

ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ
государственное профессиональное образовательное учреждение
Ярославской области
Рыбинский полиграфический колледж

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Дипломный проект

Разработка социальной сети для гурманов

специальность 09.02.07 – Информационные системы и программирование

Пояснительная записка

Студент группы	4-ИС-2 (Код)		Н.С. Кузнецов (И.О. Фамилия)
Руководитель	Преподаватель (Должность, звание)		Е.А. Лобанова (И.О. Фамилия)
Консультант по эко- номической части	Преподаватель (Должность, звание)		М.Г. Кудряшова (И.О. Фамилия)
Консультант по охране труда	Преподаватель (Должность, звание)		Е.А. Лобанова (И.О. Фамилия)
Нормоконтроль	Преподаватель (Должность, звание)		Т.М. Моисеева (И.О. Фамилия)
Рецензент	Преподаватель (Должность, звание)		С.В. Ермолычева (И.О. Фамилия)
К защите допустить Зам. директора по учебной работе			И.Н. Осокина (И.О. Фамилия)

г. Рыбинск
2023

СОДЕРЖАНИЕ

Введение.....	3
1 Исследовательский раздел	5
2 Конструкторский раздел.....	12
2.1 Проектирование информационной модели данных	12
2.2 Проектирование серверной части приложения.....	13
2.2.1 Разработка схемы базы данных	13
2.2.2 Разработка структуры сущностей базы данных.....	16
2.3 Проектирование клиентской части приложения	18
2.3.1 Разработка модульной схемы	18
2.3.2 Разработка пользовательского интерфейса.....	20
2.3.3 Организация доступа к объектам базы данных	35
2.3.4 Разработка блох-схем алгоритмов процедур и функций	38
2.4 Обеспечение коллективного доступа. Защита информации	41
3 Технологическая часть	45
3.1 Тестирование и отладка приложения.....	45
3.2 Инструкция администратора базы данных.....	49
3.3 Инструкция по эксплуатации приложения.....	55
4 Техничко-экономический раздел.....	62
5 Раздел охраны труда	73
Заключение	75
Список используемых источников.....	77
Приложение А	78

					ДП.0902.10.000000.00 ПЗ		
Изм	Лист	№ докум.	Подп.	Дата			
Разраб.	Кузнецов Н.С.				Разработка социальной сети для гур- маров	Лит.	Лист
Провер.	Лобанова Е.А.					У	К
						П	2
Н.контр.	Моисеева Т.М.					РПК, ГР. 4-ИС-2	
						Листов	85

ВВЕДЕНИЕ

Сегодня онлайн сервисы, предоставляющие доступ к различной информации, имеют огромную популярность. Сервис выбора рецептов не является исключением, и его реализация имеет ряд преимуществ при использовании вычислительной техники. Этот сервис предоставляет пользователю возможность найти и выбрать рецепты в соответствии с их предпочтениями, диетическими ограничениями и индивидуальными потребностями из огромного количества рецептов из различных источников. Благодаря интернету и развитию цифровых платформ, пользователи могут получить доступ к миллионам рецептов на одной платформе. Такой объем информации позволяет пользователям исследовать новые блюда, экспериментировать с ингредиентами и находить идеи для разнообразных приемов пищи.

Сервис выбора рецептов предоставляет ряд преимуществ как для рядовых пользователей программных продуктов, так и для прикладных программистов.

Для рядовых пользователей программных продуктов:

- расширенные возможности выбора: сервис выбора рецептов позволит пользователям получать рецепты по различным выбранным категориям и фильтрам, а также предоставит возможность удобной сортировки по различным критериям. Такой подход значительно упрощает процесс выбора рецептов.

- удобство и эффективность: пользователи смогут получать доступ к огромной базе данных рецептов из различных источников в одном месте; Это сэкономит время, которое обычно тратится на поиск рецептов в разных книгах или на различных веб-сайтах.

Для прикладных программистов:

- развитие навыков программирования: в период реализации сервиса прикладные программисты смогут развивать свои навыки в области фронтенда, бэкенда и баз данных; Это позволит им расширить свой технический набор и приобрести ценный опыт в разработке комплексных программных решений.

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		3

- развитие навыков разработки дизайна приложения: прикладные программисты смогут получить опыт в разработке дизайна приложения используя разные приемы и развивая чувство вкуса.

					ДП.0902.10.000000.00 ПЗ	Лист
						4
Изм	Лист	№ докум.	Подпись	Дата		

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

Процесс разработки программного обеспечения – набор правил, согласно которым построена разработка программного обеспечения. Приложение является клиент серверным если оно включает в себя клиент-серверную архитектуру. Разработку клиент-серверного приложения необходимо начинать с выбора архитектуры клиент-сервера.

Для разработки клиент-серверных систем имеется два подхода. Первый подход построение систем на основе двухзвенной архитектуры. Состоит из клиентской и серверной части. Как правило, серверная часть представляет собой сервер БД, на котором расположены общие данные. А клиентская часть представляет приложение, которое связывается с сервером БД, осуществляет к нему запросы и получает ответы. Такие системы используются в локальных сетях, т.к. нет затруднений с установкой клиентской части. Также системы с такой архитектурой более безопасны, т.к. могут использовать собственные протоколы передачи данных, не известные злоумышленникам. Поэтому многие крупные компании, которые располагаются не в едином месте и для соединения подразделений используют глобальную сеть Интернет, выбирают именно такую архитектуру построения клиент-серверных систем.

При разработке информационных систем, рассчитанных на широкую аудиторию, возникают проблемы с использованием двухзвенной архитектуры. Во-первых, пользователю необходимо иметь в наличии клиентскую часть, а, во-вторых, у неопытного пользователя, могут возникнуть проблемы с конфигурированием такой системы. Поэтому в последнее время, более часто разрабатывают приложения на базе трехзвенной архитектуры.

Второй подход построение систем на основе трехзвенной архитектуры. Серверная часть в этой архитектуре представляет собой сервер приложений и сервер БД. А в качестве клиента выступает web-браузер. Такая система очень проста для пользователя. Ему необходимо знать только адрес сервера приложения и наличие web-браузера на рабочем компьютере. Все данные представляются в виде

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		5

html-разметки, с использованием графики (jpeg, gif, flash) и JavaScript. Передача запросов от клиента к серверу приложений происходит по средствам CGI-интерфейса. Сервер приложений общается с сервером БД, используя другой интерфейс, зависящий от того, на основе каких средств строится конкретная информационная система. Недостатками такой архитектуры является использование общеизвестных протоколов и интерфейсов передачи данных. Злоумышленник может осуществить взлом системы, если она будет недостаточно хорошо проверять поступившие запросы от клиента.

При разработке клиент-серверных приложений необходимо учитывать:

- на каких пользователей будет рассчитана данная информационная система.
- какие требования предъявляются к безопасности.

Если информационная система должна быть общедоступной и рассчитана на широкую аудиторию, то необходимо использовать трехзвенную архитектуру.

Если информационная система используется внутри предприятия, доступ имеют к ней ограниченные пользователи и требуется создать максимально безопасную и защищенную систему, то следует отдать предпочтение двухзвенной архитектуре [1].

В рамках дипломного проекта был выбран второй способ для разработки клиент-серверной системы на основе трехзвенной архитектуры.

Для реализации трехзвенной архитектуры для клиента был выбран фреймворк Vue.js, а для сервера Express.

При выборе фреймворка Vue.js можно выделить следующие преимущества:

- простота в использовании: Vue.js обладает простым и понятным API, что делает его легким для изучения и использования даже для новичков; Он также предлагает простую и интуитивно понятную структуру компонентов, что упрощает разработку и поддержку приложений.

- гибкость и масштабируемость: Vue.js предоставляет гибкие инструменты для создания масштабируемых приложений; Он позволяет построить приложение постепенно, добавляя и настраивая компоненты по мере необходимости. Vue.js также интегрируется легко с существующими проектами и библиотеками.

- реактивность: одной из ключевых особенностей Vue.js является его реактивность; Изменение данных в модели автоматически обновляет представление, что позволяет создавать динамические и отзывчивые пользовательские интерфейсы без необходимости явного обновления DOM.

- компонентный подход: Vue.js основан на компонентном подходе разработки, позволяя создавать приложения, состоящие из множества независимых и переиспользуемых компонентов; Это упрощает организацию кода, повторное использование компонентов и обеспечивает легкость сопровождения проекта.

- активное сообщество и экосистема: Vue.js имеет большое и активное сообщество разработчиков, что обеспечивает доступ к обширной базе знаний, документации и ресурсам для изучения и разработки; Также существует обширная экосистема плагинов и расширений, которые облегчают разработку и добавление новых функций в приложение.

- высокая производительность: Vue.js обладает быстрым виртуальным DOM и эффективной системой рендеринга, что обеспечивает высокую производительность и отзывчивость приложений; Оптимизации, такие как ленивая загрузка компонентов и асинхронная загрузка ресурсов, помогают улучшить время отклика и общую производительность приложения.

При выборе фреймворка Express для разработки веб-приложений можно выделить следующие преимущества:

- минималистичность и гибкость: Express является минималистичным фреймворком, который предоставляет только основные инструменты для создания веб-приложений; Он не навязывает жестких правил и позволяет разработчикам гибко определять структуру и логику своих приложений; Это особенно полезно для проектов, требующих высокой степени настраиваемости.

					ДП.0902.10.000000.00 ПЗ	Лист
						7
Изм	Лист	№ докум.	Подпись	Дата		

- удобство и простота использования: Express имеет простой и интуитивно понятный API, что делает его отличным выбором для начинающих разработчиков или тех, кто хочет быстро создать прототип или маленькое веб-приложение; Он предоставляет лаконичные методы и функции для маршрутизации, обработки запросов и ответов, упрощая разработку.

- расширяемость и поддержка сторонних модулей: Express является популярным и широко используемым фреймворком, что приводит к наличию обширной экосистемы модулей и плагинов; Разработчики могут использовать сторонние модули для добавления дополнительных функций, таких как аутентификация, сессии, обработка файлов и многое другое, что значительно упрощает разработку и расширение функциональности приложения.

- мощная система маршрутизации: Express предоставляет гибкую и мощную систему маршрутизации, которая позволяет определить обработчики для различных маршрутов и HTTP-методов; Это облегчает организацию и обработку запросов, а также позволяет создавать RESTful API с легкостью.

- обширная документация и сообщество: Express имеет обширную и хорошо документированную базу знаний, которая облегчает изучение и использование фреймворка; Кроме того, есть активное сообщество разработчиков, которые готовы помочь и поделиться опытом; Это позволяет быстро находить ответы на вопросы и решения проблем во время разработки.

При выборе сред разработки были рассмотрены Visual Studio Code и WebStorm. Visual Studio Code (VS Code) - это бесплатная и открытая среда разработки, разработанная компанией Microsoft; Она предоставляет разработчикам широкий набор инструментов для написания кода, отладки, управления версиями и других задач разработки; Вот некоторые особенности и причины выбора Visual Studio Code:

- множество расширений: VS Code имеет обширную экосистему расширений, которые позволяют настраивать среду разработки под свои нужды и до-

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		8

бавлять новые функции; Расширения доступны для различных языков программирования, фреймворков и инструментов разработки, что делает VS Code гибким и мощным инструментом для разработчиков.

- легковесность и производительность: одним из главных преимуществ VS Code является его легковесность; Он потребляет меньше ресурсов системы по сравнению с некоторыми другими средами разработки, что позволяет ему работать быстро и эффективно, даже на старых или медленных компьютерах.

- кроссплатформенность: VS Code доступен для Windows, macOS и Linux, что обеспечивает возможность работы на различных операционных системах; Это особенно полезно для команд разработчиков, где каждый может использовать предпочитаемую платформу без проблем совместимости.

- интеграция с Git: VS Code предоставляет интегрированные инструменты для работы с системой контроля версий Git; Разработчики могут легко выполнять коммиты, слияния, ветвления и просматривать изменения в своем проекте, не покидая среду разработки.

WebStorm - это интегрированная среда разработки (IDE) для веб-разработки, разработанная компанией JetBrains. Она предоставляет разработчикам широкий набор инструментов для эффективного написания кода, отладки, тестирования и управления проектами; Вот некоторые особенности и причины выбора WebStorm:

- поддержка различных языков и фреймворков: WebStorm обладает мощной интеграцией с различными языками программирования и фреймворками, включая JavaScript, TypeScript, HTML, CSS, Angular, React, Vue.js и многие другие; Это обеспечивает разработчикам удобство и продуктивность при работе с разными технологиями.

- интеллектуальные инструменты и автоматическое дополнение кода: WebStorm предоставляет широкий набор инструментов, которые помогают разработчикам писать код быстрее и без ошибок; Это включает автоматическое допол-

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		9

нение кода, быструю навигацию по проекту, статический анализ кода, рефакторинг и другие инструменты, которые повышают производительность и качество кода.

- отладка и тестирование: WebStorm предоставляет интегрированные инструменты для отладки и тестирования кода; Разработчики могут удобно выполнять отладку кода, наблюдать за значением переменных, устанавливать точки останова и тестировать свои приложения прямо из среды разработки.

- интеграция с системами контроля версий: WebStorm интегрируется с популярными системами контроля версий, такими как Git, SVN и Mercurial; Разработчики могут легко выполнять операции коммита, слияния, обновления и просматривать историю изменений прямо из среды разработки.

- удобная настройка проекта: WebStorm позволяет разработчикам удобно настраивать и управлять проектом; Он поддерживает различные типы проектов и предоставляет возможность настройки среды разработки под индивидуальные потребности разработчика.

Из этих двух сред разработки был выбран Visual Studio Code, так как он является очень гибкой средой разработки, позволяющей разработчикам настраивать ее под свои потребности. Он предоставляет широкий выбор расширений и настроек, позволяющих настроить среду разработки под конкретные требования проекта или личные предпочтения разработчика.

В интернете существует аналог разрабатываемого приложения [2], у которого есть ряд ключевых отличий. Аналог обладает устаревшим дизайном, что может отталкивать некоторых пользователей. Также у аналога отсутствует возможности добавления видео с рецептом, и он является обычным сайтом. Разрабатываемый проект будет веб-приложением, что даёт ему преимущество в удобстве использовании при навигации. Можно отметить хороший функционал у аналога, например, в отличии от разрабатываемого приложения у аналога есть возможность увидеть список ингредиентов и пищевую/энергетическую ценность продуктов.

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		10

Приложение должно предоставлять возможность пользователю найти интересные рецепты максимально удобным образом с помощью фильтров и сортировки, а также, если пользователь авторизован, добавлять свои собственные, что позволит сервису развиваться быстрее. У авторизованного пользователя также будет возможность добавлять рецепты в избранные и просматривать их в личном кабинете. Пользователи с повышенными правами доступа будут иметь возможность редактировать все существующие рецепты или получать доступ к панели администратора.

Основным процессом будет выборка записей с рецептами из базы данных. Выборка будет происходить в зависимости от выбранных фильтров и сортировки данных. Для экономии ресурсов, данные будут разбиты на части с определенным количеством.

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		11

2 КОНСТРУКТОРСКИЙ РАЗДЕЛ

2.1 Проектирование информационной модели данных

Черная сфера представляет собой систему, внутреннее устройство которой не важно. В эту систему подаются входные данные, а на выходе из системы поступают выходные данные. Черная сфера представлена на рисунке 2.1.



Рисунок 2.1 – Модель «Черная сфера»

Представим наше приложение в виде черной сферы. В приложении будут присутствовать такие входные данные, как пользователь и рецепты. На выходе из приложения будет - рецепт. Пользователем будет выступать человек, планирующий найти интересующий рецепт. Рецепты – это то, с чем будет взаимодействовать пользователь для поиска рецепта. Рецепт будет результатом, полученный после взаимодействия пользователя с рецептами. Черная сфера с перечисленными входными и выходными параметрами представлена на рисунке 2.2.

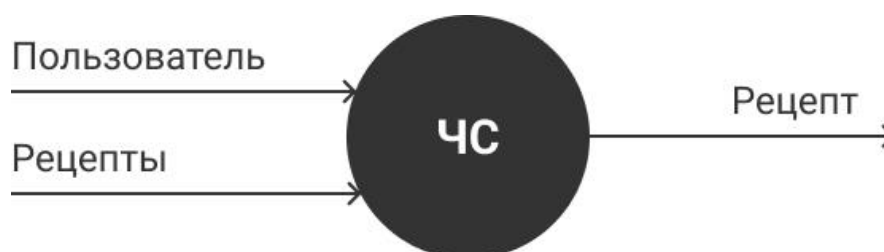


Рисунок 2.2 – Черная сфера с параметрами

В приложении существует основной процесс, отвечающий за генерацию списка рецептов. Список рецептов получается путем учета всех настроек фильтрации, сортировки, лимита и страницы. Эти данные поступают от клиента и по-

падают на сервер, создающий на основе данных запрос к базе данных и возвращающий результирующий список рецептов с необходимыми данными для отображения и взаимодействия с ними.

2.2 Проектирование серверной части приложения

2.2.1 Разработка схемы базы данных

Для выявления всех возможных сущностей будущей базы и получения концептуальной модели данных будет проведено несколько серий нормализации.

На первом этапе нормализации можно представить модель как связь между клиентом и рецептами. Первый этап нормализации представлен на рисунке 2.3.

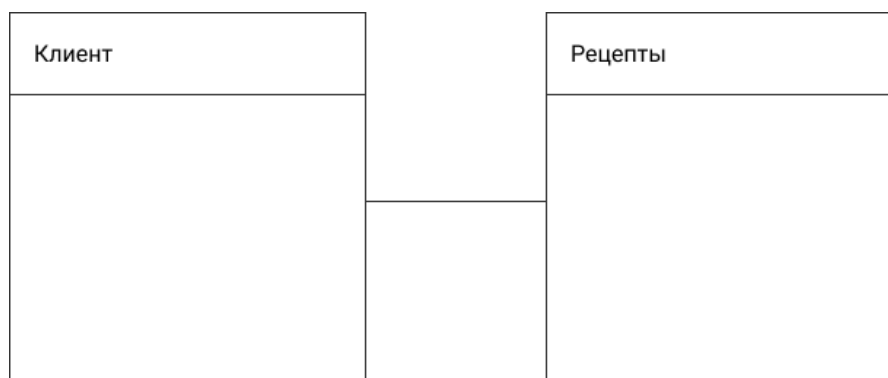


Рисунок 2.3 – Первый этап нормализации

Во втором этапе нормализации разобьём сущность клиент на 3 сущности: Пользователи, Роли и рефрешь токены. Сущность рецепты будет разбита на 2 сущности: Рецепты и категории. Второй этап нормализации на рисунке 2.4.

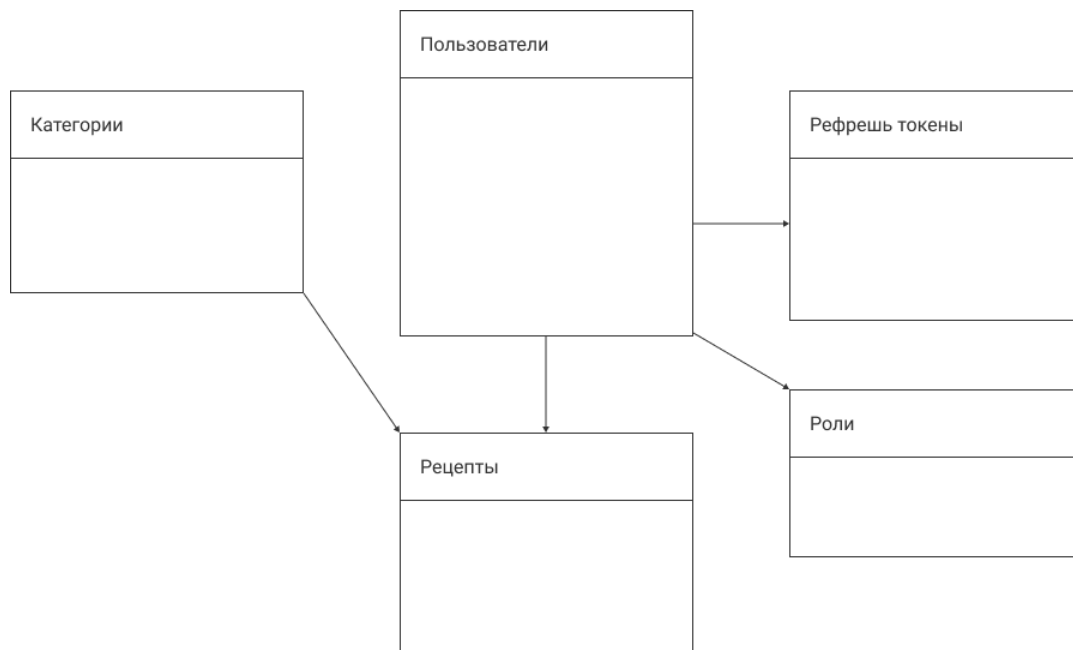


Рисунок 2.4 – Второй этап нормализации

В третьем заключительном этапе нормализации получим полную концептуальную схему разбив сущность категорий на две сущности категории и группы категорий, а сущность рецептов на рецепты и понравившиеся рецепты. Связь рецептов и категорий будет проходить через сущность Рецепты категории. Третий этап нормализации представлена на рисунке 2.5.

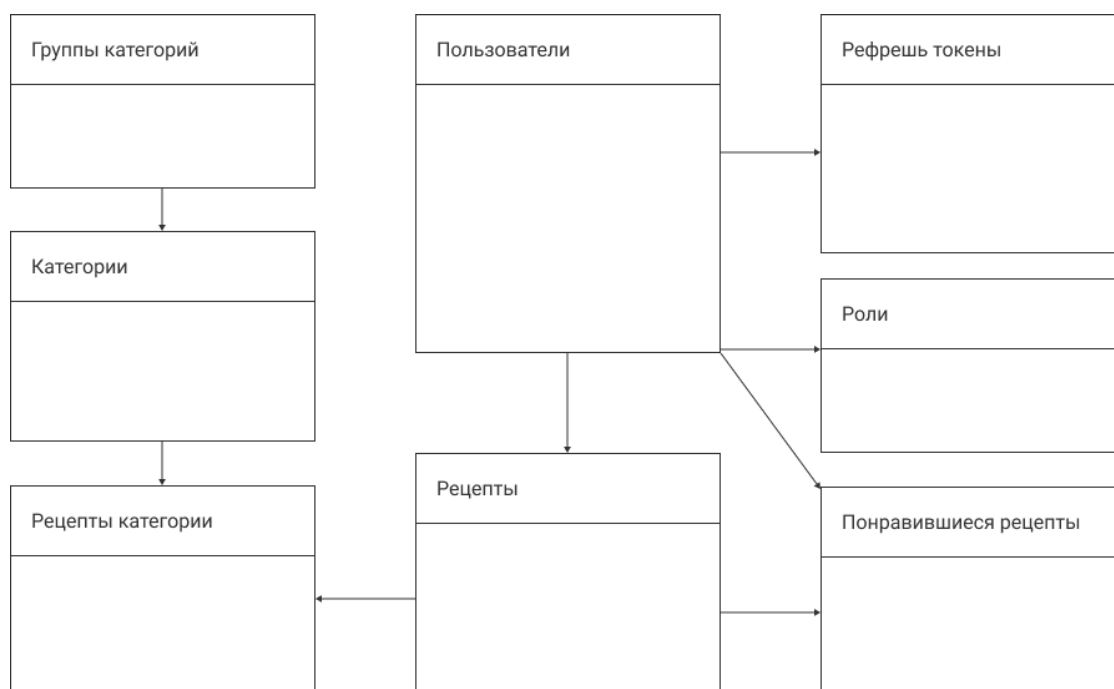


Рисунок 2.5 – Третий этап нормализации

Получим логическую модель данных с содержанием всех сущностей, связей и атрибутов данных. Логическая модель данных представлена на рисунке 2.6.

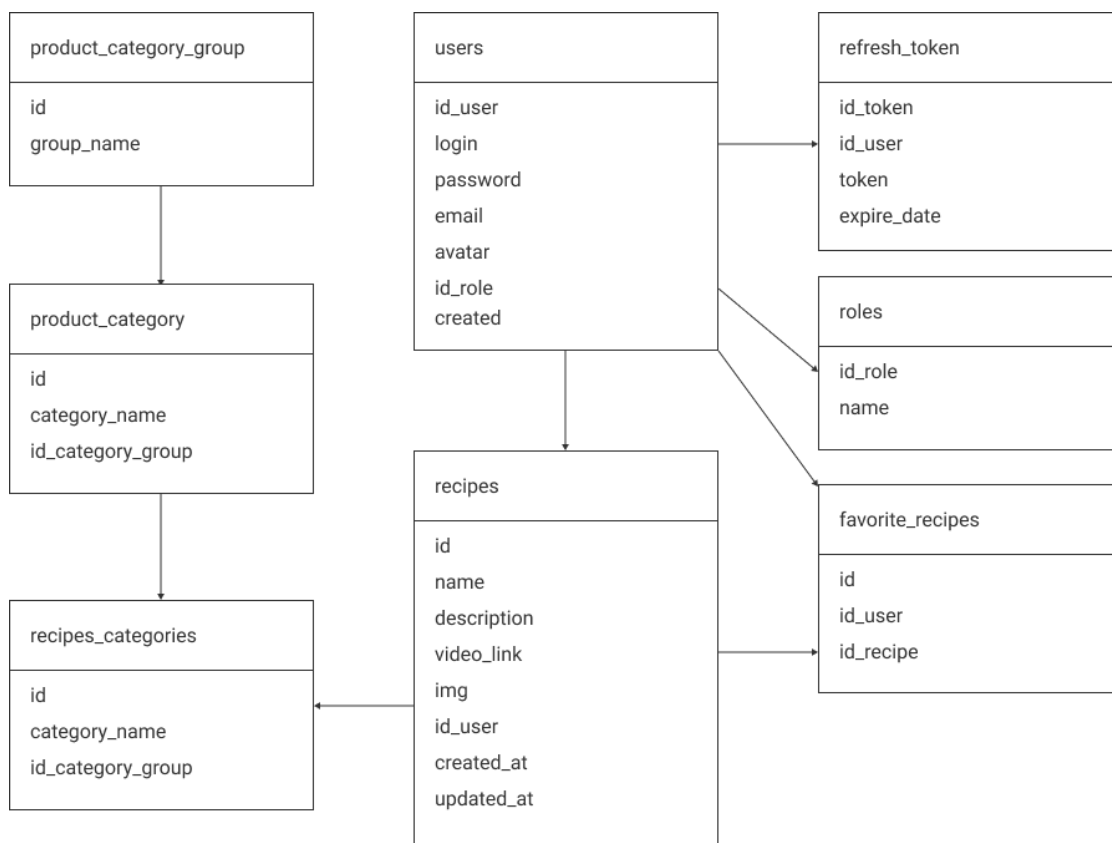


Рисунок 2.6 – Логическая модель данных

Получим физическую модель данных, включающая ассоциативные таблицы, которые иллюстрируют отношения между сущностями, а также первичные и внешние ключи для связи данных. Физическая модель данных представлена на рисунке 2.7.

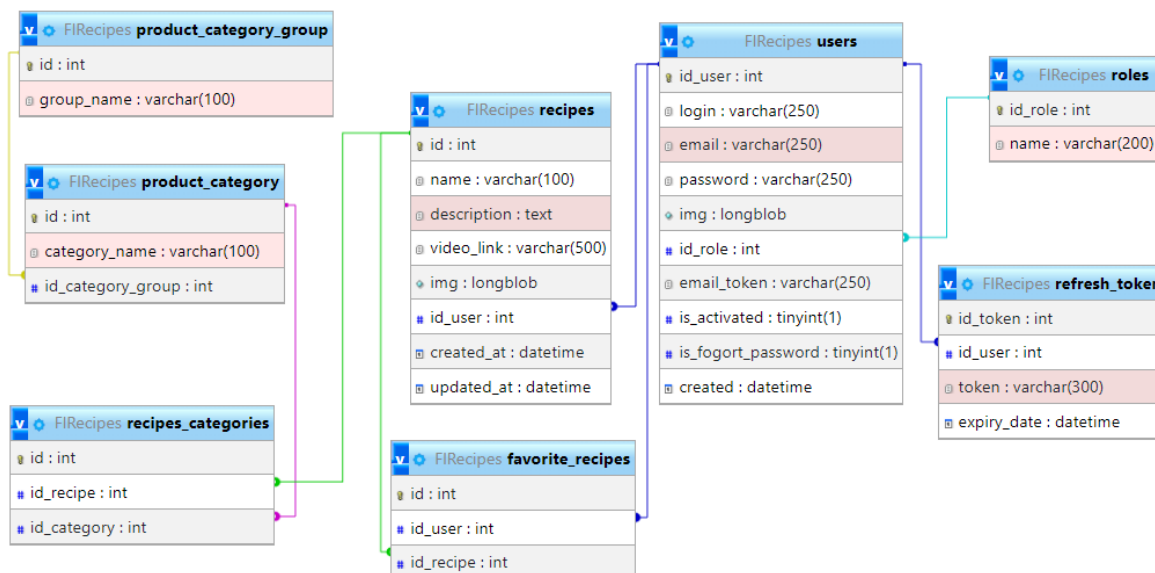


Рисунок 2.7 – Физическая модель данных

2.2.2 Разработка структуры сущностей базы данных

В результатах раздела «Разработка схемы базы данных» получена схема базы данных, из которой следует необходимость присутствия определенных сущностей необходимых для полноценной работы приложения. Для удобства все сущности сведены в табличном виде. Сущности схемы базы данных представлены в таблице 2.1.

Таблица 2.1 – Сущности схемы базы данных

Имя сущности	Назначение сущности	Типы данных	Перечисление наименований сущностей, которые подчиняются текущей сущности	Перечисление наименований сущностей, которым подчиняется текущая сущность
Users	Содержит данные о пользователях	int, varchar(250), tinyint, longblob, datetime	Roles, Refresh_token, Recipes, Favorite_recipes	-
Roles	Содержит данные о ролях	int, varchar(200)		Users
Refresh_token	Содержит данные о токенах	int, varchar(250), datetime		Users
Recipes	Содержит данные о рецептах	int, varchar(100), varchar(500), text, longblob, datetime	Favorite_recipes, Recipes_categories	Users
Favorite_recipes	Содержит данные о понравившихся рецептах	int, date, time(7)		Users, Recipes
Product_category_group	Содержит данные о группах категорий	int, varchar(100)	Product_category	
Product_category	Содержит данные о категориях	int, varchar(100)	Recipes_categories	Product_category_group
Recipes_categories	Содержит данные о связях категорий с рецептами	int		Product_category, Recipes

2.3 Проектирование клиентской части приложения

2.3.1 Разработка модульной схемы

Vue представляет собой инновационный фреймворк для разработки веб-приложений. Встраиваясь в парадигму реактивного программирования, Vue предлагает разработчикам эффективный и гибкий способ создания мощных пользовательских интерфейсов.

Основные особенности построения приложений на Vue включают:

- Composition API: Vue представляет Composition API, которая предоставляет гибкий и масштабируемый способ организации компонентов; Она позволяет разделять логику на более мелкие функциональные блоки, называемые композициями, что упрощает повторное использование кода и облегчает его понимание и тестирование.

- повышенная производительность: Vue внедряет новый реактивный движок, который значительно повышает производительность веб-приложений; С использованием Proxy API, Vue обеспечивает эффективное отслеживание изменений и реагирование на них, что улучшает общую отзывчивость приложения.

- модульность: Vue предлагает модульную структуру, которая позволяет использовать только необходимые функциональные возможности фреймворка; Это помогает сократить размер окончательного бандла и улучшить время загрузки приложения.

Эти особенности делают Vue мощным инструментом для разработки веб-приложений, обеспечивая гибкость, производительность и удобство использования. Разработчики могут использовать Vue для создания современных и интерактивных пользовательских интерфейсов, повышая качество и эффективность своих проектов.

Представим клиентскую часть приложения в виде модульной схемы показывающая связь между окнами, классами и страницами при организации клиентской части приложения. Модульная схема клиентской части приложения представлена на рисунке 2.8.



Рисунок 2.8 – Модульная схема клиентской части приложения

В составе модульной схемы присутствуют следующие элементы:

- главная страница, первая страница при запуске приложения.
- меню, вызывается на всех страницах, предназначено для навигации по приложению.
- панель администратора, страница для редактирования записей групп категорий, категорий и пользователей.
- страница «о нас», содержит информацию о создателе приложения.
- страница рецептов, содержит рецепты по выбранным фильтрам, сортировки и страницам.
- страница добавления рецепта, содержит необходимые поля для добавления нового рецепта.
- страница рецепта, содержит данные об рецепте.

- страница редактирования рецепта, содержит необходимые поля для редактирования рецепта.
- страница личного кабинета, содержит информацию об пользователе, а также об добавленных и избранных рецептах.
- модальное окно регистрации, окно для регистрации пользователя.
- модальное окно авторизации, окно для авторизации пользователя.
- страница восстановления пароля, необходимо для отправки письма для восстановления пароля пользователя.
- страница изменения пароля, необходима для создания нового пароля пользователя.
- страница 404, отображается при переходе по несуществующему пути.

2.3.2 Разработка пользовательского интерфейса

Пользовательский интерфейс – это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы программной системы.

Графический пользовательский интерфейс (Graphical User Interface или GUI). Самый популярный тип UI. Представляет собой окошко с различными элементами управления. Пользователи взаимодействуют с ними с помощью клавиатуры, мыши и голосовых команд: нажимают на кнопки, кликают мышкой, смахивают пальцами.

HTML (HyperText Markup Language) — это язык разметки, который используется для создания и структурирования содержимого веб-страниц. Он представляет собой основу для построения веб-страниц и веб-приложений, определяя структуру, семантику и визуальное представление контента. HTML является основным языком для разработки веб-страниц и тесно связан с другими технологиями, такими как CSS и JavaScript. Современные версии HTML (например,

					ДП.0902.10.000000.00 ПЗ	Лист
						20
Изм	Лист	№ докум.	Подпись	Дата		

HTML5) предлагают расширенные возможности и элементы, которые улучшают функциональность и взаимодействие на веб-страницах.

Главная страница, содержит только информационный контент без организации ввода и вывода информации и элементов управления. Пример блока информации представлен на рисунке 2.9.

```
<div class="info-cards__item info-cards__item--reverse">
  <div class="info-cards__img-wrapper">
    
  </div>
  <div class="info-cards__content-wrapper">
    <div class="info-cards__content">
      <h3>Рецепты</h3>
      <p>
        Выбирайте рецепты по интересующим категориям. У вас полная
        свобода действий во время добавления рецептуры. Делитесь
        рецептами в социальных сетях.
      </p>
    </div>
  </div>
</div>
```

Рисунок 2.9 – Пример блока информации

Меню – это компонент, содержащий в себе ссылки для навигации по приложению. Навигация выполнена в виде списка в зависимости от роли пользователя есть ограничения по доступу в некоторых ссылках. Меню вызывается на всех страницах, что позволяет организовывать навигацию по приложению из любого места. Разметка меню представлена на рисунке 2.10.

```

<template>
<div class="side-menu" v-if="show" @click.stop="hideMenu">
  <div @click.stop class="side-menu__content">
    <p class="logo"><span>LF</span>Resipes</p>
    <nav class="side-menu__nav">
      <ul class="side-menu__elements">
        <li class="side-menu__el">
          <button class="side-menu__button" @click="goByRoute('/')">
            Приветствие
          </button>
        </li>
        <li class="side-menu__el">
          <button class="side-menu__button" @click="goByRoute('/recipes')">
            Рецепты
          </button>
        </li>
        <li class="side-menu__el" v-if="userData.user">
          <button class="side-menu__button" @click="goByRoute('/add-recipe')">
            Добавить рецепт
          </button>
        </li>
        <li class="side-menu__el">
          <button class="side-menu__button" @click="goByRoute('/about')">
            О нас
          </button>
        </li>
        <li class="side-menu__el" v-if="userData.user?.role === 'admin'">
          <button class="side-menu__button" @click="goByRoute('/admin')">
            Панель администратора
          </button>
        </li>
      </ul>
    </nav>
  </div>
</div>
</template>

```

Рисунок 2.10 – Разметка меню

Страница панели администратора отображает навигацию по компонентам, позволяющим редактировать таблицы базы данных. Страница панели администратора представлена на рисунке 2.11.

```

<template>
  <<section class="admin">
    <<div class="container">
      <<h1>Панель администратора</h1>
      <<div class="tab-menu">
        <<button
          <<v-for="tab in tabs"
          <<:key="tab.name"
          <<@click="handlerTab(tab)"
          <<class="tab-menu__button"
          <<:class="tab.isSelected ? 'tab-menu__button--active' : ''"
          <>
            <<{{ tab.name }}
          <</button>
        <</div>
        <<component :is="selectedTab.component"></component>
      <</div>
    <</section>
  </template>

```

Рисунок 2.11 – Страница панели администратора

Фрагмент компонента для редактирования базы данных представлен на рисунке 2.12.

```

<template>
<div class="categories-table">
  <template v-if="categoryGroups">
    <DataTable
      v-model:editingRows="editingRows"
      :value="categoryGroups"
      resizableColumns
      columnResizeMode="fit"
      showGridlines
      editMode="row"
      dataKey="id"
      @row-edit-save="onRowEditSave"
      scrollable
      scrollHeight="80vh"
      tableClass="editable-cells-table"
      tableStyle="min-width: 50rem"
    >
      <template #header>
        <Button
          type="button"
          icon="pi pi-plus"
          label="Добавить"
          outlined
          @click="isShowDialog = !isShowDialog"
        />
      </template>
      <Column field="group_name" header="Group name">
        <template #editor="{ data, field }">
          <InputText v-model="data[field]" maxLength="100" />
        </template>
      </Column>
      <Column
        :rowEditor="true"
        style="min-width: 5rem"
        headerStyle="width: 5rem; text-align: center"
        bodyStyle="text-align:center"
      ></Column>
    </div>
  </template>
</div>

```

Рисунок 2.12 – Фрагмент компонента для редактирования базы данных

Страница «О нас» содержит только информацию о проекте и ссылки на контактные без организации ввода и вывода информации. Страница «О нас» представлена на рисунке 2.13.


```

<template>
  <<section class="about">
    <<div class="container">
      <<<h1>О нас</h1>
      <<<<p class="desc">
        <<<<Данный проект разработан и поддерживается одним человеком в качестве
        <<<<дипломной работы.
      <<<<</p>
      <<<<<div class="contact-wrapper">
        <<<<<p>Контакты для связи:</p>
        <<<<<<ul>
          <<<<<<li>Электронная почта: NiKuzfiz@yandex.ru</li>
          <<<<<<li>
            <<<<<<Мой телеграм:
            <<<<<<<a target="_blank" href="https://t.me/Nikiuz"
            <<<<<<>https://t.me/Nikiuz</a>
          <<<<<<>
        <<<<<<</li>
        <<<<<<</ul>
      <<<<<<</div>
    <<<<<<</div>
  <<<<<<</section>
</template>

```

Рисунок 2.13 – Страница «О нас»

Страница рецептов содержит фильтры по категориям и с помощью поиска, сортировку, пагинацию и список самих рецептов. Категории, пагинация и рецепты отображаются при помощи разработанных компонентов, в шаблоне им передаются данные и изменяется видимость. Фрагмент с содержимым представлен на рисунке 2.14.

```

<form @submit.prevent class="form-filters">
  <input v-model="searchInput" type="text" placeholder="Поиск..." />
  <select v-model="sortSelect">
    <option v-for="sort in sorts" :key="sort.name" :value="sort">
      {{ sort.name }}
    </option>
  </select>
  
</form>

<div class="row">
  <div class="col-md-3" v-show="isShowCategories">
    <category-filter :selectedCategories="selectedCategories" />
  </div>

  <div class="offset-md-1 col-md-8">
    <template v-if="recipes.length">
      <recipe-cards
        class="recipes-offset"
        :recipes="recipes"
      ></recipe-cards>
    </template>
    <p v-else>Рецепты не найдены...</p>
    <main-pagination
      class="pagination"
      v-model:page="page"
      :countOfRecords="countOfRecords"
      :limit="limit"
    ></main-pagination>
  </div>
</div>

```

Рисунок 2.14 – Фрагмент с содержанием

Страница добавления рецепта содержит форму с полями для добавления рецепта пользователем. Пример полей для заполнения представлен на рисунке 2.15.

```

<label for="name">Название рецепта</label>
<Field
|  <class="add-recipe-form__form-input"
|  <type="text"
|  <name="name"
|></Field>
<ErrorMessage name="name" class="add-recipe-form__error" />

<label for="video_link">Ссылка на видео с рецептом</label>
<Field
|  <class="add-recipe-form__form-input"
|  <type="text"
|  <name="video_link"
|></Field>
<ErrorMessage name="video_link" class="add-recipe-form__error" />

<label for="description">Описание</label>
<Field
|  <as="textarea"
|  <class="add-recipe-form__form-input"
|  <name="description"
|  <rows="3"
|>
|>
<ErrorMessage name="description" class="add-recipe-form__error" />

```

Рисунок 2.15 – Пример полей для заполнения представлен

Страница рецепта содержит фрейм для отображения видео, текстовые поля с информацией об рецепте и блок с кнопками для добавления/удаления из избранных, а также изменение и удаления рецепта. Страница рецепта представлена на рисунке 2.16.

```

<h1>{{ recipe.name }}</h1>
<div class="recipe__content-wrapper">
  <iframe
    width="560"
    height="315"
    :src="videLink"
    frameborder="0"
    allowfullscreen
    class="recipe__video"
  ></iframe>
  <p>{{ recipe.description }}</p>
  <p>Список категорий: {{ recipe?.categories?.join(", ") }}</p>
  <p>
    Добавлен пользователем:
    <span class="recipe__accent-content">{{ recipe.user_login }}</span>
  </p>
  <p>
    Дата добавления:
    <span class="recipe__accent-content">
      >{{
        recipe?.created_at?.slice(0, 10).split("-").reverse().join("-")
      }}
    </span>
  </p>
  <div v-if="userData.status.loggedIn" class="recipe__edit-wrapper">
    <button v-if="!isFavoriteRecipe" @click="handlerAddFavoriteRecipe">
      Добавить в избранное
    </button>
    <button v-else @click="handlerDeleteFavoriteRecipe">
      Удалить из избранного
    </button>
    <template v-if="isShowControll">
      <button @click="$router.push(`/update-recipe/${recipe.id}`)">
        Изменить
      </button>
      <button @click="handlerDeleteRecipe">Удалить</button>
    </template>
  </div>

```

Рисунок 2.16 – Страница рецепта

Страница редактирования рецепта полями напоминает страницу добавления, но благодаря привязки данных к полям, при открытии страницы, поля заполняются данными о рецепте из базы данных. Страница редактирования рецепта представлена на рисунке 2.17.

```

* <label for="name">Название рецепта</label>
* <Field
* | * v-model="recipe.name"
* | * class="add-recipe-form__form-input"
* | * type="text"
* | * name="name"
* ></Field>
* <ErrorMessage name="name" class="add-recipe-form__error" />

* <label for="video_link">Ссылка на видео с рецептом</label>
* <Field
* | * v-model="recipe.video_link"
* | * class="add-recipe-form__form-input"
* | * type="text"
* | * name="video_link"
* ></Field>
* <ErrorMessage name="video_link" class="add-recipe-form__error" />

* <label for="description">Описание</label>
* <Field
* | * v-model="recipe.description"
* | * as="textarea"
* | * class="add-recipe-form__form-input"
* | * name="description"
* | * rows="3"
* />
* <ErrorMessage name="description" class="add-recipe-form__error" />

```

Рисунок 2.17 – Страница редактирования рецепта

Страница личного кабинета содержит поля для изменения аватара и логина пользователя, а также добавленные и избранные рецепты. Отображение списков рецептов представлено на рисунке 2.18.

```

<div class="profile__recipes-wrapper">
  <div class="tab-menu">
    <button
      v-for="tab in tabs"
      :key="tab.name"
      @click="handlerTab(tab)"
      class="tab-menu__button"
      :class="tab.isSelected ? 'tab-menu__button--active' : ''"
    >
      {{ tab.name }}
    </button>
  </div>
  <template v-if="recipes.length">
    <recipe-cards
      class="recipes-offset"
      :recipes="recipes"
    ></recipe-cards>
  </template>
  <p v-else>Рецепты не найдены...</p>
  <main-pagination
    class="pagination"
    v-model:page="page"
    :countOfRecords="countOfRecords"
    :limit="limit"
  ></main-pagination>
</div>

```

Рисунок 2.18 – Отображение списков рецептов

Модальное окно регистрации содержит в себе форму регистрации пользователя, в которой есть поля для ввода логина, почты, пароля и повторения пароля, и кнопка для регистрации пользователя. Форма регистрации представлена на рисунке 2.19.

```

<template>
<Form @submit="handlerRegister" :validation-schema="schema" class="reg-form">
<h3>Регистрация</h3>
<div class="reg-form__content-wrapper">
<label for="login">Логин</label>
<Field class="reg-form__form-input" type="text" name="login"></Field>
<ErrorMessage name="login" class="reg-form__error" />
<label for="email">E-mail</label>
<Field class="reg-form__form-input" type="email" name="email"></Field>
<ErrorMessage name="email" class="reg-form__error" />
<label for="password">Пароль</label>
<Field
class="reg-form__form-input"
type="password"
name="password"
></Field>
<ErrorMessage name="password" class="reg-form__error" />
<label for="repeatPassword">Повторите пароль</label>
<Field
class="reg-form__form-input"
type="password"
name="repeatPassword"
></Field>
<ErrorMessage name="repeatPassword" class="reg-form__error" />
</div>
<p v-if="errorMessage" class="reg-form__error">{{ errorMessage }}</p>
<div v-if="successMessage" class="alert alert-success" role="alert">
{{ successMessage }}
</div>
<div class="reg-form__btn-wrapper">
<form-button :disabled="isLoading">Отправить</form-button>
</div>
</Form>
</template>

```

Рисунок 2.19 – Форма регистрации

Модальное окно авторизации содержит в себе форму авторизации, в которой находятся поля для ввода почты и пароля, и кнопка для авторизации пользователя. Также есть ссылка для восстановления пароля. Форма авторизации представлена на рисунке 2.20.

```

<template>
  <Form @submit="handleLogin" :validation-schema="schema" class="auth-form">
    <h3>Авторизация</h3>
    <div class="auth-form__content-wrapper">
      <label for="email">E-mail</label>
      <Field class="auth-form__form-input" type="text" name="email"></Field>
      <ErrorMessage name="email" class="auth-form__error" />
      <div class="auth-form__fogort-wrapper">
        <label for="password">Пароль</label>
        <button @click="handleFogortPassword">Забыли пароль?</button>
      </div>
      <Field
        class="auth-form__form-input"
        type="password"
        name="password"
      ></Field>
      <ErrorMessage name="password" class="auth-form__error" />
    </div>
    <p v-if="errorMessage" class="auth-form__error">{{ errorMessage }}</p>
    <div class="auth-form__btn-wrapper">
      <form-button :disabled="isLoading">Войти</form-button>
    </div>
  </Form>
</template>

```

Рисунок 2.20 – Форма авторизации

Страница восстановления пароля содержит в себе обязательное поле для ввода почты и кнопку для отправки письма сброса пароля на почту. Страница восстановления пароля представлена на рисунке 2.21.


```

<template>
  <section class="fogort-password">
    <div class="container">
      <h1>Сбросить пароль</h1>
      <div class="fogort-password_form-wrapper">
        <Form
          @submit="handleLogin"
          :validation-schema="schema"
          class="fogort-form"
        >
          <div class="fogort-form__content-wrapper">
            <label for="email">E-mail</label>
            <Field
              class="fogort-form__form-input"
              type="text"
              name="email"
            ></Field>
            <ErrorMessage name="email" class="fogort-form__error" />
          </div>
          <p v-if="errorMessage" class="fogort-form__error">
            {{ errorMessage }}
          </p>
          <div v-if="successMessage" class="alert alert-success" role="alert">
            {{ successMessage }}
          </div>
          <div class="fogort-form__butn-wrapper">
            <form-button :disabled="isLoading">Отправить</form-button>
          </div>
        </Form>
      </div>
    </div>
  </section>
</template>

```

Рисунок 2.21 – Страница восстановления пароля

Страница изменения пароля, содержит три поля для ввода почты, пароля и подтверждения пароля, и кнопку для изменения пароля. Форма изменения пароля представлена на рисунке 2.22.

```

<Form
  @submit="handleLogin"
  :validation-schema="schema"
  class="change-form"
>
  <div class="change-form__content-wrapper">
    <label for="email">E-mail</label>
    <Field
      class="change-form__form-input"
      type="text"
      name="email"
    ></Field>
    <ErrorMessage name="email" class="change-form__error" />
    <label for="password">Придумайте новый пароль</label>
    <Field
      class="change-form__form-input"
      type="password"
      name="password"
    ></Field>
    <ErrorMessage name="password" class="change-form__error" />
    <label for="repeatPassword">Повторите пароль</label>
    <Field
      class="change-form__form-input"
      type="password"
      name="repeatPassword"
    ></Field>
    <ErrorMessage name="repeatPassword" class="change-form__error" />
  </div>
  <p v-if="errorMessage" class="change-form__error">
    {{ errorMessage }}
  </p>
  <div class="change-form__btn-wrapper">
    <form-button :disabled="isLoading">Отправить</form-button>
  </div>
</Form>

```

Рисунок 2.22 – Форма изменения пароля

Страница 404 содержит только сообщение о переходе по несуществующему пути и ссылку на главную страницу без организации ввода и вывода информации. Страница 404 представлена на рисунке 2.23.

```

<template>      You, 3 недели назад • 28/04 ...
<section class="not-found">
  <div class="container">
    <h1>Ошибка 404</h1>
    <p class="desc">Нет информации по данному маршруту</p>
    <div class="contact-wrapper">
      <router-link to="/">Вернуться на главную страницу</router-link>
    </div>
  </div>
</section>
</template>

```

Рисунок 2.23 – Страница 404

2.3.3 Организация доступа к объектам базы данных

В приложение доступ к объектам данных выполняется благодаря REST API. REST API (Representational State Transfer Application Programming Interface) – это архитектурный стиль для построения веб-сервисов, основанных на протоколе HTTP. Он определяет набор ограничений и правил, которые позволяют создавать распределенные системы с высокой степенью масштабируемости, надежности и простоты взаимодействия.

Возьмем для примера страницу с рецептами, где есть компонент для отображения списка рецептов. Компонент для отображения списка рецептов представлен на рисунке 2.24.

```

<template>
  <div class="recipe-cards recipes-offset">
    <recipe-card
      v-for="recipe in recipes"
      :key="recipe.id"
      :recipe="recipe"
    ></recipe-card>      You, 3 недели назад
  </div>
</template>

```

Рисунок 2.24 – Компонент для отображения списка рецептов

Для получения рецептов, клиент отправляет GET запрос на сервер. На стороне сервера запрос обрабатывается в контроллере и обращается к модели. Модель

осуществляет SQL запрос к базе данных и получает данные. Схема GET запроса представлена на рисунке 2.25.

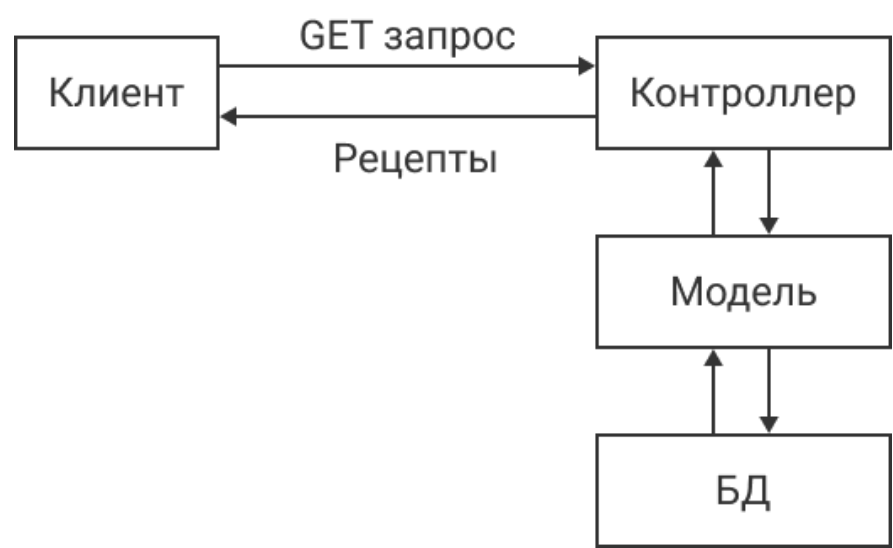


Рисунок 2.25 – Схема GET запроса

Компонент для отображения списка рецептов получает рецепты благодаря запросу к серверному API, который реализован следующим образом. Реализация запроса к серверному API представлена на рисунке 2.26.

```
· async getRecipes(data) {
·   try {
·     const result = await api.get(`/recipes`, {
·       params: {
·         limit: data.limit,
·         page: data.page,
·         name: data.name,
·         sort: data.sort,
·         filters: data.filters,
·         sortParam: data.sortParam,
·       },
·     });
·     return result;
·   } catch (error) {
·     return Promise.reject(error);
·   }
· }
```

Рисунок 2.26 – Реализация запроса к серверному API

В этой реализации выполняется GET запрос для получения рецептов передавая параметры. После данный запрос будет обрабатываться контроллером на стороне сервера. Реализация контроллера представлена на рисунке 2.27

```

async getRecipes(req, res, next) {
  try {
    const data = {
      page: req.query.page ? +req.query.page : 1,
      limit: req.query.limit ? +req.query.limit : 30,
      name: req.query.name ?? "",
      sort: req.query.sort ?? "name",
      sortParam: req.query.sortParam ?? "asc",
      filters: req.query.filters ?? "",
    };

    // Получаем рецепты по параметрам из data
    const recipes = await RecipeModel.getRecipes(data);
    // Получаем количество записей по параметрам из data
    const totalRecords = await RecipeModel.getTotalRecords(data);

    res.set("access-control-expose-headers", "X-Total-Count");
    res.set("x-total-count", totalRecords);
    return res.json(recipes);
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

```

Рисунок 2.27 – Реализация контроллера

В этом контроллере обрабатываются полученные данные с клиента и на их основе выполняем метод модели для получения рецептов и количества записей. Реализация метода модели для получения рецептов представлена на рисунке 2.28.

```

getRecipes(data) {
  return new Promise((resolve, reject) => {
    const inner = data.filters
      ? `INNER JOIN recipes_categories AS rc ON r.id = rc.id_recipe AND FIND_IN_SET(rc.id_category, "${data.filters}") > 0`
      : "";
    const where = `WHERE name LIKE "%${data.name}%"`;
    const order = `ORDER BY r.${data.sort} ${data.sortParam}`;
    const limit = `LIMIT ${data.page - 1} * data.limit, ${data.limit}`;
    db.query(
      `SELECT DISTINCT r.id, r.name, r.description, r.video_link, r.id_user, r.created_at, r.updated_at FROM recipes AS r ` +
      inner +
      where +
      order +
      limit,
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results);
        }
      }
    );
  });
}

```

Рисунок 2.28 – Реализация метода модели для получения рецептов

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		37

2.3.4 Разработка блок-схем алгоритмов процедур и функций

Основной функциональной задачей приложения является взаимодействие с рецептами. Взаимодействие происходит путем настройки фильтрации и сортировки.

Метод `getRecipes` на стороне клиента. Этот метод предназначен для получения рецептов с сервера. В качестве входных данных поступает лимит, страница, название рецепта, вид сортировки, направление сортировки, список категорий. Выходными данными выступает список рецептов. Блок-схема метода `getRecipes` на стороне клиента представлена на рисунке 2.29.

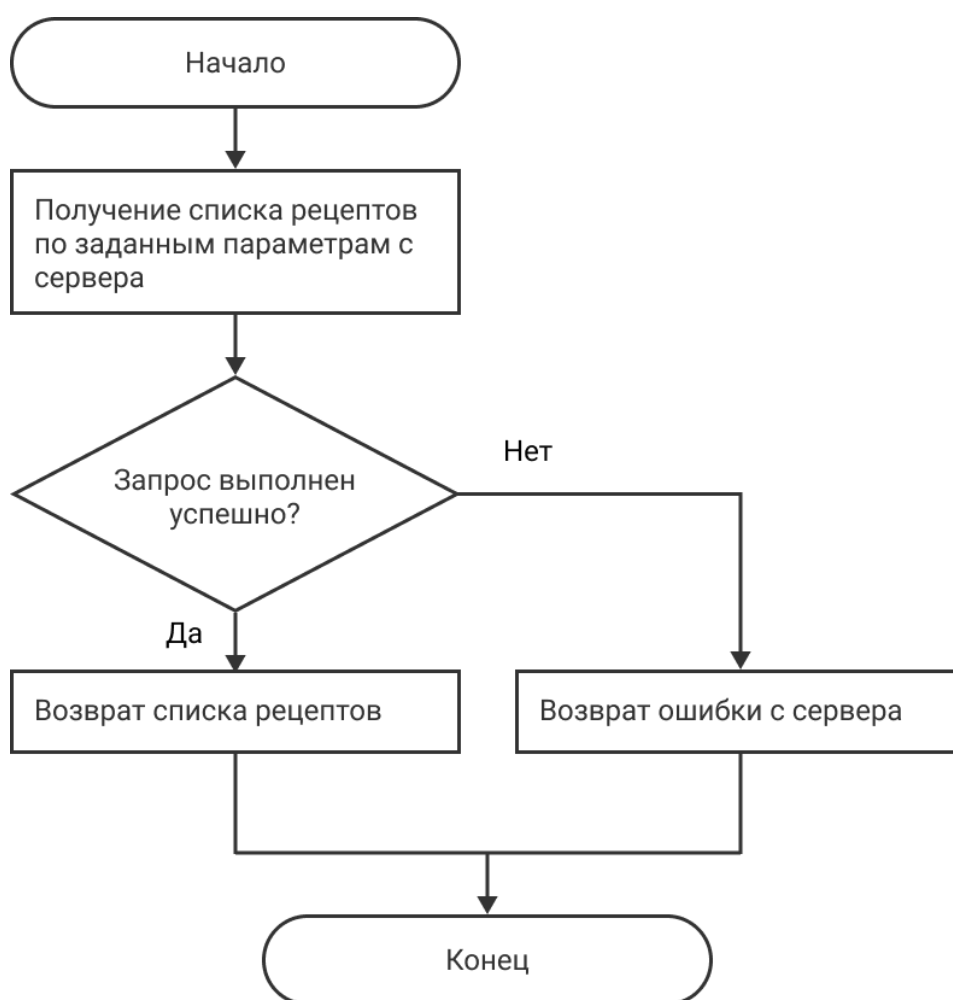


Рисунок 2.29 – Блок-схема метода `getRecipes` на стороне клиента

Метод `getRecipes` – контроллер на стороне сервера предназначен для отправки списка рецептов. Метод вызывается в методе `getRecipes` на стороне клиента. В качестве входных данных поступает лимит, страница, название рецепта,

вид сортировки, направление сортировки, список категорий. Выходными данными выступает список рецептов. Блок-схема метода контроллера getRecipes представлена на рисунке 2.30.

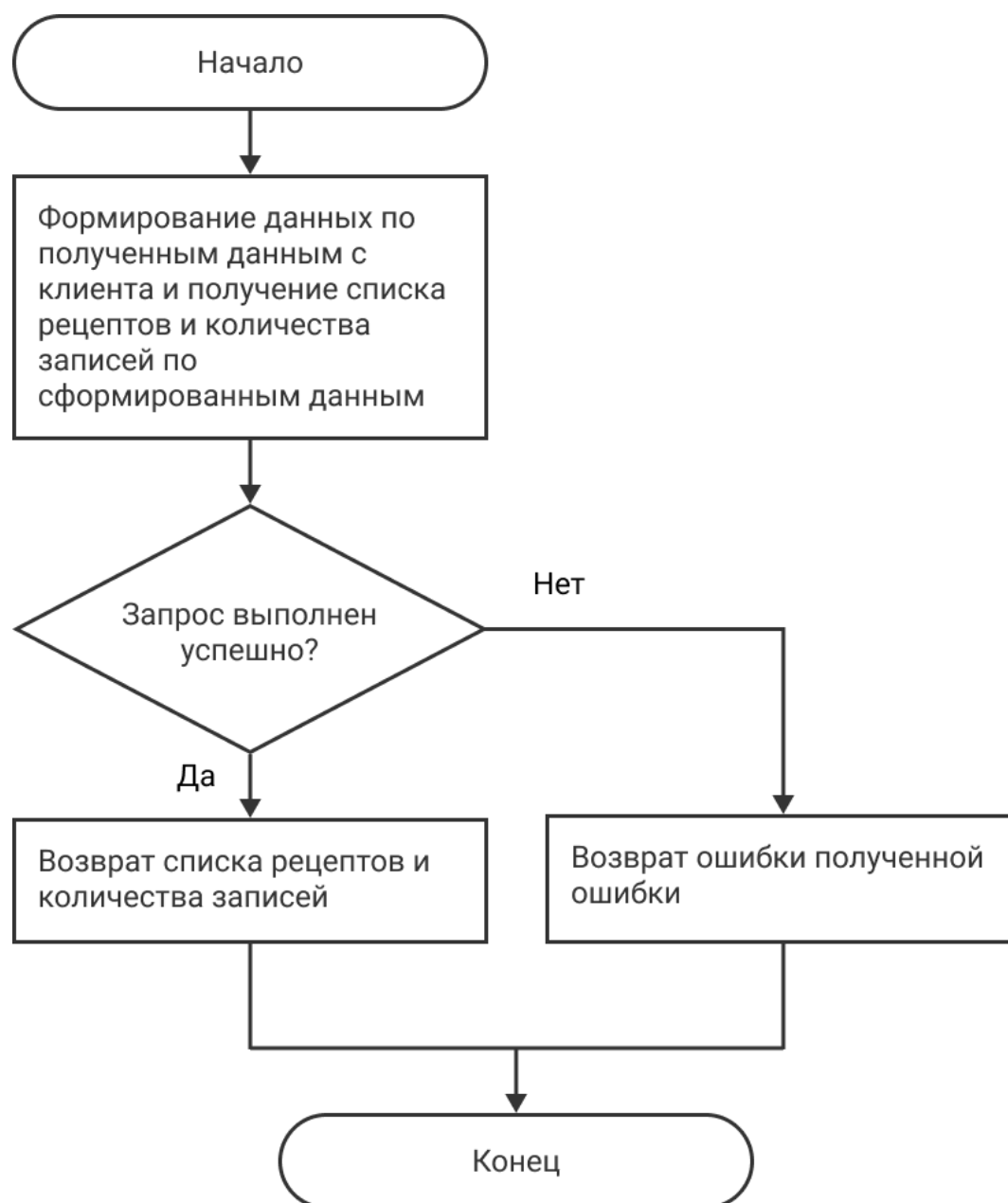


Рисунок 2.30 – Блок-схема метода контроллера getRecipes

Метод модели getRecipes предназначен для запроса к базе данных и формирования списка рецептов. Метод вызывается в методе контроллере getRecipes. В качестве входных данных поступает лимит, страница, название рецепта, вид сортировки, направление сортировки, список категорий. Выходными данными выступает сформированный список рецептов. Блок-схема метода контроллера getRecipes представлена на рисунке 2.31.

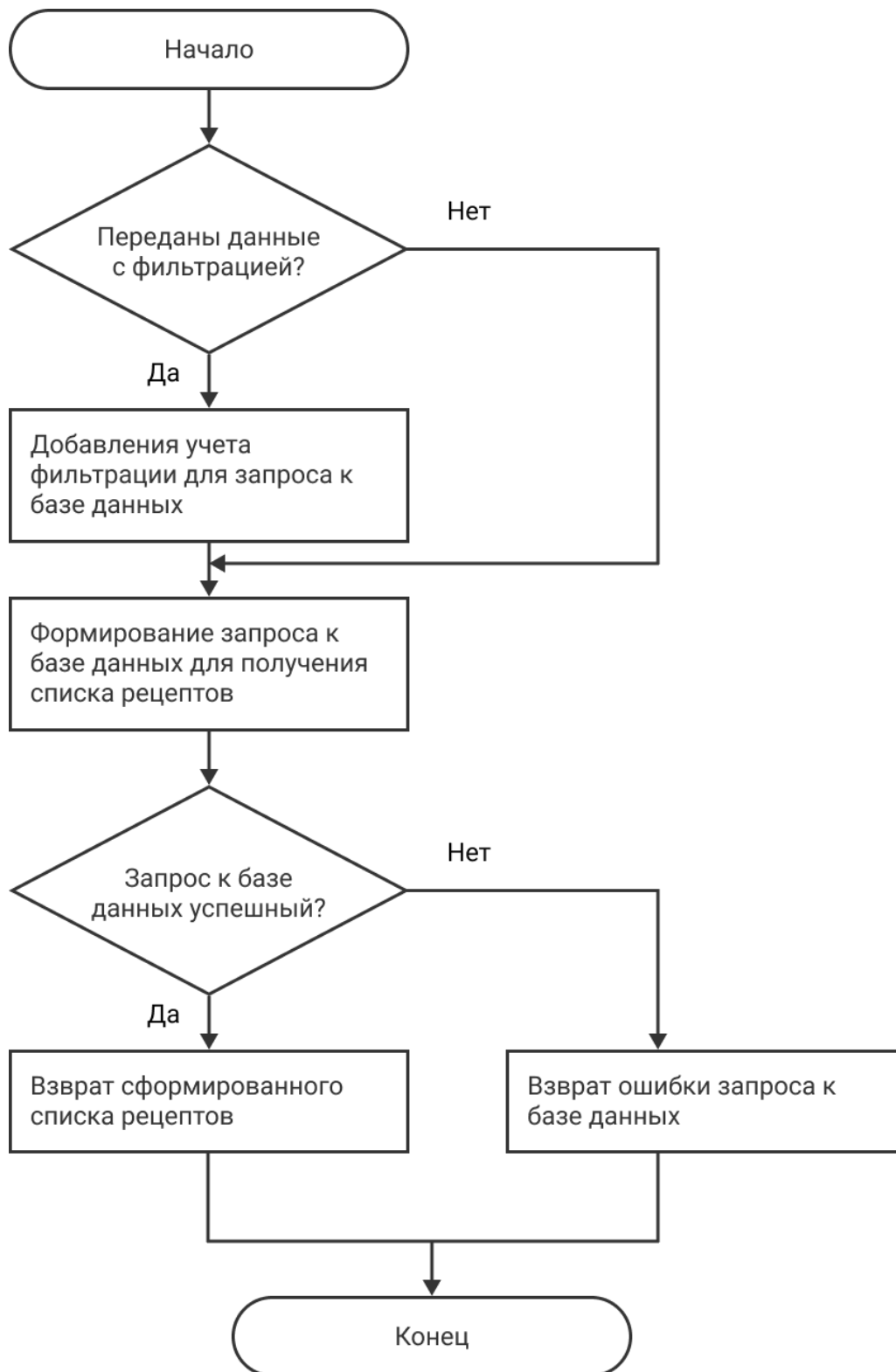


Рисунок 2.31 – Блок-схема метода модели getRecipes

На основании разработанных блок-схем был написан программный код, приведенный в Приложении А к пояснительной записке.

2.4 Обеспечение коллективного доступа. Защита информации

Основная идея ролевой модели контроля за доступом (Role-Based Access Control – RBAC) основана на максимальном приближении логики работы системы к реальному разделению функций персонала в организации.

Ролевой метод управления доступом контролирует доступ пользователей к информации на основе типов их активностей в системе. Применение данного метода подразумевает определение ролей в системе. Понятие роль можно определить, как совокупность действий и обязанностей, связанных с определенным видом деятельности. Таким образом, вместо того, чтобы указывать все типы доступа для каждого пользователя к каждому объекту, достаточно указать тип доступа к объектам для роли. А пользователям, в свою очередь, указать их роли. Пользователь, «выполняющий» роль, имеет доступ, определенный для роли [3].

В системе доступно три роли: администратор, модератор и обычный пользователь. Обычный пользователь способен управлять только своими рецептами, но также просматривать и добавлять в избранные другие рецепты. Модератор обладает всеми правами обычного пользователя за исключением возможности редактирования всех существующих рецептов. У администратора в отличие от модератора есть одно отличие – это наличие доступа к панели администратора, где можно редактировать категории и пользователей.

Для авторизации пользователю необходимо ввести почту и пароль. Авторизоваться пользователь сможет только в том случае если аккаунт был зарегистрирован в системе. На случай если пользователь забыл свой пароль, он сможет восстановить его, нажав на ссылку «Забыли пароль?». Окно авторизации представлено на рисунке 2.32.

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		41

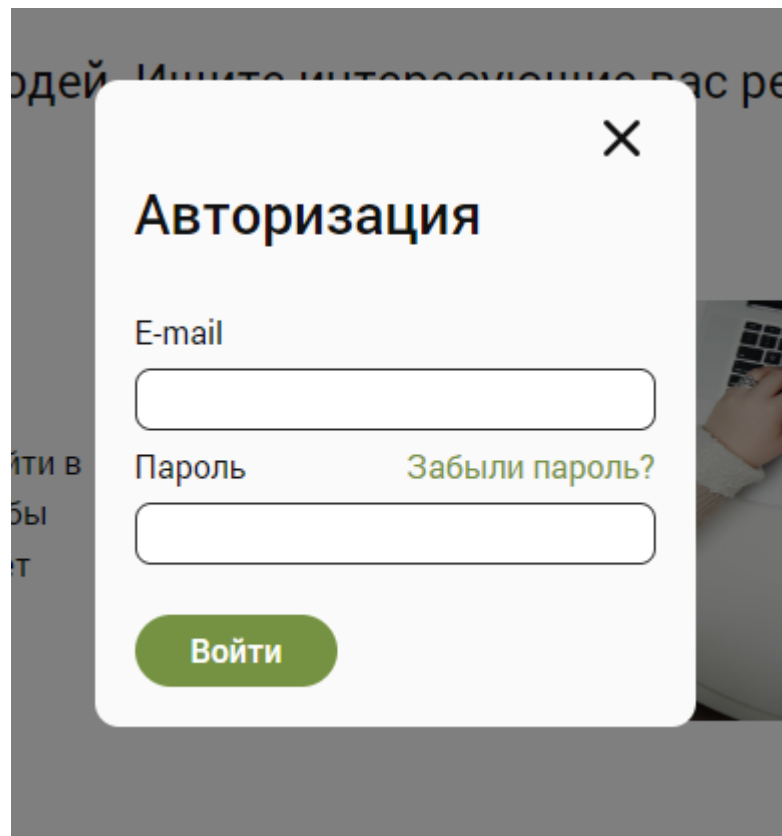


Рисунок 2.32 – Окно авторизации

Для регистрации пользователю необходимо ввести логин и почту, а также придумать и подтвердить пароль путем его повторного ввода. Все поля обладают своей валидацией и в случае некорректного ввода данных, появится подпись с предупреждением. После успешной регистрации пользователь будет оповещен о поступившем письме для подтверждения почты. Без подтверждения почты у пользователя не будет возможности авторизоваться в системе. Окно регистрации представлено на рисунке 2.33.

Регистрация

Логин

E-mail

Пароль

Повторите пароль

Отправить

Рисунок 2.33 – Окно регистрации

В случае если пользователь забыл свой пароль и хочет его восстановить он перенаправляется в окно, где необходимо ввести свою почту для оправки письма, которое необходимо для подтверждения смены пароля. Окно ввода почты представлено на рисунке 2.34.

Sign in

Sign up

Сбросить пароль

E-mail

Отправить

Copyright © 2023 LFRRecipes - рецепты от вас для вас

Рисунок 2.34 – Окно ввода почты

В письме будет ссылка, после перехода по которой произойдет перенаправление на окно изменения пароля, где пользователь сможет придумать новый пароль. Окно изменения пароля почты представлено на рисунке 2.35.

Sign in

Sign up

Изменение пароля

E-mail

Придумайте новый пароль

Повторите пароль

Отправить

Copyright © 2023 LFRRecipes - рецепты от вас для вас

Рисунок 2.35 – Окно изменения пароля

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		44

3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1 Тестирование и отладка приложения

Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки, информация из работы [4]. В связи с тем, что почти невозможно составить реальную программу без ошибок, и почти невозможно для достаточно сложной программы быстро найти и устранить все имеющиеся в ней ошибки. Разумно уже при разработке программы на этапах алгоритмизации и программирования готовиться к обнаружению ошибок на стадии отладки принимать профилактические меры по их предупреждению, информация из работы [5].

Тестирование будет происходить через тест кейсы. Тест кейс – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. При передаче тестировщику тест-кейсов, он должен пройти по всем его пунктам и выполнить описанные действия, которые должны привести к определенным результатам. Информация из работы [6]. Тест кейс для функций представлен в таблице 3.1.

Таблица 3.1 – Тест-кейс для методов

Имя метода	Управляющее воздействие	Результат воздействия
SignUp	Вызывается при POST запросе по пути /api/auth/signup	Регистрация нового пользователя в БД
Signin	Вызывается при POST запросе по пути /api/auth/signin	Возвращение необходимых данные для использования приложения авторизованным пользователем
Activate	Вызывается при GET запросе по пути /api/auth/activate/:link	Активация пользователя для возможности авторизоваться

Продолжение таблицы 3.1

Имя метода	Управляющее воздействие	Результат воздействия
RefreshToken	Вызывается при POST запросе по пути /api/auth/refresh-token	Обновление токена в БД
ForgotPassword	Вызывается при POST запросе по пути /api/forgot-password	Отправление письма для сброса пароля
IsUserForgotPassword	Вызывается при GET запросе по пути /api/forgot-password/:link	Перенаправление на страницу смены пароля
ChangeUserPassword	Вызывается при PATCH запросе по пути /api/change-password	Изменение пароля пользователя
GetRoles	Вызывается при GET запросе по пути /api/roles	Возвращение существующих ролей пользователей
GetUserWithoutAdmins	Вызывается при GET запросе по пути /api/users/without-admins	Возвращение пользователей исключая с ролью администратора
UploadImage	Вызывается при POST запросе по пути /api/user/upload-image	Обновление аватара пользователя в БД
GetImage	Вызывается при GET запросе по пути /api/user/get-image/:id	Возвращение изображения по идентификатору пользователя
UpdateUser	Вызывается при PATCH запросе по пути /api/user/:id	Обновление данных пользователя в БД
DeleteUser	Вызывается при DELETE запросе по пути /api/user/:id	Удаление пользователя из БД
AddUser	Вызывается при PUT запросе по пути /api/user/:id	Добавление нового пользователя в БД
GetCategories	Вызывается при GET запросе по пути /api/categories	Возвращение категорий
AddCategory	Вызывается при PUT запросе по пути /api/categories	Добавление категории в БД
UpdateCategory	Вызывается при PATCH запросе по пути /api/categories/:id	Обновление категории в БД

Продолжение таблицы 3.1

Имя метода	Управляющее воздействие	Результат воздействия
DeleteCategory	Вызывается при DELETE запросе по пути /api/categories/:id	Удаление категории из БД
GetCategoryGroups	Вызывается при GET запросе по пути /api/category-groups	Возвращение группы категорий
AddCategoryGroup	Вызывается при PUT запросе по пути /api/category-groups/:id	Добавление группы категорий в БД
UpdateCategoryGroup	Вызывается при PATCH запросе по пути /api/category-groups/:id	Обновление группы категорий в БД
DeleteCategoryGroup	Вызывается при DELETE запросе по пути /api/category-groups/:id	Удаление группы категорий из БД
AddRecipe	Вызывается при POST запросе по пути /api/recipes	Добавление рецепта в базу данных
GetRecipes	Вызывается при GET запросе по пути /api/recipes	Возвращение рецептов по заданным параметрам
GetUserRecipes	Вызывается при GET запросе по пути /api/recipes/user-recipes	Возвращение рецептов добавленных определенным пользователем
GetUserFavoriteRecipes	Вызывается при GET запросе по пути /api/recipes/user-favorite-recipes	Возвращение избранных рецептов пользователя
GetRecipeCategories	Вызывается при GET запросе по пути /api/recipes/categories	Возвращение категорий рецепта
GetImage	Вызывается при GET запросе по пути /api/recipes/get-image/:id	Возвращение изображения рецепта
GetRecipeById	Вызывается при GET запросе по пути /api/recipe/:id	Возвращение рецепта по его идентификатору
DeleteRecipe	Вызывается при DELETE запросе по пути /api/recipe/:recipe/:user	Удаление рецепта по идентификатору рецепта и пользователя

Окончание таблицы 3.1

Имя метода	Управляющее воздействие	Результат воздействия
UpdateRecipe	Вызывается при PATCH запросе по пути /api/recipe/:recipe/:user	Обновление рецепта по идентификатору рецепта и пользователя
AddFavoriteRecipe	Вызывается при POST запросе по пути /api/recipes/favorite-recipe	Добавление рецепта в избранные
GetFavoriteRecipes	Вызывается при GET запросе по пути /api/recipes/favorite-recipe/:id	Возвращение избранных рецептов по идентификатору пользователя
DeleteFavoriteRecipe	Вызывается при DELETE запросе по пути /api/recipes/favorite-recipe/:user/:recipe	Удаление рецепта из избранных по идентификатору пользователя и рецепта
CheckDuplicateLoginOrEmail	Вызывается при POST запросе по пути /api/auth/signup	Возвращение ошибки в случае дубликата почты
CheckRoleExisted	Вызывается при POST запросе по пути /api/auth/signup	Возвращение ошибки в случае указания несуществующей роли
VerifyToken	Вызывается в роутах требующих авторизации пользователя	Возвращение ошибки в случае, если пользователь не авторизован
IsAdmin	Вызывается в роутах требующих прав администратора	Возвращение ошибки в случае, если пользователь не обладает правами администратора
VerifyRecipeChanged	Вызывается в роутах, требующих прав администратора либо модератора либо пользователя-создателя рецепта	Возвращение ошибки в случае, если пользователь не обладает правами модератора либо администратора либо пользователя-создателя рецепта

3.2 Инструкция администратора базы данных

Перед началом работы с приложением необходимо установить и настроить Openserver, в которой отдельным компонентом необходимо установить phpMyAdmin - это бесплатная и открытая система управления базами данных (СУБД), написанная на языке PHP и предназначенная для управления базами данных MySQL.

Для установки необходимо перейти на официальный сайт Ospanel и скачать дистрибутив Openserver. Далее нужно запустить установщик где в поле «Папка назначения» задать путь, куда будет установлен локальный сервер, после нужно начать извлечение при помощи нажатия кнопки «Извлечь». Как на рисунке 3.1.

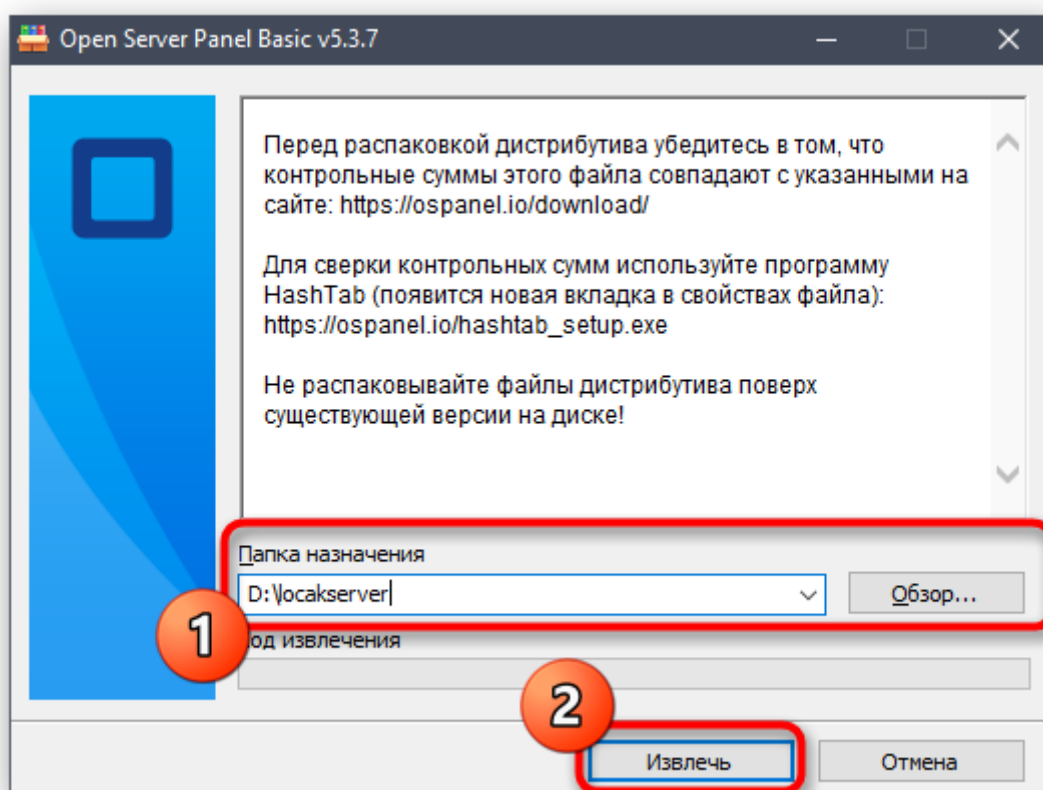


Рисунок 3.1 – Указание папки назначения

После окончания извлечения нужно выполнить переход по пути установки программы, где нужно выполнить запуск находящегося там исполняемого файла.

После чего начнется процесс загрузки. Изображение папки с исполняемым файлом представлено на рисунке 3.2.

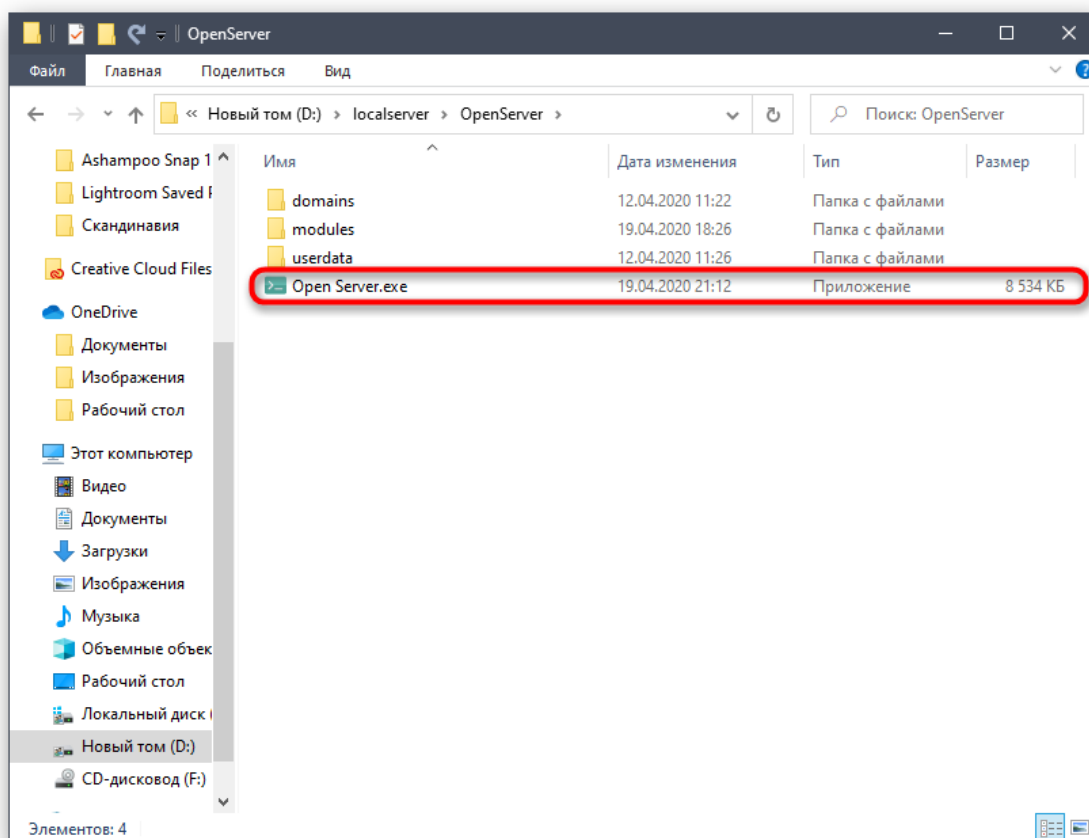


Рисунок 3.2 – Изображение папки с исполняемым файлом
Выбор удобного языка интерфейса, как показано на рисунке 3.3.

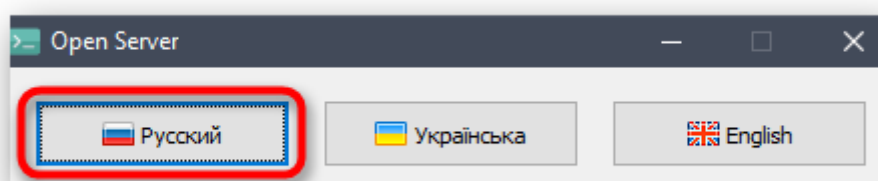


Рисунок 3.3 – Выбор языка интерфейса

Если будет уведомление, что запуск осуществляется впервые, значит нужно установить Microsoft Visual C++. Операция обязательна к подтверждению. Рисунок 3.4.

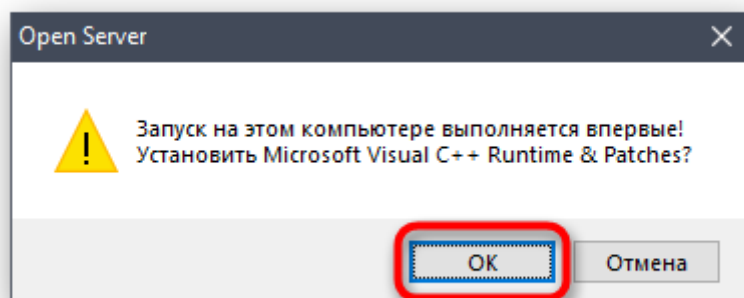


Рисунок 3.4 – Установка Microsoft Visual C++

После необходимо выполнить перезагрузку компьютера и запустить Open-server, после открыть меню путем нажатия на значок Openserve. В появившемся меню перейти в настройки. Изображения меню Openserver представлено на рисунке 3.5.

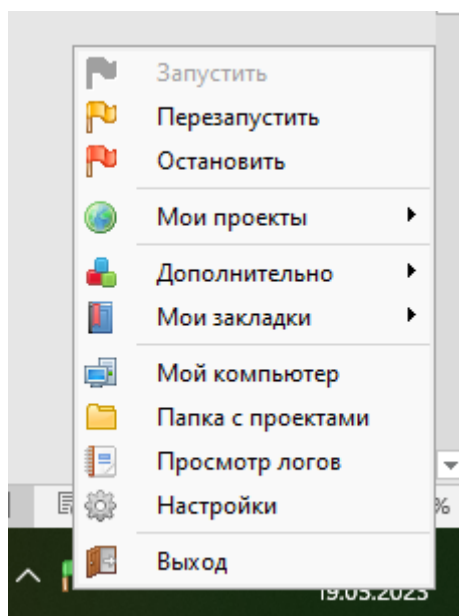


Рисунок 3.5 – Изображения меню Openserver

В настройках в разделе «Модули», выставляются настройки HTTP, PHP и MySQL, как показано на рисунке 3.6.

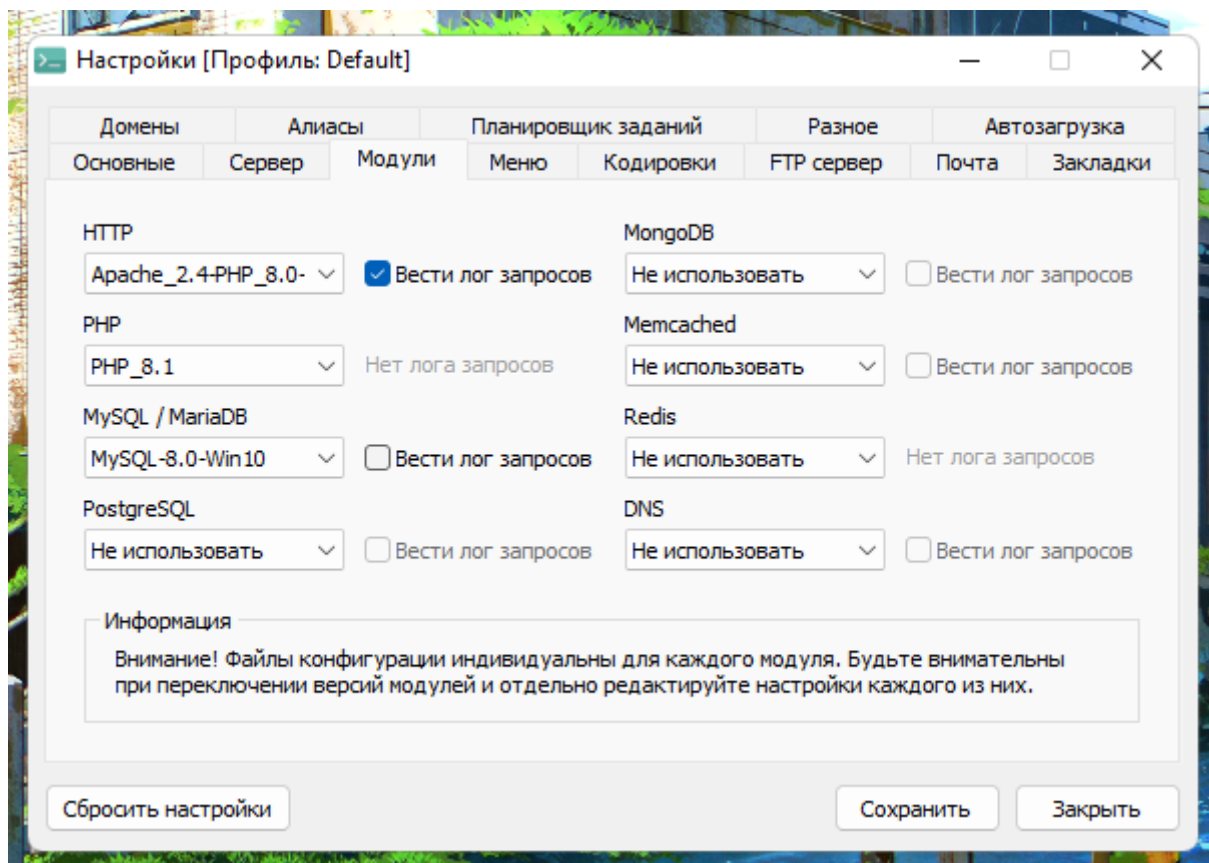


Рисунок 3.6 – Настройки OpenServer

В меню OpenServer во вкладке «Дополнительно» нужно запустить «PhpMyAdmin», как показано на рисунке 3.7.

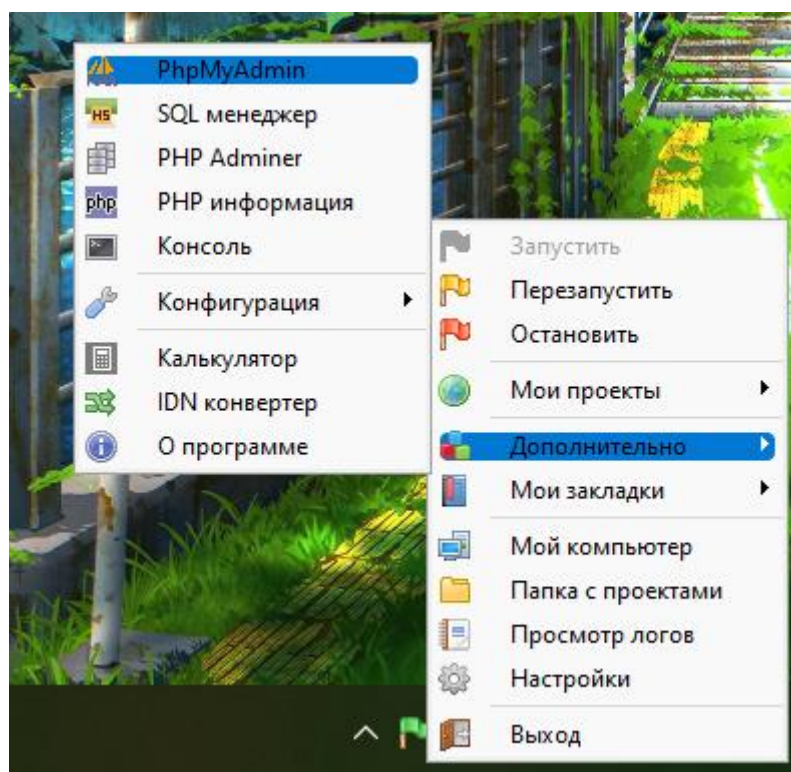


Рисунок 3.7 – Запуск PhpMyAdmin

Выполнится переход на окно авторизации, где необходимо нажать на кнопку «Авторизация», как показано на рисунке 3.8.

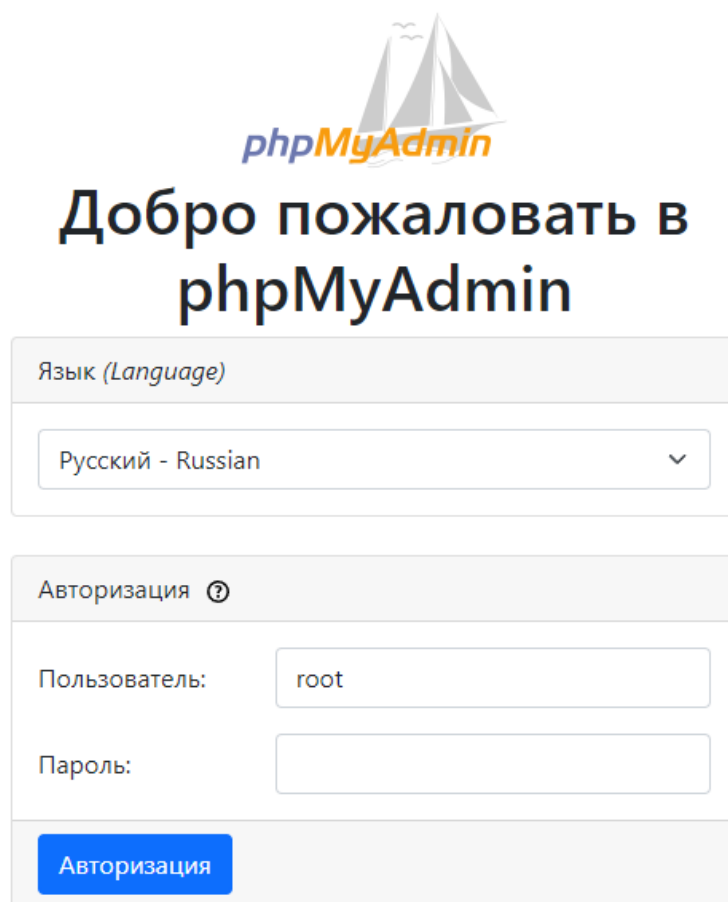


Рисунок 3.8 – Окно авторизации

После авторизации откроется главное окно СУБД, где можно начать администрировать базу данных. Главное окно СУБД представлено на рисунке 3.9.

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		53

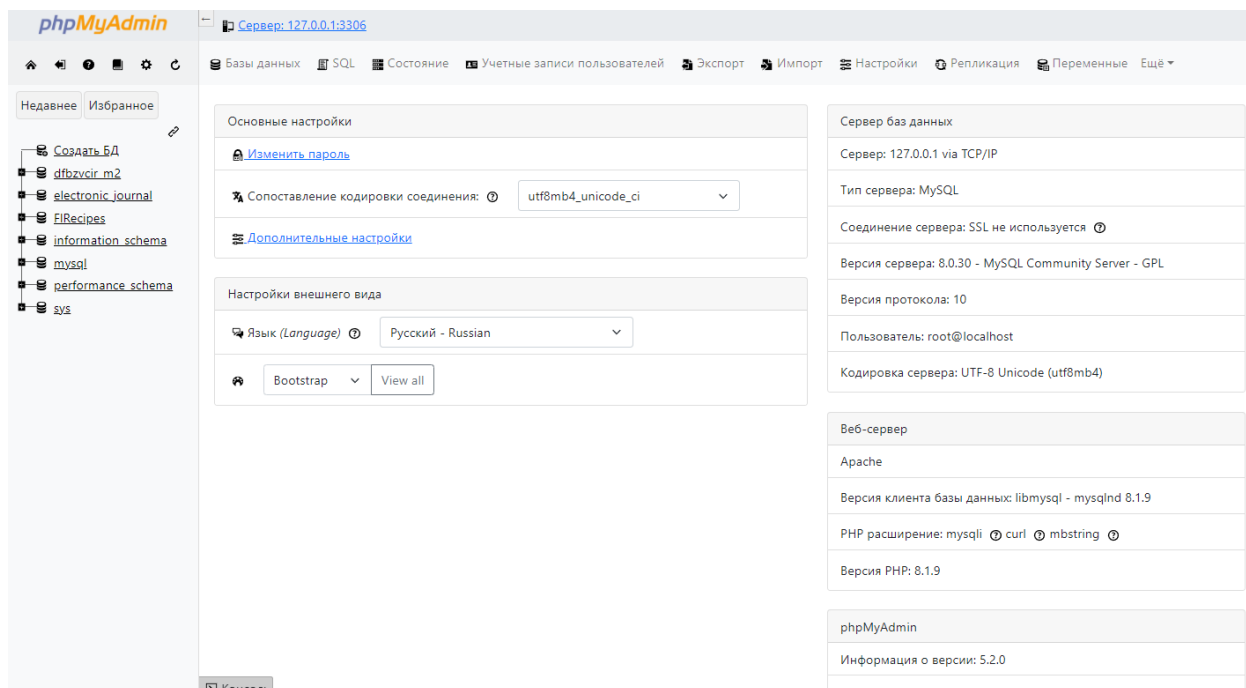


Рисунок 3.9 – Главное окно СУБД

Реляционная схема базы данных представлена на рисунке 3.10.

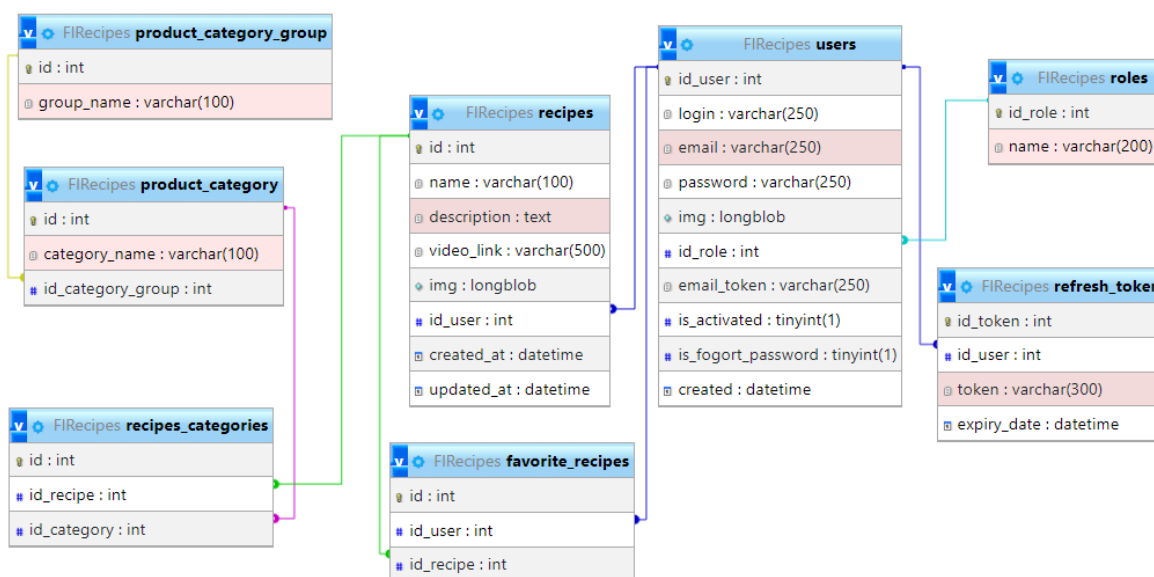


Рисунок 3.10 – Реляционная схема базы данных

Схема базы данных состоит из восьми, описывающих сущности, отношений:

- users– Пользователи.
- roles – Роли.
- refresh_token – Токены.
- recipes – Рецепты.

- favorite_recipes – Понравившиеся рецепты.
- product_category_group – Группа категорий рецептов.
- product_category – Категории рецептов.
- recipes_categories – Связь категорий и рецептов.

3.3 Инструкция по эксплуатации приложения

Приложение предназначено для удобного поиска рецептов. Пользователь может искать рецепты без авторизации, но тогда он потеряет возможность добавлять свои собственные рецепты и добавлять рецепты в избранные, где они бы отображались в личном кабинете. Для удобного поиска рецептов можно настроить параметры фильтрации и сортировки. Уже добавленные рецепты можно редактировать и удалять.

Приложение запущено при помощи хостинга, для его использования необходимо перейти по домену, который использует приложение. Для доступа к взаимодействию с базой данных необходимо обладать правами администратора. Пользователю с необходимыми правами при переходе в меню будет доступна возможность перейти к редактированию таблиц. Меню с панелью администратора представлено на рисунке 3.11.

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		55

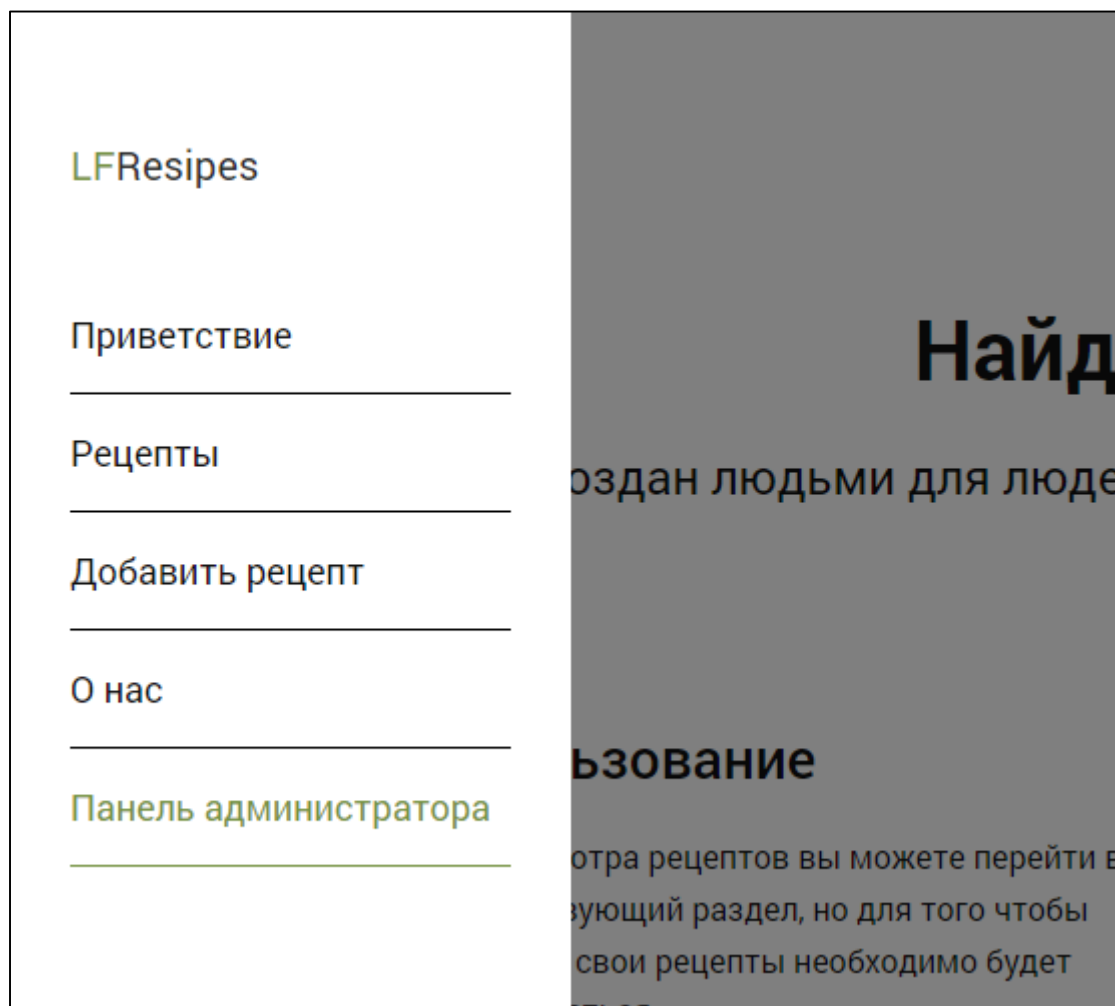


Рисунок 3.11 – Меню с панелью администратора

При нажатии на кнопку «Панель администратора», пользователь перейдет на страницу, где сможет редактировать таблицы. Панель администратора представлена на рисунке 3.12.

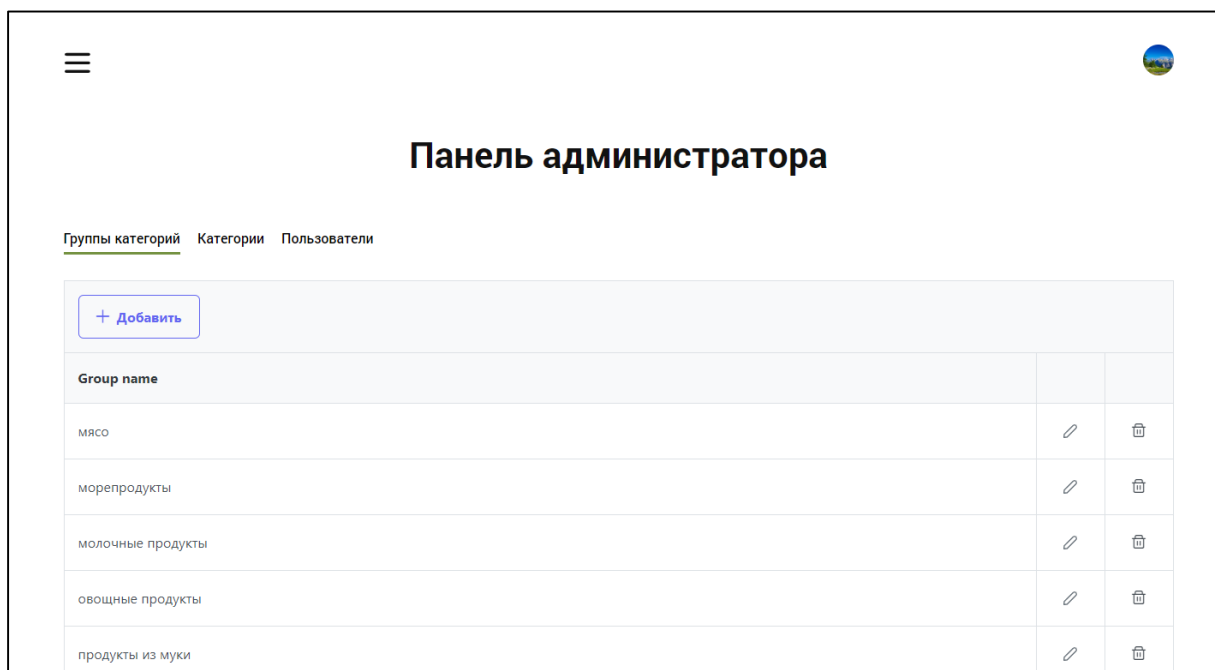


Рисунок 3.12 – Панель администратора

Для взаимодействия с панелью администратора необходимо выбрать нужную вкладку из навигации над таблицами. Список состоит из вкладок:

- группы категорий – таблица с группами категорий.
- категории – таблица с категориями.
- пользователи – таблица с пользователями.

Таблица состоит из заполненных данных, кнопкой для добавления новой записи в верхнем углу и двумя столбцами для редактирования и удаления записи. При нажатии на кнопку добавления записи появится модальное окно для заполнения данных. Модальное окно представлено на рисунке 3.13.

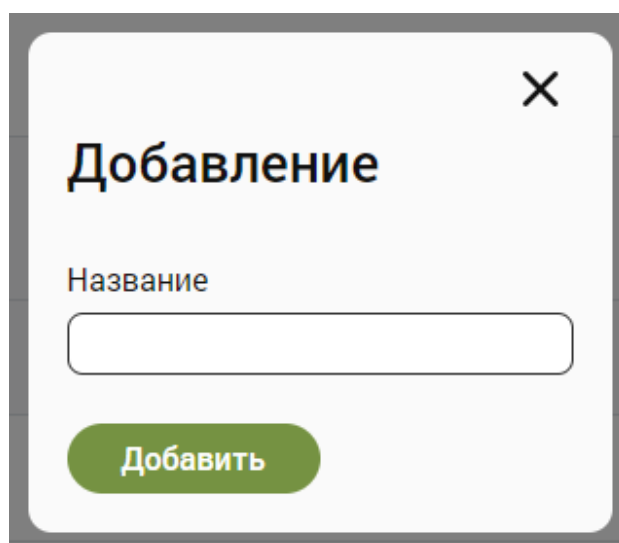


Рисунок 3.13 – Модальное окно

Для редактирования необходимо нажать на карандаш в предпоследнем столбце. После этого на месте записи появятся поля для ввода новых данных. После изменения записи необходимо подтвердить или отменить изменения. Пример редактирования записи в таблице категорий представлен на рисунке 3.14.

Category name	Group name		
<input type="text" value="мясо убойных животных"/>	<input type="text" value="Выберите группу категорий"/>	<input checked="" type="radio"/>	<input type="radio"/>
мясо диких животных	мясо	<input type="radio"/>	<input type="radio"/>
мясо птицы	мясо	<input type="radio"/>	<input type="radio"/>
субпродукты	мясо	<input type="radio"/>	<input type="radio"/>

Рисунок 3.14 – Справочник бронеи

Для удаления записи необходимо нажать на корзину в последнем столбце. После нажатия вам необходим подтвердить свои действия. Подтверждение удаления представлено на рисунке 3.15.

Подтвердите действие на странице localhost:5173

Удалить категорию?

ОК Отмена

Рисунок 3.15 – Таблица со временем сеансов

Для поиска рецептов пользователю необходимо перейти на странице с рецептами, после можно будет настроить необходимые фильтры и выбрать метод сортировки. Страница с рецептами представлена на рисунке 3.16.

Ищите интересные вас рецепты

Здесь вы сможете просмотреть рецепты пользуясь фильтрами по разным категориям и сортировкой.

Поиск...

По названию

Категории

мясо

рыба

морепродукты

яйца

молочные продукты

соевые продукты

овощные продукты

зелень

фрукты

жирны

Вкусный торт

Описание мтпаимст

Дата добавления: 01-05-2023

Жареный морж

Жареный морж

Дата добавления: 15-05-2023

Рисунок 3.16 – Пример выбора сеанса

Если пользователь авторизован он сможет добавлять свои собственные рецепты заполняя необходимые поля. Форма добавления рецепта представлена на рисунке 3.17.

Добавить рецепт

Название рецепта

Ссылка на видео с рецептом

Описание

Категории

мясо

мясо диких животных

мясо птицы

субпродукты

Изображение

Выберите файл

Файл не выбран

Добавить

Рисунок 3.17 – Форма добавления рецепта

После добавления рецепт можно изменить и удалить. Пример добавленного рецепта представлен на рисунке 3.18.

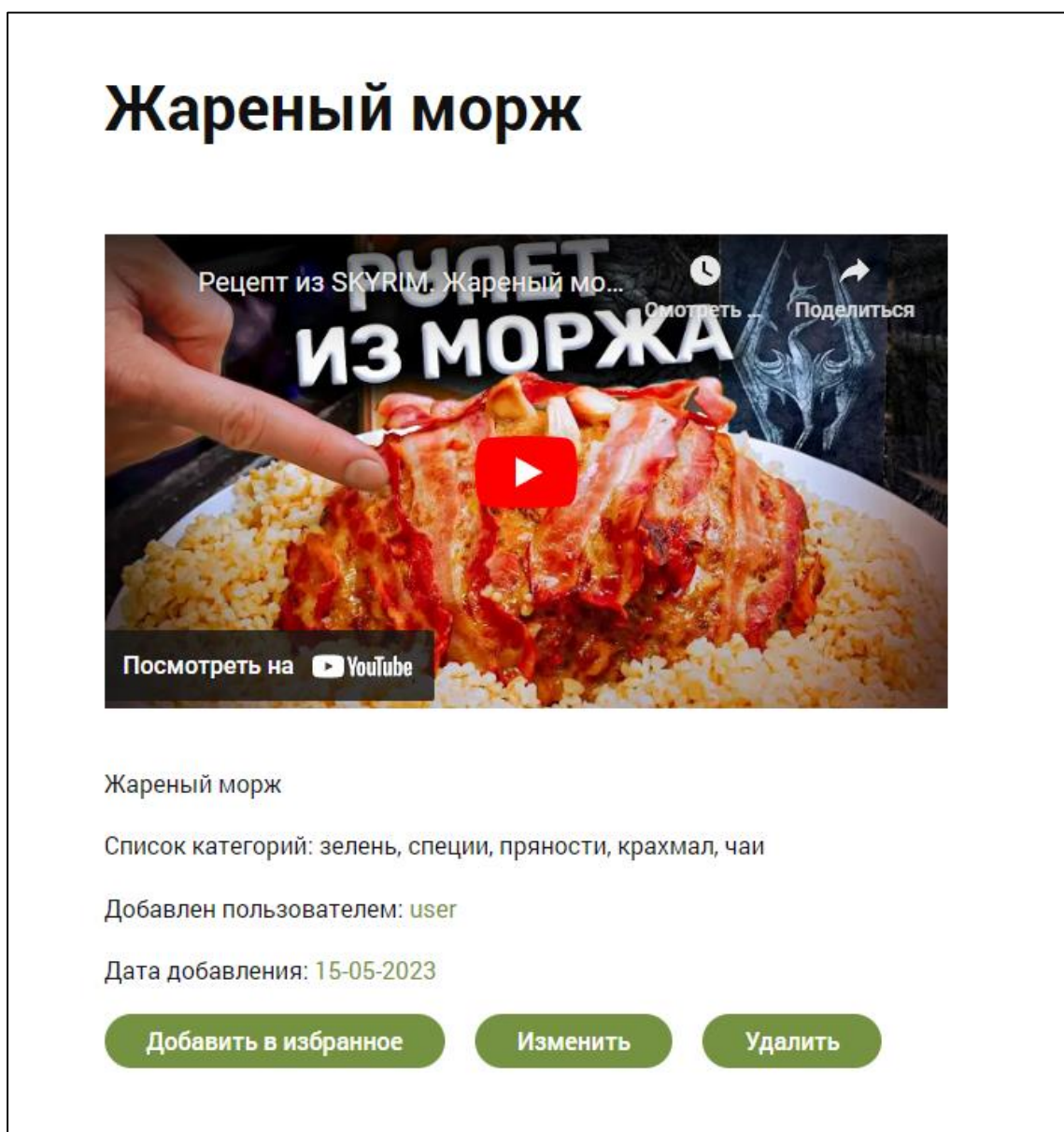


Рисунок 3.18 – Пример добавленного рецепта

Для удаления необходимо только подтвердить действие, а для изменения необходимо заполнить форму похожую на форму добавления, но форма изменения будет заполнена данными рецепта. Форма изменения рецепта представлена на рисунке 3.19.

Изменить рецепт

Название рецепта

Жареный морж

Ссылка на видео с рецептом

https://www.youtube.com/watch?v=hpCphsD1_kg

Описание

Жареный морж

Категории

мясо

мясо диких животных

мясо птицы

субпродукты

Изображение

Выберите файл

Файл не выбран

Изменить

Рисунок 3.19 – Форма изменения рецепта

4 ТЕХНИКО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

Основой расчета затрат на любой производственный процесс обычно является смета затрат. Смета затрат представляет собой сводный план всех расходов предприятия или организации на рассматриваемый период деятельности. Она определяет общую сумму издержек производства по видам используемых ресурсов, стадиям производственной деятельности, уровням управления предприятием и другим направлениям расходов. В смету включаются затраты основного и вспомогательного производства, связанные с изготовлением и продажей рассматриваемого продукта, а также на содержание административно-управленческого персонала, выполнение различных работ и услуг, в том числе и не входящих в основную производственную деятельность предприятия или организации.

Первым компонентом, входящим в сметный расчет, является материалы. В состав этих затрат принято включать стоимость материалов, которые будут проданы заказчику вместе с программным продуктом.

Поскольку кроме программного продукта никакие материалы продаваться не будут, то стоимость материалов будет равным нулю.

Вторым компонентом, включаемым в сметный расчет, являются затраты на вспомогательные материалы. В состав этих затрат принято включать стоимость расходуемых за период работ покупных инструментов и малоценного хозяйственного инвентаря.

Совокупные затраты на вспомогательные материалы и малоценный инвентарь BM , руб. рассчитываются по формуле 4.1:

$$BM = \sum_{i=1}^N BM_i, \quad (4.1)$$

где BM_i – стоимость, затраченная на каждый вспомогательный материал, руб.

Все вспомогательные материалы, используемые в рамках работы непосредственно над проектом, их количество и стоимость включены в таблицу 4.1.

Таблица 4.1 – Вспомогательные материалы

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		62

Наименование	Цена за единицу, руб., коп.	Количество, ед. изм.	Стоимость, руб., коп.
Бумажный лист формата А4.	4,00	90	360,00
Ватман формата А0, шт.	150,00	3	450,00
Картонная папка, шт.	20,00	1	20,00
Итого			830,00

Стоимость отдельного вспомогательного материала BM_i , руб. рассчитывается по формуле 4.2:

$$BM_i = N_i \cdot K_i, \quad (4.2)$$

где N_i – стоимость отдельного вспомогательного материала за штуку, руб.;

K_i – количество отдельного вспомогательного материала, руб.

Подстановкой указанных выше значений в формулу 4.1 получено:

$$BM = 830,00 \text{ руб}$$

Третьим компонентом затрат на разработку являются затраты на энергетические ресурсы ЭР, руб, состоящие из затрат на электроэнергию и затрат на топливо, и рассчитываемые по формуле 4.3:

$$\mathcal{E}P = \mathcal{E} + T, \quad (4.3)$$

где \mathcal{E} – затраты на электроэнергию, руб.;

T – затраты на топливо, руб.

Так как разработка программного продукта не требует затрат на топливо, то стоимость топлива принимается равным нулю.

Затраты на электроэнергию вычисляются по формуле 4.4:

$$\mathcal{E}L = W \cdot \mathcal{C}_{\mathcal{E}}, \quad (4.4)$$

где W – совокупная потребленная мощность, кВт×ч;

$\mathcal{C}_{\mathcal{E}}$ – стоимость одного кВт×ч электроэнергии, руб.; составляет 3.62руб.

Совокупная потребленная мощность W рассчитывается по формуле 4.5:

$$W = \sum W_i, \quad (4.5)$$

где W_i – мощность, потребляемая отдельным устройством, кВт.

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		63

В таблице 4.2 представлены все возможные устройства, используемые в разработке продукта, потребляющие электроэнергию.

Таблица 4.2 – Используемые устройства

Наименование	Потребляемая мощность, КВт	Продолжительность эксплуатации в смену, час.	Количество смен в цикле производства (разработки), шт.	Потребленная мощность, КВт×ч
Компьютер	0,5	8	15	60
Монитор	0,05	8	15	6
Итого				66

Потребленная мощность W_i , кВт×ч для каждого из устройств рассчитывается по формуле 4.6:

$$W_i = P_i \cdot t_i \cdot KС_i, \quad (4.6)$$

где P_i – мощность устройства, КВт;

t_i – продолжительность эксплуатации устройства за одну рабочую смену. час.;

$KС_i$ – количество смен, в течение которых использовалось устройство, шт.

Подстановкой значений из таблицы 4.2 в формулу 4.5 получено значение потребленной мощности:

$$W=66$$

Подстановкой указанных выше значений в формулу 4.4, получаем затраты на электроэнергию:

$$\mathcal{E}Л=66 \cdot 3,62=23892руб$$

Подстановкой указанных выше значений в формулу 4.3, получаем затраты на энергетические ресурсы:

$$\mathcal{E}Р=23892+0=23892руб$$

Четвертым этапом сметного расчета является расчет амортизации использованного в проекте оборудования.

Амортизация – это процесс постепенного переноса стоимости средств, по мере их износа, на стоимость продукции, которая производится. Амортизация распространяется на основные средства, основные фонды и нематериальные активы. Она исчисляется линейным, нелинейным методом, способом уменьшения остаточной стоимости, способом списания стоимости по сумме лет использования.

Амортизационные отчисления – отчисления части стоимости основных средств для возмещения их износа. Амортизационные отчисления включаются в издержки производства и производятся коммерческими организациями на основе установленных норм и балансовой стоимости основных средств.

Общий объем амортизационных отчислений AM , руб., включаемый в расчет по проекту для всех устройств можно вычислить по формуле 4.7:

$$AM = \sum_{i=1}^N A_i, \quad (4.7)$$

где A_i – амортизационное отчисление для отдельного основного средства, руб.

Для разработки программного продукта потребовалось оборудование и нематериальные активы, перечисленные в таблице 4.3.

Таблица 4.3 – Информация по расчету амортизационных отчислений

Наименование основного средства	Изначальная стоимость, руб., коп.	Срок службы, лет	Сумма ежегодного амортизационного отчисления, руб., коп.	Период использования в проекте, кален. дн.	Сумма амортизационного отчисления, руб., коп.
Офисный стол	2 317,00	4	579,25	15	23,80
Офисное кресло	3 680,00	5	736,00	15	30,24
Компьютер	60 999,00	5	12 199,80	15	501,36
Монитор	6 590,00	5	1 318,00	15	54,16
Итого					609,56

Амортизационное отчисление A_i , руб., включаемое в расходы по проекту, для каждого из основных средств можно вычислить по формуле 4.8:

$$A_i = AQ_i \cdot K_d / 365, \quad (4.8)$$

где AO_i – сумма планируемого амортизационного отчисления для основного средства за год, руб.;

$KД_i$ – количество календарных дней, в течение которых использовалось основное средство, дн.;

365 – это количество календарных дней в году, дн.

Для всех основных средств и нематериальных активов рассчитывается сумма годового амортизационного отчисления AO_i , руб. по формуле 4.9, которое каждый год будет составлять одну и ту же величину для одного и того же устройства.

$$AO_i = HC_i / Cp_i, \quad (4.9)$$

где HC_i – начальная стоимость основного средства, руб.;

Cp_i – срок службы основного средства, лет.

Далее для каждого основного средства рассчитывается период его использования в проекте в календарных днях $KД_i$, формула 4.10:

$$KД = 7 \cdot KC_i / 5, \quad (4.10)$$

где KC_i – это количество рабочих дней (фактически, при односменном режиме работы – количество рабочих смен);

5 – это количество рабочих дней на неделе;

7 – это количество календарных дней в неделе.

Подстановкой значений из таблицы 4.2 в формулу 4.7 получено:

$$AM = 609,5 \text{ руб}$$

Следующим этапом сметного расчета является затраты на оплату труда.

Фонд оплаты труда $ЗП$, руб., включаемый в затраты по проекту, можно рассчитать по формуле 4.11:

$$ЗП = ОЗП + ДЗП, \quad (4.11)$$

где $ОЗП$ – основная заработная плата всех работников, участвовавших в проекте, руб.;

					ДП.0902.10.000000.00 ПЗ	Лист
						66
Изм	Лист	№ докум.	Подпись	Дата		

ДЗП – дополнительная заработная плата всех работников, участвовавших в проекте, руб.;

Основная заработная плата *ОЗП*, руб., включаемая в затраты проекта, будет складываться из двух частей, для расчёта можно воспользоваться формулой 4.12:

$$ОЗП = ОкЗП + СдЗП, \quad (4.12)$$

где *ОкЗП* – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по окладной системе оплаты труда, руб.;

СдЗП – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по сдельной системе оплаты труда;

Заработная плата всех рабочих *ОкЗП*, руб., участвовавших в проекте, оплачиваемых по окладной системе оплаты труда, вычисляется по формуле 4.13:

$$ОкЗП = \sum_{i=1}^{N1} ОкЗП_i, \quad (4.13)$$

где *ОкЗП_i* – заработная плата отдельного рабочего, участвовавшего в проекте, оплачиваемого по окладной системе оплаты труда, руб.

В таблице 4.4 сведены данные по работникам, принимавшим участие в проекте, оплачиваемым по окладной системе оплаты труда.

Таблица 4.4 – Заработная плата

Должность, профессия работника	Месячный оклад, руб., коп.	Отработано смен, шт.	Заработная плата, руб., коп.
Фулстек разработчик	50 000,00	15	36 764,70
Итого			36 764,70

В рамках проекта заработная плата *ОкЗП_i*, руб., заработанная каждым из работников, оплачиваемых по окладной системе оплаты труда, рассчитывается по формуле 4.14:

$$ОкЗП_i = O_i \cdot КС_i / 204, \quad (4.14)$$

где *O_i* – оклад работника, руб.,

КС_i – количество смен (рабочих дней), которое он работал по проекту, шт., составляет 15 дней;

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		67

20,4 – это среднемесячное количество смен (рабочих дней), шт.

Подстановкой указанных выше значений в формулу 4.13 получено:

$$ОкЗП=36764,7\text{руб}$$

Заработная плата всех рабочих, оплачиваемых по сдельной системе оплаты труда, $CдЗП$, руб., рассчитывается по формуле 4.15:

$$CдЗП=\sum_{i=1}^{N2}CдЗП_i, \quad (4.15)$$

где $CдЗП_i$ – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по сдельной системе оплаты труда, руб.

Так как в разработке проекта не участвовали рабочие, оплачиваемые по сдельной системе оплаты труда, сдельная заработная плата *равна нулю*.

Подстановкой указанных выше значений в формулу 4.12 получено:

$$ОЗП=36764,7\text{руб}$$

Предполагается, что в оплату труда по проекту должна быть заложена в том числе, и дополнительная заработная плата ДЗП, которую необходимо будет выплачивать тем же самым работникам, например, в связи с очередным отпуском, то ее расчет будем осуществлять исходя из того, что ее размер должен составлять какую-то разумную обоснованную долю от основной заработной платы.

Известно, что большинство работников находятся в очередных плановых отпусках в среднем около одного месяца в году, то логично предположить, что доля дополнительной заработной платы ДЗП будет около 1/12 от основной заработной платы, что составляет приблизительно 8-9%. Логично допустить, что кроме отпусков работникам в соответствии с действующим законодательством предстоит делать еще какие-либо выплаты в виде дополнительной заработной платы. По этой причине полученную долю можно несколько увеличить и выбрать ее в размере 9-12%. Таким образом, сумму затрат на дополнительную заработную плату $ДЗП$, руб., можно рассчитать по формуле 4.16:

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		68

$$ДЗП = ОЗП \cdot K_{ДЗП} / 100, \quad (4.16)$$

где $ОЗП$ – основная заработная плата всех работников, участвовавших в проекте, руб.;

$K_{ДЗП}$ – коэффициент, %.

В рамках проекта $K_{ДЗП}$ выбран равным 10%.

Подстановкой указанных выше значений в формулу 4.16 получено:

$$ДЗП = 36764,70 / 100 = 367,64$$

Подстановкой указанных выше значений в формулу 4.11 получено:

$$ЗП = 36764,70 + 367,64 = 40441,7 \text{ руб}$$

Шестым компонентом, включаемым в сметный расчет, являются страховые взносы.

Социальное страхование – это система социальной защиты, задача которой – обеспечивать реализацию конституционного права экономически активных граждан на материальное обеспечение в старости, в случае болезни, полной или частичной утраты трудоспособности, потери кормильца, безработицы.

Уплата страховых взносов является прямой обязанностью организации-работодателя, при этом часть выплат носит персональный характер. Так, например, выплат на лицевой счет каждого из работников в пенсионном фонде зависит от размера его персональной заработной платы.

Общий размер страховых взносов $СВ$, руб., необходимых для перечисления во все страховые фонды и включаемый в состав затрат при производстве, можно вычислить как некоторый процент от фонда заработной платы, для расчёта можно воспользоваться формулой 4.17:

$$СВ = ЗП \cdot C_{СВ} / 100, \quad (4.17)$$

где $ЗП$ – фонд оплаты труда по проекту, рассчитанный ранее (руб.), составляет 40 441,17 руб.;

$C_{СВ}$ – ставка (размер налога) страховых взносов, %.

Ставка страховых взносов $C_{СВ}$ на 2023 год составляет 30%.

					ДП.0902.10.000000.00 ПЗ	Лист
						69
Изм	Лист	№ докум.	Подпись	Дата		

Подстановкой указанных выше значений в формулу 4.17 получено:

$$CB=4044,17 \cdot 30/100=1213,35 \text{руб}$$

Следующим этапом сметного расчета является накладные расходы.

В зависимости от размеров предприятия или организации накладные расходы HP , руб., могут варьироваться в достаточно широком диапазоне. Поэтому в обычной практике в сметных расчетах их принято отражать в процентном отношении к заработной плате основных работников, для расчёта можно воспользоваться формулой 4.18:

$$HP = ЗП \cdot C_{HP} / 100, \quad (4.18)$$

где $ЗП$ – фонд оплаты труда по проекту, руб.;

C_{HP} – размер накладных расходов, %.

Размер накладных расходов C_{HP} составляет 10%.

Подстановкой указанных выше значений в формулу 4.18 получено:

$$HP=4044,17 \cdot 10/100=404,41 \text{руб}$$

Восьмым этапом сметного расчета является расчет себестоимости.

Себестоимость – это текущие затраты на производство товара или услуги, запуск их в обращение и реализацию. Расчет и анализ себестоимости продукции является важнейшей задачей любого предприятия и входит в систему управленческого учета, так как именно себестоимость лежит в основе большинства управленческих решений.

Себестоимость $Сб$, руб. программного продукта определяется по формуле 4.19.

$$Сб = М + ВМ + ЭР + АМ + ЗП + СВ + HP, \quad (4.19)$$

где $М$ – затраты на материалы руб.;

$ВМ$ – затраты на вспомогательные материалы, руб.;

$ЭР$ – энергетические затраты, руб.;

$АМ$ – затраты на амортизацию, руб.;

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		70

ЗП – затраты на заработную плату, руб.;

СВ – страховые взносы, уплаченные работодателем с фонда оплаты труда, руб.;

НР – накладные расходы, возникающие как непредусмотренные всеми предшествующими статьями, руб.;

Подстановкой указанных выше значений в формулу 4.19 получено:

$$Сб = 0 + 83000 + 23892 + 60956 + 404417 + 121335 + 40441 = 582961 \text{ руб}$$

После расчета себестоимости (всех возможных затрат на разработку и производство), к ней добавляется необходимый объем прибыли, он и будет являться девятым продуктом сметного расчета.

Объем прибыли обычно определяется как процент от рассчитанной себестоимости. Объем прибыли *П*, руб. можно рассчитать по формуле 4.20:

$$П = Сб \cdot C_{П} / 100, \quad (4.20)$$

где *Сб* – себестоимость производства или разработки, руб.;

С_П – процент прибыли, на которую рассчитывает организатор коммерческого предприятия, %.

Процент прибыли *С_П*, на которую рассчитывает организатор коммерческого предприятия составляет 30%.

Подстановкой указанных выше значений в формулу 4.20 получено:

$$П = 582961 \cdot 30 / 100 = 174888 \text{ руб}$$

Следующим этапом сметного расчета является налог на добавленную стоимость.

Налог на добавленную стоимость (НДС) – косвенный налог, форма изъятия в бюджет государства части стоимости товара, работы или услуги, которая создаётся на всех стадиях процесса производства товаров, работ и услуг и вносится в бюджет по мере реализации.

Для расчета налога на добавленную стоимость *НДС*, руб. можно воспользоваться формулой 4.21:

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		71

$$НДС=(Сб+П) \cdot C_{НДС}/100, \quad (4.21)$$

где $Сб$ – себестоимость разработки, руб.;

$П$ – прибыль, руб.;

$C_{НДС}$ – ставка налога на добавленную стоимость, %.

Ставка налога $C_{НДС}$ на добавленную стоимость составляет 20%.

Подстановкой указанных выше значений в формулу 4.21 получено:

$$НДС=(5829,61+174883) \cdot 20/100=1515,88 \text{руб}$$

Конечная стоимость разработки $С$, руб. рассчитывается как сумма себестоимости, прибыли и НДС, для расчёта можно воспользоваться формулой 4.22:

$$C=Сб+П+НДС, \quad (4.22)$$

где $Сб$ – себестоимость разработки, руб.

$П$ – прибыль. руб.;

$НДС$ – сумма налога на добавленную стоимость, руб.;

Подстановкой указанных выше значений в формулу 4.22 получено:

$$C=5829,61+174883+1515,88=90941,32 \text{руб}$$

Исходя из вышеперечисленных расчетов следует, что программный продукт является конкурентоспособным, его стоимость составляет 90941,32 руб.

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		72

5 РАЗДЕЛ ОХРАНЫ ТРУДА

Охрана труда – это целая система законодательных и нормативно-правовых актов, технических, гигиенических, лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранение здоровья и работоспособности человека в процессе труда. В наши дни труд стал более интенсивным и требует огромных затрат умственной, эмоциональной и физической нагрузок.

На рабочем месте программист осуществляет трудовую деятельность и проводит большую часть рабочего времени. Правильная организация рабочего места программиста повышает производительность труда от 8 до 20%. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям. Главные элементы рабочего места программиста – это письменный стол и кресло. Рабочее место организуется в соответствии с ГОСТ 12.2.032-78, информация из работы [7].

Площадь рабочего места с компьютером с жидкокристаллическим или плазменным экраном должна быть не менее 4,5 кв. м, а расстояние между столами с мониторами (от тыла одного монитора до экрана другого) не менее 2 м. Монитор должен располагаться на расстоянии 50-70 см от глаз программиста. Параметры рабочего стола сотрудника: возможность регулировки высоты рабочего стола, или точная высота – 72,5 см, ширина – 80, 100, 120 или 140 см, глубина рабочего стола 80 или 100 см, высота и ширина пространства под столешницей (для ног) – не менее 50 см, глубина на уровне колен не менее 45 см, а на уровне вытянутых ног не менее 65 см.

Правильное освещение рабочего места – это очень важный момент в трудовой деятельности человека, влияющий на эффективность труда, при этом такой момент предупреждает травматизм и профессиональные заболевания. При недостаточном освещении приходится напрягать зрение, при этом ослабляется внимание и это приводит к наступлению преждевременной утомленности. Слишком яркое освещение тоже плохо, так как оно вызывает ослепление, раздражение и резь

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		73

в глазах. При искусственном освещении, источниками света служат два вида ламп: лампы накаливания и люминесцентные.

Известно, что шум ухудшает условия труда и оказывает вредное воздействие на организм человека. Согласно ГОСТ 12.1.003-88 «Шум для помещений расчетчиков и программистов, уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ», информация из работы [8].

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей. В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных «В». При пожаре люди должны покинуть помещение в течение минимального времени. В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования. Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях, информация из работы [9].

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		74

ЗАКЛЮЧЕНИЕ

По итогу работы было разработано веб-приложение, позволяющее пользователям находить рецепты при помощи удобных фильтров и сортировки, также у пользователей есть возможность добавлять своих рецепты и добавлять рецепты других пользователей в избранные. Добавленные и избранные рецепты отображаться в личном кабинете, внутри которого есть возможность изменения своего аватара, отображающегося в шапке приложения, и изменения своего логина, отображающегося в рецептах. Ознакомиться с рецептами можно и без авторизации, но пользователь будет терять возможность как-либо взаимодействовать с ними. Для использования приложения необходимо указывать свою реальную почту, так как без подтверждения почты, не будет возможности авторизоваться. При помощи почты есть и возможность восстановить пароль. Таким образом только пользователь, обладающий почтой зарегистрированного аккаунта, сможет изменить пароль, что защитит его от злоумышленников. Помимо обычных пользователей, есть с правами модератора, которые позволяют также изменять все имеющиеся рецепты и с правами администратора, получающие доступ к панели администратора. Панель администратора дает возможность пользователю, с соответствующими правами, редактировать, добавлять и удалять группы категорий, сами категории и пользователей.

Главным достоинством можно выделить приятный интерфейс и простоту в использовании приложения. Все действия выполняются на интуитивно понятном уровне. Также приложение безопасно в использовании благодаря технологиям, которые в нем используются.

Веб-приложение можно использовать в коммерческих целях реализовав подписки для доступа или разместив в нем рекламу. Можно также интегрировать его в ресторан или кафе, сделав его ориентированным на репертуар определенного заведения.

Разработанное веб-приложение можно доработать, добавив возможность пользователям добавлять ингредиенты к рецептам и систему лайков для оценки

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		75

популярности рецепта. Также для конечного продукта необходимо интегрировать защиту от спама путем добавления капчи, для защиты веб-приложения от спама.

					ДП.0902.10.000000.00 ПЗ	Лист
						76
Изм	Лист	№ докум.	Подпись	Дата		

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. <https://kazedu.com/referat/133091/5>
2. <https://www.povarenok.ru/recipes/find>
3. <https://steptosleep.ru/ролевая-модель/#:~:text=Основная%20идея%20ролевой%20модели%20контроля,типов%20их%20активностей%20в%20системе>
4. https://studopedia.ru/22_29871_neobhodimost-otladki-programmnogo-produkta.html
5. <https://infopedia.su/4x1ec5.html>
6. <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-4-ch3.html>
7. <https://www.retail.ru/rbc/pressreleases/tsentr-povysheniya-kvalifikatsii-lider-organizatsiya-rabochego-mesta-ofisnogo-rabotnika/>
8. <https://xn--d1aux.xn--p1ai/opisanie-rabochego-mesta-programmista-na-predpriyatii/>
9. https://studopedia.ru/8_107307_osveshchenie-pomeshcheniy-vichislitelnih-tsentrov.html

ПРИЛОЖЕНИЕ А

Программный код файла auth.config

```
module.exports = {  
  secret: "locking-for-recipes-diploma",  
  jwtExpiration: 3600, // 1 hour  
  jwtRefreshExpiration: 86400, // 24 hours  
};
```

Программный код файла db.config

```
const mysql = require("mysql2");  
// create the connection to database  
const db = mysql.createConnection({  
  host: "localhost",  
  user: "root",  
  password: "root",  
  database: "FLRecipes",  
});  
  
module.exports = db;
```

Программный код файла roles.config

```
module.exports = ["user", "admin", "moderator"];
```

Программный код файла model.categories

```
const db = require("../config/db.config.js");  
  
class CategoryModel {  
  insertCategory(data) {  
    return new Promise((resolve, reject) => {  
      db.query("INSERT INTO product_category SET ?", [data], (err, results) => {  
        if (err) {  
          console.log(err);  
          reject(err);  
        } else {  
          resolve(results);  
        }  
      });  
    });  
  }  
  
  getCategories() {  
    return new Promise((resolve, reject) => {  
      db.query(  
        `SELECT pc.id, pc.category_name, pcg.group_name`  

```

					ДП.0902.10.000000.00 ПЗ	Лист
						78
Изм	Лист	№ докум.	Подпись	Дата		

```

FROM product_category AS pc
LEFT JOIN product_category_group AS pcg
ON pc.id_category_group = pcg.id;`,
(err, results) => {
  if (err) {
    console.log(err);
    reject(err);
  } else {
    resolve(results);
  }
}
);
});
}

getCategoriesByGroupId(id) {
return new Promise((resolve, reject) => {
  db.query(
    `SELECT * FROM product_category WHERE id_category_group = ?`,
    [id],
    (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results);
      }
    }
  );
});
}

getCategoryById(id) {
return new Promise((resolve, reject) => {
  db.query(
    `SELECT category_name FROM product_category WHERE id = ?`,
    [id],
    (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results[0]);
      }
    }
  );
});
}

updateCategory(id, data) {
return new Promise((resolve, reject) => {
  db.query(
    `UPDATE product_category SET ? WHERE id = ?`,
    [data, id],
    (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results[0]);
      }
    }
  );
});
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						79
Изм	Лист	№ докум.	Подпись	Дата		

```

    });
  }

  deleteCategory(id) {
    return new Promise((resolve, reject) => {
      db.query(
        "DELETE FROM product_category WHERE id = ?",
        [id],
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {
            resolve(results);
          }
        }
      );
    });
  }
}

module.exports = new CategoryModel();

```

Программный код файла model.categoryGroups

```

const db = require("../config/db.config.js");

class CategoryGroupModel {
  insertCategoryGroup(data) {
    return new Promise((resolve, reject) => {
      db.query(
        "INSERT INTO product_category_group SET ?",
        [data],
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {
            resolve(results);
          }
        }
      );
    });
  }

  getCategoryGroups() {
    return new Promise((resolve, reject) => {
      db.query("SELECT * FROM product_category_group", (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results);
        }
      });
    });
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						80
Изм	Лист	№ докум.	Подпись	Дата		


```

updateCategoryGroup(id, data) {
  return new Promise((resolve, reject) => {
    db.query(
      `UPDATE product_category_group SET ? WHERE id = ?`,
      [data, id],
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results[0]);
        }
      }
    );
  });
}

deleteCategoryGroup(id) {
  return new Promise((resolve, reject) => {
    db.query(
      "DELETE FROM product_category_group WHERE id = ?",
      [id],
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results);
        }
      }
    );
  });
}

module.exports = new CategoryGroupModel();

```

Программный код файла model.favoriteRecipes

```

const db = require("../config/db.config.js");

class FavoriteRecipeModel {
  addFavoriteRecipe(data) {
    return new Promise((resolve, reject) => {
      db.query("INSERT INTO favorite_recipes SET ?", [data], (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results);
        }
      });
    });
  }

  getUserFavoriteRecipes(id) {
    return new Promise((resolve, reject) => {
      db.query(
        "SELECT id_recipe FROM favorite_recipes WHERE id_user = ?",

```

					ДП.0902.10.000000.00 ПЗ	Лист
						81
Изм	Лист	№ докум.	Подпись	Дата		

```

[id],
(err, results) => {
  if (err) {
    console.log(err);
    reject(err);
  } else {
    resolve(results);
  }
}
);
});
}

deleteUserFavoriteRecipe(id_user, id_recipe) {
return new Promise((resolve, reject) => {
  db.query(
    "DELETE FROM favorite_recipes WHERE id_user = ? AND id_recipe = ?",
    [id_user, id_recipe],
    (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results);
      }
    }
  );
});
}

deleteAllUserFavoriteRecipes(id) {
return new Promise((resolve, reject) => {
  db.query(
    "DELETE FROM favorite_recipes WHERE id_user = ?",
    [id],
    (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results);
      }
    }
  );
});
}

deleteFavoriteRecipe(id) {
return new Promise((resolve, reject) => {
  db.query(
    "DELETE FROM favorite_recipes WHERE id_recipe = ?",
    [id],
    (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results);
      }
    }
  );
});
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						82
Изм	Лист	№ докум.	Подпись	Дата		

```

    }
  }

  module.exports = new FavoriteRecipeModel();

```

Программный код файла model.recipes

```

const db = require("../config/db.config.js");

class RecipeModel {
  insertRecipe(data) {
    return new Promise((resolve, reject) => {
      db.query("INSERT INTO recipes SET ?", [data], (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results);
        }
      });
    });
  }

  updateRecipe(id, data) {
    return new Promise((resolve, reject) => {
      db.query(
        `UPDATE recipes SET ? WHERE id = ?`,
        [data, id],
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {
            resolve(results[0]);
          }
        }
      );
    });
  }

  getAllUserRecipes(id) {
    return new Promise((resolve, reject) => {
      db.query(
        `SELECT id FROM recipes WHERE id_user = ?`,
        [id],
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {
            resolve(results);
          }
        }
      );
    });
  }

  getRecipes(data) {
    return new Promise((resolve, reject) => {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						83
Изм	Лист	№ докум.	Подпись	Дата		

```

const inner = data.filters
? `INNER JOIN recipes_categories AS rc ON r.id = rc.id_recipe AND FIND_IN_SET(rc.id_category, "${data.filters}") > 0`
: "";
const where = `WHERE name LIKE "%${data.name}%"`;
const order = `ORDER BY r.${data.sort} ${data.sortParam}`;
const limit = `LIMIT ${data.page - 1} * data.limit, ${data.limit}`;
db.query(
`SELECT DISTINCT r.id, r.name, r.description, r.video_link, r.id_user, r.created_at, r.updated_at FROM recipes AS r ` +
inner +
" " +
where +
" " +
order +
" " +
limit,
(err, results) => {
if (err) {
console.log(err);
reject(err);
} else {
resolve(results);
}
}
);
});
}

async getUserRecipes(data) {
const where = data.id_user ? `WHERE id_user = ${data.id_user}` : "";
const limit = `LIMIT ${data.page - 1} * data.limit, ${data.limit}`;
let recipes = [];
let totalRecords = 0;

return new Promise((resolve, reject) => {
db.query(
`SELECT id, name, description, video_link, id_user, created_at, updated_at FROM recipes ` +
where +
" " +
limit,
(err, results) => {
if (err) {
console.log(err);
reject(err);
} else {
recipes = results;
db.query(
`SELECT COUNT(*) AS total_records FROM recipes ` + where,
(err, results) => {
if (err) {
console.log(err);
reject(err);
} else {
totalRecords = results[0].total_records;
resolve({
recipes: recipes,
totalRecords: totalRecords,
});
}
}
);
}
}
)
}
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						84
Изм	Лист	№ докум.	Подпись	Дата		

```

    });
  });
}

async getUserFavoriteRecipes(data) {
  const where = data.recipes
    ? `WHERE FIND_IN_SET(id, "${data.recipes}") > 0`
    : "";
  const limit = `LIMIT ${data.page - 1} * data.limit, ${data.limit}`;
  let recipes = [];
  let totalRecords = 0;

  return new Promise((resolve, reject) => {
    db.query(
      `SELECT id, name, description, video_link, id_user, created_at, updated_at FROM recipes ` +
        where +
        " " +
        limit,
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          recipes = results;
          db.query(
            `SELECT COUNT(*) AS total_records FROM recipes ` + where,
            (err, results) => {
              if (err) {
                console.log(err);
                reject(err);
              } else {
                totalRecords = results[0].total_records;
                resolve({
                  recipes: recipes,
                  totalRecords: totalRecords,
                });
              }
            }
          );
        }
      }
    );
  });
}

getTotalRecords(data) {
  return new Promise((resolve, reject) => {
    const inner = data.filters
      ? `INNER JOIN recipes_categories AS rc ON r.id = rc.id_recipe AND FIND_IN_SET(rc.id_category, "${data.filters}") > 0`
      : "";
    const where = `WHERE name LIKE "%${data.name}%"`;
    const order = `ORDER BY r.${data.sort} ${data.sortParam}`;
    db.query(
      `SELECT COUNT(r.id) AS count_pages FROM recipes AS r ` +
        inner +
        " " +
        where +
        " " +
        order +
        " ",
      (err, results) => {
        if (err) {

```

```

        console.log(err);
        reject(err);
    } else {
        resolve(results[0].count_pages);
    }
    }
    );
    });
}

findRecipeByExtend(extend, data) {
    return new Promise((resolve, reject) => {
        db.query(
            `SELECT * from recipes WHERE ${extend} = "${data}"`,
            (err, results) => {
                if (err) {
                    console.log(err);
                    reject(err);
                } else {
                    resolve(results[0]);
                }
            }
        );
    });
}

updateImageRecipeById(image, id) {
    return new Promise((resolve, reject) => {
        db.query(
            `UPDATE recipes SET img = ? WHERE id = ?`,
            [image, id],
            (err, results) => {
                if (err) {
                    console.log(err);
                    reject(err);
                } else {
                    resolve(results[0]);
                }
            }
        );
    });
}

deleteRecipe(id) {
    return new Promise((resolve, reject) => {
        db.query("DELETE FROM recipes WHERE id = ?", [id], (err, results) => {
            if (err) {
                console.log(err);
                reject(err);
            } else {
                resolve(results);
            }
        });
    });
}

deleteUserRecipes(id) {
    return new Promise((resolve, reject) => {
        db.query(
            "DELETE FROM recipes WHERE id_user = ?",
            [id],
            (err, results) => {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						86
Изм	Лист	№ докум.	Подпись	Дата		

```

        if (err) {
            console.log(err);
            reject(err);
        } else {
            resolve(results);
        }
    }
    );
});
}
}

```

```
module.exports = new RecipeModel();
```

Программный код файла model.recipesCategories

```
const db = require("../config/db.config.js");
```

```

class RecipeModel {
  insertRecipe(data) {
    return new Promise((resolve, reject) => {
      db.query("INSERT INTO recipes SET ?", [data], (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results);
        }
      });
    });
  }
}

```

```

  updateRecipe(id, data) {
    return new Promise((resolve, reject) => {
      db.query(
        `UPDATE recipes SET ? WHERE id = ?`,
        [data, id],
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {
            resolve(results[0]);
          }
        }
      );
    });
  }
}

```

```

  getAllUserRecipes(id) {
    return new Promise((resolve, reject) => {
      db.query(
        `SELECT id FROM recipes WHERE id_user = ?`,
        [id],
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						87
Изм	Лист	№ докум.	Подпись	Дата		

```

        resolve(results);
    }
}
);
});
}

getRecipes(data) {
    return new Promise((resolve, reject) => {
        const inner = data.filters
            ? `INNER JOIN recipes_categories AS rc ON r.id = rc.id_recipe AND FIND_IN_SET(rc.id_category, "${data.filters}") > 0`
            : "";
        const where = `WHERE name LIKE "%${data.name}%"`;
        const order = `ORDER BY r.${data.sort} ${data.sortParam}`;
        const limit = `LIMIT ${data.page - 1} * data.limit, ${data.limit}`;
        db.query(
            `SELECT DISTINCT r.id, r.name, r.description, r.video_link, r.id_user, r.created_at, r.updated_at FROM recipes AS r ` +
            inner +
            " " +
            where +
            " " +
            order +
            " " +
            limit,
            (err, results) => {
                if (err) {
                    console.log(err);
                    reject(err);
                } else {
                    resolve(results);
                }
            }
        );
    });
}

async getUserRecipes(data) {
    const where = data.id_user ? `WHERE id_user = ${data.id_user}` : "";
    const limit = `LIMIT ${data.page - 1} * data.limit, ${data.limit}`;
    let recipes = [];
    let totalRecords = 0;

    return new Promise((resolve, reject) => {
        db.query(
            `SELECT id, name, description, video_link, id_user, created_at, updated_at FROM recipes ` +
            where +
            " " +
            limit,
            (err, results) => {
                if (err) {
                    console.log(err);
                    reject(err);
                } else {
                    recipes = results;
                    db.query(
                        `SELECT COUNT(*) AS total_records FROM recipes ` + where,
                        (err, results) => {
                            if (err) {
                                console.log(err);
                                reject(err);
                            } else {
                                totalRecords = results[0].total_records;
                            }
                        }
                    );
                }
            }
        );
    });
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						88
Изм	Лист	№ докум.	Подпись	Дата		


```

        resolve({
          recipes: recipes,
          totalRecords: totalRecords,
        });
      }
    }
  );
}
};
});
}

async getUserFavoriteRecipes(data) {
  const where = data.recipes
    ? `WHERE FIND_IN_SET(id, "${data.recipes}") > 0`
    : "";
  const limit = `LIMIT ${data.page - 1} * data.limit, ${data.limit}`;
  let recipes = [];
  let totalRecords = 0;

  return new Promise((resolve, reject) => {
    db.query(
      `SELECT id, name, description, video_link, id_user, created_at, updated_at FROM recipes ` +
        where +
        " " +
        limit,
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          recipes = results;
          db.query(
            `SELECT COUNT(*) AS total_records FROM recipes ` + where,
            (err, results) => {
              if (err) {
                console.log(err);
                reject(err);
              } else {
                totalRecords = results[0].total_records;
                resolve({
                  recipes: recipes,
                  totalRecords: totalRecords,
                });
              }
            }
          );
        }
      }
    );
  });
}

getTotalRecords(data) {
  return new Promise((resolve, reject) => {
    const inner = data.filters
      ? `INNER JOIN recipes_categories AS rc ON r.id = rc.id_recipe AND FIND_IN_SET(rc.id_category, "${data.filters}") > 0`
      : "";
    const where = `WHERE name LIKE "%${data.name}%"`;
    const order = `ORDER BY r.${data.sort} ${data.sortParam}`;
    db.query(

```

					ДП.0902.10.000000.00 ПЗ	Лист
						89
Изм	Лист	№ докум.	Подпись	Дата		

```

`SELECT COUNT(r.id) AS count_pages FROM recipes AS r ` +
    inner +
    " " +
    where +
    " " +
    order +
    " ",
    (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results[0].count_pages);
      }
    }
  );
});
}

```

```

findRecipeByExtend(extend, data) {
  return new Promise((resolve, reject) => {
    db.query(
      `SELECT * from recipes WHERE ${extend} = "${data}"`,
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results[0]);
        }
      }
    );
  });
}

```

```

updateImageRecipeById(image, id) {
  return new Promise((resolve, reject) => {
    db.query(
      `UPDATE recipes SET img = ? WHERE id = ?`,
      [image, id],
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results[0]);
        }
      }
    );
  });
}

```

```

deleteRecipe(id) {
  return new Promise((resolve, reject) => {
    db.query("DELETE FROM recipes WHERE id = ?", [id], (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results);
      }
    });
  });
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						90
Изм	Лист	№ докум.	Подпись	Дата		

```

    });
  }

  deleteUserRecipes(id) {
    return new Promise((resolve, reject) => {
      db.query(
        "DELETE FROM recipes WHERE id_user = ?",
        [id],
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {
            resolve(results);
          }
        }
      );
    });
  }
}

module.exports = new RecipeModel();

```

Программный код файла model.refreshToken

```

const db = require("../config/db.config.js");
const config = require("../config/auth.config");
const { v4: uuidv4 } = require("uuid");

class RefreshTokenModel {
  insertToken(data) {
    let expiredAt = new Date();
    expiredAt.setSeconds(expiredAt.getSeconds() + config.jwtRefreshExpiration);
    const newData = {
      id_user: data.id_user,
      token: uuidv4(),
      expiry_date: expiredAt,
    };
    return new Promise((resolve, reject) => {
      db.query("INSERT INTO refresh_token SET ?", [newData], (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results);
        }
      });
    });
  }

  findTokenByExtend(extend, data) {
    return new Promise((resolve, reject) => {
      db.query(
        `SELECT * from refresh_token WHERE ${extend} = "${data}"`,
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {
            resolve(results);
          }
        }
      );
    });
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						91
Изм	Лист	№ докум.	Подпись	Дата		

```

        resolve(results[0]);
    }
}
);
});
}

destroyTokenByExtend(extend, data) {
    return new Promise((resolve, reject) => {
        db.query(
            `DELETE from refresh_token WHERE ${extend} = "${data}"`,
            (err, results) => {
                if (err) {
                    console.log(err);
                    reject(err);
                } else {
                    resolve(results);
                }
            }
        );
    });
}

verifyExpiration(token) {
    return token.expiry_date.getTime() < new Date().getTime();
}
}

module.exports = new RefreshTokenModel();

```

Программный код файла model.roles

```

const db = require("../config/db.config.js");

class RoleModel {
    getRoles() {
        return new Promise((resolve, reject) => {
            db.query("SELECT * FROM roles", (err, results) => {
                if (err) {
                    console.log(err);
                    reject(err);
                } else {
                    resolve(results);
                }
            });
        });
    }

    getRole(data) {
        return new Promise((resolve, reject) => {
            db.query(
                `SELECT name from Roles WHERE id_role = ?`,
                [data],
                (err, results) => {
                    if (err) {
                        console.log(err);
                        reject(err);
                    } else {
                        resolve(results[0]);
                    }
                }
            );
        });
    }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						92
Изм	Лист	№ докум.	Подпись	Дата		

```

    }
  }
  );
});
}

getIdRole(data) {
  return new Promise((resolve, reject) => {
    db.query(
      `SELECT id_role from Roles WHERE name = ?`,
      [data],
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results[0]);
        }
      }
    );
  });
}

module.exports = new RoleModel();

```

Программный код файла model.users

```

const db = require("../config/db.config.js");

class UserModel {
  getUsersWithoutAdmins() {
    return new Promise((resolve, reject) => {
      db.query(
        "SELECT id_user, login, email, id_role, is_activated FROM users WHERE id_role != 3",
        (err, results) => {
          if (err) {
            console.log(err);
            reject(err);
          } else {
            resolve(results);
          }
        }
      );
    });
  }

  insertUser(data) {
    return new Promise((resolve, reject) => {
      db.query("INSERT INTO Users SET ?", [data], (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results);
        }
      });
    });
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						93
Изм	Лист	№ докум.	Подпись	Дата		

```

findUserByExtend(extend, data) {
  return new Promise((resolve, reject) => {
    db.query(
      `SELECT * from USERS WHERE ${extend} = "${data}"`,
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results[0]);
        }
      }
    );
  });
}

updateUser(id, data) {
  return new Promise((resolve, reject) => {
    db.query(
      `UPDATE users SET ? WHERE id_user = ?`,
      [data, id],
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results[0]);
        }
      }
    );
  });
}

updateImageUserById(image, id) {
  return new Promise((resolve, reject) => {
    db.query(
      `UPDATE USERS SET img = ? WHERE id_user = ?`,
      [image, id],
      (err, results) => {
        if (err) {
          console.log(err);
          reject(err);
        } else {
          resolve(results[0]);
        }
      }
    );
  });
}

deleteUser(id) {
  return new Promise((resolve, reject) => {
    db.query("DELETE FROM users WHERE id_user = ?", [id], (err, results) => {
      if (err) {
        console.log(err);
        reject(err);
      } else {
        resolve(results);
      }
    });
  });
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						94
Изм	Лист	№ докум.	Подпись	Дата		

```

    }
}

module.exports = new UserModel();

```

Программный код файла controller.auth

```

const config = require("../config/auth.config");
const jwt = require("jsonwebtoken");
const bcrypt = require("bcryptjs");
const uuid = require("uuid");
const mailService = require("../service/mail-service.js");
const userModel = require("../models/model.users.js");
const roleModel = require("../models/model.roles.js");
const refreshTokenModel = require("../models/model.refreshToken.js");
const ApiError = require("../exceptions/api-error");

class AuthController {
  async signup(req, res, next) {
    try {
      const data = {
        login: req.body.login,
        email: req.body.email,
        password: bcrypt.hashSync(req.body.password, 8),
        email_token: uuid.v4(),
        created: new Date(),
        id_role: 1,
      };
      const role = await roleModel.getIdRole(req.body.role);
      data.id_role = role.id_role;
      await userModel.insertUser(data);
      await mailService.sendActivationMail(
        data.email,
        `${process.env.API_URL}/api/activate/${data.email_token}`,
        "Активация аккаунта"
      );
      return res.send("Пользователь создан");
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async signin(req, res, next) {
    try {
      const user = await userModel.findUserByExtend("email", req.body.email);
      if (!user) {
        return next(ApiError.Error(401, "Email не найден"));
      }
      if (!user.is_activated) {
        return next(ApiError.Error(401, "Email не подтверждён"));
      }
      const passwordIsValid = bcrypt.compareSync(
        req.body.password,
        user.password
      );
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						95
Изм	Лист	№ докум.	Подпись	Дата		

```

    if (!passwordIsValid) {
        return next(ApiError.Error(401, "Неверный пароль"));
    }

    const token = jwt.sign({ id: user.id_user }, config.secret, {
        expiresIn: config.jwtExpiration,
    });

    let refreshToken = await refreshTokenModel.findTokenByExtend(
        "id_user",
        user.id_user
    );

    if (refreshToken != null) {
        await refreshTokenModel.destroyTokenByExtend(
            "id_token",
            refreshToken.id_token
        );
    }

    await refreshTokenModel.insertToken(user);

    refreshToken = await refreshTokenModel.findTokenByExtend(
        "id_user",
        user.id_user
    );

    const role = await roleModel.getRole(user.id_role);

    return res.status(200).send({
        id: user.id_user,
        username: user.login,
        email: user.email,
        role: role.name,
        accessToken: token,
        refreshToken: refreshToken.token,
    });
} catch (err) {
    if (err instanceof ApiError) {
        return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
}

}

async refreshToken(req, res, next) {
    const { refreshToken: requestToken } = req.body;

    if (requestToken == null) {
        return next(ApiError.Error(403, "Refresh Token обязательный"));
    }

    try {
        let refreshToken = await refreshTokenModel.findTokenByExtend(
            "token",
            requestToken
        );

        if (!refreshToken) {
            return next(
                ApiError.Error(403, "Refresh token отсутствует в базе данных!")
            );
        }
    }

```

					ДП.0902.10.000000.00 ПЗ	Лист
						96
Изм	Лист	№ докум.	Подпись	Дата		


```

    }

    if (refreshTokenModel.verifyExpiration(refreshToken)) {
      await refreshTokenModel.destroyTokenByExtend("token", refreshToken);
      return next(
        ApiError.Error(
          403,
          "Срок действия Refresh token истек. Пожалуйста сделайте новый signin request"
        )
      );
    }
  }

  const user = await userModel.findUserByExtend(
    "id_user",
    refreshToken.id_user
  );
  let newAccessToken = jwt.sign({ id: user.id_user }, config.secret, {
    expiresIn: config.jwtExpiration,
  });

  return res.status(200).json({
    accessToken: newAccessToken,
    refreshToken: refreshToken.token,
  });
} catch (err) {
  if (err instanceof ApiError) {
    return next(err);
  }
  next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
}
}

async activate(req, res, next) {
  try {
    const emailToken = req.params.link;
    const user = await userModel.findUserByExtend("email_token", emailToken);
    if (!user) {
      return next(ApiError.Error(404, "Некорректная ссылка активации"));
    }
    await userModel.updateUser(user.id_user, { is_activated: 1 });
    return res.redirect(process.env.CLIENT_URL);
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async forgotPassword(req, res, next) {
  try {
    const user = await userModel.findUserByExtend("email", req.body.email);
    if (!user) {
      return next(ApiError.Error(401, "Email не найден"));
    }
    const emailToken = uuid.v4();
    await userModel.updateUser(user.id_user, { email_token: emailToken });
    await mailService.sendActivationMail(
      req.body.email,
      `${process.env.API_URL}/api/fogort-password/${emailToken}`,
      "Сброс пароля"
    );
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						97
Изм	Лист	№ докум.	Подпись	Дата		

```

        return res.status(200).json({
            emailToken: emailToken,
        });
    } catch (err) {
        if (err instanceof ApiError) {
            return next(err);
        }
        next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
}

async isUserForgotPassword(req, res, next) {
    try {
        const emailToken = req.params.link;
        const user = await userModel.findUserByExtend("email_token", emailToken);
        if (!user) {
            return next(ApiError.Error(404, "Некорректная ссылка активации"));
        }
        if (user.is_fogort_password) {
            return res.redirect(process.env.CLIENT_URL + "/change-password");
        }

        await userModel.updateUser(user.id_user, { is_fogort_password: 1 });
        return res.redirect(process.env.CLIENT_URL + "/change-password");
    } catch (err) {
        if (err instanceof ApiError) {
            return next(err);
        }
        next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
}

async changeUserPassword(req, res, next) {
    try {
        const user = await userModel.findUserByExtend("email", req.body.email);
        if (!user) {
            return next(ApiError.Error(401, "Email не найден"));
        }
        if (!user.is_fogort_password) {
            return next(ApiError.Error(401, "Email не проверен"));
        }
        const password = bcrypt.hashSync(req.body.password, 8);
        await userModel.updateUser(user.id_user, { password });
        await userModel.updateUser(user.id_user, { is_fogort_password: 0 });
        return res.send("Пароль изменен");
    } catch (err) {
        if (err instanceof ApiError) {
            return next(err);
        }
        next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
}

module.exports = new AuthController();

```

Программный код файла controller.categories

```
const CategoryModel = require("../models/model.categories.js");
```

					ДП.0902.10.000000.00 ПЗ	Лист
						98
Изм	Лист	№ докум.	Подпись	Дата		

```

const RecipeCategoryModel = require("../models/model.recipesCategories.js");

class CategoryController {
  async getCategories(req, res, next) {
    try {
      const categorys = await CategoryModel.getCategories();
      return res.json(categorys);
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async addCategory(req, res, next) {
    try {
      await CategoryModel.insertCategory(req.body);
      return res.send("Категория добавлена");
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async updateCategory(req, res, next) {
    try {
      await CategoryModel.updateCategory(req.params.id, req.body);
      return res.send("Категория обновлена");
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async deleteCategory(req, res, next) {
    try {
      await RecipeCategoryModel.deleteRecipeCategoriesByldCategory(
        req.params.id
      );
      await CategoryModel.deleteCategory(req.params.id);
      return res.send("Категория удалена");
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }
}

module.exports = new CategoryController();

```

Программный код файла controller.categoryGroups

					ДП.0902.10.000000.00 ПЗ	Лист
						99
Изм	Лист	№ докум.	Подпись	Дата		

```

const CategoryGroupModel = require("../models/model.categoryGroups.js");
const CategoryModel = require("../models/model.categories.js");
const RecipeCategoryModel = require("../models/model.recipesCategories.js");

class CategoryGroupController {
  async getCategoryGroups(req, res, next) {
    try {
      const categoryGroups = await CategoryGroupModel.getCategoryGroups();
      return res.json(categoryGroups);
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async addCategoryGroup(req, res, next) {
    try {
      await CategoryGroupModel.insertCategoryGroup(req.body);
      return res.send("Группа категорий добавлена");
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async updateCategoryGroup(req, res, next) {
    try {
      await CategoryGroupModel.updateCategoryGroup(req.params.id, req.body);
      return res.send("Группа категорий обновлена");
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async deleteCategoryGroup(req, res, next) {
    try {
      const categories = await CategoryModel.getCategoriesByGroupId(
        req.params.id
      );
      if (categories.length) {
        for (let category of categories) {
          await RecipeCategoryModel.deleteRecipeCategoriesByIdCategory(
            category.id
          );
          await CategoryModel.deleteCategory(category.id);
        }
      }
      await CategoryGroupModel.deleteCategoryGroup(req.params.id);
      return res.send("Группа категорий удалена");
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						100
Изм	Лист	№ докум.	Подпись	Дата		

```

}
}

```

```

module.exports = new CategoryGroupController();

```

Программный код файла controller.recipes

```

const RecipeModel = require("../models/model.recipes.js");
const FavoriteRecipeModel = require("../models/model.favoriteRecipes.js");
const RecipeCategoryModel = require("../models/model.recipesCategories.js");
const CategoryModel = require("../models/model.categories.js");
const UserModel = require("../models/model.users.js");
const ApiError = require("../exceptions/api-error");
const sharp = require("sharp");

async function convertImage(image) {
  try {
    const dataUrl = image;
    const imageData = dataUrl.replace(/^data:image\/\w+;base64/, "");
    const buffer = Buffer.from(imageData, "base64");
    const metadata = await sharp(buffer).metadata();
    if (!metadata.format) {
      return ApiError.Error(400, "Недопустимый формат изображения");
    }
    return buffer;
  } catch (err) {
    return ApiError.BadRequest(500, "Недопустимый запрос изображения", err);
  }
}

class RecipeController {
  async addRecipe(req, res, next) {
    try {
      // Конвертируем изображение
      const img = await convertImage(req.body.img);
      if (img instanceof ApiError) {
        return next(img);
      }

      const data = {
        name: req.body.name,
        description: req.body.description,
        video_link: req.body.video_link,
        img: img,
        id_user: req.body.id_user,
        created_at: new Date(),
        updated_at: new Date(),
      };

      // Добавляем рецепт
      const recipe = await RecipeModel.insertRecipe(data);

      // Добавляем связь категорий с рецептом
      req.body.categories.forEach(async (categoryId) => {
        await RecipeCategoryModel.insertRecipeCategory({
          id_recipe: recipe.insertId,
          id_category: categoryId,
        });
      });
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						101
Изм	Лист	№ докум.	Подпись	Дата		

```

        return res.send("Рецепт добавлен");
    } catch (err) {
        if (err instanceof ApiError) {
            return next(err);
        }
        next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
}

async updateRecipe(req, res, next) {
    try {
        let img;
        if (req.body.img) {
            // Конвертируем изображение
            img = await convertImage(req.body.img);
            if (img instanceof ApiError) {
                return next(img);
            }
        }

        const data = {
            name: req.body.name,
            description: req.body.description,
            video_link: req.body.video_link,
            img: img,
            updated_at: new Date(),
        };

        if (!img) {
            delete data.img;
        }

        // Изменяем рецепт
        const recipe = await RecipeModel.updateRecipe(req.params.id_recipe, data);

        // Удаление категорий рецепта
        await RecipeCategoryModel.deleteRecipeCategories(req.params.id_recipe);

        // Добавляем связь категорий с рецептом
        req.body.categories.forEach(async (categoryId) => {
            await RecipeCategoryModel.insertRecipeCategory({
                id_recipe: req.params.id_recipe,
                id_category: categoryId,
            });
        });
        return res.send("Рецепт обновлен");
    } catch (err) {
        if (err instanceof ApiError) {
            return next(err);
        }
        next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
}

async deleteRecipe(req, res, next) {
    try {
        const recipe = await RecipeModel.findRecipeByExtend(
            "id",
            req.params.id_recipe
        );
        if (!recipe) {
            return next(ApiError.Error(400, "Рецепта не существует"));
        }
    }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						102
Изм	Лист	№ докум.	Подпись	Дата		

```

    }
    // Удаление рецепта из избранных
    await FavoriteRecipeModel.deleteFavoriteRecipe(req.params.id_recipe);

    // Удаление категорий рецепта
    await RecipeCategoryModel.deleteRecipeCategories(req.params.id_recipe);

    // Удаление рецепта
    await RecipeModel.deleteRecipe(req.params.id_recipe);
    return res.send("Рецепт удален");
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async addFavoriteRecipe(req, res, next) {
  try {
    // Проверка на существования рецепта в связи с пользователем
    (
      await FavoriteRecipeModel.getUserFavoriteRecipes(req.body.id_user)
    ).forEach((recipe) => {
      if (recipe.id_recipe === req.body.id_recipe) {
        return next(ApiError.Error(400, "Рецепт уже существует"));
      }
    });

    const data = {
      id_user: req.body.id_user,
      id_recipe: req.body.id_recipe,
    };

    // Добавление рецепта
    await FavoriteRecipeModel.addFavoriteRecipe(data);

    return res.send("Рецепт добавлен");
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async getFavoriteRecipes(req, res, next) {
  try {
    const userFavoriteRecipes = [];
    (await FavoriteRecipeModel.getUserFavoriteRecipes(req.params.id)).forEach(
      (recipe) => {
        userFavoriteRecipes.push(recipe.id_recipe);
      }
    );

    return res.json(userFavoriteRecipes);
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						103
Изм	Лист	№ докум.	Подпись	Дата		

```

}

async deleteFavoriteRecipe(req, res, next) {
  try {
    // Проверка на существования рецепта в связи с пользователем
    const userFavoriteRecipes = [];
    (
      await FavoriteRecipeModel.getUserFavoriteRecipes(req.params.id_user)
    ).forEach((recipe) => {
      userFavoriteRecipes.push(recipe.id_recipe);
    });
    if (!userFavoriteRecipes.includes(+req.params.id_recipe)) {
      return next(ApiError.Error(400, "Рецепта не существует"));
    }

    // Удаление рецепта
    await FavoriteRecipeModel.deleteUserFavoriteRecipe(
      req.params.id_user,
      req.params.id_recipe
    );
    return res.json("Рецепт удален");
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async getRecipes(req, res, next) {
  try {
    const data = {
      page: req.query.page ? +req.query.page : 1,
      limit: req.query.limit ? +req.query.limit : 30,
      name: req.query.name ?? "",
      sort: req.query.sort ?? "name",
      sortParam: req.query.sortParam ?? "asc",
      filters: req.query.filters ?? "",
    };

    // Получаем рецепты по параметрам из data
    const recipes = await RecipeModel.getRecipes(data);
    // Получаем количество записей по параметрам из data
    const totalRecords = await RecipeModel.getTotalRecords(data);

    res.set("access-control-expose-headers", "X-Total-Count");
    res.set("x-total-count", totalRecords);
    return res.json(recipes);
  } catch (err) {
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async getUserRecipes(req, res, next) {
  try {
    const data = {
      page: req.query.page ? +req.query.page : 1,
      limit: req.query.limit ? +req.query.limit : 30,
      id_user: req.query.id_user ?? 1,
    };

    // Получаем рецепты по параметрам из data

```

					ДП.0902.10.000000.00 ПЗ	Лист
						104
Изм	Лист	№ докум.	Подпись	Дата		


```

const result = await RecipeModel.getUserRecipes(data);
const recipes = result.recipes;
const totalRecords = result.totalRecords;

res.set("access-control-expose-headers", "X-Total-Count");
res.set("x-total-count", totalRecords);
return res.json(recipes);
} catch (err) {
  if (err instanceof ApiError) {
    return next(err);
  }
  next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
}
}

async getUserFavoriteRecipes(req, res, next) {
  try {
    const userFavoriteRecipes = [];
    (
      await FavoriteRecipeModel.getUserFavoriteRecipes(req.query.id_user)
    ).forEach((recipe) => {
      userFavoriteRecipes.push(recipe.id_recipe);
    });
    if (!userFavoriteRecipes.length) return res.json(userFavoriteRecipes);
    const data = {
      page: req.query.page ? +req.query.page : 1,
      limit: req.query.limit ? +req.query.limit : 30,
      recipes: userFavoriteRecipes.join(","),
    };

    // Получаем рецепты по параметрам из data
    const result = await RecipeModel.getUserFavoriteRecipes(data);
    const recipes = result.recipes;
    const totalRecords = result.totalRecords;

    res.set("access-control-expose-headers", "X-Total-Count");
    res.set("x-total-count", totalRecords);
    return res.json(recipes);
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async getRecipeById(req, res, next) {
  try {
    // Получаем рецепт по его идентификатору
    const recipe = await RecipeModel.findRecipeByExtend("id", req.params.id);
    delete recipe.img;

    // Получаем категории рецепта
    const categories =
      await RecipeCategoryModel.getRecipeCategoriesByRecipeId(req.params.id);

    // Формируем массив с категориями рецепта
    recipe.categories = [];
    categories.forEach(async (category) => {
      recipe.categories.push(
        (await CategoryModel.getCategoryById(category.id_category))
          .category_name
      );
    });
  }
}

```

```

    );
  });

  // Получаем логин пользователя добавивший рецепт
  const user = await UserModel.findUserByExtend("id_user", recipe.id_user);
  recipe.user_login = user.login;

  return res.json(recipe);
} catch (err) {
  if (err instanceof ApiError) {
    return next(err);
  }
  next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
}
}

async getRecipeCategories(req, res, next) {
  try {
    const recipeCategories = await RecipeCategoryModel.getRecipeCategories();
    return res.json(recipeCategories);
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async getImage(req, res, next) {
  try {
    const result = await RecipeModel.findRecipeByExtend("id", req.params.id);
    if (!result.img) {
      return res.status(200).send(result.img);
    }
    const metadata = await sharp(result.img).metadata();
    res.set("Content-Type", `image/${metadata.format}`);
    return res.status(200).send(result.img);
  } catch (err) {
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

module.exports = new RecipeController();

```

Программный код файла controller.roles

```

const RoleModel = require("../models/model.roles.js");
class RoleController {
  async getRoles(req, res, next) {
    try {
      const roles = await RoleModel.getRoles();
      return res.json(roles);
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						106
Изм	Лист	№ докум.	Подпись	Дата		

```

    }
}

module.exports = new RoleController();

```

Программный код файла controller.users

```

const UserModel = require("../models/model.users.js");
const FavoriteRecipeModel = require("../models/model.favoriteRecipes.js");
const RecipeModel = require("../models/model.recipes.js");
const RecipeCategoryModel = require("../models/model.recipesCategories.js");
const RefreshTokenModel = require("../models/model.refreshToken.js");
const ApiError = require("../exceptions/api-error");
const sharp = require("sharp");
const bcrypt = require("bcryptjs");
const uuid = require("uuid");
const mailService = require("../service/mail-service.js");
class UserController {
  async getUsersWithoutAdmins(req, res, next) {
    try {
      const users = await UserModel.getUsersWithoutAdmins();
      return res.json(users);
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async addUser(req, res, next) {
    try {
      console.log(req.body);
      const data = {
        login: req.body.login,
        email: req.body.email,
        password: bcrypt.hashSync(req.body.password, 8),
        email_token: uuid.v4(),
        created: new Date(),
        id_role: req.body.id_role,
      };
      await UserModel.insertUser(data);
      await mailService.sendActivationMail(
        data.email,
        `${process.env.API_URL}/api/activate/${data.email_token}`,
        "Активация аккаунта"
      );
      return res.send("Пользователь добавлен");
    } catch (err) {
      if (err instanceof ApiError) {
        return next(err);
      }
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  }

  async updateUser(req, res, next) {
    try {
      await UserModel.updateUser(req.params.id, req.body.data);
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						107
Изм	Лист	№ докум.	Подпись	Дата		

```

const user = await UserModel.findUserByExtend("id_user", req.params.id);
return res.json(user);
} catch (err) {
  if (err instanceof ApiError) {
    return next(err);
  }
  next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
}
}

async deleteUser(req, res, next) {
  try {
    const recipes = await RecipeModel.getAllUserRecipes(req.params.id);
    if (recipes.length) {
      const recipesId = recipes.map((el) => el.id).join(",");
      await RecipeCategoryModel.deleteAllRecipeCategories(recipesId);
      await RecipeModel.deleteUserRecipes(req.params.id);
    }

    await FavoriteRecipeModel.deleteAllUserFavoriteRecipes(req.params.id);
    await RefreshTokenModel.destroyTokenByExtend("id_user", req.params.id);
    await UserModel.deleteUser(req.params.id);
    return res.send("Пользователь удален");
  } catch (err) {
    if (err instanceof ApiError) {
      return next(err);
    }
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async uploadImage(req, res, next) {
  try {
    const dataUrl = req.body.image;
    const imageData = dataUrl.replace(/^data:image\/\w+;base64/, "");
    const buffer = Buffer.from(imageData, "base64");
    const metadata = await sharp(buffer).metadata();
    if (!metadata.format) {
      return next(ApiError.Error(400, "Недопустимый формат изображения"));
    }
    await UserModel.updateImageUserById(buffer, req.body.id_user);
    return res.send("Изображение сохранено");
  } catch (err) {
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}

async getImage(req, res, next) {
  try {
    const result = await UserModel.findUserByExtend("id_user", req.params.id);
    if (!result.img) {
      return res.status(200).send(result.img);
    }
    const metadata = await sharp(result.img).metadata();
    res.set("Content-Type", `image/${metadata.format}`);
    return res.status(200).send(result.img);
  } catch (err) {
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
}
}

```

```
module.exports = new UserController();
```

Программный код файла api-error

```
module.exports = class ApiError extends Error {
  status;
  error;

  constructor(status, message, error = null) {
    super(message);
    this.status = status;
    this.error = error;
  }

  static Error(status, message) {
    return new ApiError(status, message);
  }

  static BadRequest(status, message, error) {
    return new ApiError(status, message, error);
  }
};
```

Программный код файла authJwt

```
const jwt = require("jsonwebtoken");
const config = require("../config/auth.config.js");
const userModel = require("../models/model.users.js");
const roleModel = require("../models/model.roles.js");
const ApiError = require("../exceptions/api-error");
const { TokenExpiredError } = jwt;
const ROLES = require("../config/roles.config.js");
class AuthJwt {
  verifyToken = (req, res, next) => {
    try {
      const token = req.headers["x-access-token"];

      if (!token) {
        return next(ApiError.Error(403, "Токен не предоставлен!"));
      }

      jwt.verify(token, config.secret, (err, decoded) => {
        if (err) {
          if (err instanceof TokenExpiredError) {
            return next(
              ApiError.Error(
                401,
                "Неавтоизованы! Срок действия Access Token истек!"
              )
            );
          }
        }
        return next(ApiError.Error(401, "Неавтоизованы"));
      });
      req.id_user = decoded.id;
      next();
    });
  };
}
```

					ДП.0902.10.000000.00 ПЗ	Лист
						109
Изм	Лист	№ докум.	Подпись	Дата		

```

    } catch (err) {
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  };

#verifyRole = async (req, res, next) => {
  try {
    const user = await userModel.findUserByExtend("id_user", req.id_user);
    const role = await roleModel.getRole(user.id_role);
    if (role.name !== req.role) {
      return next(ApiError.Error(403, `Только для ${req.role}`));
    }
    next();
  } catch (err) {
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
};

isAdmin = async (req, res, next) => {
  req.role = ROLES[1]; //admin
  await this.#verifyRole(req, res, next);
};

isModerator = async (req, res, next) => {
  req.role = ROLES[2]; //moderator
  await this.#verifyRole(req, res, next);
};

isUser = async (req, res, next) => {
  req.role = ROLES[0]; //user
  await this.#verifyRole(req, res, next);
};

verifyRecipeChanged = async (req, res, next) => {
  try {
    const user = await userModel.findUserByExtend("id_user", req.id_user);
    const role = await roleModel.getRole(user.id_role);
    if (
      req.id_user !== req.params.id_user &&
      role.name !== ROLES[2] &&
      role.name !== ROLES[1]
    ) {
      return next(ApiError.Error(403, `Запрещено для вас`));
    }
    next();
  } catch (error) {
    next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
  }
};

module.exports = new AuthJwt();

```

Программный код файла error-middleware

```

const ApiError = require("../exceptions/api-error");

module.exports = function (err, req, res, next) {
  console.log(err);

```

					ДП.0902.10.000000.00 ПЗ	Лист
						110
Изм	Лист	№ докум.	Подпись	Дата		

```

if (err instanceof ApiError) {
  return res
    .status(err.status)
    .json({ message: err.message, error: err.error });
}
return res.status(500).json({ message: "Неожиданная ошибка" });
};

```

Программный код файла verifySignUp

```

const ROLES = require("../config/roles.config.js");
const ApiError = require("../exceptions/api-error");
const userModel = require("../models/model.users.js");

class VerifySignUp {
  checkDuplicateLoginOrEmail = async (req, res, next) => {
    try {
      let user = await userModel.findUserByExtend("login", req.body.login);
      if (user) {
        return next(ApiError.Error(400, "Логин уже используется"));
      }
      user = await userModel.findUserByExtend("email", req.body.email);
      if (user) {
        return next(ApiError.Error(400, "Email уже используется"));
      }
      next();
    } catch (err) {
      next(ApiError.BadRequest(500, "Недопустимый запрос к базе данных", err));
    }
  };

  checkRoleExisted = (req, res, next) => {
    if (!req.body.role) {
      return next(ApiError.Error(400, "Роль не найдена"));
    }
    if (!ROLES.includes(req.body.role)) {
      return next(ApiError.Error(400, "Роль не существует = " + req.body.role));
    }
    next();
  };
}

module.exports = new VerifySignUp();

```

Программный код файла mail-service

```

const nodemailer = require("nodemailer");
class MailService {
  constructor() {
    this.transporter = nodemailer.createTransport({
      host: process.env.SMTP_HOST,
      port: process.env.SMTP_PORT,
      secure: false,
      auth: {
        user: process.env.SMTP_USER,
        pass: process.env.SMTP_PASSWORD,
      },
    });
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						111
Изм	Лист	№ докум.	Подпись	Дата		

```

    },
  });
}

async sendActivationMail(to, link, subject) {
  await this.transporter.sendMail({
    from: process.env.SMTP_USER,
    to,
    subject,
    text: "",
    html: `
      <div>
        <h1>${subject}</h1>
        <a href="${link}">${link}</a>
      </div>
    `,
  });
}

module.exports = new MailService();

```

Программный код файла routes

```

const express = require("express");
const router = express.Router();
const authJwt = require("../middleware/authJwt.js");
const verifySignUp = require("../middleware/verifySignUp.js");
const authController = require("../controllers/controller.auth.js");
const userController = require("../controllers/controller.users.js");
const roleController = require("../controllers/controller.roles.js");
const categoryController = require("../controllers/controller.categories.js");
const categoryGroupController = require("../controllers/controller.categoryGroups.js");
const recipeController = require("../controllers/controller.recipes.js");

// Auth controllers

router.post(
  "/api/auth/signup",
  [verifySignUp.checkDuplicateLoginOrEmail, verifySignUp.checkRoleExisted],
  authController.signup
);

router.post("/api/auth/signin", authController.signin);

router.get("/api/activate/:link", authController.activate);

router.post("/api/auth/refreshToken", authController.refreshToken);

router.post("/api/fogort-password", authController.forgotPassword);

router.get("/api/fogort-password/:link", authController.isUserForgotPassword);

router.patch("/api/change-password", authController.changeUserPassword);

// Roles

router.get(
  "/api/roles",

```

					ДП.0902.10.000000.00 ПЗ	Лист
						112
Изм	Лист	№ докум.	Подпись	Дата		


```

    [authJwt.verifyToken, authJwt.isAdmin],
    roleController.getRoles
  );

  // User controllers

  router.get(
    "/api/users/without-admins",
    [authJwt.verifyToken, authJwt.isAdmin],
    userController.getUsersWithoutAdmins
  );

  router.post(
    "/api/user/upload-image",
    [authJwt.verifyToken],
    userController.uploadImage
  );
  router.get("/api/user/get-image/:id", userController.getImage);
  router.patch("/api/user/:id", [authJwt.verifyToken], userController.updateUser);
  router.delete(
    "/api/user/:id",
    [authJwt.verifyToken, authJwt.isAdmin],
    userController.deleteUser
  );
  router.put(
    "/api/user/",
    [authJwt.verifyToken, authJwt.isAdmin],
    userController.addUser
  );

  // Categorys controllers

  router.get("/api/categories", categoryController.getCategories);
  router.put(
    "/api/categories",
    [authJwt.verifyToken, authJwt.isAdmin],
    categoryController.addCategory
  );
  router.patch(
    "/api/categories/:id",
    [authJwt.verifyToken, authJwt.isAdmin],
    categoryController.updateCategory
  );
  router.delete(
    "/api/categories/:id",
    [authJwt.verifyToken, authJwt.isAdmin],
    categoryController.deleteCategory
  );

  // CategoryGroups controllers

  router.get("/api/category-groups", categoryGroupController.getCategoryGroups);
  router.put(
    "/api/category-groups",
    [authJwt.verifyToken, authJwt.isAdmin],
    categoryGroupController.addCategoryGroup
  );
  router.patch(
    "/api/category-groups/:id",
    [authJwt.verifyToken, authJwt.isAdmin],
    categoryGroupController.updateCategoryGroup
  );

```

					ДП.0902.10.000000.00 ПЗ	Лист
						113
Изм	Лист	№ докум.	Подпись	Дата		

```

router.delete(
  "/api/category-groups/:id",
  [authJwt.verifyToken, authJwt.isAdmin],
  categoryGroupController.deleteCategoryGroup
);

// Recipes controllers

router.post("/api/recipes", [authJwt.verifyToken], recipeController.addRecipe);
router.get("/api/recipes", recipeController.getRecipes);
router.get(
  "/api/recipes/user-recipes",
  [authJwt.verifyToken],
  recipeController.getUserRecipes
);
router.get(
  "/api/recipes/user-favorite-recipes",
  [authJwt.verifyToken],
  recipeController.getUserFavoriteRecipes
);
router.get("/api/recipes/categories", recipeController.getRecipeCategories);
router.get("/api/recipes/get-image/:id", recipeController.getImage);

router.get("/api/recipe/:id", recipeController.getRecipeById);
router.delete(
  "/api/recipe/:id_recipe/:id_user",
  [authJwt.verifyToken, authJwt.verifyRecipeChanged],
  recipeController.deleteRecipe
);
router.patch(
  "/api/recipe/:id_recipe/:id_user",
  [authJwt.verifyToken, authJwt.verifyRecipeChanged],
  recipeController.updateRecipe
);

router.post(
  "/api/recipes/favorite-recipe/",
  [authJwt.verifyToken],
  recipeController.addFavoriteRecipe
);
router.get(
  "/api/recipes/favorite-recipe/:id",
  [authJwt.verifyToken],
  recipeController.getFavoriteRecipes
);
router.delete(
  "/api/recipes/favorite-recipe/:id_user/:id_recipe",
  [authJwt.verifyToken],
  recipeController.deleteFavoriteRecipe
);

module.exports = router;

```

Программный код файла index

```

require("dotenv").config();
const express = require("express");
const cors = require("cors");
const Router = require("../routes/routes.js");

```

					ДП.0902.10.000000.00 ПЗ	Лист
						114
Изм	Лист	№ докум.	Подпись	Дата		

```

const errorMiddleware = require("./middleware/error-middleware.js");
const PORT = process.env.PORT || 5000;
const app = express();

const corsOptions = {
  origin: "http://localhost:5173",
};

app.use(cors(corsOptions));
app.use(express.json({ limit: "50mb" }));
app.use(
  express.urlencoded({ limit: "50mb", extended: true, parameterLimit: 50000 })
);
app.use(function (req, res, next) {
  res.header(
    "Access-Control-Allow-Headers",
    "x-access-token, Origin, Content-Type, Accept"
  );
  next();
});
app.use(Router);
app.use(errorMiddleware);

app.listen(PORT, () =>
  console.log(`Server running at http://localhost:${PORT}`)
);

```

Программный код файла App

```

<template>
  <div class="fl-container">
    <main-navbar
      style="margin: 40px 0 60px 0"
      @show-dialog="handlerShowDialog"
      @show-menu="showMenu = true"
    ></main-navbar>
    <side-menu v-model:show="showMenu"></side-menu>
    <main>
      <main-dialog v-model:show="showDialog">
        <div v-if="typeConn == 'signin'">
          <authorization-form
            @hide-dialog="showDialog = false"
          ></authorization-form>
        </div>
        <div v-else-if="typeConn == 'signup'">
          <registration-form
            @hide-dialog="showDialog = false"
          ></registration-form>
        </div>
      </main-dialog>
      <router-view></router-view>
    </main>
    <main-footer></main-footer>
  </div>
</template>

<script>
import { RouterView } from "vue-router";
import MainNavbar from "@components/MainNavbar.vue";

```

					ДП.0902.10.000000.00 ПЗ	Лист
						115
Изм	Лист	№ докум.	Подпись	Дата		

```

import MainFooter from "@/components/MainFooter.vue";
import MainDialog from "@/components/MainDialog.vue";
import SideMenu from "@/components/SideMenu.vue";
import AuthorizationForm from "@/components/AuthorizationForm.vue";
import RegistrationForm from "@/components/RegistrationForm.vue";
import EventBus from "@/common/EventBus";
import { useAuthStore } from "@/stores/auth";
export default {
  components: {
    MainNavbar,
    MainFooter,
    MainDialog,
    SideMenu,
    RouterView,
    AuthorizationForm,
    RegistrationForm,
  },
  setup() {
    const auth = useAuthStore();
    const { logout } = auth;
    return {
      logout,
    };
  },
  data() {
    return {
      showDialog: false,
      showMenu: false,
      typeConn: "",
    };
  },
  methods: {
    handlerShowDialog(typeConn) {
      this.typeConn = typeConn;
      this.showDialog = true;
    },
  },
  mounted() {
    EventBus.on("logout", () => {
      this.$router.push("/");
      this.logout();
    });
  },
  beforeDestroy() {
    EventBus.remove("logout");
  },
};
</script>

<style scoped>
.fl-container {
  position: relative;
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

main {
  flex: 1 1 auto;
  height: 100%;
}
</style>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						116
Изм	Лист	№ докум.	Подпись	Дата		

Программный код файла AboutView

```
<template>
<section class="about">
  <div class="container">
    <h1>О нас</h1>
    <p class="desc">
      Данный проект разработан и поддерживается одним человеком в качестве
      дипломной работы.
    </p>
    <div class="contact-wrapper">
      <p>Контакты для связи:</p>
      <ul>
        <li>Электронная почта: NiKufiz@yandex.ru</li>
        <li>
          Мой телеграм:
          <a target="_blank" href="https://t.me/Nikiuz">
            https://t.me/Nikiuz</a>
          </li>
        </li>
      </ul>
    </div>
  </div>
</section>
</template>

<style scoped>
.about {
  position: relative;
}

.contact-wrapper {
  margin-top: 60px;
  margin-bottom: 150px;
}
</style>
```

Программный код файла AddRecipeView

```
<template>
<section class="not-found">
  <div class="container">
    <h1>Добавить рецепт</h1>
    <div class="add-recipe">
      <Form
        @submit="handleAddRecipe"
        :validation-schema="schema"
        class="add-recipe-form"
      >
        <div class="add-recipe-form__content-wrapper">
          <label for="name">Название рецепта</label>
          <Field
            class="add-recipe-form__form-input"
            type="text"
            name="name"
          ></Field>
          <ErrorMessage name="name" class="add-recipe-form__error" />
        </div>
      </Form>
    </div>
  </div>
</section>
</template>
```

					ДП.0902.10.000000.00 ПЗ	Лист
						117
Изм	Лист	№ докум.	Подпись	Дата		

```

<label for="video_link">Ссылка на видео с рецептом</label>
<Field
  class="add-recipe-form__form-input"
  type="text"
  name="video_link"
>/Field>
<ErrorMessage name="video_link" class="add-recipe-form__error" />

<label for="description">Описание</label>
<Field
  as="textarea"
  class="add-recipe-form__form-input"
  name="description"
  rows="3"
/>
<ErrorMessage name="description" class="add-recipe-form__error" />

<label for="categories">Категории</label>
<Field
  as="select"
  name="categories"
  class="add-recipe-form__form-select"
  multiple
>
  <optgroup
    v-for="[groupKey, groupValue] of Object.entries(groups)"
    :key="groupKey"
    :label="groupKey"
  >
    <option
      v-for="category of groupValue.categories"
      :key="category.id"
      :value="category.id"
    >
      {{ category.category_name }}
    </option>
  </optgroup>
</Field>
<p v-if="errorCategories" class="add-recipe-form__error">
  {{ errorCategories }}
</p>

<label for="image">Изображение</label>
<input
  type="file"
  name="image"
  ref="image"
  accept="image/png, image/jpeg, image/jpg"
/>
<p v-if="errorImage" class="add-recipe-form__error">
  {{ errorImage }}
</p>
</div>
<div class="add-recipe-form__btn-wrapper">
  <form-button :disabled="isLoading">Добавить</form-button>
</div>
</Form>
</div>
</div>
</section>
</template>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						118
Изм	Лист	№ докум.	Подпись	Дата		

```

<script>
import FormInput from "@/components/UI/FormInput.vue";
import FormButton from "@/components/UI/FormButton.vue";
import { Form, Field, ErrorMessage } from "vee-validate";
import * as yup from "yup";
import { storeToRefs } from "pinia";
import { useAuthStore } from "@/stores/auth";
import { useRecipeStore } from "@/stores/recipe";
import { useCategoryStore } from "@/stores/category";
export default {
  name: "add-recipeorization-form",
  components: {
    FormInput,
    FormButton,
    Form,
    Field,
    ErrorMessage,
  },
  setup() {
    const auth = useAuthStore();
    const recipe = useRecipeStore();
    const category = useCategoryStore();
    const { userData } = storeToRefs(auth);
    const { addRecipe } = recipe;
    const { getCategories } = category;
    const { categories } = storeToRefs(category);
    return {
      userData,
      addRecipe,
      categories,
      getCategories,
    };
  },
  data() {
    return {
      groups: {},
      isLoading: false,
      errorCategories: "",
      errorImage: "",
    };
  },
  computed: {
    schema() {
      return yup.object({
        name: yup.string().trim().required("Обязательное поле"),
        video_link: yup.string().trim().max(500).required("Обязательное поле"),
        description: yup.string().trim().required("Обязательное поле"),
      });
    },
  },
  mounted() {
    if (!this.userData.status.loggedIn) this.$router.push("/");
    this.loadData();
  },
  methods: {
    async loadData() {
      if (!this.categories.lenght) await this.getCategories();
      this.loadGroups();
    },

    loadGroups() {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						119
Изм	Лист	№ докум.	Подпись	Дата		

```

try {
  // Преобразуем категории в другую структуру для удобной отрисовки
  this.groups = this.categories.reduce((acc, category) => {
    if (category.group_name) {
      if (!acc[category.group_name]) {
        acc[category.group_name] = {
          categories: [],
        };
      }
      acc[category.group_name].categories.push(category);
    } else {
      if (!acc["другие"]) {
        acc["другие"] = {
          categories: [],
        };
      }
      acc["другие"].categories.push(category);
    }
    return acc;
  }, {});
} catch (error) {
  console.log(error);
}
},

async handleAddRecipe(values) {
  this.errorCategories = "";
  this.errorImage = "";
  if (!values.categories) {
    this.errorCategories = "Нужно выбрать хотя бы одну категорию";
    return;
  }
  values.img = this.$refs.image.files[0];
  if (!values.img) {
    this.errorImage = "Необходимо добавить изображение";
    return;
  }
  const fileFormats = ["png", "jpg", "jpeg", "svg"];
  const fromatOfFile = values.img.name.split(".").at(-1);
  if (!fileFormats.includes(fromatOfFile)) {
    this.errorImage = "Формат не поддерживается";
    return;
  }
  console.log(values.img);
  values.id_user = this.userData.user.id;
  this.isLoading = true;
  try {
    await this.addRecipe(values);
    this.errorCategories = "";
    this.errorImage = "";
    this.$router.push("/recipes");
  } catch (error) {
    console.log(error);
  }
  this.isLoading = false;
},
};
</script>

```

```

<style scoped>
h1 {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						120
Изм	Лист	№ докум.	Подпись	Дата		


```

margin-bottom: 60px;
}
.add-recipe {
display: flex;
justify-content: center;
margin-bottom: 150px;
}
.add-recipe-form {
display: flex;
flex-direction: column;
width: 100%;
max-width: 400px;
}

.add-recipe-form__fogort-wrapper {
display: flex;
justify-content: space-between;
}
.add-recipe-form__fogort-wrapper button {
color: var(--color-accent);
}

.add-recipe-form__error {
color: red;
}
.add-recipe-form__content-wrapper {
display: flex;
flex-direction: column;
margin-top: 10px;
}
.add-recipe-form__form-input {
padding: 2px 10px;
border: 1px solid var(--color-light-black);
border-radius: 8px;
margin: 5px 0;
}
.add-recipe-form__butn-wrapper {
display: flex;
gap: 20px;
margin-top: 30px;
}

.add-recipe-form__form-select {
padding: 10px 15px;
border: 1px solid var(--color-light-black);
border-radius: 8px;
margin: 5px 0;
}

label {
font-weight: 500;
margin-top: 5px;
}

option {
padding: 5px 0;
padding-left: 20px;
}

optgroup {
padding: 5px 0;
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						121
Изм	Лист	№ докум.	Подпись	Дата		

```

button:disabled {
  background-color: red;
}
</style>

```

Программный код файла AdminView

```

<template>
  <section class="admin">
    <div class="container">
      <h1>Панель администратора</h1>
      <div class="tab-menu">
        <button
          v-for="tab in tabs"
          :key="tab.name"
          @click="handlerTab(tab)"
          class="tab-menu__button"
          :class="tab.isSelected ? 'tab-menu__button--active' : ''"
        >
          {{ tab.name }}
        </button>
      </div>
      <component :is="selectedTab.component"></component>
    </div>
  </section>
</template>

<script>
import AdminUsers from "@/components/admin/AdminUsers.vue";
import AdminCategories from "@/components/admin/AdminCategories.vue";
import AdminGroupsCategories from "@/components/admin/AdminGroupsCategories.vue";
import { storeToRefs } from "pinia";
import { useUserStore } from "@/stores/user";
export default {
  components: {
    AdminUsers,
    AdminCategories,
    AdminGroupsCategories,
  },

  setup() {
    const user = useUserStore();
    const { userData } = storeToRefs(user);
    return {
      userData,
    };
  },

  data() {
    return {
      tabs: [
        {
          name: "Группы категорий",
          isSelected: true,
          component: "AdminGroupsCategories",
        },
        {
          name: "Категории",

```

					ДП.0902.10.000000.00 ПЗ	Лист
						122
Изм	Лист	№ докум.	Подпись	Дата		

```

        isSelected: false,
        component: "AdminCategories",
      },
      {
        name: "Пользователи",
        isSelected: false,
        component: "AdminUsers",
      },
    ],
  };
},

mounted() {
  if (this.userData?.user?.role !== "admin") {
    this.$router.push("/");
  }
},

computed: {
  selectedTab() {
    let selectedTab = this.tabs[0];
    this.tabs.forEach((tab) => {
      if (tab.isSelected) selectedTab = tab;
    });
    return selectedTab;
  },
},

methods: {
  async handlerTab(tab) {
    this.tabs.forEach((el) => {
      el.isSelected = false;
      if (el.name === tab.name) el.isSelected = true;
    });
  },
},
};
</script>

```

```

<style scoped>
.tab-menu {
  display: flex;
  gap: 20px;
  margin-bottom: 30px;
}
.tab-menu__button {
  font-weight: 500;
  padding-bottom: 3px;
  border-bottom: 3px solid transparent;
  transition: border-bottom 0.25s ease-out;
}
.tab-menu__button:hover,
.tab-menu__button--active {
  border-bottom: 3px solid var(--color-accent);
}

```

```

.admin {
  margin-bottom: 150px;
}

```

```

h1 {
  margin-bottom: 60px;
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						123
Изм	Лист	№ докум.	Подпись	Дата		

```
}  
</style>
```

Программный код файла ChangePasswordView

```
<template>  
  <section class="change-password">  
    <div class="container">  
      <h1>Изменение пароля</h1>  
      <div class="change-password_form-wrapper">  
        <Form  
          @submit="handleLogin"  
          :validation-schema="schema"  
          class="change-form"  
        >  
          <div class="change-form__content-wrapper">  
            <label for="email">E-mail</label>  
            <Field  
              class="change-form__form-input"  
              type="text"  
              name="email"  
            ></Field>  
            <ErrorMessage name="email" class="change-form__error" />  
            <label for="password">Придумайте новый пароль</label>  
            <Field  
              class="change-form__form-input"  
              type="password"  
              name="password"  
            ></Field>  
            <ErrorMessage name="password" class="change-form__error" />  
            <label for="repeatPassword">Повторите пароль</label>  
            <Field  
              class="change-form__form-input"  
              type="password"  
              name="repeatPassword"  
            ></Field>  
            <ErrorMessage name="repeatPassword" class="change-form__error" />  
          </div>  
          <p v-if="errorMessage" class="change-form__error">  
            {{ errorMessage }}  
          </p>  
          <div class="change-form__btn-wrapper">  
            <form-button :disabled="isLoading">Отправить</form-button>  
          </div>  
        </Form>  
      </div>  
    </div>  
  </section>  
</template>  
  
<script>  
import FormButton from "@/components/UI/FormButton.vue";  
import { Form, Field, ErrorMessage } from "vee-validate";  
import * as yup from "yup";  
import { storeToRefs } from "pinia";  
import { useAuthStore } from "@/stores/auth";  
export default {  
  components: {  
    FormButton,  

```

					ДП.0902.10.000000.00 ПЗ	Лист
						124
Изм	Лист	№ докум.	Подпись	Дата		

```

Form,
Field,
ErrorMessage,
},
setup() {
  const auth = useAuthStore();
  const { changeUserPassword } = auth;
  return {
    changeUserPassword,
  };
},
data() {
  return {
    errorMessage: "",
    isLoading: false,
  };
},
computed: {
  schema() {
    return yup.object({
      email: yup
        .string()
        .trim()
        .required("Обязательное поле")
        .email("Не верный формат"),
      password: yup
        .string()
        .trim()
        .min(8, "Минимум 8 символов")
        .required("Обязательное поле"),
      repeatPassword: yup
        .string()
        .trim()
        .min(8, "Минимум 8 символов")
        .required("Обязательное поле"),
    });
  },
},
methods: {
  async handleLogin(values) {
    this.errorMessage = "";
    if (values.password !== values.repeatPassword) {
      this.errorMessage = "Пароли не совпадают";
      return;
    }
    this.isLoading = true;
    try {
      await this.changeUserPassword(values.email, values.password);
      this.$router.push("/");
    } catch (error) {
      this.errorMessage = error.response.data.message;
    }
    this.isLoading = false;
  },
},
mounted() {},
};
</script>

<style scoped>
h1 {
  text-align: center;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						125
Изм	Лист	№ докум.	Подпись	Дата		

```

}

.change-password_form-wrapper {
  display: flex;
  justify-content: center;
}

.change-form {
  display: flex;
  flex-direction: column;
  margin-top: 30px;
  width: 100%;
  max-width: 400px;
}

.change-form__error {
  color: red;
}

.change-form__content-wrapper {
  display: flex;
  flex-direction: column;
  margin-top: 10px;
}

.change-form__form-input {
  padding: 2px 10px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}

.change-form__btn-wrapper {
  display: flex;
  gap: 20px;
  margin-top: 20px;
}
}
</style>

```

Программный код файла FogortPasswordView

```

<template>
<section class="fogort-password">
  <div class="container">
    <h1>Сбросить пароль</h1>
    <div class="fogort-password_form-wrapper">
      <Form
        @submit="handleLogin"
        :validation-schema="schema"
        class="fogort-form"
      >
        <div class="fogort-form__content-wrapper">
          <label for="email">E-mail</label>
          <Field
            class="fogort-form__form-input"
            type="text"
            name="email"
          ></Field>
          <ErrorMessage name="email" class="fogort-form__error" />
        </div>
        <p v-if="errorMessage" class="fogort-form__error">
          {{ errorMessage }}
        </p>
      </Form>
    </div>
  </section>
</template>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						126
Изм	Лист	№ докум.	Подпись	Дата		

```

<div v-if="successMessage" class="alert alert-success" role="alert">
  {{ successMessage }}
</div>
<div class="fogort-form__butn-wrapper">
  <form-button :disabled="isLoading">Отправить</form-button>
</div>
</Form>
</div>
</div>
</section>
</template>

```

```

<script>
import FormButton from "@/components/UI/FormButton.vue";
import { Form, Field, ErrorMessage } from "vee-validate";
import * as yup from "yup";
import { storeToRefs } from "pinia";
import { useAuthStore } from "@/stores/auth";
export default {
  components: {
    FormButton,
    Form,
    Field,
    ErrorMessage,
  },
  setup() {
    const auth = useAuthStore();
    const { fogortPassword } = auth;
    return {
      fogortPassword,
    };
  },
  data() {
    return {
      successMessage: "",
      errorMessage: "",
      isLoading: false,
    };
  },
  computed: {
    schema() {
      return yup.object({
        email: yup
          .string()
          .trim()
          .required("Обязательное поле")
          .email("Не верный формат"),
      });
    },
  },
  methods: {
    redirectToMain(delay) {
      return new Promise((resolve, reject) => {
        setTimeout(() => {
          resolve(this.$router.push("/"));
        }, delay);
      });
    },
    async handleLogin(values) {
      this.successMessage = "";
      this.errorMessage = "";
    },
  },
};

```

					ДП.0902.10.000000.00 ПЗ	Лист
						127
Изм	Лист	№ докум.	Подпись	Дата		

```

        this.isLoading = true;
        try {
            await this.fogortPassword(values.email);
            this.successMessage = "На почту отправлено письмо для сброса пароля";
            await this.redirectToMain(4000);
        } catch (error) {
            this.errorMessage = error.response.data.message;
        }
        this.isLoading = false;
    },
    },
    mounted() {},
};
</script>

```

```
<style scoped>
```

```

h1 {
  text-align: center;
}

```

```

.fogort-password_form-wrapper {
  display: flex;
  justify-content: center;
  margin-bottom: 150px;
}

```

```

.fogort-form {
  display: flex;
  flex-direction: column;
  margin-top: 30px;
  width: 100%;
  max-width: 400px;
}

```

```

.fogort-form__error {
  color: red;
}

```

```

.fogort-form__content-wrapper {
  display: flex;
  flex-direction: column;
  margin-top: 10px;
}

```

```

.fogort-form__form-input {
  padding: 2px 10px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}

```

```

.fogort-form__butn-wrapper {
  display: flex;
  gap: 20px;
  margin-top: 20px;
}

```

```
</style>
```

Программный код файла HomeView

```

<template>
  <section class="home">
    <div class="container">

```

					ДП.0902.10.000000.00 ПЗ	Лист
						128
Изм	Лист	№ докум.	Подпись	Дата		


```

<h1>Найдём рецепты по вкусу</h1>
<p class="desc">
    Этот сервис создан людьми для людей. Ищите интересующие вас рецепты и
    добавляйте собственные.
</p>

```

```

<div class="info-cards">
  <div class="info-cards__item">
    <div class="info-cards__content-wrapper">
      <div class="info-cards__content">
        <h3>Использование</h3>
        <p>
            Для просмотра рецептов вы можете перейти в соответствующий
            раздел, но для того чтобы добавлять свои рецепты необходимо
            будет авторизоваться.
          </p>
        </div>
      </div>
      <div class="info-cards__img-wrapper">
        
      </div>
    </div>
  </div>

```

```

<div class="info-cards__item info-cards__item--reverse">
  <div class="info-cards__img-wrapper">
    
  </div>
  <div class="info-cards__content-wrapper">
    <div class="info-cards__content">
      <h3>Рецепты</h3>
      <p>
          Выбирайте рецепты по интересующим категориям. У вас полная
          свобода действий во время добавления рецептуры. Делитесь
          рецептами в социальных сетях.
        </p>
      </div>
    </div>
  </div>

```

```

<div class="info-cards__item">
  <div class="info-cards__content-wrapper">
    <div class="info-cards__content">
      <h3>Заключение</h3>
      <p>
          Упростим себе поиск рецептов и поможем друг другу упростить себе
          жизнь создав для себя удобную среду где можно найти чем вкусно
          покушать.
        </p>
      </div>
    </div>
    <div class="info-cards__img-wrapper">
      
    </div>
  </div>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						129
Изм	Лист	№ докум.	Подпись	Дата		

```

        class="info-cards__img"
      />
    </div>
  </div>
</div>
</section>
</template>

```

```
<script></script>
```

```
<style scoped>
```

```
.home {
  position: relative;
}
```

```
.home .info-cards {
  margin-top: 90px;
  margin-bottom: 150px;
}
```

```
.info-cards {
  display: flex;
  flex-wrap: wrap;
  gap: 80px;
}
```

```
.info-cards__item {
  display: flex;
  width: 100%;
  padding: 0 20px;
}
```

```
.info-cards__content-wrapper {
  display: flex;
  flex: 1 1 auto;
  justify-content: center;
  align-items: center;
}
```

```
.info-cards__img-wrapper {
  display: flex;
  flex: 1 1 auto;
  justify-content: center;
}
```

```
.info-cards__content {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  max-width: 400px;
}
```

```
.info-cards__content p {
  margin-bottom: 25px;
}
```

```
.info-cards__img {
  height: 210px;
  width: 400px;
  border-radius: 30px;
}
```

```
@media (max-width: 991px) {
  .info-cards__item {
    flex-wrap: wrap;
  }
}
```

					ДП.0902.10.000000.00 ПЗ	Лист
						130
Изм	Лист	№ докум.	Подпись	Дата		

```

        justify-content: center;
    }

    .info-cards__content-wrapper {
        order: 2;
    }
    .info-cards__img-wrapper {
        order: 1;
        margin-bottom: 30px;
    }

    .info-cards__item--reverse .info-cards__content-wrapper {
        order: 2;
    }

    .info-cards__item--reverse .info-cards__img-wrapper {
        order: 1;
    }
}

@media (max-width: 480px) {
    .info-cards__img {
        height: auto;
        width: 100%;
    }

    .desc {
        font-size: 18px;
    }
}
</style>

```

Программный код файла NotFoundView

```

<template>
<section class="not-found">
<div class="container">
<h1>Ошибка 404</h1>
<p class="desc">Нет информации по данному маршруту</p>
<div class="contanct-wrapper">
<router-link to="/">Вернуться на главную страницу</router-link>
</div>
</div>
</section>
</template>

```

Программный код файла PersonView

```

<template>
<section class="profile">
<div class="container">
<h1>Личный кабинет</h1>
<div class="profile__forms-wrapper">
<form @submit.prevent class="file-upload">
<label>
<input

```

					ДП.0902.10.000000.00 ПЗ	Лист
						131
Изм	Лист	№ докум.	Подпись	Дата		

```

        name="file"
        type="file"
        ref="imageInput"
        @change="uploadImage"
      />
    </label>
    
  </form>
  <form @submit.prevent class="form-login">
    <label for="login">Логин</label>
    <input v-model="login" type="text" name="login" />
    <button
      v-if="login != userData.user.username"
      class="button"
      @click="handlerChangeLogin"
    >
      Сохранить
    </button>
  </form>
</div>
<div class="profile__recipes-wrapper">
  <div class="tab-menu">
    <button
      v-for="tab in tabs"
      :key="tab.name"
      @click="handlerTab(tab)"
      class="tab-menu__button"
      :class="tab.isSelected ? 'tab-menu__button--active' : ''"
    >
      {{ tab.name }}
    </button>
  </div>
  <template v-if="recipes.length">
    <recipe-cards
      class="recipes-offset"
      :recipes="recipes"
    ></recipe-cards>
  </template>
  <p v-else>Рецепты не найдены...</p>
  <main-pagination
    class="pagination"
    v-model:page="page"
    :countOfRecords="countOfRecords"
    :limit="limit"
  ></main-pagination>
</div>
</div>
</section>
</template>

<script>
import CategoryFilter from "@components/CategoryFilter.vue";
import RecipeCards from "@components/RecipeCards.vue";
import MainPagination from "@components/MainPagination.vue";
import RecipeService from "@services/recipe.service.js";
import { storeToRefs } from "pinia";
import { useUserStore } from "@stores/user";
export default {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						132
Изм	Лист	№ докум.	Подпись	Дата		

```

components: {
  RecipeCards,
  MainPagination,
},

setup() {
  const user = useUserStore();
  const { uploadImage, getImage, updateUser } = user;
  const { userData } = storeToRefs(user);
  return {
    userData,
    uploadImage,
    getImage,
    updateUser,
  };
},

data() {
  return {
    tabs: [
      {
        name: "Добавленные",
        isSelected: true,
      },
      {
        name: "Избранные",
        isSelected: false,
      },
    ],
    login: this.userData.user.username,
    recipes: [],
    limit: 10,
    page: 1,
    countOfRecords: 0,
  };
},

mounted() {
  if (!this.userData.user) {
    this.$router.push("/");
  }
  this.getImage();
  this.loadRecipes();
},

methods: {
  async uploadImage() {
    if (!this.userData.user) return;
    const input = this.$refs.imageInput;
    const file = input.files[0];
    await this.uploadImage(file, this.userData.user.id);
    this.userData.imageUrl =
      "http://localhost:5000/api/user/get-image/" + this.userData.user.id;
    setTimeout(() => {
      location.reload();
    }, 300);
  },

  async loadRecipes() {
    try {
      const data = {
        page: this.page,

```

					ДП.0902.10.000000.00 ПЗ	Лист
						133
Изм	Лист	№ докум.	Подпись	Дата		

```

        limit: this.limit,
        id_user: this.userData.user.id,
    });
    let result = [];
    if (this.tabs[0].isSelected)
        result = await RecipeService.getUserRecipes(data);
    else if (this.tabs[1].isSelected)
        result = await RecipeService.getUserFavoriteRecipes(data);
    this.countOfRecords = +result.headers["x-total-count"];
    this.recipes = result.data;
} catch (error) {
    console.log(error);
}
},

async handlerTab(tab) {
    this.tabs.forEach((el) => {
        el.isSelected = false;
        if (el.name === tab.name) el.isSelected = true;
    });
},

async handlerChangeLogin() {
    await this.updateUser(this.userData.user.id, { login: this.login });
},
},

watch: {
    tabs: {
        handler() {
            this.loadRecipes();
        },
        deep: true,
    },
},
};
</script>

<style scoped>
.tab-menu {
    display: flex;
    gap: 20px;
    margin-bottom: 30px;
}
.tab-menu__button {
    font-weight: 500;
    padding-bottom: 3px;
    border-bottom: 3px solid transparent;
    transition: border-bottom 0.25s ease-out;
}
.tab-menu__button:hover,
.tab-menu__button--active {
    border-bottom: 3px solid var(--color-accent);
}

h1 {
    text-align: center;
    margin-bottom: 60px;
}

.profile {
    margin-bottom: 150px;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						134
Изм	Лист	№ докум.	Подпись	Дата		

```

}

.profile__forms-wrapper {
  display: flex;
  justify-content: start;
  flex-wrap: wrap;
  gap: 20px;
}

.form-login {
  flex: 4 1 auto;
}

.file-upload {
  flex: 1 1 auto;
}

.profile__recipes-wrapper {
  margin-top: 100px;
}

.form-login {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
}

.file-upload input[type="file"] {
  display: none;
}

.file-upload {
  position: relative;
  height: 100px;
  min-width: 100px;
}

.file-upload img {
  display: block;
  object-fit: cover;
  position: absolute;
  top: 0;
  left: 0;
  width: 100px;
  height: 100px;
  overflow: hidden;
  border-radius: 30px;
  z-index: 1;
}

.file-upload label::before {
  content: "";
  position: absolute;
  display: block;
  top: 0;
  left: 0;
  width: 100px;
  height: 100px;
  overflow: hidden;
  border-radius: 30px;
  background: #000;
  opacity: 0;
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						135
Изм	Лист	№ докум.	Подпись	Дата		

```

z-index: 10;
transition: opacity 0.25s ease-out;
}

.file-upload label:hover::before {
  opacity: 0.3;
}

.file-upload label {
  display: block;
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  cursor: pointer;
  background: transparent;
  z-index: 2;
}

input {
  margin-top: 10px;
  padding: 2px 15px;
  border: 1px solid var(--color-light-black);
  border-radius: 30px;
}

.button {
  margin-top: 30px;
  padding: 10px 28px;
  line-height: 1;
  border-radius: 30px;
  font-size: 16px;
  font-weight: 500;
  color: var(--color-white);
  background: var(--color-accent);
}

.button:disabled {
  color: var(--color-white);
  background: var(--color-accent-disabled);
}

@media (max-width: 991px) {
  .form-login {
    flex: 1 1 auto;
  }
}
</style>

```

Программный код файла RecipesView

```

<template>
<section class="recipes">
  <div class="container-md">
    <h1>Ищите интересные для вас рецепты</h1>
    <p class="desc">
      Здесь вы сможете просмотреть рецепты пользуясь фильтрами по разным
      категориям и сортировкой.
    </p>
  </div>
</section>
</template>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						136
Изм	Лист	№ докум.	Подпись	Дата		


```

</p>
<form @submit.prevent class="form-filters">
  <input v-model="searchInput" type="text" placeholder="Поиск..." />
  <select v-model="sortSelect">
    <option v-for="sort in sorts" :key="sort.name" :value="sort">
      {{ sort.name }}
    </option>
  </select>
  
</form>

<div class="row">
  <div class="col-md-3" v-show="isShowCategories">
    <category-filter :selectedCategories="selectedCategories" />
  </div>

  <div class="offset-md-1 col-md-8">
    <template v-if="recipes.length">
      <recipe-cards
        class="recipes-offset"
        :recipes="recipes"
      ></recipe-cards>
    </template>
    <p v-else>Рецепты не найдены...</p>
    <main-pagination
      class="pagination"
      v-model:page="page"
      :countOfRecords="countOfRecords"
      :limit="limit"
    ></main-pagination>
  </div>
</div>
</div>
</section>
</template>

<script>
import CategoryFilter from "@components/CategoryFilter.vue";
import RecipeCards from "@components/RecipeCards.vue";
import MainPagination from "@components/MainPagination.vue";
import RecipeService from "@services/recipe.service.js";
import { storeToRefs } from "pinia";
import { useRecipeStore } from "@stores/recipe";
import { useCategoryStore } from "@stores/category";
export default {
  components: {
    CategoryFilter,
    RecipeCards,
    MainPagination,
  },

  setup() {
    const recipe = useRecipeStore();
    const category = useCategoryStore();
    const { getRecipeCategories } = recipe;
    const { recipeCategories } = storeToRefs(recipe);
    const { getCategories } = category;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						137
Изм	Лист	№ докум.	Подпись	Дата		

```

const { categories } = storeToRefs(category);

const sorts = [
  {
    name: "По названию",
    sort: "name",
    sortParam: "asc",
  },
  {
    name: "По названию (убывание)",
    sort: "name",
    sortParam: "desc",
  },
  {
    name: "По дате добавления",
    sort: "created_at",
    sortParam: "asc",
  },
  {
    name: "По дате добавления (убывание)",
    sort: "created_at",
    sortParam: "desc",
  },
];

return {
  categories,
  getCategories,
  getRecipeCategories,
  recipeCategories,
  sorts,
};
},

data() {
  return {
    searchInput: "",
    sortSelect: this.sorts[0],
    recipes: [],
    selectedCategories: [],
    isShowCategories: true,
    limit: 10,
    page: 1,
    countOfRecords: 0,
  };
},

mounted() {
  this.loadData();
},

methods: {
  async loadData() {
    if (!this.categories.lenght) await this.getCategories();
    if (!this.recipeCategories.lenght) await this.getRecipeCategories();
    await this.loadRecipes();
  },

  async loadRecipes() {
    try {
      const data = {
        name: this.searchInput,

```

					ДП.0902.10.000000.00 ПЗ	Лист
						138
Изм	Лист	№ докум.	Подпись	Дата		

```

        page: this.page,
        limit: this.limit,
        sort: this.sortSelect.sort,
        sortParam: this.sortSelect.sortParam,
        filters: this.selectedCategories,
    };
    const result = await RecipeService.getRecipes(data);
    this.countOfRecords = +result.headers["x-total-count"];
    this.recipes = result.data;
  } catch (error) {
    console.log(error);
  }
},
},

watch: {
  sortSelect() {
    this.loadRecipes();
  },

  searchInput() {
    this.loadRecipes();
  },

  page() {
    this.loadRecipes();
  },

  selectedCategories: {
    handler() {
      this.loadRecipes();
    },
    deep: true,
  },
},
};
</script>

<style scoped>
.recipes {
  margin-bottom: 150px;
}

select {
  padding: 2px 15px;
  border-radius: 30px;
}

input {
  padding: 2px 15px;
  border: 1px solid var(--color-light-black);
  border-radius: 30px;
}

.form-filters__filter {
  cursor: pointer;
}

.desc {
  margin-bottom: 90px;
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		139

```

.recipes-offset {
  margin-top: 54px;
}

.form-filters {
  display: flex;
  gap: 20px;
  margin-bottom: 50px;
  flex-wrap: wrap;
}

.pagination {
  margin-top: 60px;
}
</style>

```

Программный код файла RecipeView

```

<template>
  <section class="recipe">
    <div class="container">
      <h1>{{ recipe.name }}</h1>
      <div class="recipe__content-wrapper">
        <iframe
          width="560"
          height="315"
          :src="videLink"
          frameborder="0"
          allowfullscreen
          class="recipe__video"
        ></iframe>
        <p>{{ recipe.description }}</p>
        <p>Список категорий: {{ recipe?.categories?.join(", ") }}</p>
        <p>
          Добавлен пользователем:
          <span class="recipe__accent-content">{{ recipe.user_login }}</span>
        </p>
        <p>
          Дата добавления:
          <span class="recipe__accent-content">
            >{{
              recipe?.created_at?.slice(0, 10).split("-").reverse().join("-")
            }}
          </span>
        </p>
        <div v-if="userData.status.loggedIn" class="recipe__edit-wrapper">
          <button v-if="!isFavoriteRecipe" @click="handlerAddFavoriteRecipe">
            Добавить в избранное
          </button>
          <button v-else @click="handlerDeleteFavoriteRecipe">
            Удалить из избранного
          </button>
          <template v-if="isShowControll">
            <button @click="$router.push('/update-recipe/${recipe.id}')">
              Изменить
            </button>
            <button @click="handlerDeleteRecipe">Удалить</button>
          </template>
        </div>
      </div>
    </div>
  </section>
</template>

```

```

</div>
</div>
</section>
</template>

<script>
import RecipeService from "@/services/recipe.service.js";
import { storeToRefs } from "pinia";
import { useAuthStore } from "@/stores/auth";
export default {
  setup() {
    const auth = useAuthStore();
    const { userData, userFavoriteRecipes } = storeToRefs(auth);
    const { getUserFavoriteRecipes } = auth;
    return {
      userData,
      userFavoriteRecipes,
      getUserFavoriteRecipes,
    };
  },

  data() {
    return {
      recipe: {},
    };
  },

  mounted() {
    if (this.userData.status.loggedIn) this.getUserFavoriteRecipes();
    RecipeService.getRecipe(this.$route.params.id)
      .then((res) => {
        this.recipe = res;
      })
      .catch((error) => console.log(error));
  },

  computed: {
    videLink() {
      const videold = this.recipe?.video_link
        ?.replace("https://www.youtube.com/", "")
        .split("=")[1];
      return "https://www.youtube.com/embed/" + videold;
    },

    isFavoriteRecipe() {
      return this.userFavoriteRecipes.includes(this.recipe.id);
    },

    isShowControll() {
      return (
        this.userData.user.id === this.recipe.id_user ||
        this.userData.user.role !== "user"
      );
    },
  },

  methods: {
    async handlerAddFavoriteRecipe() {
      try {
        await RecipeService.addFavoriteRecipe({
          id_user: this.userData.user.id,
          id_recipe: this.recipe.id,
        });
      } catch (error) {
        console.log(error);
      }
    },
  },
};

```

					ДП.0902.10.000000.00 ПЗ	Лист
						141
Изм	Лист	№ докум.	Подпись	Дата		

```

    });
    await this.getUserFavoriteRecipes();
  } catch (error) {
    console.log(error);
  }
},

async handlerDeleteFavoriteRecipe() {
  try {
    await RecipeService.deleteFavoriteRecipe({
      id_user: this.userData.user.id,
      id_recipe: this.recipe.id,
    });
    await this.getUserFavoriteRecipes();
  } catch (error) {
    console.log(error);
  }
},

async handlerDeleteRecipe() {
  try {
    if (confirm("Удалить рецепт?")) {
      await RecipeService.deleteRecipe(this.recipe.id, this.recipe.id_user);
      this.$router.push("/recipes");
    }
  } catch (error) {
    console.log(error);
  }
},
},
};
</script>

```

```

<style scoped>
.recipe {
  margin-bottom: 150px;
}

h1 {
  text-align: left;
  margin-bottom: 60px;
}

.recipe__video {
  margin-bottom: 30px;
}

.recipe__accent-content {
  color: var(--color-accent);
}

.recipe__edit-wrapper {
  display: flex;
  gap: 20px;
}

button {
  padding: 10px 28px;
  line-height: 1;
  border-radius: 30px;
  font-size: 16px;
  font-weight: 500;
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						142
Изм	Лист	№ докум.	Подпись	Дата		

```

color: var(--color-white);
background: var(--color-accent);
}

button:disabled {
color: var(--color-white);
background: var(--color-accent-disabled);
}
</style>

```

Программный код файла UpdateRecipeView

```

<template>
<section class="not-found">
<div class="container">
<h1>Изменить рецепт</h1>
<div class="add-recipe">
<Form
  @submit="handleUpdateRecipe"
  :validation-schema="schema"
  class="add-recipe-form"
>
<div class="add-recipe-form__content-wrapper">
<label for="name">Название рецепта</label>
<Field
  v-model="recipe.name"
  class="add-recipe-form__form-input"
  type="text"
  name="name"
></Field>
<ErrorMessage name="name" class="add-recipe-form__error" />

<label for="video_link">Ссылка на видео с рецептом</label>
<Field
  v-model="recipe.video_link"
  class="add-recipe-form__form-input"
  type="text"
  name="video_link"
></Field>
<ErrorMessage name="video_link" class="add-recipe-form__error" />

<label for="description">Описание</label>
<Field
  v-model="recipe.description"
  as="textarea"
  class="add-recipe-form__form-input"
  name="description"
  rows="3"
/>
<ErrorMessage name="description" class="add-recipe-form__error" />

<label for="categories">Категории</label>
<Field
  v-model="selectedCategories"
  as="select"
  name="categories"
  class="add-recipe-form__form-select"
  multiple
>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						143
Изм	Лист	№ докум.	Подпись	Дата		

```

<optgroup
  v-for="[groupKey, groupValue] of Object.entries(groups)"
  :key="groupKey"
  :label="groupKey"
>
  <option
    v-for="category of groupValue.categories"
    :key="category.id"
    :value="category.id"
  >
    {{ category.category_name }}
  </option>
</optgroup>
</Field>
<p v-if="errorCategories" class="add-recipe-form__error">
  {{ errorCategories }}
</p>

<label for="image">Изображение</label>
<input
  type="file"
  name="image"
  ref="image"
  accept="image/png, image/jpeg, image/jpg"
/>
<p v-if="errorImage" class="add-recipe-form__error">
  {{ errorImage }}
</p>
</div>
<div class="add-recipe-form__btn-wrapper">
  <form-button :disabled="isLoading">Изменить</form-button>
</div>
</Form>
</div>
</div>
</section>
</template>

<script>
import RecipeService from "@services/recipe.service.js";
import FormInput from "@components/UI/FormInput.vue";
import FormButton from "@components/UI/FormButton.vue";
import { Form, Field, ErrorMessage } from "vee-validate";
import * as yup from "yup";
import { storeToRefs } from "pinia";
import { useAuthStore } from "@stores/auth";
import { useRecipeStore } from "@stores/recipe";
import { useCategoryStore } from "@stores/category";
export default {
  name: "add-recipeorization-form",
  components: {
    FormInput,
    FormButton,
    Form,
    Field,
    ErrorMessage,
  },
  setup() {
    const auth = useAuthStore();
    const recipe = useRecipeStore();
    const category = useCategoryStore();
    const { userData } = storeToRefs(auth);

```

					ДП.0902.10.000000.00 ПЗ	Лист
						144
Изм	Лист	№ докум.	Подпись	Дата		


```

const { updateRecipe } = recipe;
const { getCategories } = category;
const { categories } = storeToRefs(category);
return {
  userData,
  updateRecipe,
  categories,
  getCategories,
};
},
data() {
  return {
    recipe: {},
    groups: {},
    selectedCategories: [],
    isLoading: false,
    errorCategories: "",
    errorImage: "",
  };
},
computed: {
  schema() {
    return yup.object({
      name: yup.string().trim().required("Обязательное поле"),
      video_link: yup.string().trim().max(500).required("Обязательное поле"),
      description: yup.string().trim().required("Обязательное поле"),
    });
  },
},
mounted() {
  if (!this.userData.status.loggedIn) this.$router.push("/");
  this.loadData();
},
methods: {
  async loadData() {
    if (!this.categories.length) await this.getCategories();
    this.recipe = await RecipeService.getRecipe(this.$route.params.id);
    this.categories.forEach((category) => {
      if (this.recipe.categories.includes(category.category_name))
        this.selectedCategories.push(category.id);
    });
    this.loadGroups();
  },

  loadGroups() {
    try {
      // Преобразуем категории в другую структуру для удобной отрисовки
      this.groups = this.categories.reduce((acc, category) => {
        if (category.group_name) {
          if (!acc[category.group_name]) {
            acc[category.group_name] = {
              categories: [],
            };
          }
          acc[category.group_name].categories.push(category);
        } else {
          if (!acc["другие"]) {
            acc["другие"] = {
              categories: [],
            };
          }
          acc["другие"].categories.push(category);
        }
      }, {});
    } catch (error) {
      console.error(error);
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						145
Изм	Лист	№ докум.	Подпись	Дата		

```

    }
    return acc;
  }, {});
} catch (error) {
  console.log(error);
}
},

async handleUpdateRecipe(values) {
  if (!values.categories) {
    this.errorCategories = "Нужно выбрать хотя бы одну категорию";
    return;
  }
  values.img = this.$refs.image.files[0];
  this.isLoading = true;
  try {
    await this.updateRecipe(
      this.$route.params.id,
      this.recipe.id_user,
      values
    );
    this.errorCategories = "";
    this.errorImage = "";
    this.$router.push("/recipes");
  } catch (error) {
    console.log(error);
  }
  this.isLoading = false;
},
},
};
</script>

```

```

<style scoped>
h1 {
  margin-bottom: 60px;
}
.add-recipe {
  display: flex;
  justify-content: center;
  margin-bottom: 150px;
}
.add-recipe-form {
  display: flex;
  flex-direction: column;
  width: 100%;
  max-width: 400px;
}

.add-recipe-form__fogort-wrapper {
  display: flex;
  justify-content: space-between;
}
.add-recipe-form__fogort-wrapper button {
  color: var(--color-accent);
}

.add-recipe-form__error {
  color: red;
}
.add-recipe-form__content-wrapper {
  display: flex;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						146
Изм	Лист	№ докум.	Подпись	Дата		

```

flex-direction: column;
margin-top: 10px;
}
.add-recipe-form__form-input {
padding: 2px 10px;
border: 1px solid var(--color-light-black);
border-radius: 8px;
margin: 5px 0;
}
.add-recipe-form__butn-wrapper {
display: flex;
gap: 20px;
margin-top: 30px;
}

.add-recipe-form__form-select {
padding: 10px 15px;
border: 1px solid var(--color-light-black);
border-radius: 8px;
margin: 5px 0;
}

label {
font-weight: 500;
margin-top: 5px;
}

option {
padding: 5px 0;
padding-left: 20px;
}

optgroup {
padding: 5px 0;
}

button:disabled {
background-color: red;
}
</style>

```

Программный код файла auth

```

import { defineStore } from "pinia";
import AuthService from "@/services/auth.service";
import RecipeService from "@/services/recipe.service.js";

const user = JSON.parse(localStorage.getItem("user"));
const DEFAULT_AVATAR = "http://localhost:5173/src/assets/avatar.png"; //Путь к дефолтному аватару
const USER_AVATAR = "http://localhost:5000/api/user/get-image/"; //Путь к аватару с сервера

export const useAuthStore = defineStore("auth", {
  state: () => ({
    userData: user
    ? {
      status: { loggedIn: true },
      user,
      imageUrl: USER_AVATAR + user.id,
    }
  })

```

					ДП.0902.10.000000.00 ПЗ	Лист
						147
Изм	Лист	№ докум.	Подпись	Дата		

```

: {
  status: { loggedIn: false },
  user: null,
  imageUrl: DEFAULT_AVATAR,
},
userFavoriteRecipes: [],
}),
actions: {
  login(user) {
    return AuthService.login(user).then(
      async (user) => {
        this.userData.status.loggedIn = true;
        this.userData.user = user;
        this.userData.imageUrl = USER_AVATAR + user.id;
        this.userFavoriteRecipes = await RecipeService.getFavoriteRecipes(
          this.userData.user.id
        );
        return await Promise.resolve(user);
      },
      (error) => {
        this.userData.status.loggedIn = false;
        this.userData.user = null;
        this.userData.imageUrl = DEFAULT_AVATAR;
        return Promise.reject(error);
      }
    );
  },

  logout() {
    AuthService.logout();
    this.userData.status.loggedIn = false;
    this.userData.user = null;
    this.userData.imageUrl = DEFAULT_AVATAR;
    this.userFavoriteRecipes = [];
  },

  register(user) {
    return AuthService.register(user).then(
      (response) => {
        this.userData.status.loggedIn = false;
        this.userData.imageUrl = DEFAULT_AVATAR;
        return Promise.resolve(response.data);
      },
      (error) => {
        this.userData.status.loggedIn = false;
        this.userData.imageUrl = DEFAULT_AVATAR;
        return Promise.reject(error);
      }
    );
  },

  refreshToken(access_token) {
    this.userData.status.loggedIn = true;
    this.userData.user = { ...this.userData.user, accessToken: access_token };
  },

  forgotPassword(email) {
    return AuthService.forgotPassword(email).catch((error) =>
      Promise.reject(error)
    );
  },
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		148

```

changeUserPassword(email, password) {
  return AuthService.changeUserPassword(email, password).catch((error) =>
    Promise.reject(error)
  );
},

async getUserFavoriteRecipes() {
  try {
    this.userFavoriteRecipes = await RecipeService.getFavoriteRecipes(
      this.userData.user.id
    );
  } catch (error) {
    return await Promise.reject(error);
  }
},
},
});

```

Программный код файла category

```

import { ref, computed, reactive, readonly } from "vue";
import { defineStore } from "pinia";
import CategoryService from "@/services/category.service.js";

export const useCategoryStore = defineStore("category", () => {
  const categories = ref([]);

  function $reset() {
    categories.value = [];
  }

  async function getCategories() {
    try {
      categories.value = await CategoryService.getCategories();
    } catch (error) {
      return Promise.reject(error);
    }
  }

  return {
    categories,
    $reset,
    getCategories,
  };
});

```

Программный код файла recipe

```

import { ref, computed, reactive, readonly } from "vue";
import { defineStore } from "pinia";
import { useAuthStore } from "../auth";
import RecipeService from "@/services/recipe.service.js";

export const useRecipeStore = defineStore("recipe", () => {
  const auth = useAuthStore();
  const { userData } = auth;

```

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		149

```

const recipes = ref([]);
const recipeCategories = ref([]);

function $reset() {
  recipes.value = [];
  recipeCategories.value = [];
}

async function addRecipe(data) {
  if (!userData.status.loggedIn) return;
  try {
    return await RecipeService.addRecipe(data);
  } catch (error) {
    return Promise.reject(error);
  }
}

async function updateRecipe(id_recipe, id_user, data) {
  if (!userData.status.loggedIn) return;
  try {
    return await RecipeService.updateRecipe(id_recipe, id_user, data);
  } catch (error) {
    return Promise.reject(error);
  }
}

async function getRecipes() {
  if (!userData.status.loggedIn) return;
  try {
    recipes.value = await RecipeService.getRecipes();
  } catch (error) {
    return Promise.reject(error);
  }
}

async function getRecipeCategories() {
  if (!userData.status.loggedIn) return;
  try {
    recipeCategories.value = await RecipeService.getRecipeCategories();
  } catch (error) {
    return Promise.reject(error);
  }
}

return {
  recipes,
  recipeCategories,
  $reset,
  getRecipes,
  getRecipeCategories,
  addRecipe,
  updateRecipe,
};
});

```

Программный код файла user

```

import { defineStore } from "pinia";
import { useAuthStore } from "../auth";

```

					ДП.0902.10.000000.00 ПЗ	Лист
						150
Изм	Лист	№ докум.	Подпись	Дата		

```

import UserService from "../services/user.service";
import TokenService from "../services/token.service.js";

const DEFAULT_AVATAR = "src/assets/avatar.png"; //Путь к дефолтному аватару
const USER_AVATAR = "http://localhost:5000/api/user/get-image/"; //Путь к аватару с сервера

export const useUserStore = defineStore("user", {
  state: () => ({
    auth: useAuthStore(),
  }),
  getters: {
    userData(state) {
      return state.auth.userData;
    },
  },
  actions: {
    //Обновление пользователя
    async updateUser(id, data) {
      try {
        const result = await UserService.updateUser(id, data);
        const user = TokenService.getUser();
        user.username = result.login;
        TokenService.setUser(user);
        this.auth.userData.user.username = result.login;
      } catch (error) {
        return Promise.reject(error);
      }
    },

    //Загрузка изображения на сервер
    async uploadImage(file, id) {
      try {
        return await UserService.uploadImage(file, id);
      } catch (error) {
        return Promise.reject(error);
      }
    },

    //Получение изображения с сервера и изменение url в случае его отсутствия
    async getImage() {
      const id_user = this.auth.userData.user.id;
      const res = await UserService.getImage(id_user);
      //Если изображение не найдено, ставим дефолтный аватар
      if (!res.data) {
        return (this.auth.userData.imageUrl = DEFAULT_AVATAR);
      } else {
        return (this.auth.userData.imageUrl =
          USER_AVATAR + this.auth.userData.user.id);
      }
    },
  },
});

```

Программный код файла api

```

import axios from "axios";

const instance = axios.create({
  baseURL: "http://localhost:5000/api",

```

					ДП.0902.10.000000.00 ПЗ	Лист
						151
Изм	Лист	№ докум.	Подпись	Дата		

```

headers: {
  "Content-Type": "application/json",
},
});

```

```
export default instance;
```

Программный код файла auth.service

```

import api from "./api";
import TokenService from "./token.service";

class AuthService {
  login({ email, password }) {
    return api
      .post("/auth/signin", {
        email,
        password,
      })
      .then((response) => {
        if (response.data.accessToken) {
          TokenService.setUser(response.data);
        }
        return response.data;
      });
  }
  logout() {
    TokenService.removeUser();
  }
  register({ login, email, password, role }) {
    return api.post("/auth/signup", {
      login,
      email,
      password,
      role,
    });
  }
  forgotPassword(email) {
    return api.post("/forgot-password", { email: email });
  }
  changeUserPassword(email, password) {
    return api.patch("/change-password", { email: email, password: password });
  }
}

export default new AuthService();

```

Программный код файла category.service

```

import api from "./api";

class CategoryService {
  async getCategories() {
    try {
      const result = await api.get("/categories");
      return result.data;
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						152
Изм	Лист	№ докум.	Подпись	Дата		


```

    } catch (error) {
        return Promise.reject(error);
    }
}

async addCategory(data) {
    try {
        const result = await api.put("/categories", data);
        return result.data;
    } catch (error) {
        return Promise.reject(error);
    }
}

async updateCategory(id, data) {
    try {
        const result = await api.patch("/categories/" + id, data);
        return result.data;
    } catch (error) {
        return Promise.reject(error);
    }
}

async deleteCategory(id) {
    try {
        const result = await api.delete("/categories/" + id);
        return result.data;
    } catch (error) {
        return Promise.reject(error);
    }
}

export default new CategoryService();

```

Программный код файла categoryGroups.service

```

import api from "./api";

class CategoryGroupService {
    async getCategoryGroups() {
        try {
            const result = await api.get("/category-groups");
            return result.data;
        } catch (error) {
            return Promise.reject(error);
        }
    }

    async addCategoryGroup(data) {
        try {
            const result = await api.put("/category-groups", data);
            return result.data;
        } catch (error) {
            return Promise.reject(error);
        }
    }

    async updateCategoryGroup(id, data) {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						153
Изм	Лист	№ докум.	Подпись	Дата		

```

    try {
      const result = await api.patch("/category-groups/" + id, data);
      return result.data;
    } catch (error) {
      return Promise.reject(error);
    }
  }
}

async deleteCategoryGroup(id) {
  try {
    const result = await api.delete("/category-groups/" + id);
    return result.data;
  } catch (error) {
    return Promise.reject(error);
  }
}
}

export default new CategoryGroupService();

```

Программный код файла recipe.service

```

import api from "./api";

class RecipeService {
  //Конвертация изображения
  async convertImage(file) {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => {
      const dataUrl = reader.result;
      console.log(dataUrl);
      try {
        return Promise.resolve(dataUrl);
      } catch (error) {
        return Promise.reject(error);
      }
    };
  }

  async addRecipe(data) {
    try {
      const reader = new FileReader();
      reader.readAsDataURL(data.img);
      reader.onload = async () => {
        data.img = reader.result;
        try {
          return await api.post("/recipes", data);
        } catch (error) {
          return Promise.reject(error);
        }
      };
    } catch (error) {
      return Promise.reject(error);
    }
  }

  async updateRecipe(id_recipe, id_user, data) {
    try {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						154
Изм	Лист	№ докум.	Подпись	Дата		

```

    if (!data.img)
        return await api.patch(`/recipe/${id_recipe}/${id_user}`, data);
    const reader = new FileReader();
    reader.readAsDataURL(data.img);
    reader.onload = async () => {
        data.img = reader.result;
        try {
            return await api.patch(`/recipe/${id_recipe}/${id_user}`, data);
        } catch (error) {
            return Promise.reject(error);
        }
    };
} catch (error) {
    return Promise.reject(error);
}
}

async addFavoriteRecipe(data) {
    try {
        return await api.post("/recipes/favorite-recipe/", data);
    } catch (error) {
        return Promise.reject(error);
    }
}

async deleteFavoriteRecipe(data) {
    try {
        return await api.delete(
            `/recipes/favorite-recipe/${data.id_user}/${data.id_recipe}`
        );
    } catch (error) {
        return Promise.reject(error);
    }
}

async deleteRecipe(id_recipe, id_user) {
    try {
        return await api.delete(`/recipe/${id_recipe}/${id_user}`);
    } catch (error) {
        return Promise.reject(error);
    }
}

async getFavoriteRecipes(id) {
    try {
        const result = await api.get("/recipes/favorite-recipe/" + id);
        return result.data;
    } catch (error) {
        return Promise.reject(error);
    }
}

async getRecipes(data) {
    try {
        const result = await api.get(`/recipes`, {
            params: {
                limit: data.limit,
                page: data.page,
                name: data.name,
                sort: data.sort,
                filters: data.filters,
                sortParam: data.sortParam,
            }
        });
    }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						155
Изм	Лист	№ докум.	Подпись	Дата		

```

    },
  });
  return result;
} catch (error) {
  return Promise.reject(error);
}
}

async getUserRecipes(data) {
  try {
    const result = await api.get(`/recipes/user-recipes`, {
      params: {
        limit: data.limit,
        page: data.page,
        id_user: data.id_user,
      },
    });
    return result;
  } catch (error) {
    return Promise.reject(error);
  }
}

async getUserFavoriteRecipes(data) {
  try {
    const result = await api.get(`/recipes/user-favorite-recipes`, {
      params: {
        limit: data.limit,
        page: data.page,
        id_user: data.id_user,
      },
    });
    return result;
  } catch (error) {
    return Promise.reject(error);
  }
}

async getRecipe(id) {
  try {
    const result = await api.get(`/recipe/${id}`);
    return result.data;
  } catch (error) {
    return Promise.reject(error);
  }
}

async getRecipeCategories() {
  try {
    const result = await api.get("/recipes/categories");
    return result.data;
  } catch (error) {
    return Promise.reject(error);
  }
}

//Проверка ответа с сервера на наличие изображения
async getImage(id) {
  try {
    const result = await api.get(`/recipes/get-image/${id}`);
    return result.data;
  } catch (error) {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						156
Изм	Лист	№ докум.	Подпись	Дата		

```

        return null;
    }
}
}

export default new RecipeService();

```

Программный код файла role.service

```

import api from "../api";

class RoleService {
  async getRoles() {
    try {
      const result = await api.get("/roles");
      return result.data;
    } catch (error) {
      console.log(error);
    }
  }
}

export default new RoleService();

```

Программный код файла serupInterceptors

```

import axiosInstance from "../api";
import TokenService from "../token.service";

const setup = (auth) => {
  const { refreshToken } = auth;
  axiosInstance.interceptors.request.use(
    (config) => {
      const token = TokenService.getLocalAccessToken();
      if (token) {
        config.headers["x-access-token"] = token;
      }
      return config;
    },
    (error) => {
      return Promise.reject(error);
    }
  );

  axiosInstance.interceptors.response.use(
    (res) => {
      return res;
    },
    async (err) => {
      const originalConfig = err.config;
      if (
        originalConfig.url !== "/auth/signin" &&
        originalConfig.url !== "/change-password" &&
        originalConfig.url !== "/forgot-password" &&
        err.response
      ) {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						157
Изм	Лист	№ докум.	Подпись	Дата		

```

// Срок действия токена доступа истек
if (err.response.status === 401 && !originalConfig._retry) {
  originalConfig._retry = true;

  try {
    const rs = await axiosInstance.post("/auth/refreshToken", {
      refreshToken: TokenService.getLocalRefreshToken(),
    });

    const { accessToken } = rs.data;

    refreshToken(accessToken);
    TokenService.updateLocalAccessToken(accessToken);

    return axiosInstance(originalConfig);
  } catch (_error) {
    return Promise.reject(_error);
  }
}
return Promise.reject(err);
};

export default setup;

```

Программный код файла token.service

```

class TokenService {
  getLocalRefreshToken() {
    const user = JSON.parse(localStorage.getItem("user"));
    return user?.refreshToken;
  }

  getLocalAccessToken() {
    const user = JSON.parse(localStorage.getItem("user"));
    return user?.accessToken;
  }

  updateLocalAccessToken(token) {
    let user = JSON.parse(localStorage.getItem("user"));
    user.accessToken = token;
    localStorage.setItem("user", JSON.stringify(user));
  }

  getUser() {
    return JSON.parse(localStorage.getItem("user"));
  }

  setUser(user) {
    localStorage.setItem("user", JSON.stringify(user));
  }

  removeUser() {
    localStorage.removeItem("user");
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		158

```
export default new TokenService();
```

Программный код файла user.service

```
import api from "./api";

class UserService {
  async getUsersWithoutAdmins() {
    try {
      const result = await api.get("/users/without-admins");
      return result.data;
    } catch (error) {
      console.log(error);
    }
  }

  //Загрузка изображения на сервер
  async uploadImage(file, id) {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = async () => {
      const dataUrl = reader.result;
      try {
        await api.post("/user/upload-image", {
          image: dataUrl,
          id_user: id,
        });
      } catch (error) {
        console.log(error);
      }
    };
  }

  //Проверка ответа с сервера на наличие изображения
  async getImage(id) {
    try {
      const result = await api.get(`/user/get-image/${id}`);
      return result;
    } catch (error) {
      return null;
    }
  }

  async addUser(data) {
    try {
      const result = await api.put(`/user/`, data);
      return result.data;
    } catch (error) {
      return Promise.reject(error);
    }
  }

  async updateUser(id, data) {
    try {
      const result = await api.patch(`/user/${id}`, { data });
      return result.data;
    } catch (error) {
      return Promise.reject(error);
    }
  }
}
```

					ДП.0902.10.000000.00 ПЗ	Лист
						159
Изм	Лист	№ докум.	Подпись	Дата		

```

async deleteUser(id) {
  try {
    const result = await api.delete(`/user/${id}`);
    return result.data;
  } catch (error) {
    return Promise.reject(error);
  }
}
}

export default new UserService();

```

Программный код файла index

```

import { createRouter, createWebHistory } from "vue-router";
import HomeView from "@/views/HomeView.vue";
import PersonView from "@/views/PersonView.vue";
import FogortPasswordView from "@/views/FogortPasswordView.vue";
import ChangePasswordView from "@/views/ChangePasswordView.vue";
import AboutView from "@/views/AboutView.vue";
import NotFoundView from "@/views/NotFoundView.vue";
import AddRecipeView from "@/views/AddRecipeView.vue";
import UpdateRecipeView from "@/views/UpdateRecipeView.vue";
import RecipesView from "@/views/RecipesView.vue";
import RecipeView from "@/views/RecipeView.vue";
import AdminView from "@/views/AdminView.vue";

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: "/",
      name: "home",
      component: HomeView,
    },
    {
      path: "/person",
      name: "person",
      component: PersonView,
    },
    {
      path: "/fogort-password",
      name: "fogort-password",
      component: FogortPasswordView,
    },
    {
      path: "/change-password",
      name: "change-password",
      component: ChangePasswordView,
    },
    {
      path: "/about",
      name: "about",
      component: AboutView,
    },
    {
      path: "/add-recipe",
      name: "add-recipe",

```

					ДП.0902.10.000000.00 ПЗ	Лист
						160
Изм	Лист	№ докум.	Подпись	Дата		


```

        component: AddRecipeView,
      },
      {
        path: "/update-recipe/:id",
        name: "update-recipe",
        component: UpdateRecipeView,
      },
      {
        path: "/recipes",
        name: "recipes",
        component: RecipesView,
      },
      {
        path: "/recipe/:id",
        name: "recipe",
        component: RecipeView,
      },
      {
        path: "/admin",
        name: "admin",
        component: AdminView,
      },
      {
        path: "/*",
        name: "not-found",
        component: NotFoundView,
      },
    ],
  });

```

```
export default router;
```

Программный код файла AdminCategories

```

<template>
  <div class="categories-table">
    <template v-if="categories && categoryGroups">
      <DataTable
        v-model:editingRows="editingRows"
        :value="categories"
        resizableColumns
        columnResizeMode="fit"
        showGridlines
        editMode="row"
        dataKey="id"
        @row-edit-save="onRowEditSave"
        scrollable
        scrollHeight="80vh"
        tableClass="editable-cells-table"
        tableStyle="min-width: 50rem"
      >
        <template #header>
          <Button
            type="button"
            icon="pi pi-plus"
            label="Добавить"
            outlined
            @click="isShowDialog = !isShowDialog"
          />

```

					ДП.0902.10.000000.00 ПЗ	Лист
						161
Изм	Лист	№ докум.	Подпись	Дата		

```

</template>

<Column field="category_name" header="Category name">
  <template #editor="{ data, field }">
    <InputText v-model="data[field]" maxlength="100" />
  </template>
</Column>

<Column field="id_category_group" header="Group name">
  <template #editor="{ data, field }">
    <Dropdown
      v-model="data[field]"
      :options="categoryGroups"
      optionLabel="group_name"
      optionValue="id"
      placeholder="Выберите группу категории"
    >
  </Dropdown>
</template>
<template #body="slotProps">
  <p>{{ slotProps.data.group_name }}</p>
</template>
</Column>

<Column
  :rowEditor="true"
  style="min-width: 5rem"
  headerStyle="width: 5rem; text-align: center"
  bodyStyle="text-align:center"
></Column>

<Column
  style="min-width: 5rem"
  headerStyle="width: 5rem; text-align: center"
  bodyStyle="text-align: center"
>
  <template #body="slotProps">
    <i
      class="pi pi-trash"
      style="font-size: 1rem"
      @click="deleteCategory(slotProps.data)"
    ></i>
  </template>
</Column>
</DataTable>
</template>

<p v-else>Пользователи не найдены</p>
<main-dialog v-model:show="isShowDialog">
  <form-add-category @hide-dialog="handleHideDialog"></form-add-category>
</main-dialog>
</div>
</template>

<script>
import DataTable from "primevue/datatable";
import Column from "primevue/column";
import InputText from "primevue/inputtext";
import InputNumber from "primevue/inputnumber";
import Button from "primevue/button";
import Dropdown from "primevue/dropdown";
import FormAddCategory from "@/components/admin/FormAddCategory.vue";
import MainDialog from "@/components/MainDialog.vue";
import CategoryService from "@/services/category.service.js";

```

					ДП.0902.10.000000.00 ПЗ	Лист
						162
Изм	Лист	№ докум.	Подпись	Дата		

```

import CategoryGroupService from "@services/categoryGroups.service.js";
export default {
  name: "admin-categories",
  components: {
    DataTable,
    Column,
    InputText,
    InputNumber,
    Button,
    Dropdown,
    FormAddCategory,
    MainDialog,
  },

  data() {
    return {
      categories: null,
      categoryGroups: null,
      editingRows: [],
      isShowDialog: false,
    };
  },

  mounted() {
    this.LoadCategories();
    this.LoadCategoryGroups();
  },

  methods: {
    async LoadCategories() {
      this.categories = await CategoryService.getCategories();
    },

    async LoadCategoryGroups() {
      this.categoryGroups = await CategoryGroupService.getCategoryGroups();
      this.categoryGroups.unshift({ id: null, group_name: "Без группы" });
    },

    async onRowEditSave(event) {
      let { newData, index } = event;
      this.categories[index] = newData;
      try {
        await CategoryService.updateCategory(newData.id, {
          category_name: newData.category_name,
          id_category_group: newData.id_category_group,
        });
        await this.LoadCategories();
      } catch (error) {
        console.log(error);
      }
    },

    async deleteCategory(category) {
      if (confirm("Удалить категорию?")) {
        this.categories = this.categories.filter((el) => el.id !== category.id);
        try {
          await CategoryService.deleteCategory(category.id);
        } catch (error) {
          console.log(error);
        }
      }
    },
  },

```

					ДП.0902.10.000000.00 ПЗ	Лист
						163
Изм	Лист	№ докум.	Подпись	Дата		

```

    async handleHideDialog() {
        this.isShowDialog = false;
        await this.LoadCategories();
    },
},
};
</script>

```

```

<style scoped>
.pi {
    cursor: pointer;
}
</style>

```

Программный код файла AdminGroupsCategories

```

<template>
<div class="categories-table">
  <template v-if="categoryGroups">
    <DataTable
      v-model:editingRows="editingRows"
      :value="categoryGroups"
      resizableColumns
      columnResizeMode="fit"
      showGridlines
      editMode="row"
      dataKey="id"
      @row-edit-save="onRowEditSave"
      scrollable
      scrollHeight="80vh"
      tableClass="editable-cells-table"
      tableStyle="min-width: 50rem"
    >
      <template #header>
        <Button
          type="button"
          icon="pi pi-plus"
          label="Добавить"
          outlined
          @click="isShowDialog = !isShowDialog"
        />
      </template>
      <Column field="group_name" header="Group name">
        <template #editor="{ data, field }">
          <InputText v-model="data[field]" maxlength="100" />
        </template>
      </Column>
      <Column
        :rowEditor="true"
        style="min-width: 5rem"
        headerStyle="width: 5rem; text-align: center"
        bodyStyle="text-align:center"
      ></Column>
      <Column
        style="min-width: 5rem"
        headerStyle="width: 5rem; text-align: center"
        bodyStyle="text-align: center"
      >
    </div>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						164
Изм	Лист	№ докум.	Подпись	Дата		

```

<template #body="slotProps">
  <i
    class="pi pi-trash"
    style="font-size: 1rem"
    @click="deleteCategoryGroup(slotProps.data)"
  ></i>
</template>
</Column>
</DataTable>
</template>
<p v-else>Пользователи не найдены</p>
<main-dialog v-model:show="isShowDialog">
  <form-add-category-group
    @hide-dialog="handleHideDialog"
  ></form-add-category-group>
</main-dialog>
</div>
</template>

<script>
import DataTable from "primevue/datatable";
import Column from "primevue/column";
import InputText from "primevue/inputtext";
import InputNumber from "primevue/inputnumber";
import Button from "primevue/button";
import Dropdown from "primevue/dropdown";
import FormAddCategoryGroup from "@/components/admin/FormAddCategoryGroup.vue";
import MainDialog from "@/components/MainDialog.vue";
import CategoryService from "@/services/category.service.js";
import CategoryGroupService from "@/services/categoryGroups.service.js";
export default {
  name: "admin-categories",
  components: {
    DataTable,
    Column,
    InputText,
    Button,
    FormAddCategoryGroup,
    MainDialog,
  },

  data() {
    return {
      categoryGroups: null,
      editingRows: [],
      isShowDialog: false,
    };
  },

  mounted() {
    this.LoadCategoryGroups();
  },

  methods: {
    async LoadCategoryGroups() {
      this.categoryGroups = await CategoryGroupService.getCategoryGroups();
    },

    async onRowEditSave(event) {
      let { newData, index } = event;
      this.categoryGroups[index] = newData;
      try {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						165
Изм	Лист	№ докум.	Подпись	Дата		

```

        console.log(newData);
        await CategoryGroupService.updateCategoryGroup(newData.id, newData);
    } catch (error) {
        console.log(error);
    }
},

async deleteCategoryGroup(group) {
    if (confirm("Удалить категорию?")) {
        this.categoryGroups = this.categoryGroups.filter(
            (el) => el.id !== group.id
        );
        try {
            await CategoryGroupService.deleteCategoryGroup(group.id);
        } catch (error) {
            console.log(error);
        }
    }
},

async handleHideDialog() {
    this.isShowDialog = false;
    await this.LoadCategoryGroups();
},
},
};
</script>

<style scoped>
.pi {
    cursor: pointer;
}
</style>

```

Программный код файла AdminUsers

```

<template>
<div class="users-table">
<template v-if="users && roles">
<DataTable
    v-model:editingRows="editingRows"
    :value="users"
    resizableColumns
    columnResizeMode="fit"
    showGridlines
    editMode="row"
    dataKey="id_user"
    @row-edit-save="onRowEditSave"
    scrollable
    scrollHeight="80vh"
    tableClass="editable-cells-table"
    tableStyle="min-width: 50rem"
>
<template #header>
<Button
    type="button"
    icon="pi pi-plus"
    label="Добавить"
    outlined

```

					ДП.0902.10.000000.00 ПЗ	Лист
						166
Изм	Лист	№ докум.	Подпись	Дата		

```

        @click="isShowDialog = !isShowDialog"
      />
    </template>

    <Column field="login" header="Login">
      <template #editor="{ data, field }">
        <InputText v-model="data[field]" maxlength="250" />
      </template>
    </Column>

    <Column field="email" header="Email">
      <template #editor="{ data, field }">
        <InputText v-model="data[field]" maxlength="250" />
      </template>
    </Column>

    <Column field="id_role" header="Role name">
      <template #editor="{ data, field }">
        <Dropdown
          v-model="data[field]"
          :options="roles"
          optionLabel="name"
          optionValue="id_role"
          placeholder="Выберите роль"
        >
        </Dropdown>
      </template>
      <template #body="slotProps">
        <p>{{ getRole(slotProps.data.id_role) }}</p>
      </template>
    </Column>

    <Column field="is_activated" header="Is activated">
      <template #editor="{ data, field }">
        <InputNumber v-model="data[field]" :min="0" :max="1" />
      </template>
    </Column>

    <Column
      :rowEditor="true"
      style="min-width: 5rem"
      headerStyle="width: 5rem; text-align: center"
      bodyStyle="text-align:center"
    ></Column>

    <Column
      style="min-width: 5rem"
      headerStyle="width: 5rem; text-align: center"
      bodyStyle="text-align: center"
    >
      <template #body="slotProps">
        <i
          class="pi pi-trash"
          style="font-size: 1rem"
          @click="deleteUser(slotProps.data)"
        ></i>
      </template>
    </Column>
  </DataTable>
</template>
<p v-else>Пользователи не найдены</p>
<main-dialog v-model:show="isShowDialog">

```

					ДП.0902.10.000000.00 ПЗ	Лист
						167
Изм	Лист	№ докум.	Подпись	Дата		

```

        <form-add-user @hide-dialog="handleHideDialog"></form-add-user>
      </main-dialog>
    </div>
  </template>

```

```

<script>
import DataTable from "primevue/datatable";
import Column from "primevue/column";
import InputText from "primevue/inputtext";
import InputNumber from "primevue/inputnumber";
import Button from "primevue/button";
import Dropdown from "primevue/dropdown";
import FormAddUser from "@/components/admin/FormAddUser.vue";
import MainDialog from "@/components/MainDialog.vue";
import UserService from "@/services/user.service.js";
import RoleService from "@/services/role.service.js";
export default {
  name: "admin-users",
  components: {
    DataTable,
    Column,
    InputText,
    InputNumber,
    Button,
    Dropdown,
    FormAddUser,
    MainDialog,
  },

  data() {
    return {
      roles: null,
      users: null,
      editingRows: [],
      isShowDialog: false,
    };
  },

  mounted() {
    this.loadUsers();
    this.loadRoles();
  },

  methods: {
    getRole(id) {
      return this.roles.find((role) => role.id_role === id).name;
    },

    async loadUsers() {
      this.users = await UserService getUsersWithoutAdmins();
    },

    async loadRoles() {
      this.roles = await RoleService.getRoles();
      this.roles = this.roles.filter((role) => role.name !== "admin");
    },

    async onRowEditSave(event) {
      let { newData, index } = event;
      this.users[index] = newData;
      try {
        await UserService.updateUser(newData.id_user, newData);
      }
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						168
Изм	Лист	№ докум.	Подпись	Дата		


```

    } catch (error) {
      console.log(error);
    }
  },

  async deleteUser(user) {
    if (confirm("Удалить пользователя?")) {
      this.users = this.users.filter((el) => el.id_user !== user.id_user);
      try {
        await UserService.deleteUser(user.id_user);
      } catch (error) {
        console.log(error);
      }
    }
  },

  async handleHideDialog() {
    this.isShowDialog = false;
    await this.loadUsers();
  },
},
};
</script>

<style scoped>
.pi {
  cursor: pointer;
}
</style>

```

Программный код файла FormAddCategory

```

<template>
  <Form
    @submit="handlerAddCategory"
    :validation-schema="schema"
    class="add-form"
  >
    <h3>Добавление</h3>
    <div class="add-form__content-wrapper">
      <label for="category_name">Название</label>
      <Field
        class="add-form__form-input"
        type="text"
        name="category_name"
        maxlength="100"
      ></Field>
      <ErrorMessage name="category_name" class="add-form__error" />

      <label for="id_category_group">Группы katerорий</label>
      <Field as="select" name="id_category_group" class="add-form__form-select">
        <option
          v-for="categoryGroup of categoryGroups"
          :key="categoryGroup.id"
          :value="categoryGroup.id"
        >
          {{ categoryGroup.group_name }}
        </option>
      </Field>
    </div>
  </Form>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						169
Изм	Лист	№ докум.	Подпись	Дата		

```

</div>
<div v-if="errorMessage" class="alert alert-danger" role="alert">
  {{ errorMessage }}
</div>
<div class="add-form__butn-wrapper">
  <form-button :disabled="isLoading">Добавить</form-button>
</div>
</Form>
</template>

<script>
import FormInput from "@/components/UI/FormInput.vue";
import FormButton from "@/components/UI/FormButton.vue";
import { Form, Field, ErrorMessage } from "vee-validate";
import * as yup from "yup";
import CategoryService from "@/services/category.service.js";
import CategoryGroupService from "@/services/categoryGroups.service.js";
export default {
  name: "form-add-category",

  components: {
    FormInput,
    FormButton,
    Form,
    Field,
    ErrorMessage,
  },

  emits: ["hideDialog"],

  data() {
    return {
      categoryGroups: null,
      errorMessage: "",
      isLoading: false,
    };
  },

  computed: {
    schema() {
      return yup.object({
        category_name: yup.string().trim().required("Обязательное поле"),
      });
    },
  },

  mounted() {
    this.LoadCategoryGroups();
  },

  methods: {
    async LoadCategoryGroups() {
      this.categoryGroups = await CategoryGroupService.getCategoryGroups();
      this.categoryGroups.unshift({ id: null, group_name: "Без группы" });
    },

    async handlerAddCategory(values) {
      if (!values.id_category_group) values.id_category_group = null;
      this.isLoading = true;
      this.errorMessage = "";
      try {
        await CategoryService.addCategory(values);
      }
    }
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						170
Изм	Лист	№ докум.	Подпись	Дата		

```

        this.$emit("hideDialog");
    } catch (error) {
        this.errorMessage = error;
    }
    this.isLoading = false;
  },
},
};
</script>

<style scoped>
.add-form {
  display: flex;
  flex-direction: column;
  margin-top: 30px;
}

.add-form__error {
  color: red;
}

.add-form__content-wrapper {
  display: flex;
  flex-direction: column;
  margin-top: 10px;
}

.add-form__form-input {
  padding: 2px 10px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}

.add-form__btn-wrapper {
  display: flex;
  gap: 20px;
  margin-top: 20px;
}

.add-form__form-select {
  padding: 10px 15px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}
</style>

```

Программный код файла FormAddCategoryGroup

```

<template>
<Form
  @submit="handlerAddCategory"
  :validation-schema="schema"
  class="add-form"
>
  <h3>Добавление</h3>
  <div class="add-form__content-wrapper">
    <label for="group_name">Название</label>
    <Field
      class="add-form__form-input"
      type="text"
    >

```

					ДП.0902.10.000000.00 ПЗ	Лист
						171
Изм	Лист	№ докум.	Подпись	Дата		

```

        name="group_name"
        maxlength="100"
      ></Field>
      <ErrorMessage name="group_name" class="add-form__error" />
    </div>
    <div v-if="errorMessage" class="alert alert-danger" role="alert">
      {{ errorMessage }}
    </div>
    <div class="add-form__btn-wrapper">
      <form-button :disabled="isLoading">Добавить</form-button>
    </div>
  </Form>
</template>

<script>
import FormInput from "@/components/UI/FormInput.vue";
import FormButton from "@/components/UI/FormButton.vue";
import { Form, Field, ErrorMessage } from "vee-validate";
import * as yup from "yup";
import CategoryGroupService from "@/services/categoryGroups.service.js";
export default {
  name: "form-add-category-group",

  components: {
    FormInput,
    FormButton,
    Form,
    Field,
    ErrorMessage,
  },

  emits: ["hideDialog"],

  data() {
    return {
      categoryGroups: null,
      errorMessage: "",
      isLoading: false,
    };
  },

  computed: {
    schema() {
      return yup.object({
        group_name: yup.string().trim().required("Обязательное поле"),
      });
    },
  },

  methods: {
    async handlerAddCategory(values) {
      this.isLoading = true;
      this.errorMessage = "";
      try {
        await CategoryGroupService.addCategoryGroup(values);
        this.$emit("hideDialog");
      } catch (error) {
        this.errorMessage = error;
      }
      this.isLoading = false;
    },
  },
};

```

					ДП.0902.10.000000.00 ПЗ	Лист
						172
Изм	Лист	№ докум.	Подпись	Дата		

```

};
</script>

<style scoped>
.add-form {
  display: flex;
  flex-direction: column;
  margin-top: 30px;
}

.add-form__error {
  color: red;
}

.add-form__content-wrapper {
  display: flex;
  flex-direction: column;
  margin-top: 10px;
}

.add-form__form-input {
  padding: 2px 10px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}

.add-form__butn-wrapper {
  display: flex;
  gap: 20px;
  margin-top: 20px;
}

.add-form__form-select {
  padding: 10px 15px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}
</style>

```

Программный код файла FormAddUser

```

<template>
<Form @submit="handlerAddUser" :validation-schema="schema" class="add-form">
  <h3>Добавление</h3>
  <div class="add-form__content-wrapper">
    <label for="login">Логин</label>
    <Field
      class="add-form__form-input"
      type="text"
      name="login"
      maxlength="250"
    ></Field>
    <ErrorMessage name="login" class="add-form__error" />

    <label for="email">E-mail</label>
    <Field
      class="add-form__form-input"
      type="email"
      name="email"
      maxlength="250"
    >

```

					ДП.0902.10.000000.00 ПЗ	Лист
						173
Изм	Лист	№ докум.	Подпись	Дата		

```

    </Field>
    <ErrorMessage name="email" class="add-form__error" />

    <label for="password">Пароль</label>
    <Field
      class="add-form__form-input"
      type="password"
      name="password"
      maxlength="250"
    ></Field>
    <ErrorMessage name="password" class="add-form__error" />

    <label for="id_role">Роли</label>
    <Field as="select" name="id_role" class="add-form__form-select">
      <option v-for="role of roles" :key="role.id_role" :value="role.id_role">
        {{ role.name }}
      </option>
    </Field>
  </div>
  <div v-if="errorMessage" class="alert alert-danger" role="alert">
    {{ errorMessage }}
  </div>
  <div class="add-form__btn-wrapper">
    <form-button :disabled="isLoading">Добавить</form-button>
  </div>
</Form>
</template>

<script>
import FormInput from "@components/UI/FormInput.vue";
import FormButton from "@components/UI/FormButton.vue";
import { Form, Field, ErrorMessage } from "vee-validate";
import * as yup from "yup";
import UserService from "@services/user.service.js";
import RoleService from "@services/role.service.js";
export default {
  name: "form-add-user",

  components: {
    FormInput,
    FormButton,
    Form,
    Field,
    ErrorMessage,
  },

  emits: ["hideDialog"],

  data() {
    return {
      roles: [],
      errorMessage: "",
      isLoading: false,
    };
  },

  mounted() {
    this.loadRoles();
  },

  computed: {
    schema() {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						174
Изм	Лист	№ докум.	Подпись	Дата		

```

return yup.object({
  login: yup.string().trim().required("Обязательное поле"),
  email: yup
    .string()
    .trim()
    .required("Обязательное поле")
    .email("Не верный формат"),
  password: yup
    .string()
    .trim()
    .min(8, "Минимум 8 символов")
    .required("Обязательное поле"),
});
},
},
methods: {
  async loadRoles() {
    this.roles = await RoleService.getRoles();
    this.roles = this.roles.filter((role) => role.name !== "admin");
  },

  async handlerAddUser(values) {
    if (!values.id_role) values.id_role = 1;
    this.isLoading = true;
    this.errorMessage = "";
    try {
      await UserService.addUser(values);
      this.$emit("hideDialog");
    } catch (error) {
      this.errorMessage = error;
    }
    this.isLoading = false;
  },
},
};
</script>

```

```

<style scoped>
.add-form {
  display: flex;
  flex-direction: column;
  margin-top: 30px;
}

.add-form__error {
  color: red;
}

.add-form__content-wrapper {
  display: flex;
  flex-direction: column;
  margin-top: 10px;
}

.add-form__form-input {
  padding: 2px 10px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}

.add-form__btn-wrapper {
  display: flex;
  gap: 20px;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						175
Изм	Лист	№ докум.	Подпись	Дата		

```

margin-top: 20px;
}

.add-form__form-select {
padding: 10px 15px;
border: 1px solid var(--color-light-black);
border-radius: 8px;
margin: 5px 0;
}
</style>

```

Программный код файла FromButton

```

<template>
  <button><slot></slot></button>
</template>

<script>
export default {
  name: "form-button",
};
</script>

<style scoped>
button {
padding: 10px 28px;
line-height: 1;
border-radius: 30px;
font-size: 16px;
font-weight: 500;
color: var(--color-white);
background: var(--color-accent);
}

button:disabled {
color: var(--color-white);
background: var(--color-accent-disabled);
}
</style>

```

Программный код файла FromInput

```

<template>
  <input :value="modelValue" @input="updateInput" />
</template>

<script>
export default {
  name: "form-input",
  props: ["modelValue"],
  emits: ["update:modelValue"],
  methods: {
    updateInput(e) {
      this.$emit("update:modelValue", e.target.value);
    },
  },
};

```

					ДП.0902.10.000000.00 ПЗ	Лист
						176
Изм	Лист	№ докум.	Подпись	Дата		


```

};
</script>

<style scoped>
input {
padding: 2px 15px;
border: 1px solid var(--color-light-black);
border-radius: 30px;
}
</style>

```

Программный код файла AuthorizationForm

```

<template>
<Form @submit="handleLogin" :validation-schema="schema" class="auth-form">
  <h3>Авторизация</h3>
  <div class="auth-form__content-wrapper">
    <label for="email">E-mail</label>
    <Field class="auth-form__form-input" type="text" name="email"></Field>
    <ErrorMessage name="email" class="auth-form__error" />
    <div class="auth-form__fogort-wrapper">
      <label for="password">Пароль</label>
      <button @click="handleFogortPassword">Забыли пароль?</button>
    </div>
    <Field
      class="auth-form__form-input"
      type="password"
      name="password"
    ></Field>
    <ErrorMessage name="password" class="auth-form__error" />
  </div>
  <p v-if="errorMessage" class="auth-form__error">{{ errorMessage }}</p>
  <div class="auth-form__butn-wrapper">
    <form-button :disabled="isLoading">Войти</form-button>
  </div>
</Form>
</template>

```

```

<script>
import FormInput from "@/components/UI/FormInput.vue";
import FormButton from "@/components/UI/FormButton.vue";
import { Form, Field, ErrorMessage } from "vee-validate";
import * as yup from "yup";
import { storeToRefs } from "pinia";
import { useAuthStore } from "@/stores/auth";
export default {
name: "authorization-form",
components: {
FormInput,
FormButton,
Form,
Field,
ErrorMessage,
},
emits: ["hideDialog"],
setup() {
const auth = useAuthStore();
const { userData } = storeToRefs(auth);
const { login } = auth;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						177
Изм	Лист	№ докум.	Подпись	Дата		

```

return {
  userData,
  login,
};
},
data() {
  return {
    errorMessage: "",
    isLoading: false,
  };
},
computed: {
  schema() {
    return yup.object({
      email: yup
        .string()
        .trim()
        .required("Обязательное поле")
        .email("Не верный формат"),
      password: yup
        .string()
        .trim()
        .min(8, "Минимум 8 символов")
        .required("Обязательное поле"),
    });
  },
},
methods: {
  handleFogortPassword() {
    this.$emit("hideDialog");
    this.$router.push("/fogort-password");
  },

  async handleLogin(values) {
    this.errorMessage = "";
    this.isLoading = true;
    try {
      await this.login({ email: values.email, password: values.password });
      this.$emit("hideDialog");
    } catch (error) {
      this.errorMessage = error.response.data.message;
    } finally {
      this.isLoading = false;
    }
  },
},
};
</script>

```

```
<style scoped>
```

```

.auth-form {
  display: flex;
  flex-direction: column;
  margin-top: 30px;
}

```

```

.auth-form__fogort-wrapper {
  display: flex;
  justify-content: space-between;
}

```

```

.auth-form__fogort-wrapper button {
  color: var(--color-accent);
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						178
Изм	Лист	№ докум.	Подпись	Дата		

```

}

.auth-form__error {
  color: red;
}

.auth-form__content-wrapper {
  display: flex;
  flex-direction: column;
  margin-top: 10px;
}

.auth-form__form-input {
  padding: 2px 10px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}

.auth-form__btn-wrapper {
  display: flex;
  gap: 20px;
  margin-top: 20px;
}
</style>

```

Программный код файла CategoryFilter

```

<template>
  <div>
    <h4>Категории</h4>
    <div class="categories_filter">
      <!--Список категорий-->
      <ul class="categories_filter__list">
        <!--Категории-->
        <li
          v-for="[groupKey, groupValue] of Object.entries(groups)"
          :key="groupKey"
          class="categories_filter__item"
        >
          <template
            v-if="Object.keys(groupValue).some((key) => key == 'categories')"
          >
            <div class="categories_filter__category-wrapper">
              <input
                type="checkbox"
                :id="groupKey"
                :checked="groupValue.isChecked"
                @change="
                  handlerChangeInputGroup($event.target.checked, groupValue)
                "
              />
              <label
                class="categories_filter__label-group"
                @click.prevent="groupValue.isVisible = !groupValue.isVisible"
                :for="groupKey"
              >
                {{ groupKey }}
              </label>
            </div>
          <!--Подкатегории-->
          <ul

```

					ДП.0902.10.000000.00 ПЗ	Лист
						179
Изм	Лист	№ докум.	Подпись	Дата		

```

class="categories_filter__list categories_filter__list--subcategories"
v-show="groupValue.isVisible"
>
<li
  v-for="category of groupValue.categories"
  :key="category.id"
  class="categories_filter__item"
>
  <div class="categories_filter__category-wrapper">
    <input
      type="checkbox"
      :id="category.id"
      :value="category.id"
      @change="
        handlerChangeInputSubcategory(category, groupValue)
      "
      :checked="category.isChecked"
    />
    <label :for="category.category_name">
      {{ category.category_name }}
    </label>
  </div>
</li>
</ul>
</template>
<!--Категории без подкатегорий-->
<template v-else>
  <div class="categories_filter__category-wrapper">
    <input
      type="checkbox"
      :id="groupValue.id"
      :value="groupValue.id"
      @change="toggleCategory($event.target.value)"
    />
    <label :for="groupValue.category_name">{{ groupKey }}</label>
  </div>
</template>
</li>
</ul>
</div>
</div>
</template>

<script>
import { storeToRefs } from "pinia";
import { useCategoryStore } from "@/stores/category";
export default {
  props: {
    selectedCategories: {
      type: Array,
      default: () => [],
      required: true,
    },
  },
  setup() {
    const category = useCategoryStore();
    const { getCategories } = category;
    const { categories } = storeToRefs(category);
    return {
      categories,
      getCategories,

```

					ДП.0902.10.000000.00 ПЗ	Лист
						180
Изм	Лист	№ докум.	Подпись	Дата		

```

    };
  },

  data() {
    return {
      groups: {},
      isChecked: false,
    };
  },

  mounted() {
    this.loadData();
  },

  methods: {
    async loadData() {
      if (!this.categories.lenght) await this.getCategories();
      this.loadGroups();
    },

    loadGroups() {
      try {
        // Преобразуем категории в другую структуру для удобной отрисовки
        this.groups = this.categories.reduce((acc, category) => {
          if (category.group_name) {
            if (!acc[category.group_name]) {
              acc[category.group_name] = {
                categories: [],
                isVisible: false,
                isChecked: false,
              };
            }
            category["isChecked"] = false;
            acc[category.group_name].categories.push(category);
          } else {
            acc[category.category_name] = category;
          }
          return acc;
        }, {});
      } catch (error) {
        console.log(error);
      }
    },

    handlerChangeInputGroup(isChecked, group) {
      group.isChecked = !group.isChecked;
      group.isVisible = true;
      group.categories.forEach((category) => {
        const index = this.selectedCategories.indexOf(category.id);
        if (isChecked) {
          category.isChecked = true;
          if (index === -1) {
            this.selectedCategories.push(category.id);
          }
        } else {
          category.isChecked = false;
          if (index !== -1) {
            this.selectedCategories.splice(index, 1);
          }
        }
      });
    },
  },

```

					ДП.0902.10.000000.00 ПЗ	Лист
						181
Изм	Лист	№ докум.	Подпись	Дата		

```

handlerChangeInputSubcategory(category, group) {
  category.isChecked = !category.isChecked;
  let isAllChecked = true;
  group.categories.forEach((category) => {
    if (!category.isChecked) isAllChecked = false;
  });
  group.isChecked = isAllChecked;
  this.toggleCategory(category.id);
},

toggleCategory(category_id) {
  category_id = +category_id;
  const index = this.selectedCategories.indexOf(category_id);
  if (index === -1) {
    this.selectedCategories.push(category_id);
  } else {
    this.selectedCategories.splice(index, 1);
  }
},
};
</script>

```

```
<style scoped>
```

```

.categories_filter {
  height: 400px;
  max-width: 300px;
  overflow-y: scroll;
}

```

```

.categories_filter__label-group {
  font-weight: 700;
  cursor: pointer;
  width: 100%;
}

```

```

.categories_filter__list {
  display: flex;
  flex-direction: column;
  gap: 6px;
}

```

```

.categories_filter__list--subcategories {
  margin-left: 20px;
}

```

```

.categories_filter__item {
  display: flex;
  flex-direction: column;
}

```

```

.categories_filter__category-wrapper {
  display: flex;
  align-items: flex-start;
}

```

```

input {
  margin-top: 7px;
  margin-right: 10px;
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						182
Изм	Лист	№ докум.	Подпись	Дата		

```

@media (max-width: 767px) {
  .categories_filter__list {
    gap: 10px;
  }
}
</style>

```

Программный код файла MainDialog

```

<template>
  <div class="dialog" v-if="show" @click.stop="hideDialog">
    <div @click.stop class="dialog__content">
      <button class="dialog__btn-close" @click="hideDialog"></button>
      <slot></slot>
    </div>
  </div>
</template>

```

```

<script>
export default {
  name: "main-dialog",
  emits: ["update:show"],
  props: {
    show: {
      type: Boolean,
      default: false,
    },
  },
  methods: {
    hideDialog() {
      this.$emit("update:show", false);
    },
  },
};
</script>

```

```

<style scoped>
.dialog {
  display: flex;
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.5);
  z-index: 100;
}

```

```

.dialog__content {
  position: relative;
  margin: auto;
  background: #fafafa;
  border-radius: 12px;
  min-height: 50px;
  min-width: 300px;
  padding: 20px;
}

```

```

.dialog__btn-close {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						183
Изм	Лист	№ докум.	Подпись	Дата		

```

position: absolute;
top: 20px;
right: 20px;
background-image: url("../assets/close.svg");
background-position: 0 0;
background-size: contain;
background-repeat: no-repeat;
height: 18px;
width: 26px;
z-index: 1;
}
</style>

```

Программный код файла MainFooter

```

<template>
<footer class="main-footer">
  <div class="container">
    <p class="main-footer__text">
      Copyright © 2023 LFRRecipes - рецепты от вас для вас
    </p>
  </div>
</footer>
</template>

<style scoped>
p {
  margin: 0;
  color: var(--color-white);
  font-weight: 400;
}

.main-footer {
  background-color: var(--color-dark-green);
  padding: 50px 0;
  border-start-end-radius: 30px;
  border-start-start-radius: 30px;
}
</style>

```

Программный код файла MainNavbar

```

<template>
<header class="navbar-header">
<nav class="container main-navbar">
  <button class="menu-burger" @click="$emit('showMenu', true)"></button>
  <div v-if="!loggedIn" class="main-navbar__nav-wrapepr">
    <button
      class="main-navbar__button-signin"
      @click="$emit('showDialog', 'signin')"
    >
      Sign in
    </button>
    <button
      class="main-navbar__button-signup"
      @click="$emit('showDialog', 'signup')"
    >
      Sign up
    </button>
  </div>
</nav>
</header>

```

					ДП.0902.10.000000.00 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		184


```

    >
      Sign up
    </button>
  </div>
  <div class="main-navbar__nav-wrapepr" v-else>
    <button
      class="main-navbar__button-user"
      :style="styleImg"
      @click="showUserMenu = !showUserMenu"
    ></button>
    <user-menu v-model:show="showUserMenu"></user-menu>
  </div>
</nav>
</header>
</template>

```

```

<script>
import UserMenu from "@components/UserMenu.vue";
import { storeToRefs } from "pinia";
import { useUserStore } from "@stores/user";
export default {
  name: "main-navbar",
  components: {
    UserMenu,
  },
  emits: ["showDialog", "showMenu"],
  setup() {
    const user = useUserStore();
    const { getImage } = user;
    const { userData } = storeToRefs(user);
    return {
      userData,
      getImage,
    };
  },
  data() {
    return {
      showUserMenu: false,
      styleImg: {
        backgroundImage: "url('../src/assets/avatar.png')",
      },
    };
  },
  computed: {
    loggedIn() {
      return this.userData.status.loggedIn;
    },
  },
  methods: {
    getUserAvatar() {
      if (this.userData.user) {
        this.getImage()
          .then((img) => {
            this.styleImg.backgroundImage = `url(${img})`;
          })
          .catch((err) => console.log(err));
      }
    },
  },
  watch: {
    userData: {
      handler() {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						185
Изм	Лист	№ докум.	Подпись	Дата		

```

        this.getUserAvatar();
    },
    deep: true,
  },
},
mounted() {
  this.getUserAvatar();
},
};
</script>

<style scoped>
.navbar-header {
  position: relative;
}

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

.main-navbar__nav-wrapepr {
  position: relative;
}

.main-navbar__button-signin {
  margin-right: 30px;
  font-weight: 500;
  transition: color 0.25s ease-out;
}

.main-navbar__button-signin:hover {
  color: var(--color-accent);
}

.main-navbar__button-signin:focus {
  color: var(--color-dark-green);
}

.main-navbar__button-signup {
  color: var(--color-accent);
  font-weight: 600;
  line-height: 1;
  padding: 10px 20px;
  border: 2px var(--color-accent) solid;
  border-radius: 30px;
  transition: color, border 0.15s ease-out;
}

.main-navbar__button-signup:hover {
  color: var(--color-light-green);
  border-color: var(--color-light-green);
}

.main-navbar__button-signup:focus {
  color: var(--color-dark-green);
  border-color: var(--color-dark-green);
}

.menu-burger {
  background-image: url("../assets/menu.svg");
  background-position: 0 0;
  background-size: contain;
  background-repeat: no-repeat;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						186
Изм	Лист	№ докум.	Подпись	Дата		

```

height: 24px;
width: 32px;
}

.main-navbar__button-user {
width: 36px;
height: 36px;
border-radius: 10em;
background-size: cover;
background-repeat: no-repeat;
background-position: center;
}
</style>

```

Программный код файла MainPagination

```

<template>
<div v-if="countOfPages > 1" class="pagination">
  <div class="pagination__nav-wrapper">
    <button
      class="pagination__double-arrow"
      :disabled="page == 1"
      @click="handlerSelectedPage(1)"
    ></button>
    <button
      class="pagination__arrow"
      :disabled="page == 1"
      @click="handlerSelectedPage(page - 1)"
    ></button>
  </div>

  <div class="pagination__buttons-wrapper">
    <button
      v-for="numPage in pages"
      :key="numPage"
      :style="numPage == page ? selectedButton : {}"
      @click="handlerSelectedPage(numPage)"
    >
      {{ numPage }}
    </button>
  </div>

  <div class="pagination__nav-wrapper pagination__nav-wrapper--reverse">
    <button
      class="pagination__double-arrow"
      :disabled="page == countOfPages"
      @click="handlerSelectedPage(countOfPages)"
    ></button>
    <button
      class="pagination__arrow"
      :disabled="page == countOfPages"
      @click="handlerSelectedPage(page + 1)"
    ></button>
  </div>
</div>
</template>

<script>
export default {

```

					ДП.0902.10.000000.00 ПЗ	Лист
						187
Изм	Лист	№ докум.	Подпись	Дата		

```

name: "main-pagination",
props: {
  page: {
    type: Number,
    required: true,
  },
  countOfRecords: {
    type: Number,
    required: true,
  },
  limit: {
    type: Number,
    required: true,
  },
},
emits: ["update:page"],
data() {
  return {
    selectedButton: {
      backgroundColor: "var(--color-accent)",
      color: "var(--color-white)",
    },
    countOfViewPages: 5,
  };
},
computed: {
  countOfPages() {
    return Math.ceil(this.countOfRecords / this.limit);
  },
  maxPage() {
    if (this.page <= this.countOfPages) return this.countOfPages;
    if (this.page + this.countOfPages / 2 >= this.countOfPages)
      return this.countOfPages;
    return this.page + this.countOfPages - Math.ceil(this.countOfPages / 2);
  },
  pages() {
    const arrOfPages = [];
    for (
      let page = this.maxPage - (this.countOfPages - 1);
      page <= this.maxPage;
      ++page
    )
      arrOfPages.push(page);
    return arrOfPages;
  },
},
methods: {
  handlerSelectedPage(page) {
    this.$emit("update:page", page);
  },
},
};
</script>

<style scoped>
.pagination {
  display: flex;
  flex-direction: row;
  align-items: center;
  gap: 20px;
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						188
Изм	Лист	№ докум.	Подпись	Дата		

```

.pagination__nav-wrapper {
  display: flex;
  gap: 10px;
}

.pagination__arrow,
.pagination__double-arrow {
  height: 21px;
  background-repeat: no-repeat;
  background-position: center;
}

.pagination__arrow {
  width: 12px;
  background-image: url("../assets/arrow-green.svg");
}

.pagination__double-arrow {
  width: 24px;
  background-image: url("../assets/double-arrow-green.svg");
}

.pagination__arrow:disabled {
  background-image: url("../assets/arrow.svg");
}

.pagination__double-arrow:disabled {
  background-image: url("../assets/double-arrow.svg");
}

.pagination__nav-wrapper--reverse {
  transform: rotate(180deg);
}

.pagination__buttons-wrapper {
  display: flex;
  gap: 10px;
}

.pagination__buttons-wrapper button {
  padding: 2px 10px;
  border-radius: 5px;
  font-weight: 700;
}

p {
  margin-bottom: 0;
}
</style>

```

Программный код файла RecipeCard

```

<template>
<div class="recipe-card" @click="$router.push(`/recipe/${recipe.id}`)">
  
  <div class="recipe-card__content-wrapper">
    <h3>{{ recipe.name }}</h3>
    <p>
      {{

```

					ДП.0902.10.000000.00 ПЗ	Лист
						189
Изм	Лист	№ докум.	Подпись	Дата		

```

        recipe.description.length > 150
        ? recipe.description.slice(0, 150) + "..."
        : recipe.description
    }}
</p>
<p>
    Дата добавления:
    <span class="recipe-card__date">{{
        recipe.created_at.slice(0, 10).split("-").reverse().join("-")
    }}</span>
</p>
</div>
</div>
</template>

<script>
import RecipeService from "@services/recipe.service.js";
export default {
    name: "recipe-card",
    props: {
        recipe: {
            type: Object,
            default: () => {},
            required: true,
        },
    },
    data() {
        return {
            imageSrc: `http://localhost:5000/api/recipes/get-image/${this.recipe.id}`,
        };
    },
};
</script>

<style scoped>
.recipe-card {
    box-sizing: content-box;
    display: flex;
    gap: 20px;
    flex-wrap: wrap;
    cursor: pointer;
    transition: padding 0.25s ease-out;
}

.recipe-card:hover h3 {
    transition: color 0.25s ease-out;
}

.recipe-card:hover,
.recipe-card:focus {
    padding-left: 20px;
    border-left: 5px solid var(--color-accent);
}

.recipe-card:hover h3 {
    color: var(--color-accent);
}

.recipe-card__img {
    height: 160px;
    width: 230px;
    object-fit: cover;
    border-radius: 30px;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						190
Изм	Лист	№ докум.	Подпись	Дата		

```

}
.recipe-card__content-wrapper {
  max-width: 400px;
}

.recipe-card__date {
  color: var(--color-accent);
}

p {
  margin-bottom: 10px;
}

.recipe-card__content-wrapper p:last-child {
  margin-bottom: 0;
}

@media (max-width: 991px) {
  .recipe-card__img {
    height: 160px;
    width: 300px;
  }
}
</style>

```

Программный код файла RecipeCards

```

<template>
  <div class="recipe-cards recipes-offset">
    <recipe-card
      v-for="recipe in recipes"
      :key="recipe.id"
      :recipe="recipe"
    ></recipe-card>
  </div>
</template>

<script>
import RecipeCard from "@/components/RecipeCard.vue";
export default {
  name: "recipe-cards",
  components: {
    RecipeCard,
  },
  props: {
    recipes: {
      type: Array,
      default: () => [],
      required: true,
    },
  },
};
</script>

<style scoped>
.recipe-cards {
  display: flex;
  gap: 20px;
  flex-direction: column;

```

					ДП.0902.10.000000.00 ПЗ	Лист
						191
Изм	Лист	№ докум.	Подпись	Дата		

```
}  
</style>
```

Программный код файла RegistrationForm

```
<template>  
  <Form @submit="handlerRegister" :validation-schema="schema" class="reg-form">  
    <h3>Регистрация</h3>  
    <div class="reg-form__content-wrapper">  
      <label for="login">Логин</label>  
      <Field class="reg-form__form-input" type="text" name="login"></Field>  
      <ErrorMessage name="login" class="reg-form__error" />  
      <label for="email">E-mail</label>  
      <Field class="reg-form__form-input" type="email" name="email"></Field>  
      <ErrorMessage name="email" class="reg-form__error" />  
      <label for="password">Пароль</label>  
      <Field  
        class="reg-form__form-input"  
        type="password"  
        name="password"  
      ></Field>  
      <ErrorMessage name="password" class="reg-form__error" />  
      <label for="repeatPassword">Повторите пароль</label>  
      <Field  
        class="reg-form__form-input"  
        type="password"  
        name="repeatPassword"  
      ></Field>  
      <ErrorMessage name="repeatPassword" class="reg-form__error" />  
    </div>  
    <p v-if="errorMessage" class="reg-form__error">{{ errorMessage }}</p>  
    <div v-if="successMessage" class="alert alert-success" role="alert">  
      {{ successMessage }}  
    </div>  
    <div class="reg-form__btn-wrapper">  
      <form-button :disabled="isLoading">Отправить</form-button>  
    </div>  
  </Form>  
</template>
```

```
<script>  
import FormInput from "@/components/UI/FormInput.vue";  
import FormButton from "@/components/UI/FormButton.vue";  
import { Form, Field, ErrorMessage } from "vee-validate";  
import * as yup from "yup";  
import { storeToRefs } from "pinia";  
import { useAuthStore } from "@/stores/auth";  
export default {  
  name: "registration-form",  
  components: {  
    FormInput,  
    FormButton,  
    Form,  
    Field,  
    ErrorMessage,  
  },  
  emits: ["hideDialog"],  
  setup() {  
    const auth = useAuthStore();
```

					ДП.0902.10.000000.00 ПЗ	Лист
						192
Изм	Лист	№ докум.	Подпись	Дата		


```

const { userData } = storeToRefs(auth);
const { register } = auth;
return {
  userData,
  register,
};
},
data() {
  return {
    successMessage: "",
    errorMessage: "",
    isLoading: false,
  };
},
computed: {
  schema() {
    return yup.object({
      login: yup.string().trim().required("Обязательное поле"),
      email: yup
        .string()
        .trim()
        .required("Обязательное поле")
        .email("Не верный формат"),
      password: yup
        .string()
        .trim()
        .min(8, "Минимум 8 символов")
        .required("Обязательное поле"),
      repeatPassword: yup
        .string()
        .trim()
        .min(8)
        .required("Обязательное поле"),
    });
  },
},
methods: {
  hideDialog(delay) {
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        resolve(this.$emit("hideDialog"));
      }, delay);
    });
  },
},

async handlerRegister(values) {
  this.successMessage = "";
  this.errorMessage = "";
  if (values.password !== values.repeatPassword) {
    this.errorMessage = "Пароли не совпадают";
    return;
  }
  this.isLoading = true;
  try {
    await this.register({
      login: values.login,
      email: values.email,
      password: values.password,
      role: "user",
    });
    this.successMessage = "На почту отправлено письмо для активации";
    await this.hideDialog(4000);
  }
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						193
Изм	Лист	№ докум.	Подпись	Дата		

```

    } catch (error) {
      this.errorMessage = error.response.data.message;
    } finally {
      this.isLoading = false;
    }
  },
},
};
</script>

```

```

<style scoped>
.reg-form {
  display: flex;
  flex-direction: column;
  margin-top: 30px;
}

.reg-form__error {
  color: red;
}

.reg-form__content-wrapper {
  display: flex;
  flex-direction: column;
  margin-top: 10px;
}

.reg-form__form-input {
  padding: 2px 10px;
  border: 1px solid var(--color-light-black);
  border-radius: 8px;
  margin: 5px 0;
}

.reg-form__btn-wrapper {
  display: flex;
  gap: 20px;
  margin-top: 20px;
}
</style>

```

Программный код файла SideMenu

```

<template>
<div class="side-menu" v-if="show" @click.stop="hideMenu">
  <div @click.stop class="side-menu__content">
    <p class="logo"><span>LF</span>Resipes</p>
    <nav class="side-menu__nav">
      <ul class="side-menu__elements">
        <li class="side-menu__el">
          <button class="side-menu__button" @click="goByRoute('/')">
            Приветствие
          </button>
        </li>
        <li class="side-menu__el">
          <button class="side-menu__button" @click="goByRoute('/recipes')">
            Рецепты
          </button>
        </li>
        <li class="side-menu__el" v-if="userData.user">
          <button class="side-menu__button" @click="goByRoute('/add-recipe')">
            Добавить рецепт
          </button>
        </li>
      </ul>
    </nav>
  </div>
</div>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						194
Изм	Лист	№ докум.	Подпись	Дата		

```

        </button>
      </li>
      <li class="side-menu__el">
        <button class="side-menu__button" @click="goByRoute('/about')">
          О нас
        </button>
      </li>
      <li class="side-menu__el" v-if="userData.user?.role === 'admin'">
        <button class="side-menu__button" @click="goByRoute('/admin')">
          Панель администратора
        </button>
      </li>
    </ul>
  </nav>
</div>
</div>
</template>

```

```

<script>
import { storeToRefs } from "pinia";
import { useUserStore } from "@/stores/user";
export default {
  name: "side-menu",
  emits: ["update:show"],
  props: {
    show: {
      type: Boolean,
      default: false,
    },
  },

  setup() {
    const user = useUserStore();
    const { userData } = storeToRefs(user);
    return {
      userData,
    };
  },

  methods: {
    hideMenu() {
      this.$emit("update:show", false);
    },

    goByRoute(route) {
      this.hideMenu();
      this.$router.push(route);
    },
  },
};
</script>

```

```

<style scoped>
.side-menu {
  display: flex;
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.5);
  z-index: 100;
}

```

					ДП.0902.10.000000.00 ПЗ	Лист
						195
Изм	Лист	№ докум.	Подпись	Дата		

```

}

.side-menu__content {
  position: relative;
  left: 0;
  top: 0;
  bottom: 0;
  background: var(--color-white);
  min-width: 280px;
  padding: 0 30px;
}

.logo {
  margin-top: 60px;
  font-size: 20px;
  color: var(--color-light-black);
}

.logo > span {
  color: var(--color-accent);
}

.side-menu__elements {
  display: flex;
  flex-direction: column;
  margin-top: 40px;
}

.side-menu__el {
  border-bottom: 1px solid #000;
  transition: border-color 0.15s ease-out;
}

.side-menu__el:hover {
  border-color: var(--color-accent);
}

.side-menu__el:focus {
  border-color: var(--color-dark-green);
}

.side-menu__button {
  width: 100%;
  text-align: left;
  line-height: 1;
  font-size: 18px;
  padding: 20px 0;
  color: var(--color-text);
  transition: color 0.15s ease-out;
}

.side-menu__button:hover {
  color: var(--color-accent);
}

.side-menu__button:focus {
  color: var(--color-dark-green);
}
</style>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						196
Изм	Лист	№ докум.	Подпись	Дата		

Программный код файла UserMenu

```
<template>
<div class="user-menu" v-if="show" @click.stop="hideMenu">
  <div class="user-menu__content">
    <nav @click.stop class="user-menu__nav">
      <ul class="user-menu__elements">
        <li class="user-menu__el">
          <button class="user-menu__button" @click="goByRoute('/person')">
            Личный кабинет
          </button>
        </li>
        <li class="user-menu__el">
          <button class="user-menu__button" @click="logout">Выход</button>
        </li>
      </ul>
    </nav>
  </div>
</div>
</template>
```

```
<script>
import EventBus from "@common/EventBus";
export default {
  name: "user-menu",
  emits: ["update:show"],
  props: {
    show: {
      type: Boolean,
      default: false,
    },
  },
  methods: {
    hideMenu() {
      this.$emit("update:show", false);
    },
    goByRoute(route) {
      this.hideMenu();
      this.$router.push(route);
    },
    logout() {
      this.hideMenu();
      EventBus.dispatch("logout");
    },
  },
};
</script>
```

```
<style scoped>
.user-menu {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: transparent;
}
```

```
.user-menu::before {
  content: "";
```

					ДП.0902.10.000000.00 ПЗ	Лист
						197
Изм	Лист	№ докум.	Подпись	Дата		

```

display: block;
position: fixed;
top: 0;
left: 0;
right: 0;
bottom: 0;
background: transparent;
z-index: 99;
}

.user-menu__content {
position: relative;
right: 114px;
top: 45px;
background: var(--color-white);
border: 1px solid rgba(0, 0, 0, 0.6);
border-radius: 10px;
min-width: 150px;
height: auto;
padding: 10px;
z-index: 100;
}

.user-menu__elements {
display: flex;
flex-direction: column;
}

.user-menu__button {
line-height: 1;
font-size: 14px;
padding: 10px 0;
color: var(--color-text);
transition: color 0.15s ease-out;
}

.user-menu__button:hover {
color: var(--color-accent);
}

.user-menu__button:focus {
color: var(--color-dark-green);
}
</style>

```

					ДП.0902.10.000000.00 ПЗ	Лист
						198
Изм	Лист	№ докум.	Подпись	Дата		