В качестве CSS-препроцессора применялся SCSS, а для повышения удобства и безопасности разработки – TypeScript. Полный исходный код клиентской части приведён в приложении Г. Диаграмма страниц и компонентов клиентской части разрабатываемого приложения представлена на рисунке 3.3.

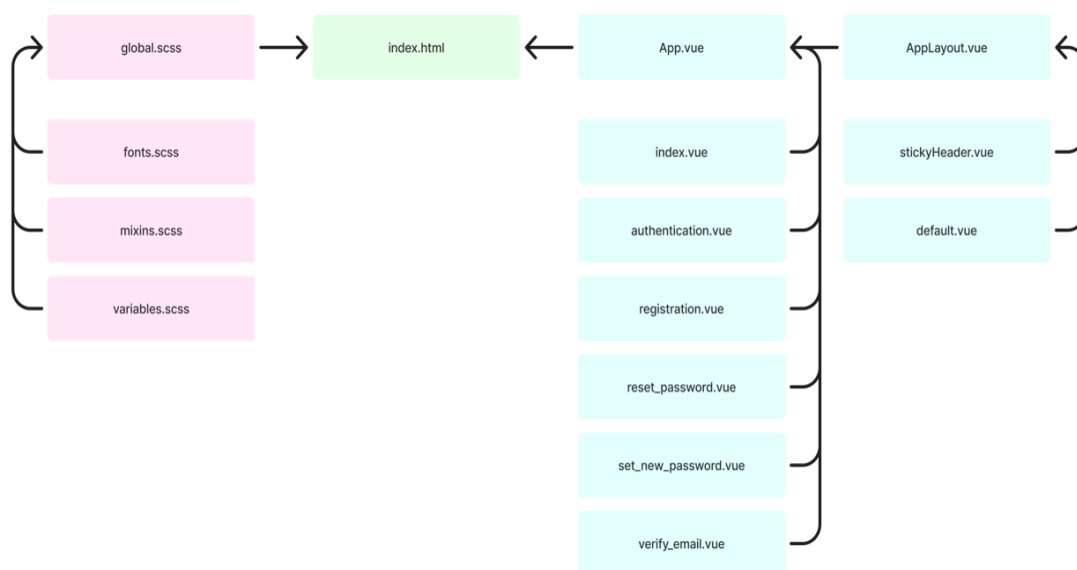


Рисунок 3.3 – Диаграмма страниц и компонентов клиентской части

Описание компонентов клиентской части содержится в таблице 3.

Таблица 3 – Описание компонентов клиентской части приложения

Компонент	Описание
index.html	Входная точка клиентского приложения
global.scss	Компонент, содержащий общие стили элементов интерфейса
fonts.scss	Компонент, содержащий подключение файлов шрифтов
mixins.scss	Компонент, реализующий стилевые функции переключения направления письма и чтения
variables.scss	Компонент, содержащий стилевые переменные
App.vue	Компонент, реализующий функционал одностраничного приложения
AppLayout.vue	Компонент, реализующий функционал автоматического переключения между каркасами страниц
stickyHeader.vue	Компонент, реализующий функционал каркаса страницы с фиксированной верхней панелью
default.vue	Компонент, реализующий функционал каркаса страницы с статической верхней панелью
index.vue	Компонент, реализующий страницу личного кабинета пользователя
authentication.vue	Компонент, реализующий страницу авторизации пользователя
registration.vue	Компонент, реализующий страницу регистрации пользователя
reset_password.vue	Компонент, реализующий страницу сброса пароля
set_new_password.vue	Компонент, реализующий страницу установки нового пароля
verify_email.vue	Компонент, реализующий страницу подтверждения почты пользователя

Для обмена данными между клиентской и серверной частью используется интерфейс Fetch API, который представляет собой актуальный и универсальный набор функций.

3.4 Реализация интерфейса приложения

В соответствии с разработанной ранее структурой интерфейса, было реализовано одностраничное web-приложение. При входе в систему пользователю доступны следующие разделы:

- а) раздел авторизации, включающий возможность сброса пароля и установку нового;
- б) раздел регистрации с обязательным подтверждением электронной почты.

После успешной авторизации пользователь получает доступ к разделу личного кабинета, содержащему основную функциональность приложения.

Для доступа к приложению пользователь должен выполнить вход в разделе авторизации, используя форму для ввода адреса электронной почты и пароля. Экранная форма окна авторизации представлена на рисунке 3.4.

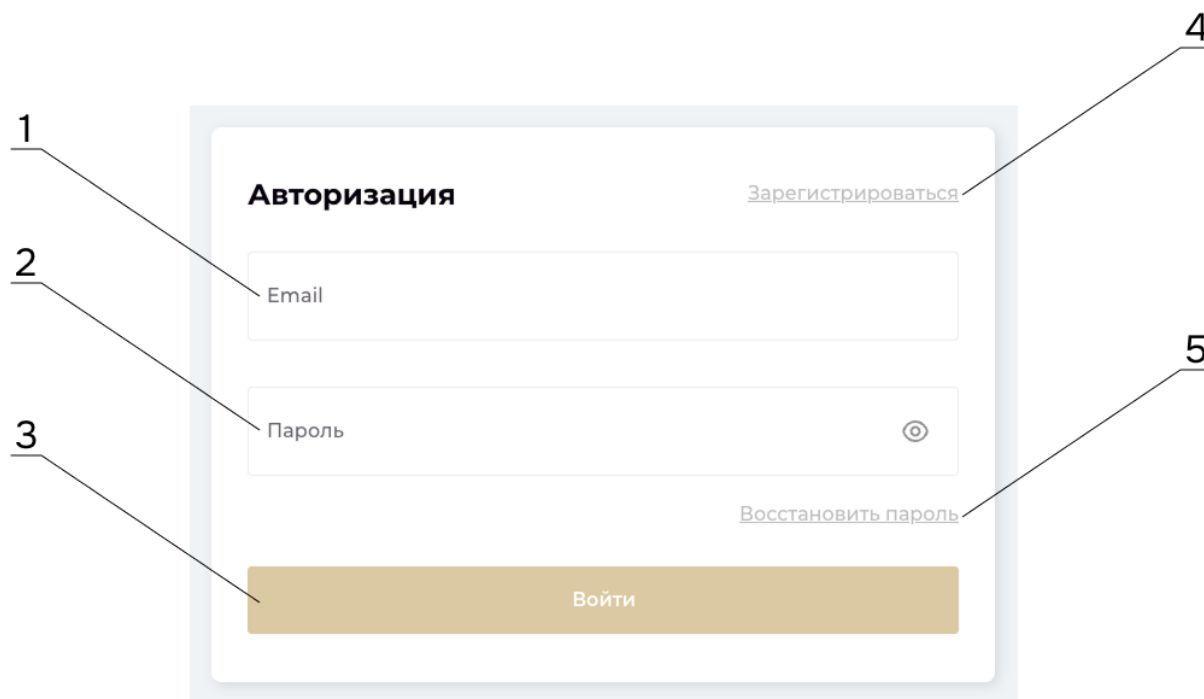


Рисунок 3.4 – Раздел авторизации

Раздел авторизации включает следующие элементы:

- 1 – поле для ввода адреса электронной почты;
- 2 – поле для ввода пароля;
- 3 – кнопка входа;
- 4 – ссылка для перехода к регистрации;
- 5 – ссылка для перехода к процедуре сброса пароля.

Когда пользователь нажимает кнопку входа, приложение начинает проверку введённых данных. Если указанный аккаунт существует, и информация введена корректно, приложение выдаёт пользователю токен доступа, после чего осуществляется вход в личный аккаунт.

Если пользователь ещё не зарегистрирован, он может перейти по ссылке для регистрации. После нажатия на неё на странице отображается главное окно раздела регистрации, представленное на рисунке 3.5.

Главное окно раздела регистрации включает следующие элементы:

- 1 – поле для ввода фамилии;
- 2 – поле для ввода имени;
- 3 – поле для ввода гражданства;
- 4 – поле для ввода адреса электронной почты;
- 5 – поле для ввода номера телефона;
- 6 – поле для ввода пароля;
- 7 – поле для повторного ввода пароля;
- 8 – флажок подтверждения согласия с политиками;
- 9 – кнопка регистрации;
- 10 – ссылка для перехода к авторизации.

После нажатия кнопки регистрации приложение проверяет корректность введённых данных и отсутствие уже существующего пользователя с такими параметрами. Если ошибки не обнаружены, пользователь успешно регистрируется и перенаправляется в раздел подтверждения электронной почты.

The image shows a registration form titled "Регистрация" (Registration). It includes a progress indicator with "Шаг 1" (Step 1) and "Шаг 2" (Step 2). The form contains several input fields: "Фамилия" (Surname), "Имя" (Name), "Гражданство" (Citizenship), "Email" (with an information icon), "Номер телефона" (Phone number), "Пароль" (Password, with eye and information icons), and "Повторите пароль" (Repeat password, with an eye icon). There is a checkbox for "Я согласен с политикой конфиденциальности" (I agree with the privacy policy) and a "Зарегистрироваться" (Register) button. A link "Авторизоваться" (Log in) is located at the top right. Numbered callouts point to various elements: 1 points to the title, 2 to the "Имя" field, 3 to the "Фамилия" field, 4 to the "Гражданство" field, 5 to the "Номер телефона" field, 6 to the "Email" field, 7 to the "Повторите пароль" field, 8 to the "Пароль" field, 9 to the checkbox, and 10 to the "Авторизоваться" link.

1

2

3

4

5

6

7

8

9

10

Регистрация

[Авторизоваться](#)

Шаг 1 Шаг 2

Фамилия

Имя

Гражданство

Email ⓘ

Номер телефона

Пароль ⓘ ⓘ

Повторите пароль ⓘ

☐ Я согласен с [политикой конфиденциальности](#)

Зарегистрироваться

Рисунок 3.5 – Главное окно раздела регистрации

Интерфейс раздела подтверждения электронной почты состоит из двух окон. Первое окно отображается сразу после успешной регистрации. Оно информирует пользователя о необходимости подтвердить адрес электронной почты, это окно представлено на рисунке 3.6.

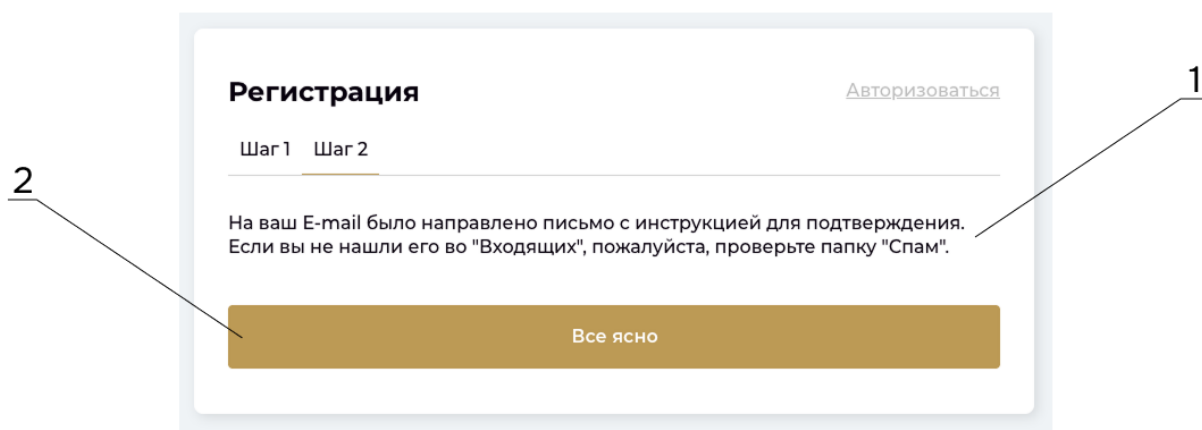


Рисунок 3.6 – Раздел подтверждения электронной почты, первое окно

Первое окно содержит следующие элементы:

- 1 – текстовое описание дальнейших действий;
- 2 – кнопка подтверждения.

После нажатия этой кнопки приложение перенаправляет пользователя на страницу авторизации.

Второе окно отображается после перехода по ссылке из письма и сообщает о статусе подтверждения электронной почты. Окно представлено на рисунке 3.7.

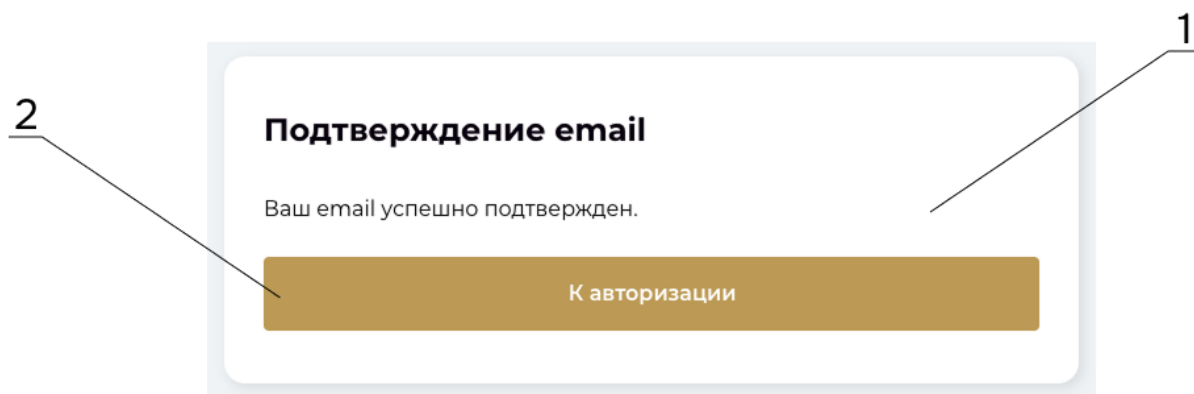


Рисунок 3.7– Раздел подтверждения электронной почты, второе окно

Структура второго окна подтверждения электронной почты включает в себя следующие элементы:

- 1 – текстовое описание результата подтверждения;
- 2 – кнопка для перехода к авторизации.

Если пользователь забыл пароль и ему необходимо восстановить доступ к аккаунту, для этого предусмотрен специальный раздел. Переход к разделу восстановления доступа осуществляется из раздела авторизации по ссылке, расположенной под формой входа (рисунок 3.4, элемент 5 раздела авторизации). После нажатия на эту ссылку пользователь попадает в раздел восстановления пароля. Подобно интерфейсу ранее описанного раздела, интерфейс раздела восстановления парольного доступа состоит из нескольких окон.

Первое окно отображается сразу после перехода к восстановлению пароля и представляет собой форму запроса, показанную на рисунке 3.8.

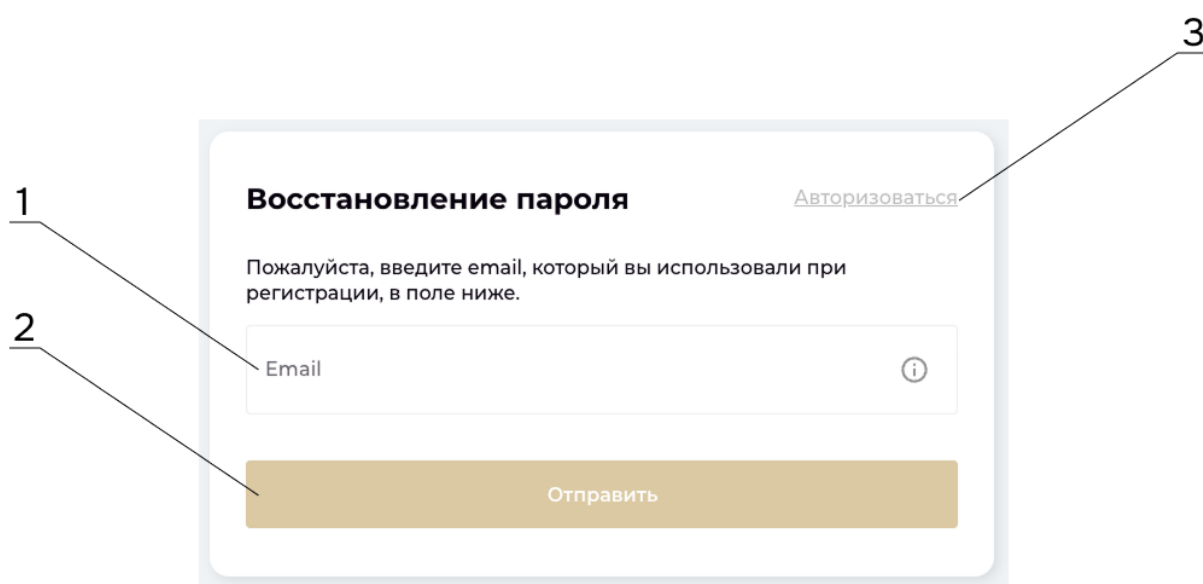


Рисунок 3.8 – Раздел восстановления пароля, первое окно

Структура первого окна раздела восстановления доступа включает в себя следующие элементы:

- 1 – поле для ввода адреса электронной почты;
- 2 – кнопка отправки данных для продолжения процесса;
- 3 – ссылка для перехода к авторизации.

Второе окно интерфейса раздела восстановления доступа отображается после отправки адреса электронной почты и используется для информирования пользователя о дальнейших шагах восстановления доступа. Это окно представлено на рисунке 3.9.

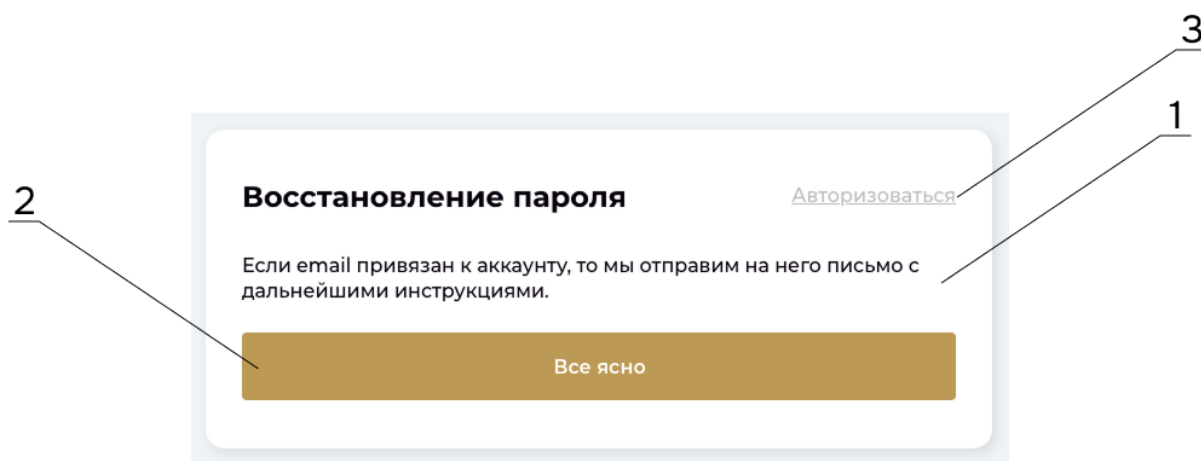


Рисунок 3.9 – Раздел восстановления пароля, второе окно

Структура второго окна раздела восстановления доступа включает в себя следующие элементы:

- 1 – текстовое описание дальнейших действий;
- 2 – кнопка подтверждения;
- 3 – ссылка для перехода к авторизации.

Следующее, третье окно становится доступным после перехода по ссылке из письма, предназначенного для сброса пароля. В этом окне пользователь может задать новый пароль для своего аккаунта. Структура третьего окна представлена на рисунке 3.10.

1

2

3

4

Восстановление пароля [Авторизоваться](#)

Шаг 1 Шаг 2

Пароль

Повторите пароль

Подтвердить

Рисунок 3.10 – Раздел восстановления пароля, третье окно

Структура третьего окна включает в себя следующие элементы:

- 1 – поле для ввода нового пароля;
- 2 – поле для повторного ввода нового пароля;
- 3 – кнопка подтверждения;
- 4 – ссылка для перехода к авторизации.

После нажатия кнопки подтверждения пользователь переходит к финальному окну интерфейса раздела восстановления пароля. Это окно информирует о завершении процесса восстановления пароля и содержит подтверждение успешного восстановления доступа. Финальное окно интерфейса раздела восстановления пароля представлено на рисунке 3.11.

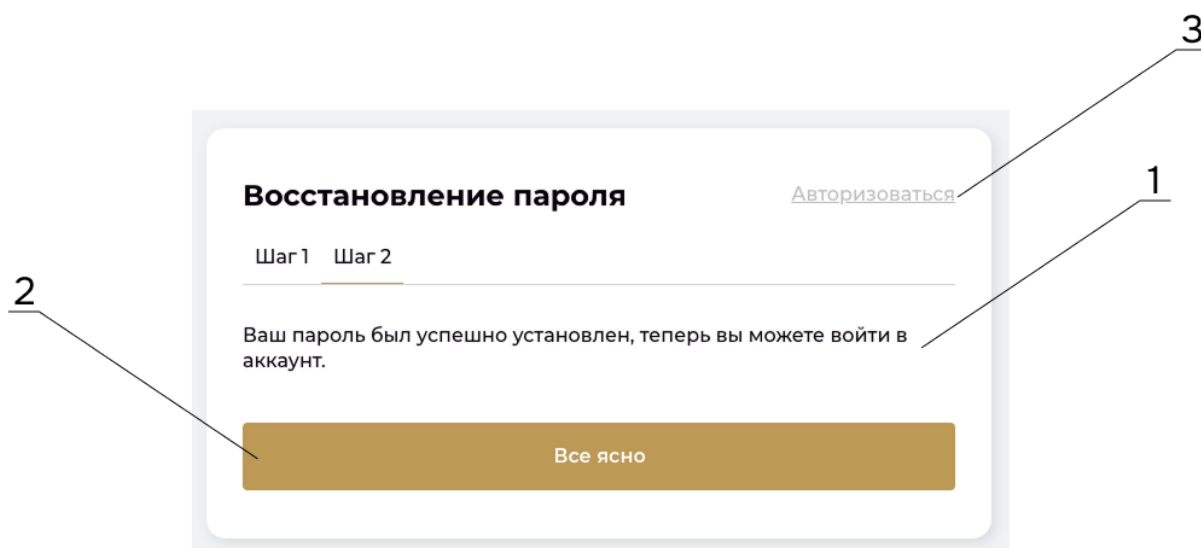


Рисунок 3.11 – Раздел восстановления пароля, финальное окно интерфейса раздела

Финальное окно интерфейса раздела восстановления пароля включает в себя следующие элементы:

- 1 – текстовое описание успешности восстановления пароля;
- 2 – кнопка подтверждения;
- 3 – ссылка для перехода к авторизации.

После завершения процесса регистрации, подтверждения электронной почты и авторизации, пользователь переходит в личный кабинет. Визуально для пользователя личный кабинет отображается как информационная панель, состоящая из четырёх блоков.

Первый блок содержит основные сведения об обучении пользователя – университет, специальность и текущий статус обучения. Визуальное представление данного блока приведено на рисунке 3.12.

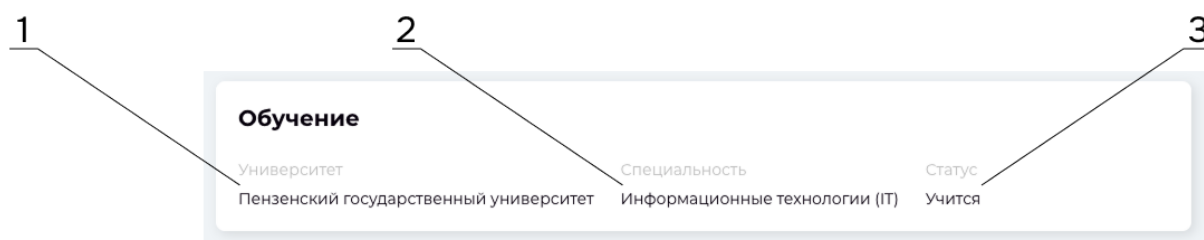


Рисунок 3.12 – Раздел личного кабинета, визуальное представление блока «Обучение»

Блок «Обучение» включает следующие элементы:

- 1 – сведения об университете;
- 2 – сведения о специальности;
- 3 – сведения о статусе обучения.

Следующий блок представляет собой карточку пользователя, содержащую его имя и фамилию. Визуальное представление этого блока приведено на рисунке 3.13.

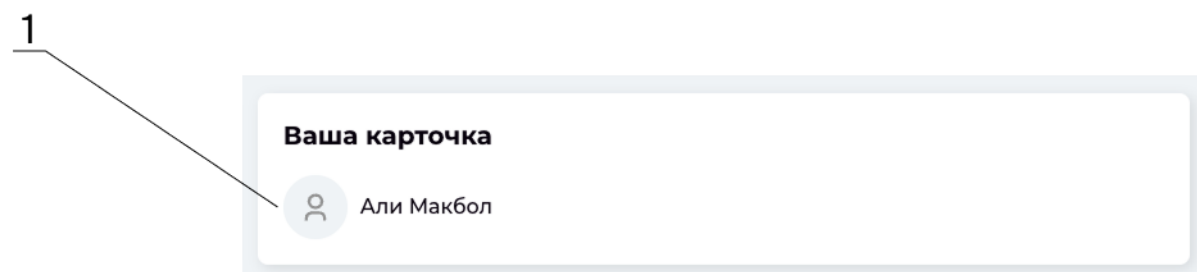


Рисунок 3.13 – Раздел личного кабинета «Карточка пользователя»

Блок «Карточка пользователя» включает в себя только один элемент:

- 1 – имя и фамилия пользователя.

Третий блок раздела личного кабинета представляет собой карточку поддержки, содержащую имя куратора, или наименование организации, и контактные данные. Визуальное представление этого блока приведено на рисунке 3.14.

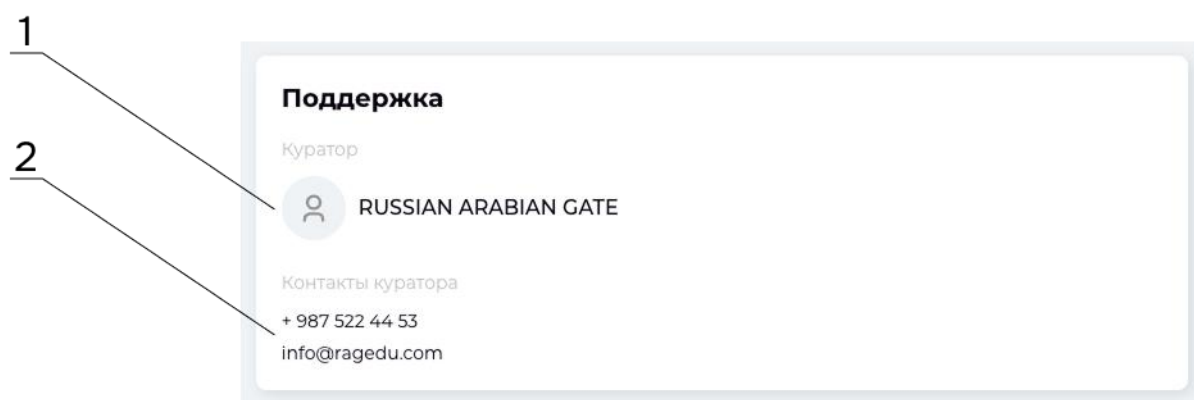


Рисунок 3.14 – Раздел личного кабинета «Карточка поддержки»

Блок «Карточка поддержки» включает следующие элементы:

- 1 – имя куратора или наименование организации;
- 2 – контактные данные, представленные номером телефона и адресом электронной почты.

Последний блок раздела личного кабинета представляет собой область для работы с документами. Он включает в себя инструменты поиска по названию и переключения режима отображения. На рисунке 3.15 представлен вариант отображения в виде «плитки», а на рисунке 3.16 – вариант отображения в виде списка.

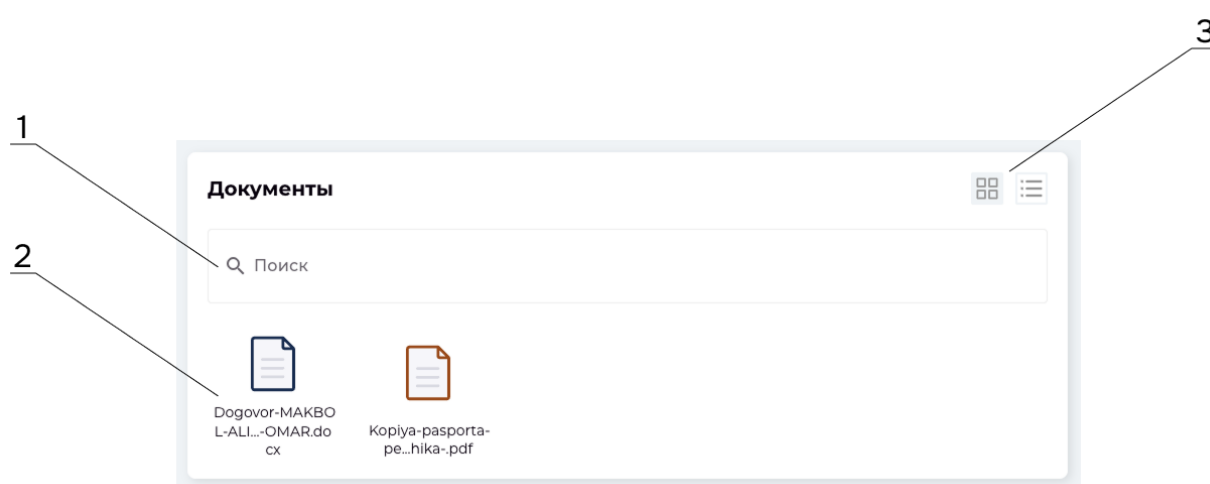


Рисунок 3.15 – Раздел личного кабинета «Документы», отображение в виде «ПЛИТКИ»

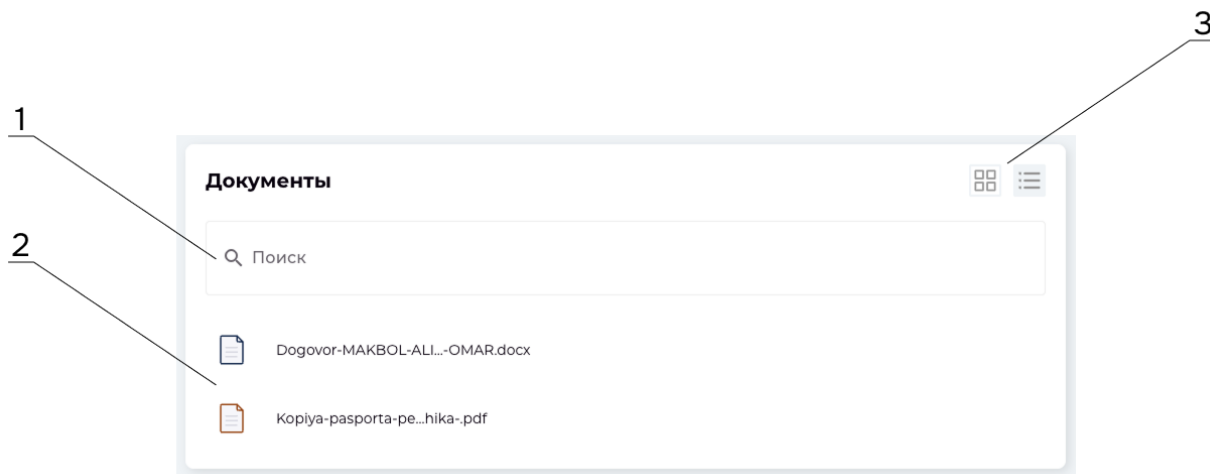


Рисунок 3.16 – Раздел личного кабинета «Документы», отображение в виде списка

Блок «Документы» в обоих вариантах отображения включает следующие элементы:

- 1 – поле поиска для ввода названия документа;
- 2 – область с документами, представление которых зависит от выбранного режима отображения и формата файла;
- 3 – переключатель режима отображения.

Кроме того, в приложении реализована возможность переключения языка интерфейса. Для этого используется библиотека **i18n**, представляющая собой инструментарий для работы с различными наборами текстов, их плюрализацией и удобным переключением между языковыми версиями.

В интерфейсе приложения эта функция реализована в виде выпадающего списка для выбора языка отображения. Данный элемент представлен на рисунке 3.17.

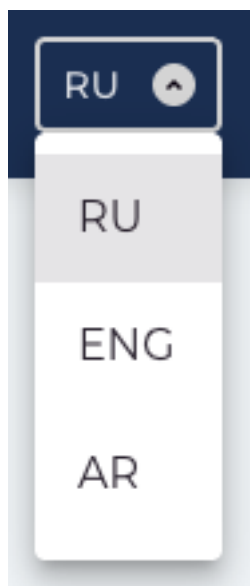


Рисунок 3.17 – Выпадающий список для выбора языка

В качестве примера приведены примеры отображения окна авторизации на английском и арабском языках, представленные на рисунках 3.18 и 3.19 соответственно.

Authorization [Register](#)

Email

Password 

[Reset password](#)

Log in

Рисунок 3.18 – Отображение окна авторизации на английском языке

The image shows a login form in Arabic. At the top right is the title 'تسجيل الدخول' (Login). At the top left is a link 'التسجيل' (Registration). Below the title is a text input field labeled 'البريد الإلكتروني' (Email). Below that is a password input field labeled 'كلمة المرور' (Password) with an eye icon for toggling visibility. Under the password field is a link 'إعادة تعيين كلمة المرور' (Reset password). At the bottom is a large orange button labeled 'تسجيل الدخول' (Login).

Рисунок 3.19 – Отображение окна авторизации на арабском языке

Все данные и функции для обращения к REST API в приложении хранятся на отдельном уровне данных (data-level). Он реализован с использованием библиотеки Pinia, которая предоставляет удобные инструменты для создания, разделения и использования специализированных хранилищ в структуре приложения.

4 Отладка и тестирование

4.1 Обоснование выбора режима тестирования

Задача тестирования заключается в анализе приложения для обнаружения ошибок и повышения качества программного продукта. Классификация видов тестирования помогает систематизировать подход, упростить планирование и разработку тест-кейсов. Среди множества направлений можно выделить наиболее распространенные [32-34]:

- функциональное тестирование;
- модульное тестирование;
- интеграционное тестирование;
- системное тестирование.

Функциональное тестирование позволяет оценить, насколько реализованные функциональные возможности соответствуют заданным требованиям. Оно может включать проверку пользовательских интерфейсов, клиент-серверного взаимодействия и общей работоспособности приложения.

Модульное тестирование направлено на проверку отдельных модулей или компонентов программы. Оно помогает оперативно выявлять и устранять ошибки при внесении изменений или добавлении новых функций, не требуя при этом полной проверки всего приложения.

Интеграционное тестирование, в свою очередь, оценивает корректность взаимодействия нескольких модулей или частей системы.

Системное тестирование представляет собой комплексную проверку готового программного продукта в целом на соответствие требованиям и обычно выполняется после завершения интеграционного этапа.

Для разработанного приложения было принято решение провести функциональное тестирование, которое позволит оценить работу приложения с точки зрения конечного пользователя и проверить соответствие заявленным требованиям.

4.2 Функциональное тестирование веб-приложения

В целях проведения функционального тестирования был выделен перечень тестируемых ситуаций, связанных с авторизацией, регистрацией, восстановлением пароля и работой пользователя в личном кабинете. Для проверки работоспособности программного обеспечения в этих ситуациях с точки зрения надежности, корректности решения поставленной задачи и устойчивости к ошибкам ввода данных, был разработан набор тестовых данных с учетом разбиения множества исходных данных на классы эквивалентности. Описание тестируемых ситуаций, тестовых данных и итогов каждого теста приведены в таблице 4.

Таблица 4 – Функциональное тестирование программного обеспечения:
тестовые данные и итоги прохождения тестов

Входные данные	Ожидаемый результат	Фактический результат	Статус
email: vvva password: 123654321qwet	Ошибка в поле: email	Ошибка в поле: email	Пройден
email: cats@mail.ru password: 123	Ошибка в поле: password	Ошибка в поле: password	Пройден
email: cats@mail.ru password: 12387653qwert	Валидация пройдена, код выполняется далее	Валидация пройдена, код выполняется далее	Пройден

Продолжение таблицы 4

Входные данные	Ожидаемый результат	Фактический результат	Статус
email: cats@ password: 12387653qwert	Ошибка в поле: email	Ошибка в поле: email	Пройден
email: (пусто) password: 12387653qwert	Ошибка: поле обязательно для заполнения	Ошибка: поле обязательно для заполнения	Пройден
email: student@gmail.co m password: Secure123qwert	Регистрация выполнена, письмо отправлено	Регистрация выполнена, письмо отправлено	Пройден
email: student@gmail.co m password: abc	Ошибка в поле: password	Ошибка в поле: password	Пройден
email: cats@mail.ru (восстановление пароля)	Письмо для сброса пароля отправлено	Письмо для сброса пароля отправлено	Пройден
email: (пусто) (восстановление пароля)	Ошибка: поле обязательно для заполнения	Ошибка: поле обязательно для заполнения	Пройден

Продолжение таблицы 4

Входные данные	Ожидаемый результат	Фактический результат	Статус
Корректный JWT токен	Данные профиля загружены из Bitrix24	Данные профиля загружены из Bitrix24	Пройден
Ввод названия документа в поле поиска	Отображаются соответствующие документы	Отображаются соответствующие документы	Пройден
Выбор языка: Arabic	Интерфейс переключается на арабский язык	Интерфейс переключается на арабский язык	Пройден
Выбор языка: English	Интерфейс переключается на английский язык	Интерфейс переключается на английский язык	Пройден
Нажатие кнопки скачать POST /downloadFile	Файл скачивается в браузер	Файл скачивается в браузер	Пройден
Переключение режима: плитки / список	Вид документов меняется	Вид документов меняется	Пройден

Итоги функционального тестирования позволяют говорить о том, что программное обеспечение корректно и устойчиво работает на тестовых данных, решая поставленную перед ним задачу.

Результаты функционального тестирования приведены в приложении Д, в приложении Е содержатся модульные тесты, использовавшиеся при проверке работоспособности отдельных модулей.

5 Внедрение и опыт эксплуатации программного обеспечения

Разработанное программное обеспечение было внедрено в Институте международного сотрудничества Пензенского государственного университета для работы с иностранными (преимущественно арабоязычными) студентами. Копия акта внедрения приведена в приложении Ж.

Опыт эксплуатации разработки позволяет говорить о росте количества пользователей системы по годам, что свидетельствует о работоспособности программного обеспечения, его надежности и удобстве. Сведения о росте количества пользователей с 2023 по 2025 годы приведены в таблице 5.

Таблица 5 – Рост количества пользователей приложения по годам эксплуатации

Год	Количество пользователей приложения
2023	55
2024	110
2025	300

Устойчивый рост количества пользователей с течением времени эксплуатации свидетельствует о востребованности приложения как с точки зрения обеспечиваемых им функциональных возможностей, так и с точки зрения удобства и надежности эксплуатации.

Таким образом, приложение обладает практической ценностью.

6 Расчет бюджета разработки. Планирование разработки и оценка бюджета

Для оценки бюджета разработки, были назначены ресурсы и стоимость их использования. Ресурсы и оценка их стоимости показаны на рисунке 6.1.

Услуги	Описание работ	Специалист	Кол-во часов	Цена, руб.	Сумма, руб.	Срок
РАБОТЫ (часы)						
Дизайн	- Прототипирование - Разработка адаптивного дизайн-макета в Figma следующих страниц: Страница авторизации, страница личного кабинета	middle UI / UX Дизайнер	50	р.1 000,00	р.50 000,00	7 дней
Аналитика	- аналитика предоставленных материалов - аналитика структуры сайта - формирование ТЗ для front-end разработчика - формирование ТЗ для back-end разработчика	Аналитик	25	р.1 000,00	р.25 000,00	4 дня
Менеджмент проекта	Ведение проекта - контроль процесса разработки - составление отчетности	Project-manager	27	р.600,00	р.16 200,00	весь период разработки
Верстка сайта	- верстка десктопной версии сайта в соответствии с утвержденным дизайном - верстка мобильной версии сайта в соответствии с утвержденным дизайном - верстка планшет версии сайта в соответствии с утвержденным дизайном - программная настройка - тестирование и отладка	middle front-end разработчик	76	р.1 500,00	р.114 000,00	23 дня
		junior front-end разработчик	10	р.800,00	р.8 000,00	
		Тестирующий	35	р.500,00	р.17 500,00	
Настройка программно-административной части	- обвязка back-end и верстки - посадка на БУС - обвязка ЛК с Б24 - тестирование и отладка	middle back-end разработчик	45	р.2 000,00	р.90 000,00	7 дней
Доп. Услуги:	Лицензия БИТРИКС				р. 17 900,00	
ИТОГО работы / проект					р.338 600,00	

Рисунок 6.1 – Данные о ресурсах и оценка бюджета

По результатам оценки бюджета, общая сумма денежных затрат на разработку составит 338600 рублей.

Заключение

В ходе выполнения выпускной квалификационной работы было разработано и реализовано приложение-личный кабинет, предназначенное для информирования студентов RagEdu.

Было проведено планирование и оценка бюджета разработки приложения. В результате выполненных расчетов предполагаемые расходы на реализацию приложения составили 338600 рублей, а общая продолжительность всех этапов разработки составила 41 день.

В результате проведения этапа проектирования были разработаны концептуальная и логическая2.2.2 Логическая модель базы данных модели базы данных, определена архитектура приложения, выделены классы серверной части и проанализированы требования к интерфейсу.

Разработка системы информационной поддержки иностранных студентов включила реализацию базы данных в MySQL, создание серверного приложения в монолитном архитектурном стиле с использованием PHP в среде PHP Storm, а также разработку клиентского приложения в среде WebStorm.

Этап отладки и тестирования приложения включил проведение функционального тестирования. Результаты тестирования подтвердили корректную работу приложения и его соответствие функциональным требованиям.

В настоящее время приложение используется в Институте международного сотрудничества Пензенского государственного университета для организации работы с иностранными студентами, имеется акт внедрения. При этом наблюдается устойчивый рост количества пользователей по годам эксплуатации. Это свидетельствует не только о работоспособности и качестве разработки, но и о ее практической ценности и востребованности.

Таким образом, главная цель выпускной квалификационной работы – разработка системы информационной поддержки иностранных студентов – была успешно достигнута, и все поставленные задачи выполнены. Разработанное приложение демонстрирует высокий уровень качества и полностью соответствует предъявленным требованиям.

Список использованных источников

1. Босов А. А. Веб-приложения: архитектура и проектирование пользовательских интерфейсов. – М.: ИНФРА-М, 2022. – 284 с.
2. Фаулер М. Архитектура корпоративных программных приложений. – СПб.: Питер, 2021. – 544 с.
3. Браун Э., Харви С. Проектирование интерфейсов. Паттерны проектирования пользовательских интерфейсов. – СПб.: Питер, 2020. – 368 с.
4. Разработка одностраничных веб-приложений / М. С. Миковски, Дж. К. Пауэлл ; пер. с англ. А. А. Слинкина. – 2-е изд., электронное. – Москва : ДМК Пресс, 2023. – 514 с.
5. Официальная документация Vue.js 3 [Электронный ресурс]. URL: <https://vuejs.org/> (дата обращения: 29.12.2025).
6. Pinia - состояние приложения для Vue.js [Электронный ресурс]. URL: <https://pinia.vuejs.org/> (дата обращения: 29.12.2025).
7. Vue I18n – библиотека локализации для Vue.js [Электронный ресурс]. URL: <https://kazupon.github.io/vue-i18n/> (дата обращения: 29.12.2025).
8. Fetch API – интерфейсы веб API // MDN Web Docs. [Электронный ресурс]. URL: https://developer.mozilla.org/ru/docs/Web/API/Fetch_API (дата обращения: 29.12.2025).
9. RESTful Web Services / L. Richardson, S. Ruby. – O'Reilly Media, 2007. – 454 p.
10. Поляков А. Веб-программирование на JavaScript. – СПб.: БХВ-Петербург, 2022. – 480 с.
11. HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт. – М.: Эксмо, 2021. – 480 с.
12. ECMAScript® 2023 Language Specification [Электронный ресурс]. URL: <https://tc39.es/ecma262/> (дата обращения: 29.12.2025).
13. SCSS – официальная документация [Электронный ресурс]. URL: <https://sass-lang.com/> (дата обращения: 29.12.2025).
14. JetBrains WebStorm – среда разработки веб-приложений [Электронный ресурс]. URL: <https://www.jetbrains.com/webstorm/> (дата обращения: 29.12.2025).
15. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. – University of California, 2000
16. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. – М.: Вильямс, 2020. – 720 с.

17. Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы. – СПб.: Питер, 2023. – 448 с.
18. Поляков А. Проектирование и разработка веб-приложений. – СПб.: Питер, 2022. – 352 с.
19. Дейт К. Введение в системы баз данных. – 8-е изд. – М.: Вильямс, 2020. – 1328 с.
20. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. – М.: Вильямс, 2003. – 1440 с.
21. Теория и практика построения баз данных / Д. Крёнке. – СПб.: Питер, 2003. – 800 с.
22. RESTful Web Services / L. Richardson, S. Ruby. – O'Reilly Media, 2007.
23. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. – University of California, 2000
24. Экскурс в REST API // Systems Education. [Электронный ресурс]. URL: <https://systems.education/rest-api-postman> (дата обращения: 29.12.2025).
25. JSON Web Tokens – официальная спецификация RFC 7519. [Электронный ресурс]. URL: <https://www.rfc-editor.org/rfc/rfc7519> (дата обращения: 29.12.2025).
26. JWT – Introduction // jwt.io. [Электронный ресурс]. URL: <https://jwt.io/introduction> (дата обращения: 29.12.2025).
27. Документация Bitrix24 REST API [Электронный ресурс]. URL: https://training.bitrix24.com/rest_help/ (дата обращения: 29.12.2025).
28. PHP. Официальная документация языка программирования PHP. [Электронный ресурс]. URL: <https://www.php.net/manual/ru/> (дата обращения: 29.12.2025).
29. Руководство по работе с PDO и MySQL в PHP. [Электронный ресурс]. URL: <https://www.php.net/manual/ru/book.pdo.php> (дата обращения: 29.12.2025).
30. OWASP Top 10 – основные уязвимости веб-приложений. [Электронный ресурс]. URL: <https://owasp.org/www-project-top-ten/> (дата обращения: 29.12.2025).
31. RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. [Электронный ресурс]. URL: <https://www.rfc-editor.org/rfc/rfc7231> (дата обращения: 29.12.2025).
32. Postman – инструмент тестирования REST API. [Электронный ресурс]. URL: <https://www.postman.com/> (дата обращения: 29.12.2025).

33. Функциональное тестирование: основы, виды и применение // Prog Academy. [Электронный ресурс].URL: <https://prog.academy/ru/blog/functional-testing> (дата обращения: 29.12.2025).

34. Что такое системное тестирование и как его проводить? // Хабр. [Электронный ресурс].URL: <https://habr.com/ru/articles/807813/> (дата обращения: 29.12.2025).

Приложение А
(справочное)
Глоссарий

БД (База данных) – это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

СУБД (Система управления базами данных) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

REST (Representational State Transfer) – это архитектурная концепция, определяющая набор ограничений для построения масштабируемых распределенных систем. Основными принципами REST является ресурсно-ориентированная архитектура, использование стандартных HTTP-методов для манипуляции ресурсами, передача данных без сохранения состояния на сервере, унифицированный интерфейс доступа.

VUE – это прогрессивный JavaScript-фреймворк с открытым исходным кодом, предназначенный для создания пользовательских интерфейсов и одностраничных веб-приложений

Приложение Б
(обязательное)
Атрибуты сущностей базы **данных**

Таблица Б.1 – Атрибуты сущности «Пользователь»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений
id	Уникальный идентификатор пользователя	bigint		0-9
deal_id	ID сделки в битрикс системе	int		0-9
name	Имя	varchar	255	
surname	Фамилия	varchar	255	
nationality	Национальность	varchar	255	
phone	Номер телефона	varchar	255	
email	Почта	varchar	255	a-z A-z 0-9 @
lang	Язык	varchar	255	
password	Пароль	varchar	255	a-z A-z 0-9 . _ - ! ? @ * +
activated	Активирован ли профиль	int		0-1

Продолжение таблицы Б.1

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений
varification_token	Токен верификации, отправленный на почту	varchar	255	
verification_finish_at	Дата окончания действия кода	datetime		
restore_token	Токен восстановления	varchar	255	
restore_finish_at	Дата истечения токена восстановления	datetime		
created_date	Дата создания пользователя	datetime		

Приложение В
(обязательное)
Листинг серверной части приложения

Листинг В.1 – Файл index.php

```

<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Methods: HEAD, GET, POST, PUT, PATCH, DELETE, OPTIONS");
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With, Content-Type,
Accept, Access-Control-Request-Method,Access-Control-Request-Headers, Authorization");
header('Content-Type: application/json');
$method = $_SERVER['REQUEST_METHOD'];
if ($method == "OPTIONS") {
    header('Access-Control-Allow-Origin: *');
    header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With, Content-Type,
Accept, Access-Control-Request-Method,Access-Control-Request-Headers, Authorization");
    header("HTTP/1.1 200 OK");
    die();
}
ini_set('display_errors', 1);

try {
    date_default_timezone_set('UTC');

    require '../vendor/autoload.php';
    require_once('../bitrix_api/crest.php');
    require_once 'utils/Config.php';
    require_once 'utils/Database.php';
    require_once 'utils/JWT.php';
    require_once 'utils/Utils.php';

    # запросы
    require_once 'utils/queries/user_queries.php';
    require_once 'utils/queries/validation_queries.php';

    # основные функции
    require_once 'functions/user.php';

    require_once 'utils/validation/Utils.php';
    require_once 'utils/validation/validation.php';

    $database = new Database();
    $db = $database->getConnection();

    $method = $_SERVER['REQUEST_METHOD'];

    $q = $_GET['q'];
    $params = explode('/', $q);

    $type = $params[0];

```

```

if (needCheckToken($type, $method)) {
    $headers = apache_request_headers();
    if ((isset($headers['authorization']) or (isset($headers['Authorization'])))) {

        if (isset($headers['authorization'])) {
            $authHeader = explode(" ", $headers['authorization']);
        } elseif (isset($headers['Authorization'])) {
            $authHeader = explode(" ", $headers['Authorization']);
        }

        $token = $authHeader[1];

        if (!empty($token)) {
            $jwt = new JWT(Config::$jwt_secret_key);
            $auth_data = $jwt->validateToken($token);

            if (is_null($auth_data)) {
                http_response_code(403);
                echo json_encode(array("status" => false, "error" => "token incorrect"));
                die;
            } else {
                $auth_user_id = $auth_data->user_id;
                if (!existUser($db, $auth_user_id)) {
                    http_response_code(404);
                    echo json_encode(array("status" => false, "error" => "Пользователя не
существует"));
                    die;
                }
            }
        } else {
            http_response_code(403);
            echo json_encode(array("status" => false, "error" => "token is empty"));
            die;
        }
    } else {
        http_response_code(403);
        echo json_encode(array("status" => false, "error" => "token not found"));
        die;
    }
} else {
    $auth_user_id = null;
}

if (isset($params[1])) {
    $id = $params[1];
    $id = intval($id);
}

if ($method != "GET") {
    $data = file_get_contents('php://input');
}

```

```

if ($data) {
    $data = json_decode($data, true);
    /*if ($type !== "webhook" and $method !== "POST") {
        $data = array_map('trim', $data);
    }*/
}
} else {
    $data = $_GET;
}

switch ($method) {
    case "GET":
        switch ($type) {
            case "users":
                echo getUser($db, $auth_user_id);
                break;

            case "bitrix":
                echo getBitrix($db, $auth_user_id);
                break;

            case "verify":
                echo verifyUser($db, $data);
                break;

            case "checkRestoreToken":
                echo checkUserRestoreToken($db, $data);
                break;

            case "restorePassword":
                echo sendMailForRestorePassword($db, $data);
                break;

            case "retryEmailTest":
                retryEmailTest();
                break;

            default: http_response_code(400); echo json_encode(array("status" => false, "error" =>
"incorrect type")); break;
        }
        break;

    case "POST":
        switch ($type) {
            case "auth":
                $jwt = new JWT(Config::$jwt_secret_key);
                echo auth($db, $jwt, $data);
                break;

```

```

        case "users":
            echo addUser($db, $data);
            break;

        case "repeatVerify":
            echo repeatVerifyUser($db, $auth_user_id);
            break;

        case "restorePassword":
            echo updatePasswordRestore($db, $data);
            break;

        case "downloadFile":
            downloadBitrixFile($data);
            break;

        default: http_response_code(400); echo json_encode(array("status" => false, "error" =>
"incorrect type")); break;
    }
    break;

    case "DELETE":
        http_response_code(400);
        echo json_encode(array("status" => false, "error" => "incorrect type"));
        break;
    case "PUT":
        switch ($type) {
            case "setLang":
                echo updateUserLang($db, $auth_user_id, $data);
                break;
            default:
                http_response_code(400);
                echo json_encode(array("status" => false, "error" => "incorrect type"));
                break;
        }
        break;
        default: http_response_code(400); echo json_encode(array("status" => false, "error" =>
"incorrect method")); break;
    }

} catch (PDOException $e) {
    http_response_code(500);
    echo json_encode(array("status" => false, "error" => "database: {$e->getMessage()} line: {$e-
>getLine()} | file: {$e->getFile()}"));
    sendErrorTg( "database: {$e->getMessage()}\nline: {$e->getLine()}\nfile: {$e->getFile()}");
} catch (Exception $e) {
    http_response_code(500);
    echo json_encode(array("status" => false, "error" => "server: {$e->getMessage()} line: {$e-
>getLine()} | file: {$e->getFile()}"));
}

```

```

    sendErrorTg( "server: {$e->getMessage()}\nline: {$e->getLine()}\nfile: {$e->getFile()}");
}

```

Листинг В.2 Файл user.php

```
<?php
```

```

function auth($db, $jwt, $data): string
{
    $validation = validatePostAuth($data);
    if ($validation !== true) {
        http_response_code(400);
        return json_encode(array("status" => false, "error" => $validation, "code" =>
"validation_error"));
    }

    $email = $data['email'];
    $password = $data['password'];
    $user_id = checkUserAuth($db, $email, $password);
    if (!is_null($user_id)) {
        if (!checkUserActivated($db, $user_id)) {
            http_response_code(403);
            return json_encode(array("status" => false, "error" => "Профиль не активирован", "code"
=> "no_activated"));
        }
        $data = [
            'user_id' => $user_id,
        ];
        $token = $jwt->generateToken($data);
        http_response_code(200);
        return json_encode(array("status" => true, "data" => array("token" => $token, 'user_id' =>
$user_id)));
    } else {
        http_response_code(404);
        return json_encode(array("status" => false, "error" => "Пользователь не найден", "code" =>
"not_found"));
    }
}

function getUser($db, $auth_user_id): string
{
    $id = $auth_user_id;
    $have_user = existUser($db, $id);
    if ($have_user) {
        $user = getUserData($db, $id);

        http_response_code(200);
        return json_encode(array("status" => true, "data" => $user));
    }
}

```

```

    } else {
        http_response_code(404);
        return json_encode(array("status" => false, "error" => "Пользователь не найден", "code" =>
"not_found"));
    }
}

function addUser($db, $data): string
{
    $validation = validatePostUsers($data);
    if ($validation !== true) {
        http_response_code(400);
        return json_encode(array("status" => false, "error" => $validation, "code" =>
"validation_error"));
    }

    if (checkUserEmailExist($db, $data['email'])) {
        http_response_code(409);
        return json_encode(array("status" => false, "error" => "Такая почта уже существует в
системе", "code" => "email_exist"));
    }

    $verification_token = genRandomString();
    $data['verification_token'] = $verification_token;
    $data['password'] = md5($data['password']);
    $last_insert_id = insertUserData($db, $data);

    sendEmail($data['email'], "{$data['name']}, click on the link to activate your account and complete
registration: <a target='_blank'
href='https://cabinet.ragedu.com/verify_email?token={$verification_token}'>https://cabinet.ragedu.c
om/verify_email?token={$verification_token}</a>", $data['name'], "Email confirmation");
    addLog("[Зарегистрирован новый пользователь]" . " | new_user_id = $last_insert_id | " .
PHP_EOL . print_r($data, true), "main.log", $_SERVER['DOCUMENT_ROOT'] .
Config::$log_functions_path);

    http_response_code(201);
    return json_encode(array("status" => true, "data" => array("id" => $last_insert_id)));
}

function verifyUser($db, $data): string
{
    $validation = validateGetVerify($data);
    if ($validation !== true) {
        http_response_code(400);
        return json_encode(array("status" => false, "error" => $validation, "code" =>
"validation_error"));
    }

    if (!checkUserVerificationTokenExist($db, $data['token'])) {

```

```

    http_response_code(403);
    return json_encode(array("status" => false, "error" => "Токен не существует или истёк срок
действия", "code" => "token_expired_or_no_exist"));
}

$user_data = getUserDataByVerificationToken($db, $data['token']);

$contact_id = bitrixCreateContact($user_data);

if (is_null($contact_id)) {
    http_response_code(400);
    return json_encode(array("status" => false, "error" => "Ошибка создания контакта", "code"
=> "bitrix_error"));
}

$deal_id = bitrixCreateDeal($user_data, $contact_id);

if (is_null($deal_id)) {
    http_response_code(400);
    return json_encode(array("status" => false, "error" => "Ошибка создания сделки", "code" =>
"bitrix_error"));
}

updateUserDealIdData($db, $data['token'], $deal_id);

updateUserVerifyData($db, $data);

$verify_user = getUserEmailByVerificationToken($db, $data['token']);
if (!is_bool($verify_user)) {
    deleteUsersByEmailNotActivated($db, $verify_user['email']);
}

http_response_code(200);
return json_encode(array("status" => true, "data" => null));
}

function repeatVerifyUser($db, $auth_user_id): string
{
    if (!canUserVerificationRepeat($db, $auth_user_id)) {
        http_response_code(403);
        return json_encode(array("status" => false, "error" => "Отправить ссылку на активацию
повторно можно не ранее, чем через 15 минут после последней попытки", "code" =>
"wait_timeout"));
    }
    $verification_token = genRandomString();
    $user = getUserData($db, $auth_user_id);
    $email = $user['email'];
    $name = $user['name'];
    updateUserVerifyRepeatData($db, $auth_user_id, $verification_token);
}

```

```

    sendEmail($email, "{$name}, click on the link to activate your account and complete registration:
    <a target='_blank'
    href='https://cabinet.ragedu.com/verify_email?token={$verification_token}'>https://cabinet.ragedu.c
    om/verify_email?token={$verification_token}</a>", $name, "Email confirmation");

```

```

    http_response_code(200);
    return json_encode(array("status" => true, "data" => null));
}

```

```

function sendMailForRestorePassword($db, $data): string

```

```

{
    $validation = validateGetRestorePassword($data);
    if ($validation !== true) {
        http_response_code(400);
        return json_encode(array("status" => false, "error" => $validation, "code" =>
"validation_error"));
    }
}

```

```

    $name = getDataForRestorePassword($db, $data['email']);

```

```

    if (!$name) {
        http_response_code(403);
        return json_encode(array("status" => false, "error" => "Почты не существует в системе,
либо вы уже отправляли недавно запрос. Отправить запрос на восстановление пароля
повторно можно не ранее, чем через 15 минут после последней попытки.", "code" =>
"wait_timeout_or_no_exist_email"));
        #http_response_code(200);
        #return json_encode(array("status" => true, "data" => null));
    }
}

```

```

    $restore_token = genRandomString();
    $data['restore_token'] = $restore_token;

```

```

    updateRestoreToken($db, $data);

```

```

    sendEmail($data['email'], "{$name}, click on the link to recover your password: <a target='_blank'
    href='https://cabinet.ragedu.com/set_new_password?token={$restore_token}'>https://cabinet.rage
    du.com/set_new_password?token={$restore_token}</a>", $name, "Password recovery");

```

```

    http_response_code(200);
    return json_encode(array("status" => true, "data" => null));
}

```

```

function updatePasswordRestore($db, $data): string

```

```

{
    $validation = validatePostRestorePassword($data);
    if ($validation !== true) {
        http_response_code(400);
    }
}

```

```

        return json_encode(array("status" => false, "error" => $validation, "code" =>
"validation_error"));
    }

    $result = updatePasswordRestoreData($db, $data);

    if (!$result) {
        http_response_code(403);
        return json_encode(array("status" => false, "error" => "Недействительная ссылка.
Возможно, она устарела. Время жизни ссылки: 15 минут.", "code" =>
"link_expired_or_no_exist"));
    }

    http_response_code(200);
    return json_encode(array("status" => true, "data" => null));
}

function checkUserRestoreToken($db, $data): string
{
    $validation = validateGetCheckRestoreToken($data);
    if ($validation !== true) {
        http_response_code(400);
        return json_encode(array("status" => false, "error" => $validation, "code" =>
"validation_error"));
    }

    if (!checkUserRestoreTokenExistData($db, $data['restore_token'])) {
        http_response_code(403);
        return json_encode(array("status" => false, "error" => "Токен не существует или истёк срок
действия", "code" => "token_expired_or_no_exist"));
    }

    http_response_code(200);
    return json_encode(array("status" => true, "data" => null));
}

function updateUserLang($db, $id, $data): string
{
    $validation = validatePostSetLang($data);
    if ($validation !== true) {
        http_response_code(400);
        return json_encode(array("status" => false, "error" => $validation, "code" =>
"validation_error"));
    }

    updateUserLangData($db, $id, $data);

    http_response_code(200);
    return json_encode(array("status" => true, "data" => null));
}

```

```

}

function getBitrix($db, $auth_user_id): string
{
    $id = $auth_user_id;
    $have_user = existUser($db, $id);
    if ($have_user) {
        $deal_id = getUserDealIdData($db, $id);

        $info = bitrixGetInfo($deal_id["deal_id"]);

        http_response_code(200);
        return json_encode(array("status" => true, "data" => $info));
    } else {
        http_response_code(404);
        return json_encode(array("status" => false, "error" => "Пользователь не найден", "code" =>
"not_found"));
    }
}

```

Листинг В.3 Файл user_queries.php

```

<?php

function getUserData($db, $id)
{
    $stmt = $db->prepare("SELECT u.id, u.name, u.surname, u.nationality, u.email, u.phone, u.lang,
u.activated, u.create_date
                        FROM users as u
                        WHERE u.id = :id
                        ");
    $stmt->bindParam(":id", $id);
    $stmt->execute();

    return $stmt->fetch(PDO::FETCH_ASSOC);
}

function checkUserActivated($db, $id): bool
{
    $stmt = $db->prepare("SELECT activated FROM users WHERE id = :id");
    $stmt->bindParam(":id", $id);
    $stmt->execute();

    return $stmt->fetchColumn() == 1;
}

function existUser($db, $id): bool

```

```

{
    $stmt_count = $db->prepare("SELECT id FROM users WHERE id = :id");
    $stmt_count->bindParam(":id", $id);
    $stmt_count->execute();

    if ($stmt_count->rowCount() > 0) {
        return true;
    } else {
        return false;
    }
}

```

```

function checkUserAuth($db, $email, $password): ?int
{
    $password = md5($password);
    $stmt = $db->prepare("SELECT id FROM users WHERE email = :email AND password = :password AND activated = 1");
    $stmt->bindParam(":email", $email);
    $stmt->bindParam(":password", $password);

    if ($stmt->execute()) {
        $user = $stmt->fetch(PDO::FETCH_ASSOC);

        if ($stmt->rowCount() > 0) {
            return $user['id'];
        } else {
            return null;
        }
    }
    return null;
}

```

```

function checkUserEmailExist($db, $email): bool
{
    $stmt = $db->prepare("SELECT * FROM users WHERE email = :email and activated = 1");
    $stmt->bindParam(":email", $email);

    if ($stmt->execute()) {
        if ($stmt->rowCount() > 0) {
            return true;
        } else {
            return false;
        }
    }
    return true;
}

```

```

function checkUserVerificationTokenExist($db, $verification_token): bool

```

```

{
    $stmt = $db->prepare("SELECT id FROM users WHERE verification_token = :verification_token
AND verification_finish_at > NOW() AND activated = 0");
    $stmt->bindParam(":verification_token", $verification_token);

    if ($stmt->execute()) {
        if ($stmt->rowCount() > 0) {
            return true;
        } else {
            return false;
        }
    }
    return true;
}

```

```

function getUserDataByVerificationToken($db, $verification_token)
{
    $stmt = $db->prepare("SELECT * FROM users WHERE verification_token = :verification_token
AND verification_finish_at > NOW() AND activated = 0");
    $stmt->bindParam(":verification_token", $verification_token);

    if ($stmt->execute()) {
        if ($stmt->rowCount() > 0) {
            $user = $stmt->fetch(PDO::FETCH_ASSOC);
            return $user;
        } else {
            return false;
        }
    }
    return false;
}

```

```

function canUserVerificationRepeat($db, $auth_user_id): bool
{
    $stmt = $db->prepare("SELECT id FROM users WHERE id = :user_id AND activated = 0 AND
(verification_finish_at is null OR verification_finish_at < NOW())");
    $stmt->bindParam(":user_id", $auth_user_id);

    if ($stmt->execute()) {
        if ($stmt->rowCount() > 0) {
            return true;
        } else {
            return false;
        }
    }
    return true;
}

```

```

function insertUserData($db, $data): int

```

```

{
    $query = "INSERT INTO users
        SET
            name = :name,
            surname = :surname,
            nationality = :nationality,
            phone = :phone,
            email = :email,
            password = :password,
            lang = :lang,
            verification_token = :verification_token,
            verification_finish_at = NOW() + INTERVAL 1 YEAR
    ";

    $stmt = $db->prepare($query);

    $stmt->bindParam(":name", $data['name']);
    $stmt->bindParam(":surname", $data['surname']);
    $stmt->bindParam(":nationality", $data['nationality']);
    $stmt->bindParam(":phone", $data['phone']);
    $stmt->bindParam(":email", $data['email']);
    $stmt->bindParam(":password", $data['password']);
    $stmt->bindParam(":lang", $data['lang']);
    $stmt->bindParam(":verification_token", $data['verification_token']);

    $stmt->execute();

    return $db->lastInsertId();
}

function updateUserVerifyData($db, $data): void
{
    $query = "UPDATE users
        SET
            activated = 1,
            verification_token = null,
            verification_finish_at = null
        WHERE verification_token = :verification_token AND activated = 0
    ";

    $stmt = $db->prepare($query);

    $stmt->bindParam(":verification_token", $data['token']);

    $stmt->execute();
}

function updateUserVerifyRepeatData($db, $auth_user_id, $verification_token): void
{

```

```

$query = "UPDATE users
        SET
            verification_token = :verification_token,
            verification_finish_at = NOW() + INTERVAL 15 MINUTE
        WHERE id = :user_id AND activated = 0
        ";

$stmt = $db->prepare($query);

$stmt->bindParam(":user_id", $auth_user_id);
$stmt->bindParam(":verification_token", $verification_token);

$stmt->execute();
}

function getDataForRestorePassword($db, $email)
{
    $stmt = $db->prepare("SELECT name FROM users WHERE email = :email AND activated = 1
    AND (restore_finish_at is null OR restore_finish_at < NOW())");
    $stmt->bindParam(":email", $email);

    if ($stmt->execute()) {
        if ($stmt->rowCount() > 0) {
            return $stmt->fetchColumn();
        } else {
            return false;
        }
    }
    return false;
}

function updateRestoreToken($db, $data): void
{
    $query = "UPDATE users
        SET
            restore_token = :restore_token,
            restore_finish_at = NOW() + INTERVAL 15 MINUTE
        WHERE email = :email AND activated = 1
        ";

    $stmt = $db->prepare($query);

    $stmt->bindParam(":email", $data['email']);
    $stmt->bindParam(":restore_token", $data['restore_token']);

    $stmt->execute();
}

function updatePasswordRestoreData($db, $data): bool

```

```

{
    $stmt = $db->prepare("SELECT id FROM users WHERE restore_token = :restore_token AND
restore_finish_at > NOW()");
    $stmt->bindParam(":restore_token", $data['restore_token']);

    if ($stmt->execute()) {
        if ($stmt->rowCount() > 0) {
            $password = md5($data['password']);

            $query = "UPDATE users
SET
    password = :password,
    restore_finish_at = null,
    restore_token = null
WHERE restore_token = :restore_token
";

            $stmt = $db->prepare($query);

            $stmt->bindParam(":password", $password);
            $stmt->bindParam(":restore_token", $data['restore_token']);

            $stmt->execute();

            return true;
        } else {
            return false;
        }
    }

    return true;
}

function checkUserRestoreTokenExistData($db, $restore_token): bool
{
    $stmt = $db->prepare("SELECT id FROM users WHERE restore_token = :restore_token AND
restore_finish_at > NOW()");
    $stmt->bindParam(":restore_token", $restore_token);

    if ($stmt->execute()) {
        if ($stmt->rowCount() > 0) {
            return true;
        } else {
            return false;
        }
    }

    return true;
}

```

```

function updateUserLangData($db, $id, $data): void
{
    $query = "UPDATE users
              SET
                lang = :lang
              WHERE id = :user_id
              ";

    $stmt = $db->prepare($query);

    $stmt->bindParam(":lang", $data['lang']);
    $stmt->bindParam(":user_id", $id);

    $stmt->execute();
}

```

```

function updateUserDealIdData($db, $verification_token, $deal_id): void
{
    $query = "UPDATE users
              SET
                deal_id = :deal_id
              WHERE verification_token = :verification_token AND activated = 0
              ";

    $stmt = $db->prepare($query);

    $stmt->bindParam(":verification_token", $verification_token);
    $stmt->bindParam(":deal_id", $deal_id);

    $stmt->execute();
}

```

```

function getUserDealIdData($db, $id)
{
    $stmt = $db->prepare("SELECT deal_id
                        FROM users as u
                        WHERE u.id = :id
                        ");
    $stmt->bindParam(":id", $id);
    $stmt->execute();

    return $stmt->fetch(PDO::FETCH_ASSOC);
}

```

```

function getUserEmailByVerificationToken($db, $verification_token)
{
    $stmt = $db->prepare("SELECT * FROM users WHERE verification_token =
:verification_token");
    $stmt->bindParam(":verification_token", $verification_token);
}

```

```

if ($stmt->execute()) {
    if ($stmt->rowCount() > 0) {
        $user = $stmt->fetch(PDO::FETCH_ASSOC);
        return $user;
    } else {
        return false;
    }
} else {
    return false;
}
}

function deleteUsersByEmailNotActivated($db, $email): void
{
    $query = "DELETE FROM users WHERE email = :email AND activated = 0";

    $stmt = $db->prepare($query);

    $stmt->bindParam(":email", $email);

    $stmt->execute();
}

```

Листинг В.4 Файл validation.php

```

<?php

# users
function validatePostUsers($data): string|bool
{
    if (!isset($data['name']) or mb_strlen($data['name']) <= 0 or mb_strlen($data['name']) > 50 or
    !is_string($data['name'])) {
        return "Ошибка в поле: name";
    }
    if (!isset($data['surname']) or mb_strlen($data['surname']) <= 0 or mb_strlen($data['surname']) >
    50 or !is_string($data['surname'])) {
        return "Ошибка в поле: surname";
    }
    if (!isset($data['nationality']) or mb_strlen($data['nationality']) <= 0 or
    mb_strlen($data['nationality']) > 100 or !is_string($data['nationality'])) {
        return "Ошибка в поле: surname";
    }
    if (isset($data['phone']) and (mb_strlen($data['phone']) <= 5 or mb_strlen($data['phone']) > 25 or
    !is_numeric($data['phone']))) {
        return "Ошибка в поле: phone";
    }
}

```

```

    if (!isset($data['email']) or mb_strlen($data['email']) < 5 or mb_strlen($data['email']) > 100 or
    !is_string($data['email']) or !validateIsEmail($data['email'])) {
        return "Ошибка в поле: email";
    }
    if (!isset($data['lang']) or ($data['lang'] !== "ru" and $data['lang'] !== "ar" and $data['lang'] !==
    "eng")) {
        return "Ошибка в поле: lang";
    }
    if (!isset($data['password']) or mb_strlen($data['password']) < 8 or mb_strlen($data['password']) >
    50 or !is_string($data['password'])) {
        return "Ошибка в поле: password";
    }

    return true;
}

function validatePostAuth($data): string|bool
{
    if (!isset($data['email']) or mb_strlen($data['email']) < 5 or mb_strlen($data['email']) > 100 or
    !is_string($data['email']) or !validateIsEmail($data['email'])) {
        return "Ошибка в поле: email";
    }
    if (!isset($data['password']) or mb_strlen($data['password']) < 8 or mb_strlen($data['password']) >
    50 or !is_string($data['password'])) {
        return "Ошибка в поле: password";
    }

    return true;
}

function validateGetVerify($data): string|bool
{
    if (!isset($data['token']) or mb_strlen($data['token']) < 15 or mb_strlen($data['token']) > 20 or
    !is_string($data['token'])) {
        return "Ошибка в поле: token";
    }

    return true;
}

function validateGetCheckRestoreToken($data): string|bool
{
    if (!isset($data['restore_token']) or mb_strlen($data['restore_token']) < 15 or
    mb_strlen($data['restore_token']) > 20 or !is_string($data['restore_token'])) {
        return "Ошибка в поле: restore_token";
    }

    return true;
}

```

```

function validateGetRestorePassword($data): string|bool
{
    if (!isset($data['email']) or mb_strlen($data['email']) < 5 or mb_strlen($data['email']) > 100 or
    !is_string($data['email']) or !validateIsEmail($data['email'])) {
        return "Ошибка в поле: email";
    }

    return true;
}

function validatePostRestorePassword($data): string|bool
{
    if (!isset($data['restore_token']) or mb_strlen($data['restore_token']) < 15 or
    mb_strlen($data['restore_token']) > 20 or !is_string($data['restore_token'])) {
        return "Ошибка в поле: restore_token";
    }
    if (!isset($data['password']) or mb_strlen($data['password']) < 8 or mb_strlen($data['password']) >
    50 or !is_string($data['password'])) {
        return "Ошибка в поле: password";
    }

    return true;
}

function validatePostSetLang($data): string|bool
{
    if (!isset($data['lang']) or ($data['lang'] !== "ru" and $data['lang'] !== "ar" and $data['lang'] !==
    "eng")) {
        return "Ошибка в поле: lang";
    }

    return true;
}

```

Database.php

```
<?php
```

```
//require "Config.php";
```

```

class Database
{
    private $host;
    private $db_name;
    private $username;
    private $password;
    public $conn;

```

```

public function __construct()
{
    $this->host = Config::$host;
    $this->db_name = Config::$db_name;
    $this->username = Config::$username;
    $this->password = Config::$password;
}

public function getConnection()
{
    $this->conn = null;

    try {
        $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name, $this-
>username, $this->password);
    } catch (PDOException $exception) {
        echo "Ошибка соединения с БД: " . $exception->getMessage();
    }

    return $this->conn;
}
}

```

Листинг В.5 Файл utils.php

```

<?php

use YooKassa\Client;
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
use PHPMailer\PHPMailer\SMTP;

//require 'PHPMailer-master/src/Exception.php';
//require 'PHPMailer-master/src/PHPMailer.php';
//require 'PHPMailer-master/src/SMTP.php';
# нужен ли для роута токен
function needCheckToken($type, $method): bool
{
    if ($type == "auth") {
        return false;
    }

    if ($type == "users" and $method == "POST") {
        return false;
    }

    if ($type == "checkRestoreToken" and $method == "GET") {

```

```

        return false;
    }

    if ($type == "restorePassword") {
        return false;
    }

    if ($type == "verify") {
        return false;
    }

    return true;
}

function addLog($message, $file, $path): void
{
    date_default_timezone_set('UTC');
    file_put_contents($path . $file, date('[Y-m-d H:i:s] ') . $message . PHP_EOL, FILE_APPEND);
}

function sendEmail($send_to, $body, $name, $subject)
{
    $mail = new PHPMailer(true);
    $mail->IsSMTP();
    //$mail->SMTPDebug = 1; // debugging: 1 = errors and messages, 2 = messages only
    $mail->SMTPAuth = true; // authentication enabled
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS; // secure transfer enabled
    REQUIRED for Gmail
    $mail->Host = 'mail.hostland.ru';
    $mail->Port = 465; // or 587
    $mail->CharSet = "utf-8";
    $mail->IsHTML(true);
    $mail->Username = "";
    $mail->Password = "";
    $mail->SetFrom("", "Russian Arabian Gate");
    $mail->Subject = $subject;
    $mail->Body = $body;
    //$mail->addAttachment($_SERVER['DOCUMENT_ROOT'] . "/certificates/$cert_name");
    $mail->AddAddress($send_to, $name);

    $mail->send();
}

function getRandomString(): string
{
    $input = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $input_length = strlen($input);

```

```

$random_string = "";
$max_rnd = mt_rand(15, 20);
for($i = 0; $i < $max_rnd; $i++) {
    $random_character = $input[mt_rand(0, $input_length - 1)];
    $random_string .= $random_character;
}

return $random_string;
}

function sendErrorTg($text) {
    $token = Config::$bot_token;
    $channel_id = Config::$channel_errors_id;
    $textMessage = "Произошла ошибка.\nПодробности: $text";
    $textMessage = urlencode($textMessage);

    $urlQuery = 'https://api.telegram.org/bot' . $token . '/sendMessage?chat_id=' . $channel_id .
    '&text=' . $textMessage . '&parse_mode=None';
    file_get_contents($urlQuery);
}

# битрикс24
function bitrixCreateContact($data)
{
    $result = CRest::call(
        'crm.contact.add',
        [
            'FIELDS' => [
                'NAME' => $data['name'],
                'LAST_NAME' => $data['surname'],
                'ASSIGNED_BY_ID' => 1,
                'TYPE_ID' => 'CLIENT',
                'SOURCE_ID' => '1',
                'OPENED' => 'Y',
                'EXPORT' => 'Y',
                '#ASSIGNED_BY_ID' => 6,
                'PHONE' => [
                    [
                        'VALUE' => $data['phone'],
                        'VALUE_TYPE' => 'WORK',
                    ]
                ],
            ],
            'EMAIL' => [
                [
                    'VALUE' => $data['email'],
                    'VALUE_TYPE' => 'WORK',
                ]
            ]
        ]
    )
}

```

```

    ]
);

if (isset($result['result']) && !isset($result['error'])) {
    $contact_id = $result['result'];
    addLog("[Контакт успешно создан]" . " | contact_id = $contact_id | " . PHP_EOL, "main.log",
$_SERVER['DOCUMENT_ROOT'] . Config::$log_functions_path);
    return $contact_id;
} else {
    if (isset($result['error'])) {
        addLog("[Ошибка при создании контакта]" . " | error_description =
".$result['error_description']. " | " . PHP_EOL . print_r($data, true), "main.log",
$_SERVER['DOCUMENT_ROOT'] . Config::$log_functions_path);
    } else {
        addLog("[Ошибка НЕИЗВЕСТНАЯ при создании контакта]" . " | " . PHP_EOL .
print_r($data, true), "main.log", $_SERVER['DOCUMENT_ROOT'] . Config::$log_functions_path);
    }
    return null;
}
}

function bitrixCreateDeal($data, $contact_id)
{
    $result = CRest::call(
        'crm.deal.add',
        [
            'FIELDS' => [
                'TITLE' => 'Сделка '.$data['name']. " ".$data['surname'],
                'TYPE_ID' => 'SALE',
                'ASSIGNED_BY_ID' => 1,
                'CATEGORY_ID' => 0,
                'STAGE_ID' => 'UC_YB52C3',
                'IS_RECURRING' => 'N',
                'IS_RETURN_CUSTOMER' => 'N',
                'IS_REPEATED_APPROACH' => 'N',
                'CURRENCY_ID' => 'RUB',
                'OPPORTUNITY' => 0.00,
                'IS_MANUAL_OPPORTUNITY' => 'N',
                'TAX_VALUE' => 0.00,
                'COMPANY_ID' => 0,
                'CONTACT_ID' => $contact_id,
                'OPENED' => 'Y',
                'CLOSED' => 'N',
                'SOURCE_ID' => "1",
                'UF_CRM_1744397507' => $data['phone'],
            ]
        ]
    );
};

```

```

if (isset($result['result']) && !isset($result['error'])) {
    $deal_id = $result['result'];
    addLog("[Сделка успешно создана]" . " | deal_id = $deal_id | " . PHP_EOL, "main.log",
$_SERVER['DOCUMENT_ROOT'] . Config::$log_functions_path);
    return $deal_id;
} else {
    if (isset($result['error'])) {
        addLog("[Ошибка при создании сделки]" . " | error_description =
".$result['error_description']." | " . PHP_EOL . print_r($data, true), "main.log",
$_SERVER['DOCUMENT_ROOT'] . Config::$log_functions_path);
    } else {
        addLog("[Ошибка НЕИЗВЕСТНАЯ при создании сделки]" . " | " . PHP_EOL .
print_r($data, true), "main.log", $_SERVER['DOCUMENT_ROOT'] . Config::$log_functions_path);
    }
    return null;
}
}

function bitrixGetInfo($deal_id)
{
    $deal = CRest::call(
        'crm.deal.get',
        [
            'ID' => $deal_id
        ]
    );
    $deal = $deal['result'];

    $assigned_id = $deal['ASSIGNED_BY_ID'];

    $assigned = CRest::call(
        'user.get',
        [
            'id' => $assigned_id
        ]
    );
    $assigned = $assigned['result'][0];

    $deal_speciality = $deal['UF_CRM_1744233946'];
    $deal_field_speciality = CRest::call(
        'crm.deal.fields',
        [
            'id' => $assigned_id
        ]
    );
    $deal_field_speciality_list = $deal_field_speciality['result']['UF_CRM_1744233946']['items'];
    $spec_value = findValueByIdDealField($deal_field_speciality_list, $deal_speciality);

    $deal_status = $deal['UF_CRM_1745361181'];

```

```
$deal_field_status_list = $deal_field_speciality['result']['UF_CRM_1745361181']['items'];
$status_value = findValueByIdDealField($deal_field_status_list, $deal_status);
```

```
$files = array(
    "UF_CRM_1746122652" => $deal['UF_CRM_1746122652']['id'] ?? null,
    "UF_CRM_1746122689" => $deal['UF_CRM_1746122689']['id'] ?? null,
    "UF_CRM_1746122808" => $deal['UF_CRM_1746122808']['id'] ?? null
);
```

```
$file_names = array();
foreach ($files as $key => $file_id) {
    if (!is_null($file_id)) {
        $f_name = getFileName($file_id, $deal_id, $key);
        $file_names[] = array(
            "file_name" => $f_name,
            "file_id" => $file_id,
            "deal_id" => $deal_id,
            "option" => $key
        );
    }
}
```

```
$data = array(
    "university" => $deal['UF_CRM_1744234559'],
    "department" => $deal['UF_CRM_1744234784'],
    "speciality" => $spec_value,
    "status" => $status_value,
    "assigned_name" => $assigned['NAME'],
    "assigned_surname" => $assigned['LAST_NAME'],
    "assigned_email" => $assigned['EMAIL'],
    "assigned_phone" => $assigned['PERSONAL_MOBILE'] ?? null,
    "files" => $file_names
);
```

```
return $data;
}
```

```
function findValueByIdDealField($items, $id) {
    foreach ($items as $item) {
        if ($item['ID'] == $id) {
            return trim($item['VALUE']);
        }
    }
    return null;
}
```

```
function findValueByIdDealStage($items, $id) {
    foreach ($items as $item) {
```

```

        if ($item['STATUS_ID'] == $id) {
            return trim($item['NAME']);
        }
    }
    return null;
}

function getFileName($file_id, $deal_id, $option)
{
    $host = "";
    $fileUrl = $host .
"/bitrix/components/bitrix/crm.deal.show/show_file.php?ownerId=$deal_id&fieldName=$option&dynamic=Y&fileId=$file_id&login=yes";

    $authData = [
        'AUTH_FORM' => 'Y',
        'TYPE' => 'AUTH',
        'USER_LOGIN' => "",
        'USER_PASSWORD' => ""
    ];

    $context = stream_context_create([
        'http' => [
            'method' => 'POST',
            'header' => 'Content-Type: application/x-www-form-urlencoded',
            'content' => http_build_query($authData),
            'follow_location' => 1,
            'ignore_errors' => true
        ],
        'ssl' => [
            'verify_peer' => false,
            'verify_peer_name' => false
        ]
    ]);

    $headers = get_headers($fileUrl, 1, $context);

    $fileName = "";
    if (isset($headers['Content-Disposition'])) {
        $contentDisposition = is_array($headers['Content-Disposition'])
            ? $headers['Content-Disposition'][0]
            : $headers['Content-Disposition'];

        if (preg_match('/filename="([^\"]+)/i', $contentDisposition, $matches)) {
            $fileName = $matches[1];
        } elseif (preg_match('/filename=([^\;]+)/i', $contentDisposition, $matches)) {
            $fileName = trim($matches[1], "\"");
        }
    }
}

```

```

    return htmlspecialchars($fileName);
}
function downloadBitrixFile($data)
{
    $file_id = $data['file_id'];
    $deal_id = $data['deal_id'];
    $option = $data['option'];

    $host = "";
    $fileUrl = $host .
"/bitrix/components/bitrix/crm.deal.show/show_file.php?ownerId=$deal_id&fieldName=$option&dynamic=Y&fileId=$file_id&login=yes";

    $authData = [
        'AUTH_FORM' => 'Y',
        'TYPE' => 'AUTH',
        'USER_LOGIN' => '',
        'USER_PASSWORD' => ""
    ];
    $ch = curl_init();
    curl_setopt_array($ch, [
        CURLOPT_URL => $fileUrl,
        CURLOPT_RETURNTRANSFER => false, // Потокная передача
        CURLOPT_POST => true,
        CURLOPT_POSTFIELDS => http_build_query($authData),
        CURLOPT_FOLLOWLOCATION => true,
        CURLOPT_SSL_VERIFYPEER => false,
        CURLOPT_SSL_VERIFYHOST => false,
        CURLOPT_HEADERFUNCTION => function($curl, $header) {
            // Перехватываем заголовки Content-Type и Content-Disposition
            if (stripos($header, 'Content-Type:') === 0) {
                header($header);
            }
            if (stripos($header, 'Content-Disposition:') === 0) {
                header($header);
            }
            return strlen($header);
        }
    ]);
    curl_exec($ch);

    if (curl_errno($ch)) {
        die('Ошибка скачивания файла: ' . curl_error($ch));
    }
    curl_close($ch);}

```

Приложение Г
(обязательное)
Листинг клиентской части

Листинг Г.1 – Файл index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>Ragedu</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.ts"></script>
  </body>
</html>
```

Листинг Г.2 – Файл global.scss

```
@import 'src/styles/variables.scss';
@import 'src/styles/fonts.scss';
@import 'src/styles/mixins.scss';

html,
body {
  max-height: 100vh;
  overflow: hidden;
}

// fixes VBtn icon positioning bug
.v-btn--icon {
  .v-btn__content {
    width: 20px;
    height: 20px;

    .v-icon {
      width: 100%;
      height: 100%;

      &::before {
        width: 100%;
        height: 100%;
      }
    }
  }
}

// list
.list-move,
.list-enter-active,
.list-leave-active {
  transition: all 0.5s ease;
}
```

```

.list-enter-from,
.list-leave-to {
  opacity: 0;
  transform: translateX(56px);
}

.list-leave-active {
  position: absolute;
}

// table
.table-move,
.table-enter-active,
.table-leave-active {
  transition: all 0.5s ease;
}

.table-enter-from,
.table-leave-to {
  opacity: 0;
}

// fade
.fade-enter-active {
  transition: opacity 0.25s;
}
.fade-leave-active {
  transition: opacity 0.4s;
}

.fade-enter-from,
.fade-leave-to {
  opacity: 0;
}

```

Листинг Г.2 – Файл fonts.scss

```

@font-face {
  font-family: Onest;
  font-display: swap;
  src: url("@/assets/fonts/Onest-Regular.woff") format("woff");
  font-weight: 400;
  font-style: normal;
}

@font-face {
  font-family: Onest;
  font-display: swap;
  src: url("@/assets/fonts/Onest-Medium.woff") format("woff");
  font-weight: 500;
  font-style: normal;
}

```

```

}

@font-face {
  font-family: Onest;
  font-display: swap;
  src: url("@/assets/fonts/Onest-SemiBold.woff") format("woff");
  font-weight: 600;
  font-style: normal;
}

@font-face {
  font-family: Onest;
  font-display: swap;
  src: url("@/assets/fonts/Onest-Bold.woff") format("woff");
  font-weight: 700;
  font-style: normal;
}

```

Листинг Г.3 – Файл mixins.scss

```

@mixin ltr() {
  html[dir="ltr"] & {
    @content;
  }
}

@mixin rtl() {
  html[dir="rtl"] & {
    @content;
  }
}

```

Листинг Г.4 – Файл variables.scss

```

$font-dark-color: #012345;

$background-accent: #1A365D;
$background-primary: #F1F5F7;
$background-secondary: #FFFFFF;
$accent-color: #C4A359;

$accent-good: #0288D1;
$accent-good-light: #D1F0FF;
$accent-good-lighten: #F6FCFF;
$accent-bad: #FF8A43;
$accent-bad-light: #F9ECE4;
$accent-bad-lighten: #FEFBFA;
$accent-worst: #FF5643;

$darkest: #04000B;
$dark: #9E9E9E;
$icon-bg-or-stroke: #D9D9D9;

```

```

$font-lighten: #FFFFFF;
$font-light: #C5C5C5;
$font-darkest: #04000B;
$font-dark: #818181;

$file-docx: #1A365D;
$file-xls: #1C6C2A;
$file-pdf: #A4540D;
$file-jpg: #C4B101;

$medium-shadow: 3px 3px 10px 0px rgba(0, 0, 0, 0.1);

$header-height-desktop: 70px;
$header-height-tablet: 70px;
$header-height-mobile: 70px;

$page-padding-top-desktop: 32px;
$page-padding-top-tablet: 24px;
$page-padding-top-mobile: 20px;

$body-font-family: 'Onest';

$icon-button-size-small: 24px;
$icon-button-icon-size-small: 14px;
$icon-button-icon-font-small: .6rem;

$icon-button-size-default: 40px;
$icon-button-icon-size-default: 24px;
$icon-button-icon-font-default: 1rem;

$button-height: 56px;
$button-border-radius: 4px;
$button-font-size: 16px;
$button-font-weight: 600;
$button-text-transform: none;
$button-text-letter-spacing: 0;

$text-field-input-padding-start: 20px;
$text-field-input-padding-start-prepend: 16px;
$text-field-input-padding-end: 20px;
$text-field-input-padding-start-append: 20px;
$field-input-padding-top: 25px;
$field-input-padding-bottom: 25px;
$field-border-radius: 4px;
$field-min-height: 74px;

$special-field-min-height: 36px;
$special-text-field-input-padding-start: 10px;
$special-text-field-input-padding-end: 6px;
$special-field-input-padding-top: 6px;
$special-field-input-padding-bottom: 6px;

```

```

<template>
  <AppLayout v-if="isMounted">
    <RouterView />
  </AppLayout>
</template>

<script lang="ts" setup>
  // Vue
  import {
    ref,
    onMounted
  } from "vue";

  // Components
  import AppLayout from "@layouts/AppLayout.vue";

  // Stores
  import {
    useAuthStore
  } from "@store/auth";
  import {
    useUserStore,
  } from "@store/user";
  import {
    storeToRefs,
  } from "pinia";

  const {
    tryAuthLocal,
    fetchUserData,
    fetchBitrixData,
  } = useAuthStore();
  const {
    isAuth,
  } = storeToRefs(useAuthStore());
  const {
    isUserData,
  } = storeToRefs(useUserStore());

  const isMounted = ref(false);

  onMounted(async () => {
    if (isAuth.value && isUserData.value) {
      isMounted.value = true;
      return;
    }

    if (!isAuth.value) {
      await authUser();
    }

    if (isAuth.value && !isUserData.value) {
      await getUserData();
    }
  });

```

```

    }

    isMounted.value = true;
  });

const authUser = async () => {
  try {
    await tryAuthLocal();
  } catch (error) {
    console.error('App mounting error', error);
  }
}

const getUserData = async () => {
  try {
    await fetchUserData();
    await fetchBitrixData();
  } catch (error) {
    console.error('App mounting error', error);
  }
}
</script>

<style lang="scss">
.loaderContainer {
  width: 100%;
  min-height: 200px;
  display: flex;
  justify-content: center;
  align-items: center;
}

html {
  font-family: $body-font-family !important;
}
</style>

```

Листинг Г.6 – Файл AppLayout.vue

```

<template>
  <component :is="layoutComponent">
    <slot/>
  </component>
</template>

<script lang="ts" setup>
  // Vue
  import {
    computed,
    onMounted,
    watch,
  } from "vue";
  import {

```

```

    useRoute,
  } from "vue-router";

  // Plugins
  import i18n from '@plugins/i18n';
  import { useLocale } from "vuetify";

  const route = useRoute();

  const {
    // Changes vuetify components direction (based on configured
    locales)
    current: currentVuetifyLocale,
  } = useLocale();

  const layoutComponent = computed(() => {
    return route.meta.layoutComponent;
  });

  watch(i18n.global.locale, (newValue) => {
    handleLocale(newValue);
  });

  onMounted(() => {
    handleLocale(i18n.global.locale.value);
  });

  const handleLocale = (value: string) => {
    if (value === 'ar') {
      document.documentElement.setAttribute('dir', 'rtl');
      currentVuetifyLocale.value = 'rtl';
      return;
    }

    document.documentElement.setAttribute('dir', 'ltr');
    currentVuetifyLocale.value = 'ltr';
  }
</script>

```

Листинг Г.7 – Файл stickyHeader.vue

```

<template>
  <div class="stickyHeaderLayout">
    <div
      class="contentWrapper"
      ref="contentWrapper"
    >
      <Header
        class="header"
        :isScrollOverPageContent="isScrollOverPageContent"
        hasShadow
        :isLogoutButton="isLogoutButton"
      />
    </div>
  </div>
</template>

```

```

        <div class="pageWrapper">
            <RouterView />
        </div>
    </div>
    <CookieModal
        v-if="!isCookieAgree"
        @agreed="getCookieUsageAgreed"
    />
</div>
</template>

<script lang="ts" setup>
    // Vue
    import {
        computed,
        onMounted,
        onUnmounted,
        ref,
    } from "vue";
    import {
        useRoute,
    } from "vue-router";

    // Helpers
    import {
        throttle,
    } from "lodash";

    // Components
    import Header from "@/components/layout/Header.vue";
    import CookieModal from "@/components/modals/CookieModal.vue";

    // Stores
    import {
        useAppStore,
    } from "@/store/app";
    import {
        storeToRefs,
    } from "pinia";

    // Scss variables
    import {
        pagePaddingTopDesktop,
    } from "@/styles/variables.module.scss";

    const {
        getCookieUsageAgreed,
    } = useAppStore();
    const {
        isCookieAgree,
    } = storeToRefs(useAppStore());

    const route = useRoute();

```

```

    const pagePaddingTopDesktopInt =
parseInt(pagePaddingTopDesktop);

    const isScrollOverPageContent = ref(false);
    const contentWrapper = ref<HTMLElement | null>(null);

    const isLogoutButton = computed(() => {
      return route.meta.isLogoutButton;
    });

    onMounted(() => {
      getCookieUsageAgreed();
      contentWrapper.value?.addEventListener("scroll",
throttle(onScroll, 100));
    });

    onUnmounted(() => {
      contentWrapper.value?.removeEventListener("scroll",
throttle(onScroll, 100));
    })

    const onScroll = () => {
      if (!contentWrapper.value) {
        return;
      }

      isScrollOverPageContent.value =
contentWrapper.value?.scrollTop > (pagePaddingTopDesktopInt - 5);
    }
</script>

<style lang="scss" scoped>
  .stickyHeaderLayout {
    max-height: 100vh;
    background-color: $background-primary;
    display: flex;
    overflow: hidden;

    .header {
      position: sticky;
      top: 0px;
      z-index: 20;
    }

    .contentWrapper {
      width: 100%;
      overflow-y: scroll;

      .pageWrapper {
        min-height: calc(100vh - $header-height-desktop);
        padding-inline: 56px;
        padding-top: 24px;
      }
    }
  }

```

```

padding-bottom: 40px;

@media screen and (min-width: 640px) and (max-width:
1039px) {
  min-height: calc(100vh - $header-height-tablet);
  padding-inline: 20px;
  padding-top: 24px;
}

@media screen and (max-width: 639px) {
  min-height: calc(100vh - $header-height-mobile);
  padding-inline: 15px;
  padding-top: 24px;
}
}
}
}
</style>

```

Листинг Г.8 – Файл default.vue

```

<template>
  <div class="baseLayout">
    <div class="contentWrapper">
      <Header
        :isScrollOverPageContent="false"
        :hasShadow="false"
        :isLogoutButton="isLogoutButton"
      />
      <div class="pageWrapper">
        <RouterView />
      </div>
    </div>
    <CookieModal
      v-if="!isCookieAgree"
      @agreed="getCookieUsageAgreed"
    />
  </div>
</template>

<script lang="ts" setup>
  // Vue
  import {
    computed,
    onMounted,
  } from "vue";
  import {
    useRoute,
  } from "vue-router";

  // Components
  import Header from "@components/layout/Header.vue";
  import CookieModal from "@components/modals/CookieModal.vue";

```

```

// Stores
import {
  useAppStore,
} from "@/store/app";
import {
  storeToRefs,
} from "pinia";

const route = useRoute();

const {
  getCookieUsageAgreed,
} = useAppStore();
const {
  isCookieAgree,
} = storeToRefs(useAppStore());

const isLogoutButton = computed(() => {
  return route.meta.isLogoutButton;
});

onMounted(() => {
  getCookieUsageAgreed();
})
</script>

<style lang="scss" scoped>
  .baseLayout {
    background-color: $background-primary;
    display: flex;

    .contentWrapper {
      width: 100%;
      overflow-y: scroll;

      .pageWrapper {
        min-height: calc(100vh - $header-height-desktop);
        padding-inline: 56px;
        padding-top: 24px;
        display: flex;
        flex-direction: column;
        justify-content: center;

        @media screen and (min-width: 640px) and (max-width:
1039px) {
          min-height: calc(100vh - $header-height-tablet);
          padding-inline: 20px;
          padding-top: 24px;
        }

        @media screen and (max-width: 639px) {
          min-height: calc(100vh - $header-height-mobile);

```

```

        padding-inline: 15px;
        padding-top: 24px;
    }
}
}
}
</style>

```

Листинг Г.9 – Файл index.vue

```

<template>
  <div class="mainPage">
    <div class="content">
      <ContentCard size="big">
        <template #title>
          {{ educationTitle }}
        </template>
        <template #content>
          <EducationCardContent />
        </template>
      </ContentCard>
      <ContentCard size="big">
        <template #title>
          {{ documentsTitle }}
        </template>
        <template #titleAfter>
          <DocumentsViewSwitch />
        </template>
        <template #content>
          <DocumentsCardContent />
        </template>
      </ContentCard>
    </div>
    <div class="aside">
      <ContentCard size="small">
        <template #title>
          {{ userTitle }}
        </template>
        <template #content>
          <UserCardContent />
        </template>
      </ContentCard>
      <ContentCard size="small">
        <template #title>
          {{ helpTitle }}
        </template>
        <template #content>
          <HelpCardContent />
        </template>
      </ContentCard>
    </div>
  </div>
</template>

```

```

<script lang="ts" setup>
  // Vue
  import {
    computed,
  } from "vue";

  // Components
  import ContentCard from
"@/components/pages/main/ContentCard.vue";
  import EducationCardContent from
"@/components/pages/main/EducationCardContent.vue";
  import DocumentsCardContent from
"@/components/pages/main/Documents/DocumentsCardContent.vue";
  import UserCardContent from
"@/components/pages/main/UserCardContent.vue";
  import HelpCardContent from
"@/components/pages/main/HelpCardContent.vue";
  import DocumentsViewSwitch from
"@/components/pages/main/Documents/DocumentsViewSwitch.vue";

  // Plugins
  import {
    useI18n,
  } from "vue-i18n";

  // Helpers
  import {
    getLocaleWithFallback,
  } from "@/composables/helpers";

  const {
    t,
  } = useI18n();

  const educationTitle = computed(() => {
    return getLocaleWithFallback(t, 'main_page.education.title');
  });

  const documentsTitle = computed(() => {
    return getLocaleWithFallback(t, 'main_page.documents.title');
  });

  const userTitle = computed(() => {
    return getLocaleWithFallback(t, 'main_page.user.title');
  });

  const helpTitle = computed(() => {
    return getLocaleWithFallback(t, 'main_page.help.title');
  });
</script>

<style lang="scss" scoped>

```

```

.mainPage {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  gap: 24px;

  @media screen and (min-width: 640px) and (max-width: 1039px) {
    flex-direction: column-reverse;
    justify-content: flex-start;
    align-items: stretch;
  }

  @media screen and (max-width: 639px) {
    flex-direction: column-reverse;
    justify-content: flex-start;
    align-items: stretch;
  }

  .content {
    width: 70%;
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
    align-items: flex-start;
    gap: 24px;

    @media screen and (min-width: 640px) and (max-width: 1039px)
{
      width: 100%;
    }

    @media screen and (max-width: 639px) {
      width: 100%;
    }
  }

  .aside {
    width: 30%;
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
    align-items: flex-start;
    gap: 24px;

    @media screen and (min-width: 640px) and (max-width: 1039px)
{
      width: 100%;
    }

    @media screen and (max-width: 639px) {
      width: 100%;
    }
  }
}

```

```

    }
  </style>

```

Листинг Г.10 – Файл authentication.vue

```

<template>
  <div class="authPage">
    <div class="authContainer">
      <div class="titleContainer">
        <span class="authTitle">
          {{ title }}
        </span>
        <RouterLink
          :to="toRegLink"
          class="toReg"
        >
          {{ toRegLinkText }}
        </RouterLink>
      </div>
      <div class="authForm">
        <div class="authFormFields">
          <BaseTextField>
            <template #field>
              <VTextField
                v-model="formModel.email"
                autocomplete="username"
                :error="Boolean(emailErrors)"
                :error-messages="emailErrors"
                type="email"
                :label="emailFieldTexts.label"
                :placeholder="emailFieldTexts.placeholder"
                @input="v$.email.$touch"
                @change="v$.email.$touch"
                @blur="v$.email.$touch"
              ></VTextField>
            </template>
          </BaseTextField>
          <BaseTextField>
            <template #field>
              <VTextField
                v-model="formModel.password"
                autocomplete="current-password"
                :error="Boolean(passwordErrors)"
                :error-messages="passwordErrors"
                :type="isShowPassword ? 'text' : 'password'"
                :label="passwordFieldTexts.label"
                :placeholder="passwordFieldTexts.placeholder"
                @input="v$.password.$touch"
                @change="v$.password.$touch"
                @blur="v$.password.$touch"
              >
                <template #append-inner>
                  <div

```

```

        class="eyeIcon"
        @click="showHidePassHandler"
      >
        <VIcon
          v-show="!isShowPassword"
          size="32px"
        >
          <EyeIcon />
        </VIcon>
        <VIcon
          v-show="isShowPassword"
          size="32px"
        >
          <EyeCloseIcon />
        </VIcon>
      </div>
    </template>
  </VTextField>
</template>
</BaseTextField>
</div>
<div class="authFormAdditional">
  <RouterLink
    :to="resetPassLink"
    class="lostPasswordLink"
  >
    {{ resetPasswordLinkText }}
  </RouterLink>
</div>
</div>
<div class="buttonWrapper">
  <ActionServerResult :serverResult="serverResult" />
  <BaseButton
    block
  >
    <VBtnGood
      :disabled="!isAllValid"
      :loading="isLoading"
      @click="authHandler"
    >
      {{ buttonText }}
    </VBtnGood>
  </BaseButton>
  <RouterLink
    :to="resetPassLink"
    class="lostPasswordLink"
  >
    {{ resetPasswordLinkText }}
  </RouterLink>
  <RouterLink
    :to="toRegLink"
    class="toReg"
  >

```

```

        {{ toRegLinkText }}
      </RouterLink>
    </div>
  </div>
</div>
</template>

<script lang="ts" setup>
  // Vue
  import {
    ref,
    reactive,
    computed,
  } from "vue";
  import {
    useRouter,
  } from "vue-router";

  // Vuelidate
  import useVuelidate from "@vuelidate/core";
  import {
    required,
    minLength,
    maxLength,
  } from "@vuelidate/validators";

  // Composables
  import {
    useFieldErrors,
  } from "@composables/inputErrorHandlers";
  import {
    emailValidator,
  } from "@composables/validators";
  import {
    useServerResult,
  } from "@composables/serverResult";

  // Components
  import BaseTextField from "@components/ui/BaseTextField.vue";
  import ActionServerResult from
"@components/ui/ActionServerResult.vue";
  import BaseButton from "@components/ui/BaseButton.vue";
  import VBtnGood from
"@components/ui/AliasedComponents/VBtnGood.vue";

  // Plugins
  import {
    useI18n,
  } from "vue-i18n";

  // Helpers
  import {
    getLocaleWithFallback,

```

```

} from "@composables/helpers";

// Icons
import EyeIcon from '@assets/icons/eye_open.svg';
import EyeCloseIcon from '@assets/icons/eye_close.svg';

// Constants
import {
  SERVER_RESULT_NOTIFY_MODIFIER,
} from "@helpers/constants";

// Stores
import {
  useAuthStore,
} from "@store/auth";

const router = useRouter();

const {
  innerType,
  serverResult,
  setServerResult,
} = useServerResult();

const {
  authentication,
} = useAuthStore();

const {
  t,
} = useI18n();

innerType.value = 'authentication';

const formModel = reactive({
  email: '',
  password: ''
});

const toRegLink = {
  name: 'Registration',
};

const resetPassLink = {
  name: 'ResetPassword',
};

const rules = {
  email: {
    required,
    emailValidator,
  },
  password: {
    required,
  },
};

```

```

        minLength: minLength(8),
        maxLength: maxLength(50),
      }
    }

    const v$ = useVuelidate(rules, formModel);
    const {
      isValid,
      emailErrors,
      passwordErrors,
    } = useFieldErrors(v$, t);

    const isShowPassword = ref(false);
    const isLoading = ref(false);

    const title = computed(() => {
      return getLocaleWithFallback(t, 'authentication.title');
    });

    const toRegLinkText = computed(() => {
      return getLocaleWithFallback(t, 'authentication.to_reg_link');
    });

    const emailFieldTexts = computed(() => {
      return {
        label: getLocaleWithFallback(t,
'authentication.email.label'),
        placeholder: getLocaleWithFallback(t,
'authentication.email.placeholder'),
      }
    });

    const passwordFieldTexts = computed(() => {
      return {
        label: getLocaleWithFallback(t,
'authentication.password.label'),
        placeholder: getLocaleWithFallback(t,
'authentication.password.placeholder'),
      }
    });

    const resetPasswordLinkText = computed(() => {
      return getLocaleWithFallback(t,
'authentication.reset_password');
    });

    const buttonText = computed(() => {
      return getLocaleWithFallback(t, 'authentication.button_text');
    });

    const showHidePassHandler = () => {
      isShowPassword.value = !isShowPassword.value;
    }

```

```

const authHandler = async () => {
  isLoading.value = true;
  setServerResult();

  const isValid = await v$.value.$validate();

  if (!isValid) {
    isLoading.value = false;
    return;
  }

  const result = await authentication(formModel);

  setServerResult(result, SERVER_RESULT_NOTIFY_MODIFIER);
  isLoading.value = false;

  if (result?.number === 200) {
    await router.replace({
      name: 'Main',
    });
  }
}
</script>

<style lang="scss" scoped>
  .authPage {
    height: 100%;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    padding-top: $page-padding-top-desktop;
    padding-bottom: calc($page-padding-top-desktop * 2);

    .authContainer {
      width: 650px;
      display: flex;
      flex-direction: column;
      gap: 32px;
      padding: 40px 30px;
      background-color: $background-secondary;
      border-radius: 8px;
      box-shadow: $medium-shadow;

      @media screen and (min-width: 640px) and (max-width: 1039px)
    {
      max-width: 100%;
    }

    @media screen and (max-width: 639px) {
      max-width: 100%;
      padding: 24px 15px;
    }
  }

```

```

}

.titleContainer {
  width: 100%;
  display: flex;
  justify-content: space-between;
  align-items: center;

  .authTitle {
    color: $darkest;
    font-size: 24px;
    font-weight: 700;
    line-height: 1.3;

    @media screen and (max-width: 639px) {
      font-size: 22px;
    }
  }

  .toReg {
    color: $font-light;
    font-size: 16px;
    font-weight: 500;
    line-height: 1.3;
    text-decoration: underline;
    cursor: pointer;

    @media screen and (max-width: 639px) {
      display: none;
    }
  }
}

.authForm {
  width: 100%;
  display: flex;
  flex-direction: column;

  .authFormFields {
    width: 100%;
    display: flex;
    flex-direction: column;
    gap: 16px;
  }

  .authFormAdditional {
    width: 100%;
    display: flex;
    flex-direction: row-reverse;
    justify-content: space-between;

    @media screen and (max-width: 639px) {
      display: none;
    }
  }
}

```

```

    }
}

.lostPasswordLink {
  color: $font-light;
  font-size: 16px;
  font-weight: 500;
  line-height: 1.3;
  text-decoration: underline;
  cursor: pointer;
}

.eyeIcon {
  color: $dark;
  width: 32px;
  height: 32px;
  cursor: pointer;
}
}

.buttonWrapper {
  width: 100%;
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: center;
  gap: 12px;

  .toReg {
    color: $font-light;
    font-size: 16px;
    font-weight: 500;
    line-height: 1.3;
    text-decoration: underline;
    cursor: pointer;

    @media screen and (min-width: 640px) {
      display: none;
    }
  }

  .lostPasswordLink {
    color: $font-light;
    font-size: 16px;
    font-weight: 500;
    line-height: 1.3;
    text-decoration: underline;
    cursor: pointer;

    @media screen and (min-width: 640px) {
      display: none;
    }
  }
}

```

```

    }
  }
}
</style>

```

Листинг Г.11 – Файл registration.vue

```

<template>
  <div class="regPage">
    <div class="regContainer">
      <div class="topContainer">
        <div class="titleContainer">
          <span class="regTitle">
            {{ title }}
          </span>
          <RouterLink
            :to="toAuthLink"
            class="toAuth"
          >
            {{ toAuthLinkText }}
          </RouterLink>
        </div>
        <BaseTabs
          :options="regTabsOptions"
          :tabs="regTabs"
        ></BaseTabs>
      </div>
      <div class="content">
        <component
          :is="regActiveTabContent"
          @success="regSuccessHandler"
          @agree="regAgreeHandler"
        ></component>
      </div>
    </div>
  </div>
</template>

<script lang="ts" setup>
  // Vue
  import {
    onMounted,
    computed,
    watch,
    reactive,
  } from "vue";
  import {
    useRouter,
    useRoute,
  } from "vue-router";

  // Components
  import BaseTabs from "@components/ui/BaseTabs.vue";

```

```

import RegistrationForm from
"@/components/pages/registration/RegistrationForm.vue";
import RegistrationConfirmationInfo from
"@/components/pages/registration/RegistrationConfirmationInfo.vue"
;

// Plugins
import {
  useI18n,
} from "vue-i18n";

// Helpers
import {
  getLocaleWithFallback,
} from "@/composables/helpers";

// Types
import type {
  Tab,
} from "@/types/Tabs";

const router = useRouter();
const route = useRoute();

const {
  t,
} = useI18n();

const toAuthLink = {
  name: 'Authentication',
};

const regTabsOptions = {
  isSelectable: false,
};

const regTabs = reactive<Array<Tab>>([
  {
    id: 1,
    isActive: true,
    isDisabled: false,
    to: {
      name: 'Registration',
      query: {
        step: '1'
      }
    },
    text: ''
  },
  {
    id: 2,
    isActive: false,
    isDisabled: true,

```

```

        to: {
          name: 'Registration',
          query: {
            step: '2'
          }
        },
        text: ''
      },
    ]);

    const activeTab = computed(): Tab | undefined => {
      return regTabs
        .find(({ isActive }) => isActive);
    }

    const regActiveTabContent = computed() => {
      return activeTab.value?.id === 1
        ? RegistrationForm
        : RegistrationConfirmationInfo;
    };

    const title = computed() => {
      return getLocaleWithFallback(t, 'registration.title');
    };

    const toAuthLinkText = computed() => {
      return getLocaleWithFallback(t, 'registration.to_auth_link');
    };

    const registrationFirstTabText = computed() => {
      return getLocaleWithFallback(t,
        'registration.tabs.first_tab_label')
    };

    const registrationSecondTabText = computed() => {
      return getLocaleWithFallback(t,
        'registration.tabs.second_tab_label')
    };

    watch(
      () => route.query?.step,
      (newValue) => {
        if (!newValue) {
          return;
        }

        activeTabHandler(newValue as string);
      }
    );

    watch(registrationFirstTabText, (newValue) => {
      regTabs[0].text = newValue;
    });

```

```

watch(registrationSecondTabText, (newValue) => {
  regTabs[1].text = newValue;
})

onMounted(() => {
  const {
    step,
  } = route.query ?? {};

  if (step) {
    activeTabHandler(step as string);
  }

  const path = activeTab.value?.to;

  if (path) {
    router.replace(path);
  }

  regTabs[0].text = registrationFirstTabText.value;
  regTabs[1].text = registrationSecondTabText.value;
});

const activeTabHandler = (newValue: string) => {
  regTabs?.forEach((tab) => {
    tab.isActive = false;
  })
  const newActiveTab = regTabs
    ?.find(({ id }) => id === +newValue);

  if (!newActiveTab) {
    return;
  }

  newActiveTab.isActive = true;
}

const regSuccessHandler = () => {
  router.replace(regTabs?.[1].to);
  regTabs[1].isDisabled = false;
}

const regAgreeHandler = () => {
  router.push({
    name: 'Authentication',
  });
}
</script>

<style lang="scss" scoped>
  .regPage {
    height: 100%;

```

```

display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
padding-top: $page-padding-top-desktop;
padding-bottom: calc($page-padding-top-desktop * 2);

@media screen and (min-width: 640px) and (max-width: 1039px) {
  padding-top: $page-padding-top-tablet;
  padding-bottom: calc($page-padding-top-tablet * 2);
}

@media screen and (max-width: 639px) {
  padding-top: $page-padding-top-mobile;
  padding-bottom: calc($page-padding-top-mobile * 2);
}

.regContainer {
  width: 740px;
  display: flex;
  flex-direction: column;
  gap: 32px;
  padding: 40px 30px;
  background-color: $background-secondary;
  border-radius: 8px;
  box-shadow: $medium-shadow;

  @media screen and (min-width: 640px) and (max-width: 1039px)
{
  max-width: 100%;
}

  @media screen and (max-width: 639px) {
    max-width: 100%;
    padding: 24px 15px;
  }

  .topContainer {
    display: flex;
    flex-direction: column;
    gap: 16px;

    .titleContainer {
      width: 100%;
      display: flex;
      justify-content: space-between;
      align-items: center;

      .regTitle {
        color: $darkest;
        font-size: 24px;
        font-weight: 700;
        line-height: 1.3;
      }
    }
  }
}

```

```

        @media screen and (max-width: 639px) {
            font-size: 22px;
        }
    }

    .toAuth {
        color: $font-light;
        font-size: 16px;
        font-weight: 500;
        line-height: 1.3;
        text-decoration: underline;
        cursor: pointer;

        @media screen and (max-width: 639px) {
            display: none;
        }
    }
}
}
}
}
</style>

```

Листинг Г.12 – Файл reset_password.vue

```

<template>
  <div class="resetPassPage">
    <div class="resetPassContainer">
      <div class="topContainer">
        <div class="titleContainer">
          <span class="resetPassTitle">
            {{ title }}
          </span>
          <RouterLink
            :to="toAuthLink"
            class="toAuth"
          >
            {{ toAuthLinkText }}
          </RouterLink>
        </div>
      </div>
      <div class="resetPassForm">
        <component
          :is="resetPassActiveTabContent"
          @success="resetPassSuccessHandler"
          @agree="resetPassAgreeHandler"
        ></component>
      </div>
    </div>
  </div>
</template>

```

```

<script lang="ts" setup>
  // Vue
  import {
    computed,
    onMounted,
    reactive,
    watch,
  } from "vue";
  import {
    useRoute,
    useRouter,
  } from "vue-router";

  // Components
  import ResetPasswordForm from
"@/components/pages/reset_password/ResetPasswordForm.vue";
  import ResetPasswordConfirmationInfo from
"@/components/pages/reset_password/ResetPasswordConfirmationInfo.v
ue";

  // Plugins
  import {
    useI18n,
  } from "vue-i18n";

  // Helpers
  import {
    getLocaleWithFallback,
  } from "@/composables/helpers";

  // Types
  import type {
    Tab,
  } from "@/types/Tabs";

  const router = useRouter();
  const route = useRoute();

  const {
    t,
  } = useI18n();

  const toAuthLink = {
    name: 'Authentication',
  };

  const resetPassTabs = reactive<Array<Tab>>([
    {
      id: 1,
      isActive: true,
      to: {
        name: 'ResetPassword',
        query: {

```

```

        step: '1'
      }
    },
  },
  {
    id: 2,
    isActive: false,
    to: {
      name: 'ResetPassword',
      query: {
        step: '2'
      }
    },
  },
],
]);

const activeTab = computed(): Tab | undefined => {
  return resetPassTabs
    .find(({ isActive }) => isActive);
}

const resetPassActiveTabContent = computed() => {
  return activeTab.value?.id === 1
    ? ResetPasswordForm
    : ResetPasswordConfirmationInfo;
});

const title = computed() => {
  return getLocaleWithFallback(t, 'reset_password.title');
});

const toAuthLinkText = computed() => {
  return getLocaleWithFallback(t,
'reset_password.to_auth_link');
});

watch(
  () => route.query?.step,
  (newValue) => {
    if (!newValue) {
      return;
    }

    activeTabHandler(newValue as string);
  }
)

onMounted(() => {
  const {
    step,
  } = route.query ?? {};

  if (step) {

```

```

        activeTabHandler(step as string);
    }

    const path = activeTab.value?.to;

    if (path) {
        router.replace(path);
    }
});

const activeTabHandler = (newValue: string) => {
    resetPassTabs?.forEach((tab) => {
        tab.isActive = false;
    })
    const newActiveTab = resetPassTabs
        ?.find(({ id }) => id === +newValue);

    if (!newActiveTab) {
        return;
    }

    newActiveTab.isActive = true;
}

const resetPassSuccessHandler = () => {
    const path = resetPassTabs?.[1].to;

    if (path) {
        router.replace(path);
    }
}

const resetPassAgreeHandler = () => {
    router.push({
        name: 'Authentication',
    });
}
</script>

<style lang="scss" scoped>
.resetPassPage {
    height: 100%;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    padding-top: $page-padding-top-desktop;
    padding-bottom: calc($page-padding-top-desktop * 2);

.resetPassContainer {
    width: 650px;
    display: flex;
    flex-direction: column;

```

```

gap: 32px;
padding: 40px 30px;
background-color: $background-secondary;
border-radius: 16px;
box-shadow: $medium-shadow;

@media screen and (max-width: 639px) {
  max-width: 100%;
  padding: 24px 15px;
  gap: 24px;
}

.topContainer {
  display: flex;
  flex-direction: column;
  gap: 16px;

  .titleContainer {
    width: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;

    .resetPassTitle {
      color: $darkest;
      font-size: 24px;
      font-weight: 700;
      line-height: 1.3;

      @media screen and (max-width: 639px) {
        font-size: 22px;
      }
    }
  }

  .toAuth {
    color: $font-light;
    font-size: 16px;
    font-weight: 500;
    line-height: 1.3;
    text-decoration: underline;
    cursor: pointer;

    @media screen and (max-width: 639px) {
      display: none;
    }
  }
}

.resetPassForm {
  width: 100%;
  display: flex;
  flex-direction: column;

```

```

    }
  }
}
</style>

```

Листинг Г.13 – Файл set_new_password.vue

```

<template>
  <div class="setNewPassPage">
    <div class="setNewPassContainer">
      <div class="topContainer">
        <div class="titleContainer">
          <span class="resetPassTitle">
            {{ title }}
          </span>
          <RouterLink
            :to="toAuthLink"
            class="toAuth"
          >
            {{ toAuthLinkText }}
          </RouterLink>
        </div>
        <BaseTabs
          :options="setNewPassTabsOptions"
          :tabs="setNewPassTabs"
        ></BaseTabs>
      </div>
      <div class="setNewPassForm">
        <component
          :is="setNewPassTabContent"
          @success="setNewPassSuccessHandler"
          @agree="setNewPassAgreeHandler"
        ></component>
      </div>
    </div>
  </div>
</template>

<script lang="ts" setup>
  // Vue
  import {
    computed,
    onMounted,
    reactive,
    watch,
  } from "vue";
  import {
    useRoute,
    useRouter,
  } from "vue-router";

  // Components
  import BaseTabs from "@components/ui/BaseTabs.vue";

```

```

import SetNewPasswordForm from
"@/components/pages/set_new_password/SetNewPasswordForm.vue";
import SetNewPasswordConfirmationInfo from
"@/components/pages/set_new_password/SetNewPasswordConfirmationInf
o.vue";

// Plugins
import {
  useI18n,
} from "vue-i18n";

// Helpers
import {
  getLocaleWithFallback,
} from "@/composables/helpers";

// Types
import type {
  Tab,
} from "@/types/Tabs";

// Stores
import {
  useAuthStore,
} from "@/store/auth";

const {
  setRestoreToken,
} = useAuthStore();

const router = useRouter();
const route = useRoute();

const {
  t,
} = useI18n();

const toAuthLink = {
  name: 'Authentication',
};

const setNewPassTabsOptions = {
  isSelectable: false,
};

const setNewPassTabs = reactive<Array<Tab>>([
  {
    id: 1,
    isActive: true,
    isDisabled: false,
    to: {
      name: 'SetNewPassword',
      query: {

```

```

        step: '1'
      }
    },
    text: ''
  },
  {
    id: 2,
    isActive: false,
    isDisabled: true,
    to: {
      name: 'SetNewPassword',
      query: {
        step: '2'
      }
    },
    text: ''
  },
]);

const activeTab = computed(): Tab | undefined => {
  return setNewPassTabs
    .find(({ isActive }) => isActive);
}

const setNewPassTabContent = computed() => {
  return activeTab.value?.id === 1
    ? SetNewPasswordForm
    : SetNewPasswordConfirmationInfo;
});

const title = computed() => {
  return getLocaleWithFallback(t, 'set_new_password.title');
});

const toAuthLinkText = computed() => {
  return getLocaleWithFallback(t,
'set_new_password.to_auth_link');
});

const setNewPassFirstTabText = computed() => {
  return getLocaleWithFallback(t,
'set_new_password.tabs.first_tab_label')
});

const setNewPassSecondTabText = computed() => {
  return getLocaleWithFallback(t,
'set_new_password.tabs.second_tab_label')
});

watch(
  () => route.query?.step,
  (newValue) => {
    if (!newValue) {

```

```

        return;
    }

    activeTabHandler(newValue as string);
}
)

watch(setNewPassFirstTabText, (newValue) => {
    setNewPassTabs[0].text = newValue;
})

watch(setNewPassSecondTabText, (newValue) => {
    setNewPassTabs[1].text = newValue;
})

onMounted(() => {
    const {
        step,
        token,
    } = route.query ?? {};

    if (token) {
        setRestoreToken(token as string);
    }

    if (step) {
        activeTabHandler(step as string);
    }

    const path = activeTab.value?.to;

    if (path) {
        router.replace(path);
    }
});

const activeTabHandler = (newValue: string) => {
    setNewPassTabs?.forEach((tab) => {
        tab.isActive = false;
    })
    const newActiveTab = setNewPassTabs
        ?.find(({ id }) => id === +newValue);

    if (!newActiveTab) {
        return;
    }

    newActiveTab.isActive = true;
}

const setNewPassSuccessHandler = () => {
    const path = setNewPassTabs?.[1].to;

```

```

    if (path) {
      router.replace(path);
      setNewPassTabs[1].isDisabled = false;
    }
  }

  const setNewPassAgreeHandler = () => {
    router.push({
      name: 'Authentication',
    });
  }
</script>

<style lang="scss" scoped>
  .setNewPassPage {
    height: 100%;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    padding-top: $page-padding-top-desktop;
    padding-bottom: calc($page-padding-top-desktop * 2);

    .setNewPassContainer {
      width: 650px;
      display: flex;
      flex-direction: column;
      gap: 32px;
      padding: 40px 30px;
      background-color: $background-secondary;
      border-radius: 16px;
      box-shadow: $medium-shadow;

      @media screen and (max-width: 639px) {
        max-width: 100%;
        padding: 24px 15px;
        gap: 24px;
      }

      .topContainer {
        display: flex;
        flex-direction: column;
        gap: 16px;

        .titleContainer {
          width: 100%;
          display: flex;
          justify-content: space-between;
          align-items: center;

          .resetPassTitle {
            color: $darkest;
            font-size: 24px;
          }
        }
      }
    }
  }

```

```

        font-weight: 700;
        line-height: 1.3;

        @media screen and (max-width: 639px) {
            font-size: 22px;
        }
    }

    .toAuth {
        color: $font-light;
        font-size: 16px;
        font-weight: 500;
        line-height: 1.3;
        text-decoration: underline;
        cursor: pointer;

        @media screen and (max-width: 639px) {
            display: none;
        }
    }
}

.setNewPassForm {
    width: 100%;
    display: flex;
    flex-direction: column;
}
}
}
</style>

```

Листинг Г.14 – Файл verify_email.vue

```

<template>
  <div class="verifyEmailPage">
    <div class="verifyEmailContainer">
      <div class="titleContainer">
        <span class="verifyEmailTitle">
          {{ title }}
        </span>
      </div>
      <div class="verifyEmailContent">
        <div class="message">
          <div
            v-if="isLoading"
            class="verifyEmailWait"
          >
            {{ verifyStatuses.wait }}
          </div>
          <div
            v-else-if="isActivated && !isError"
            class="verifyEmailDone"

```

```

        >
        {{ verifyStatuses.done }}
      </div>
      <div
        v-else-if="isError"
        class="verifyEmailError"
      >
        {{ errorMessage }}
      </div>
    </div>
    <BaseButton
      block
    >
      <VBtnGood
        @click="verifyEmailButtonHandler"
      >
        {{ buttonText }}
      </VBtnGood>
    </BaseButton>
  </div>
</div>
</div>
</template>

<script lang="ts" setup>
  // Vue
  import {
    computed,
    onMounted,
    ref,
  } from "vue";
  import {
    useRoute,
    useRouter,
  } from "vue-router";

  // Components
  import BaseButton from "@components/ui/BaseButton.vue";
  import VBtnGood from
"@components/ui/AliasedComponents/VBtnGood.vue";

  // Composables
  import {
    useServerResult,
  } from "@composables/serverResult";

  // Plugins
  import {
    useI18n,
  } from "vue-i18n";

  // Helpers
  import {

```

```

    getLocaleWithFallback,
  } from "@composables/helpers";

// Constants
import {
  SERVER_RESULT_NOTIFY_MODIFIER,
} from "@helpers/constants";

// Stores
import {
  useAuthStore,
} from "@store/auth";

const {
  verifyEmail,
  fetchUserData,
  fetchBitrixData,
} = useAuthStore();

const router = useRouter();
const route = useRoute();

const {
  t,
} = useI18n();

const {
  innerType,
  setServerResult,
} = useServerResult();

innerType.value = 'verifyEmail';

const isLoading = ref(false);
const isActivated = ref(false);
const isError = ref(false);
const isNewLinkRequired = ref(false);
const errorMessageKey = ref('');

const verifyEmailButtonTo = computed(() => {
  return 'Authentication';
});

const title = computed(() => {
  return getLocaleWithFallback(t, 'verify_email.title');
});

const verifyStatuses = computed(() => {
  return {
    "noToken": getLocaleWithFallback(t,
'verify_email.statuses.no_token'),
    "wait": getLocaleWithFallback(t,
'verify_email.statuses.wait'),

```

```

        "done": getLocaleWithFallback(t,
'verify_email.statuses.done'),
        "error": getLocaleWithFallback(t,
'verify_email.statuses.error'),
    };
});

const errorMessage = computed(() => {
    return verifyStatuses.value?.[errorMessageKey.value] ?? "";
});

const buttonText = computed(() => {
    return getLocaleWithFallback(t, 'verify_email.button_text');
});

onMounted(async () => {
    const token = route.query?.token;

    if (!token) {
        isError.value = true;
        errorMessageKey.value = "noToken";
        isNewLinkRequired.value = true;
        return;
    }

    await verifyEmailHandler(token as string);
});

const verifyEmailHandler = async (token: string) => {
    isLoading.value = true;

    const result = await verifyEmail(token);

    setServerResult(result, SERVER_RESULT_NOTIFY_MODIFIER);
    isLoading.value = false;

    if (result?.number === 200) {
        isActivated.value = true;
        await fetchUserData();
        await fetchBitrixData();
    } else {
        errorMessageKey.value = "error";
        isNewLinkRequired.value = true;
        isError.value = true;
    }
}

const verifyEmailButtonHandler = () => {
    router.push({
        name: verifyEmailButtonTo.value,
    });
}
</script>

```

```

<style lang="scss" scoped>
  .verifyEmailPage {
    height: 100%;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    padding-top: $page-padding-top-desktop;
    padding-bottom: calc($page-padding-top-desktop * 2);

    .verifyEmailContainer {
      width: 650px;
      display: flex;
      flex-direction: column;
      gap: 32px;
      padding: 40px 30px;
      background-color: $background-secondary;
      border-radius: 16px;
      box-shadow: $medium-shadow;

      @media screen and (max-width: 639px) {
        max-width: 100%;
        padding: 24px 15px;
        gap: 24px;
      }

      .titleContainer {
        width: 100%;
        display: flex;
        justify-content: space-between;
        align-items: center;

        .verifyEmailTitle {
          color: $darkest;
          font-size: 24px;
          font-weight: 700;
          line-height: 1.3;

          @media screen and (max-width: 639px) {
            font-size: 22px;
          }
        }
      }

      .toPage {
        color: $font-light;
        font-size: 16px;
        font-weight: 500;
        line-height: 1.3;
        text-decoration: underline;
        cursor: pointer;
      }
    }
  }

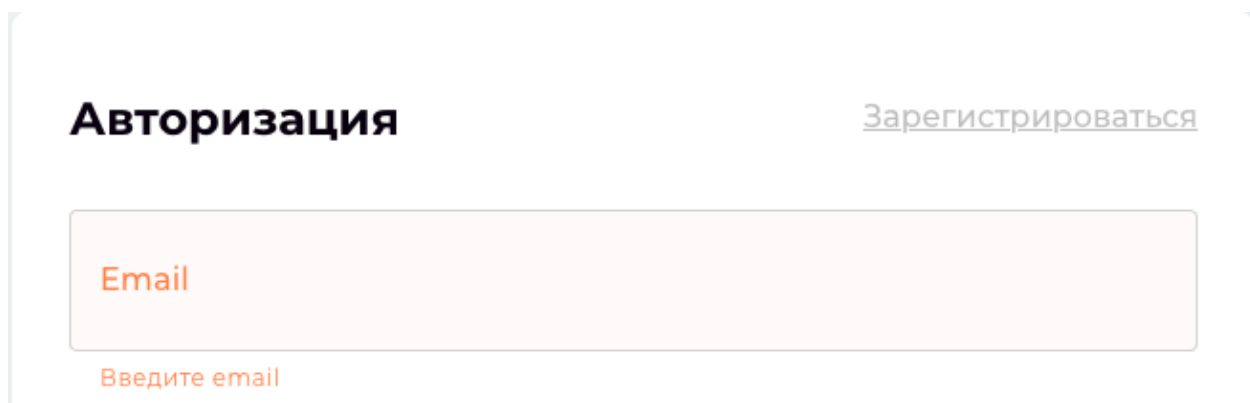
```

```
        .verifyEmailContent {  
            width: 100%;  
            display: flex;  
            flex-direction: column;  
            gap: 24px;  
        }  
    }  
}  
</style>
```

Приложение Д

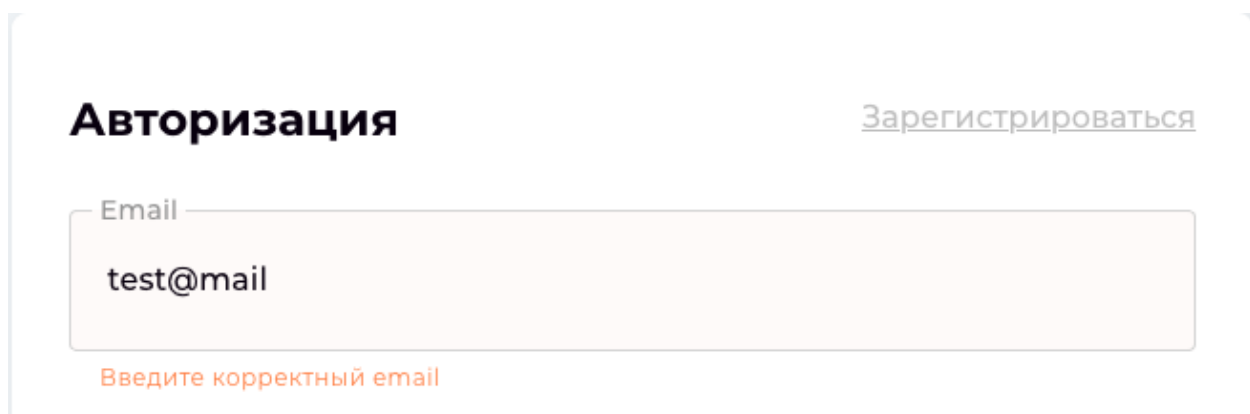
(обязательное)

Результаты тестирования клиентской части



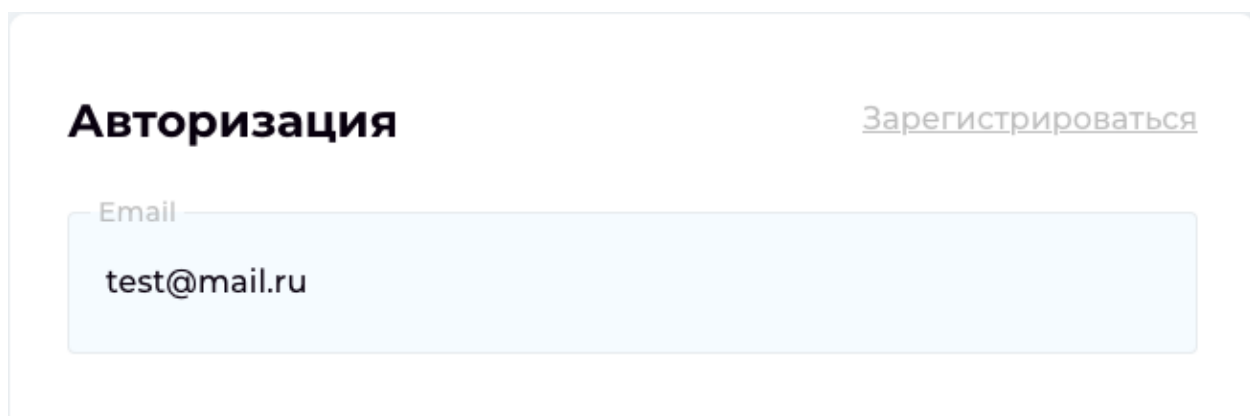
The screenshot shows a web interface for authorization. At the top left is the title "Авторизация" in bold black font. At the top right is a link "Зарегистрироваться" in blue underlined font. Below the title is a light orange rectangular input field. Inside the field, the word "Email" is written in orange. Below the field, the text "Введите email" is displayed in orange.

Рисунок Д.1 – Результаты тестирования класса эквивалентности «Пустое значение», для поля email



The screenshot shows the same authorization page as Figure D.1. The email field now contains the text "test@mail". Below the field, the text "Введите корректный email" is displayed in orange.

Рисунок Д.2 – Результаты тестирования класса эквивалентности «Некорректный формат», для поля email



The screenshot shows the same authorization page. The email field now contains the text "test@mail.ru". The field has a light blue background, and the text "test@mail.ru" is in black.

Рисунок Д.3 – Результаты тестирования класса эквивалентности «Корректный формат», для поля email

Авторизация [Зарегистрироваться](#)

Email

test@mail.ru

Пароль

Введите пароль

[Восстановить пароль](#)

Рисунок Д.4 – Результаты тестирования класса эквивалентности «Пустое значение», для поля пароль

Авторизация [Зарегистрироваться](#)

Email

test@mail.ru

Пароль

.....

Пароль должен быть не менее 8 символов

[Восстановить пароль](#)

Рисунок Д.5 – Результаты тестирования класса эквивалентности «Длина меньше 8», для поля пароль

The screenshot shows a login interface with the title "Авторизация" (Authorization) and a link "Зарегистрироваться" (Register). The "Email" field contains "test@mail.ru". The "Password" field is filled with 50 dots, indicating a password of length 50. A red error message "Пароль должен быть не более 50 символов" (Password must not be more than 50 symbols) is displayed below the password field. A "Восстановить пароль" (Reset password) link is visible at the bottom right.

Рисунок Д.6 – Результаты тестирования класса эквивалентности «Длина больше 50», для поля пароль

The screenshot shows the same login interface as Figure D.6. The "Email" field contains "test@mail.ru". The "Password" field is filled with 7 dots, indicating a password of length 7. There is no error message displayed. A "Восстановить пароль" (Reset password) link is visible at the bottom right.

Рисунок Д.7 – Результаты тестирования класса эквивалентности «Длина не меньше 8 и не больше 50», для поля пароль

Приложение Е
(обязательное)
Листинг модульных тестов

Листинг Е.1 – Файл AuthTests.сpec.js

```
import { mount } from '@vue/test-utils'
import { describe, it, expect } from 'vitest'
import Authentication from '@pages/authentication.vue'

describe('Authentication', () => {

  it('Показывает ошибку, если email пустой', async () => {
    const wrapper = mount(Authentication)

    await wrapper.find('form').trigger('submit')

    expect(wrapper.text()).toContain('Введите email')
  })

  it('Показывает ошибку при некорректном email', async () => {
    const wrapper = mount(Authentication)

    await wrapper.find('input[name="email"]').setValue('invalid-email')
    await
wrapper.find('input[name="password"]').setValue('12345678')

    await wrapper.find('form').trigger('submit')

    expect(wrapper.text()).toContain('Введите корректный email')
  })

  it('Показывает ошибку, если password пустой', async () => {
    const wrapper = mount(Authentication)

    await wrapper.find('input[name="email"]').setValue('valid-email@domain.zone')
    await wrapper.find('form').trigger('submit')

    expect(wrapper.text()).toContain('Введите пароль')
  })

  it('Показывает ошибку, если пароль меньше 8 символов', async ()
=> {
    const wrapper = mount(Authentication)

    await wrapper.find('input[name="email"]').setValue('valid-email@domain.zone')
    await
wrapper.find('input[name="password"]').setValue('1234567')

    await wrapper.find('form').trigger('submit')

    expect(wrapper.text()).toContain('Пароль должен быть не менее
8 символов')
  })
})
```

```

    it('Показывает ошибку, если пароль больше 50 символов', async ()
=> {
    const wrapper = mount(Authentication)

    const longPassword = 'a'.repeat(51)

    await wrapper.find('input[name="email"]').setValue('valid-
email@domain.zone')
    await
wrapper.find('input[name="password"]').setValue(longPassword)

    await wrapper.find('form').trigger('submit')

    expect(wrapper.text()).toContain('Пароль должен быть не более
50 символов')
  })

  it('Вызывает submit при валидных данных', async () => {
    const wrapper = mount(Authentication)

    await wrapper.find('input[name="email"]').setValue('valid-
email@domain.zone')
    await
wrapper.find('input[name="password"]').setValue('12345678')

    await wrapper.find('form').trigger('submit')

    expect(wrapper.emitted('submit')).toBeTruthy()
    expect(wrapper.emitted('submit')[0][0]).toEqual({
      email: 'valid-email@domain.zone',
      password: '12345678'
    })
  })
})
})

```

Приложение Ж
(обязательное)
Копия акта внедрения
программного обеспечения

**Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Пензенский государственный университет»
(ФГБОУ ВО «ПГУ»)**

Институт международного сотрудничества
Управление международного образования

АКТ ВНЕДРЕНИЯ

Комиссия в составе: врио начальника управления международного образования Плотников К.А. (председатель комиссии), начальник учебного отдела УМО Первушкин А.В., начальник отдела социальной адаптации и культурно-массовой работы Сягайло Е.Г.

установила, что результаты выпускной квалификационной работы студента группы 22ВИ1 (направление подготовки бакалавра 09.03.02 «Информационные системы и технологии») Макбола Али Али Омара по теме «Разработка системы информационной поддержки иностранных студентов» используется для организации работы с иностранными студентами из арабоязычных стран.

Настоящий акт не является основанием для материальных расчетов.

Председатель комиссии:

К.А. Плотников

Члены комиссии:

А.В. Первушкин

Е.Г. Сягайло

