

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра вычислительной техники

кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В. Непомнящий

подпись

« ____ » _____ 2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 — Информатика и вычислительная техника

Разработка системы управления репозиторием контента

Руководитель

подпись, дата

доцент, канд. техн. наук М. С. Медведев

Студент

подпись, дата

А. А. Горбацевич

Нормконтролер

подпись, дата

доцент, канд. техн. наук М. С. Медведев

Красноярск 2023

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В. Непомнящий

« ____ » _____ 2023 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Горбацевичу Андрею Анатольевичу
фамилия, имя, отчество

Группа КИ19-07Б Направление (специальность) 090301
номер код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы: Разработка системы
управления репозиторием контента для онлайн сервисов

Утверждена приказом по университету № _____ от _____

Руководитель ВКР: М.С. Медведев, канд. техн. наук, доцент каф. ВТ ИКИТ
инициалы, фамилия, учёная степень, должность, место работы

СФУ

Исходные данные для ВКР:

1) Ларман, К. Применение UML 2 и шаблонов проектирования / К. Ларман. – Москва: Вильямс, 2014 – 624 с.

2) Фаулер, М. UML. Основы, 3-е издание. – Перевод с англ. / М. Фаулер. – Санкт-Петербург: Символ-Плюс, 2004 – 192 с., ил.

3) Python 3 Documentation [Электронный ресурс]. URL: <https://docs.python.org/3/>

4) Dart programming language [Электронный ресурс]. URL: <https://dart.dev/>

5) Рекомендации руководителя.

Перечень разделов ВКР:

1) Спецификация требований к системе

2) Проектирование

3) Реализация и тестирование

Перечень графического материала: демонстрационное видео, презентация

Руководитель ВКР _____
подпись М.С. Медведев
инициалы, фамилия

Задание принял к исполнению _____
подпись А.А. Горбацевич
инициалы, фамилия

« _____ » _____ Г.
дата

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка системы управления репозиторием контента для онлайн-сервисов» содержит 58 страниц текстового документа, 50 иллюстраций, 1 таблицу, 18 использованных источников.

РЕПОЗИТОРИЙ КОНТЕНТА, ПРОЦЕСС ICONIX, CRUD, АРХИТЕКТУРА MVC.

Цель работы: разработать систему управления репозиторием контента с открытым исходным кодом.

Выпускная квалификационная работа делится на четыре части: введение, три главы основной части и заключение.

Во введении определяется цель работы и необходимые задачи для достижения этой цели.

В первой главе основной части проводится сравнение аналогов, выявляются требования к будущей системе, описываются прецеденты.

Во второй главе производится проектирование и составление предметной области системы на основе диаграмм пригодности и последовательности.

В третьей главе описывается разработка системы: используемые инструменты и процесс создания отдельных частей системы, а также тестирование. Также в этой главе приведены инструкции для пользователя и для разработчика.

В заключении подводятся итоги работы, приводятся примеры возможных улучшений системы и даются ссылки на исходный код компонентов системы.

СОДЕРЖАНИЕ

Введение.....	4
1 Спецификация требований к системе	5
1.1 Анализ аналогов	5
1.2 Функциональные требования.....	6
1.3 Макеты интерфейса и текстовое описание прецедентов	10
1.4 Модель предметной области.....	34
1.5 Выводы по главе.....	35
2 Проектирование.....	36
2.1 Диаграммы пригодности и последовательности	36
2.1.1 Прецедент «Вход в систему»	36
2.1.2 Прецедент «Создать новую таблицу».....	38
2.1.3 Прецедент «Создание пользователя».....	40
2.1.4 Прецедент «Создание группы пользователей».....	41
2.1.5 Прецедент «Редактирование разрешений группы на таблицы»	43
2.1.6 Прецедент «Редактирование существующего ресурса»	45
2.2 Выводы по главе.....	47
3 Реализация и тестирование	48
3.1 Выбор инструментов.....	48
3.2 Реализация	49
3.2.1 Библиотека динамического взаимодействия с базой данных Based	49
3.2.2 Программный интерфейс приложения (API) Tuuli Backend	49
3.2.3 Библиотека TuuliApi для языка программирования Dart	50
3.2.4 Графический интерфейс пользователя TuuliApp.....	51
3.3 Документация	51
3.3.1 Инструкция пользователя	51
3.3.2 Инструкция разработчика	52
3.4 Тестирование	54
3.5 Выводы по главе.....	54
Заключение	55
Список сокращений	56
Список использованных источников	57

ВВЕДЕНИЕ

В современном мире онлайн-сервисы являются неотъемлемой частью нашей повседневной жизни. Постоянно увеличивающееся число пользователей в сочетании с разнообразием контента, предоставляемого на этих платформах, создает значительную потребность в эффективной системе управления информацией как внутри компании, так и вне её.

Цель работы состоит в разработке системы управления репозиторием контента под названием Tuuli с открытым исходным кодом, удовлетворяющей потребности компаниям любого размера.

Система управления репозиторием контента – это вид ПО, которое позволяет управлять хранилищем цифрового контента. Под контентом в данном случае можно понимать различные типы информации, включая текстовые документы, изображения, видео, аудио, архивы и другие мультимедийные файлы, а также таблицы баз данных и их содержимое.

Актуальность и практическая значимость обосновывается **первой главой работы**, в которой был сделан обзор существующих решений, которые, как оказалось, не обладали нужной гибкостью и, некоторые, функционалом. Также оказалось, что не имеется отечественных аналогов систем данного типа (в частности, с открытым исходным кодом) – это тоже значительно повышает практическую значимость работы.

По результатам первой главы были выявлены необходимые требования к системе, которая затем была спроектирована по требованиям во второй главе и результат реализации описан в третьей.

1 Спецификация требований к системе

1.1 Анализ аналогов

На GitHub по запросу «headless-cms» найдено 4590 открытых репозиторийев. Являясь веб-сервисом, большинство из проектов написаны с использованием JavaScript/TypeScript, однако есть и другие, написанные с использованием C#, Java, Go или Python.

Для анализа были выбраны три популярных в сообществе репозитория: directus/directus [5], keystonejs/keystone [6] и appwrite/appwrite [7]. Сравнение данных проектов выполнено в таблице 1.

Таблица 1 – Сравнительная таблица репозиторийев

Характеристика	Репозиторий		
	directus/directus	keystonejs/keystone	appwrite/appwrite
Лицензия	Открытая, GPL-3	Открытая, MIT	Открытая, BSD-3
Лёгкость освоения	Высокая (ручная настройка после запуска сервиса почти не требуется)	Высокая (обеспечивается низкой гибкостью настроек)	Низкая (выбранный ориентир разработки так же не является простым, но имеет куда большую документацию)
Тип СУБД	Любая реляционная СУБД	Только MariaDB	Только MongoDB
Нагрузка на сервер	Низкая	Низкая	Высокая (требуется множество различных сервисов)
Гибкость настройки	Высокая (большой стандартных полей сущностей, имеется возможность расширения)	Низкая (малый набор полей для сущностей, малые возможности расширения)	Не гибкая (малый набор полей, возможность расширения отсутствует)
Интегрируемость	Сложная, документация не везде даёт точные ответы, однако имеются библиотеки почти под все крупные языки программирования	Простая, сервис автоматически реализует REST-API	Простая, реализует протокол FirebaseAPI
Docker-образ	Имеется	Отсутствует	Имеется

Исходный код всех трёх систем доступен, я смог самостоятельно их протестировать и вся информация по анализу составлена из моего опыта использования. Самой гибкой является система directus, поэтому некоторые элементы моей системы частично вдохновлены directus.

1.2 Функциональные требования

Система должна быть лёгкой в освоении и использовании, иметь гибкий функционал создания сущностей. Должна иметь простой способ открытия данных наружу, а также быть независимой от графических фреймворков – чтобы на её базе можно было построить приложение на любой платформе с любым набором инструментов. Также система должна иметь открытый исходный код и разрешающую свободную лицензию (например, «Лицензия X11/MIT»).

Диаграммы прецедентов разрабатываемой системы приведены на рисунках 1–6.

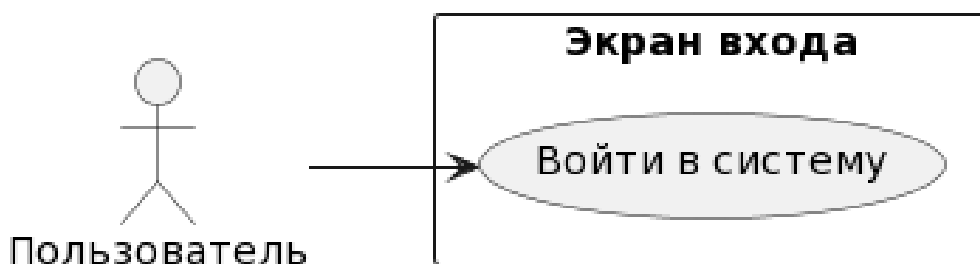


Рисунок 1 – Диаграмма вариантов использования. Вход в систему



Рисунок 2 – Диаграмма вариантов использования. Главный экран системы



Рисунок 3 – Диаграмма вариантов использования. Управление таблицами БД

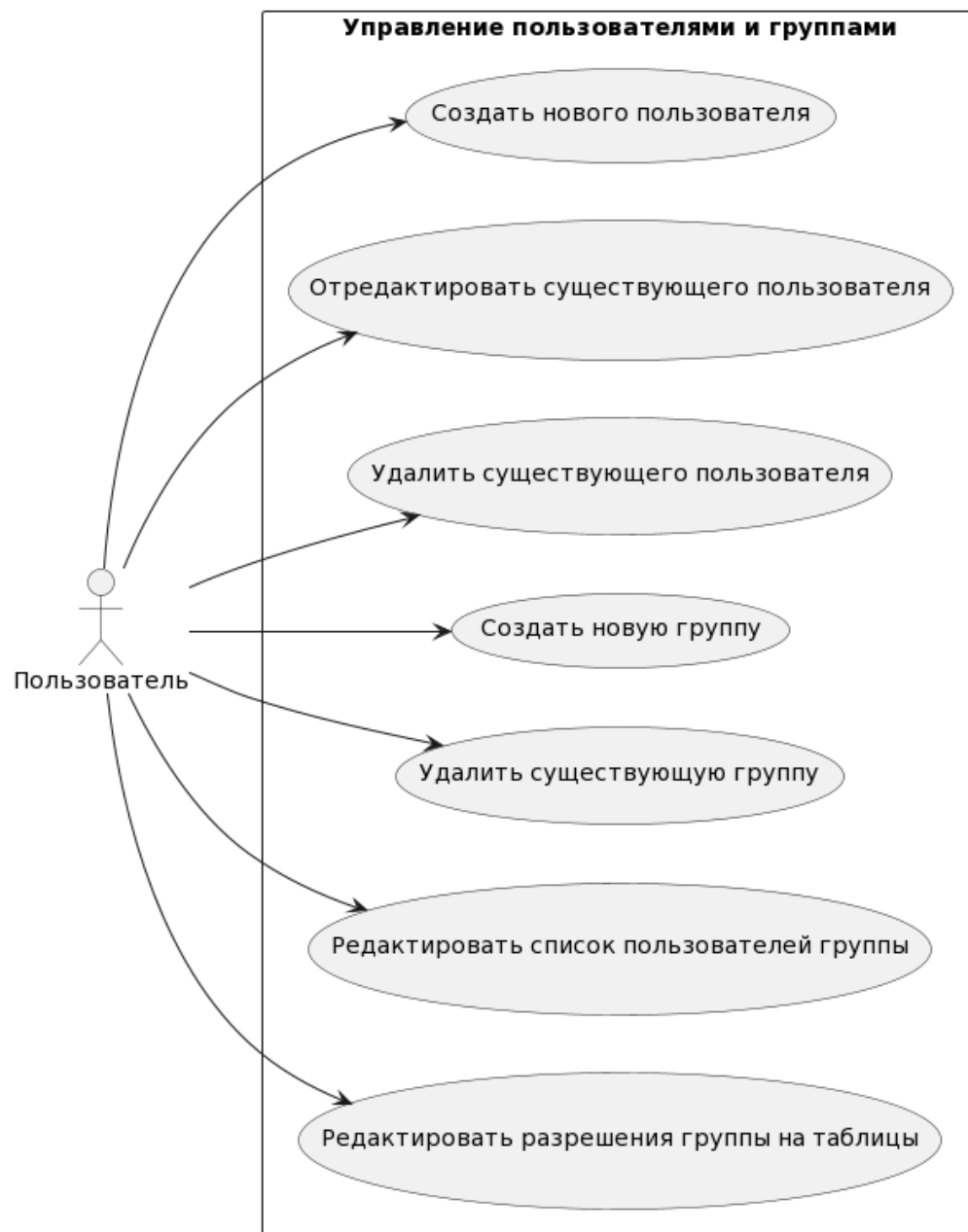


Рисунок 4 – Диаграмма вариантов использования. Управление пользователями

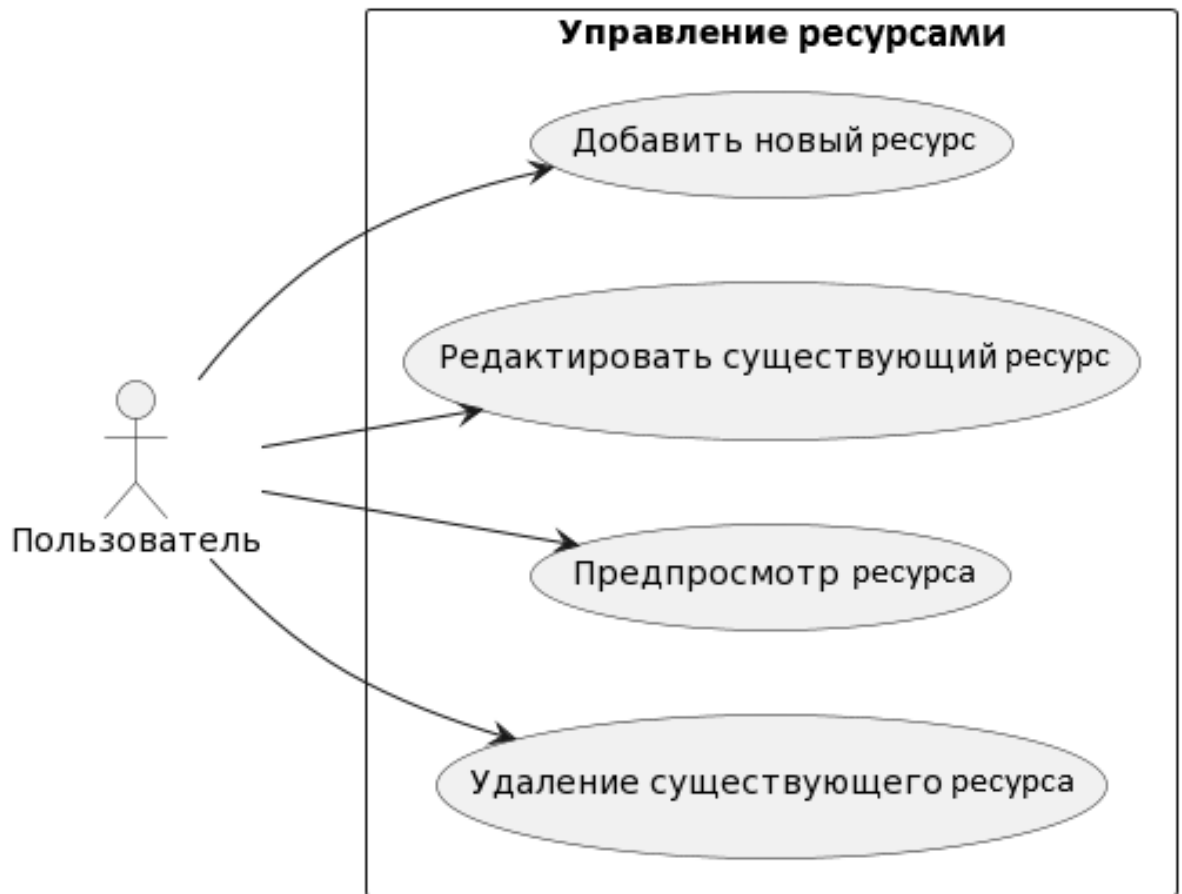


Рисунок 5 – Диаграмма вариантов использования. Управление ресурсами

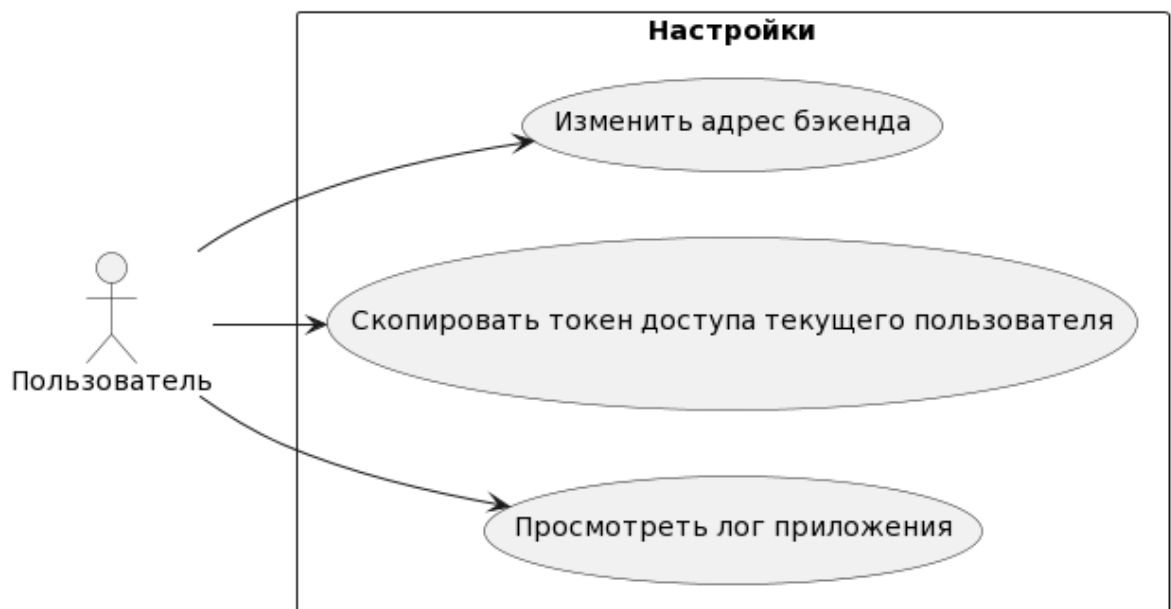


Рисунок 6 – Диаграмма вариантов использования. Настройка системы

1.3 Макеты интерфейса и текстовое описание прецедентов

Название прецедента: войти в систему.

Цель сценария: войти в систему.

Предусловия: клиент системы не содержит ключ доступа пользователя, открыта страница входа (рисунок 7).

Основной сценарий:

- 1) пользователь вводит адрес системы, свой логин и пароль;
- 2) пользователь нажимает кнопку «Войти».

Постусловия: клиент системы сохраняет полученный от системы ключ доступа.



Рисунок 7 – Экран входа клиента системы

Название прецедента: выйти из системы.

Цель сценария: завершить сессию пользователя.

Предусловия: клиент системы содержит ключ доступа пользователя.

Основной сценарий:

1) в боковой или верхней панели (рисунок 8) пользователь нажимает кнопку «Выйти».

Постусловия: клиент системы удаляет ключ доступа.

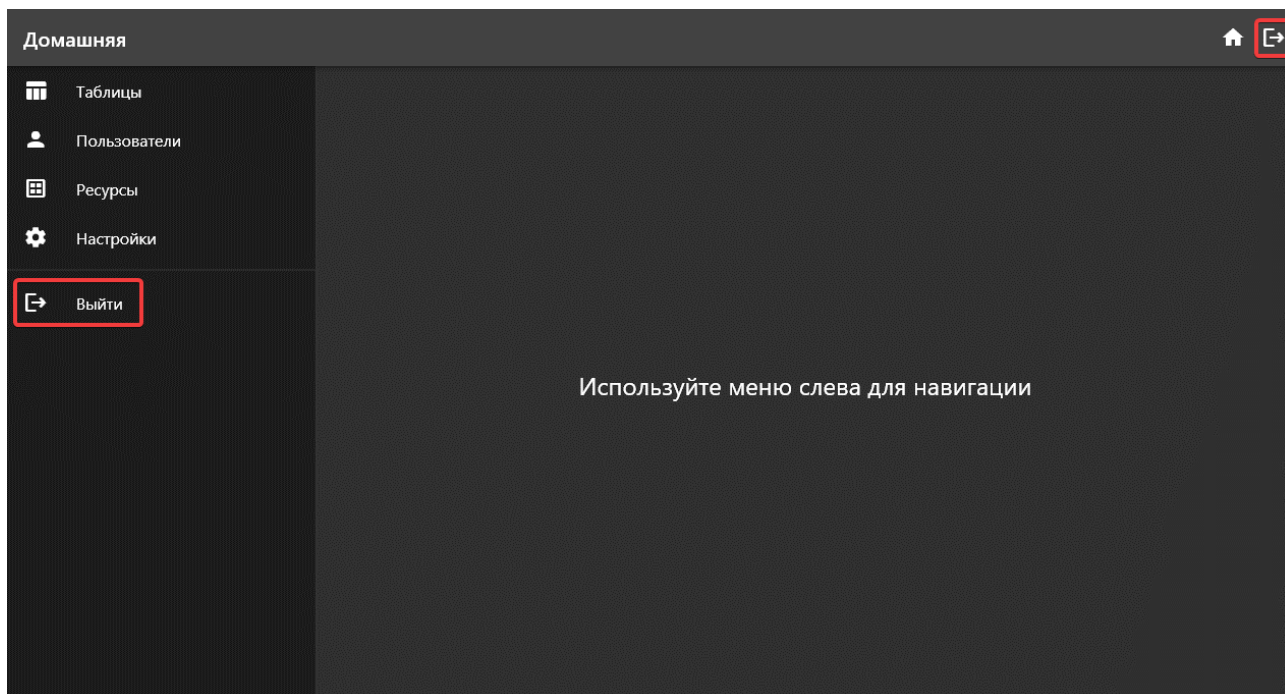


Рисунок 8 – Главный экран клиента системы

Название прецедента: создание таблицы.

Цель сценария: создать новую таблицу.

Предусловия: открыта панель таблиц на главной странице (рисунок 9), пользователь состоит в группе администраторов.

Основной сценарий:

- 1) пользователь нажимает кнопку со знаком плюс;
- 2) в открывшемся диалоге (рисунок 10) пользователь настраивает новую страницу;
- 3) пользователь нажимает кнопку «Создать».

Постусловия: диалог закрывается, в базе данных создана запись о новой таблице, также в ней создаётся соответствующая таблица, список таблиц обновляется.

Альтернативный сценарий:

- 1) пользователь нажимает кнопку со знаком плюс;
- 2) в открывшемся диалоге пользователь настраивает новую страницу;
- 3) пользователь нажимает кнопку «Закрыть».

Альтернативное постусловие: диалог закрывается, в базе данных не создаётся запись о новой таблице, новая таблица не добавляется.

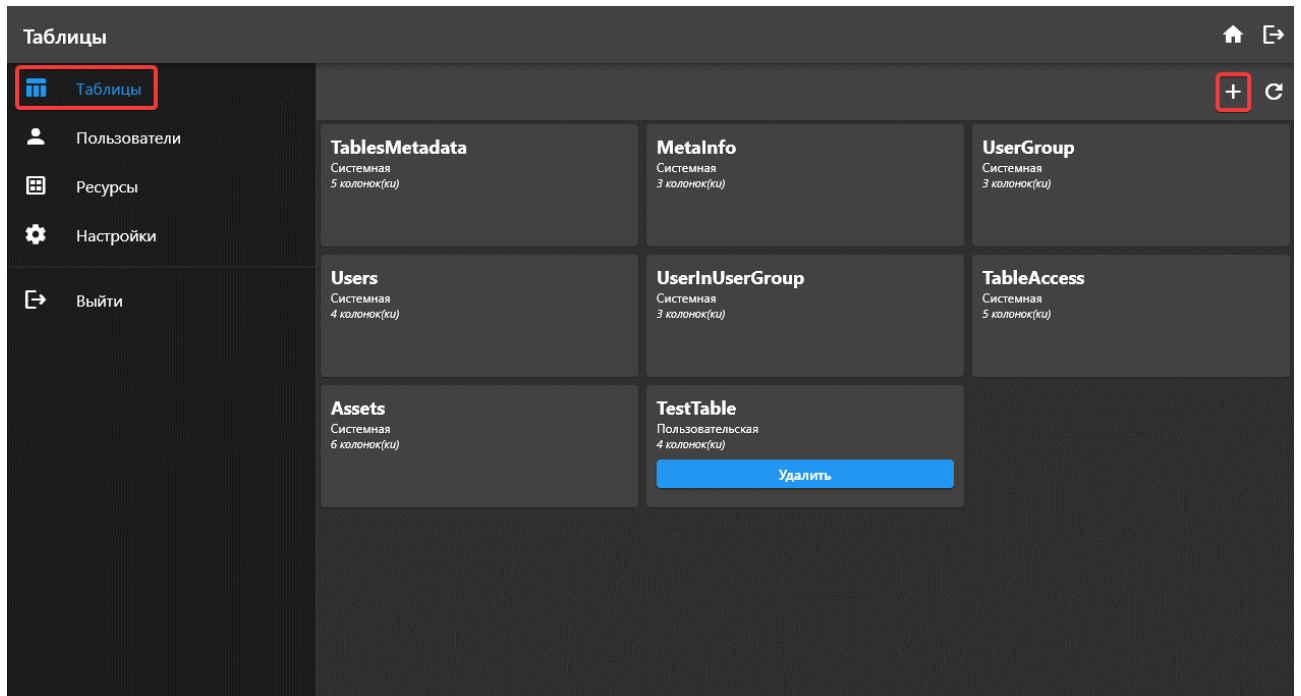


Рисунок 9 – Панель просмотра таблиц БД

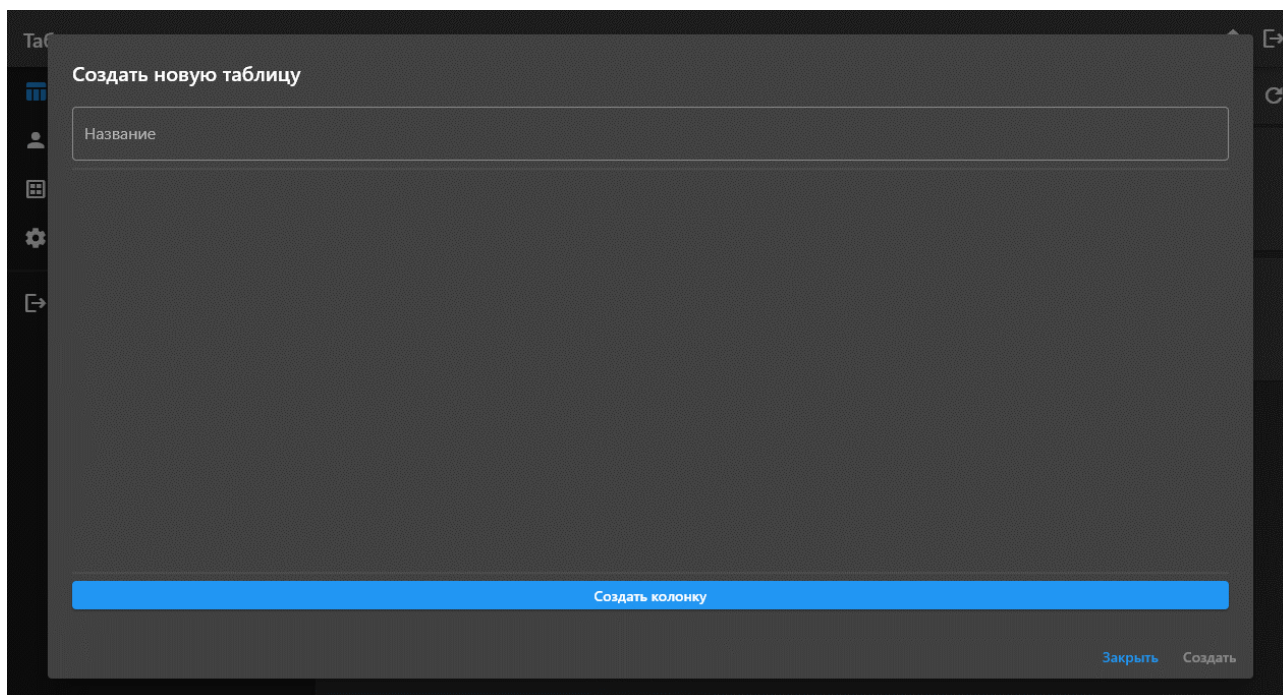


Рисунок 10 – Диалог создания новой таблицы БД

Название прецедента: удаление таблицы БД.

Цель сценария: удаление существующей таблицы.

Предусловия: таблица существует, таблица не является системной, открыта панель таблиц, пользователь состоит в группе администраторов.

Основной сценарий:

- 1) пользователь нажимает кнопку «Удалить» на карточке таблицы;
- 2) пользователь подтверждает своё намерение во всплывающем диалоге (рисунок 11).

Постусловия: из базы данных удалена запись о таблице, а также сама таблица.

Альтернативный сценарий:

- 1) пользователь нажимает кнопку «Удалить» на карточке таблицы;
- 2) пользователь отменяет своё намерение во всплывающем диалоге.

Альтернативное постусловие: из базы данных не удалена запись о таблице, как и сама таблица.

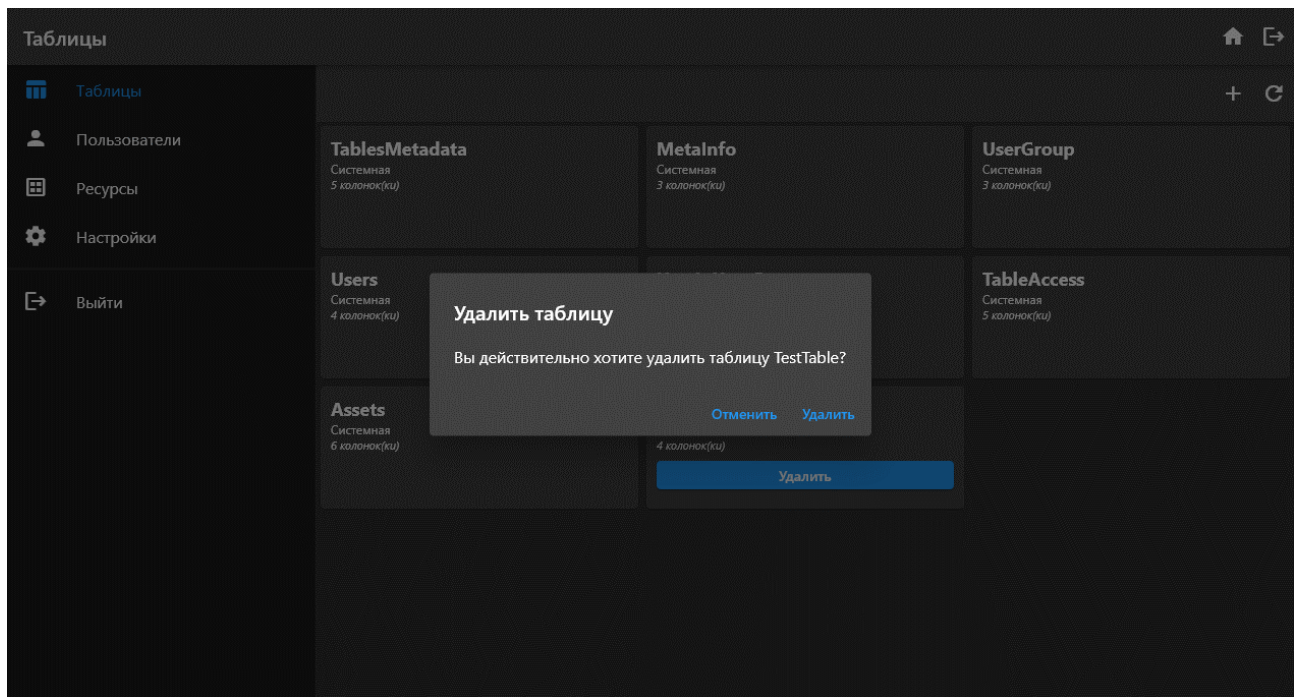


Рисунок 11 – Диалог подтверждения намерения удалить таблицу

Название прецедента: добавление нового элемента в таблицу.

Цель сценария: добавить новый элемент в таблицу.

Предусловия: открыт диалог таблицы (рисунок 12), пользователь состоит в группе с разрешением на редактирование данной контента.

Основной сценарий:

- 1) пользователь вводит необходимые данные в поля ввода;
- 2) пользователь нажимает кнопку со знаком плюс.

Постусловия: в таблице базы данных появляется новая запись.

Альтернативный сценарий:

- 1) пользователь вводит необходимые данные в поля ввода;
- 2) пользователь нажимает кнопку со знаком крестик.

Альтернативное постусловие: поля ввода очищаются, новая запись не создаётся.

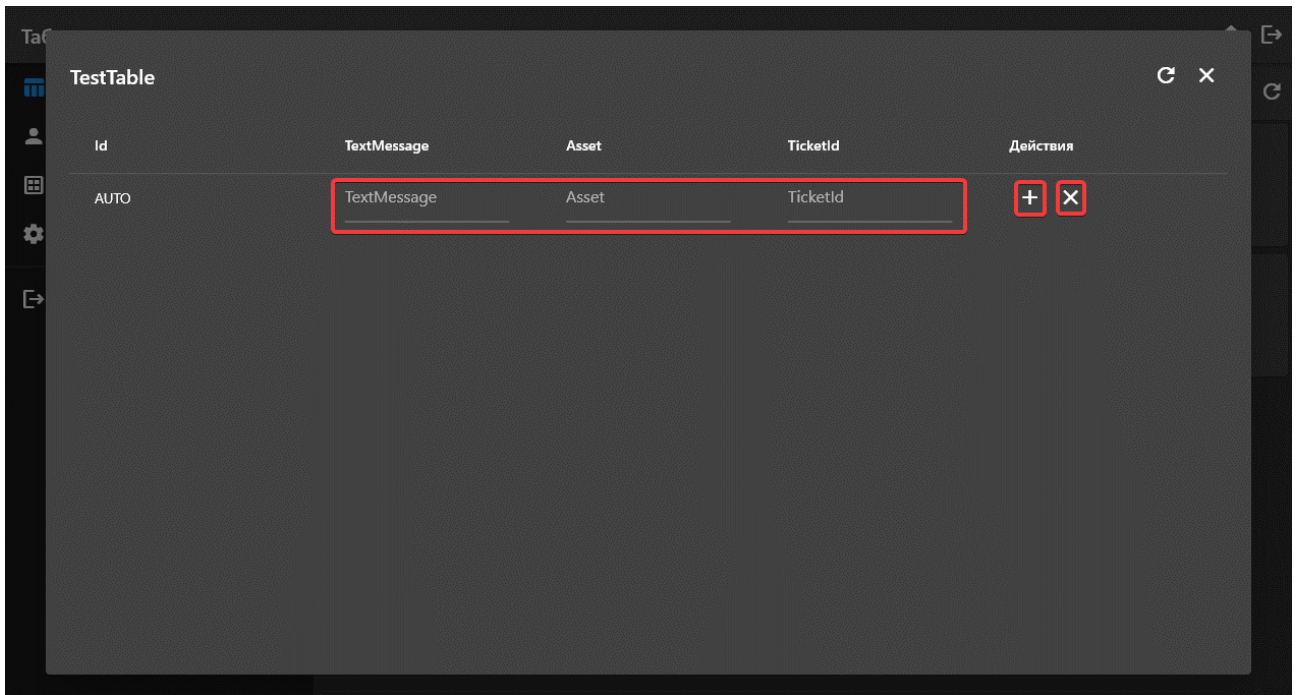


Рисунок 12 – Диалог таблицы

Название прецедента: изменение элемента в таблице.

Цель сценария: изменить данные существующего элемента в таблице.

Предусловия: открыт диалог таблицы (рисунок 13), необходимый к редактированию элемент существует, пользователь состоит в группе с разрешением на редактирование данной контента.

Основной сценарий:

1) пользователь два раза нажимает по полю, которое необходимо отредактировать;

2) в открывшемся диалоге пользователь вводит новые данные (в зависимости от типа контента открываются разные диалоги, примеры представлены на рисунках 14 и 15);

3) пользователь подтверждает своё намерение нажатием на кнопку «Ок».

Постусловия: диалог закрывается, в таблице базы данных обновляется редактируемая запись.

Альтернативный сценарий:

1) пользователь два раза нажимает по полю, которое необходимо отредактировать;

2) пользователь отменяет своё намерение нажатием кнопки «Отменить».

Альтернативное постусловие: диалог закрывается, информация в таблице базы данных не обновляется.

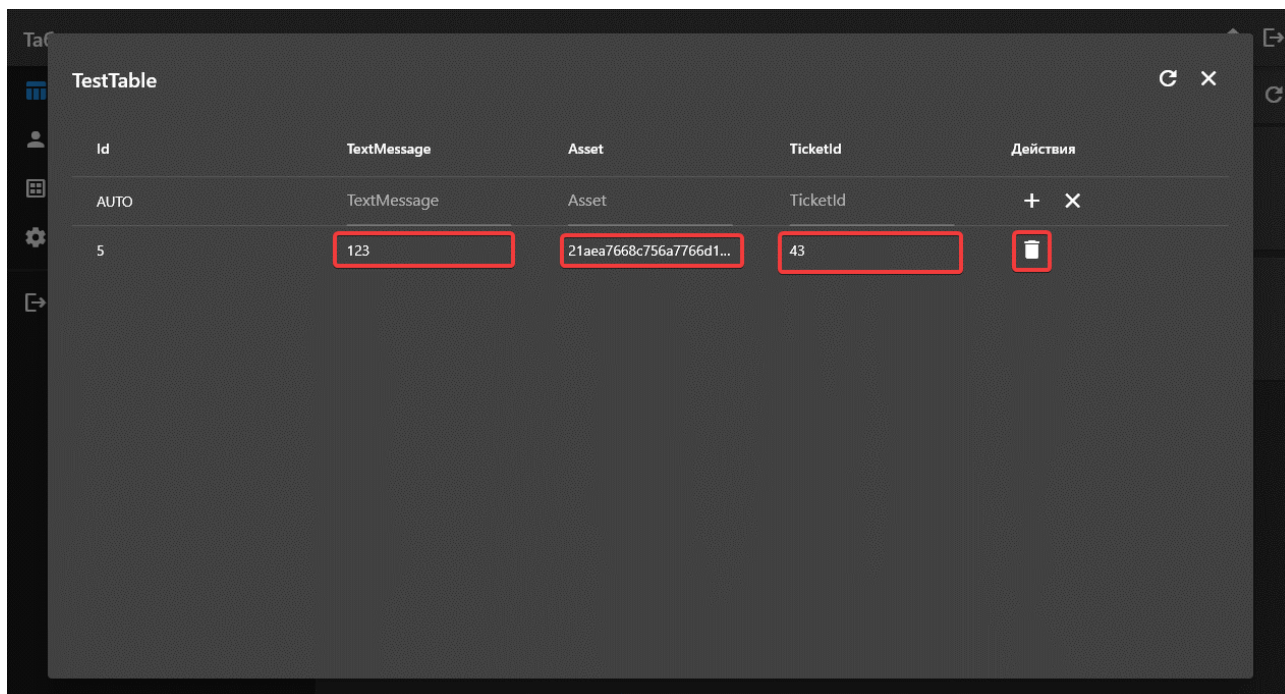


Рисунок 13 – Диалог таблицы с существующим элементом

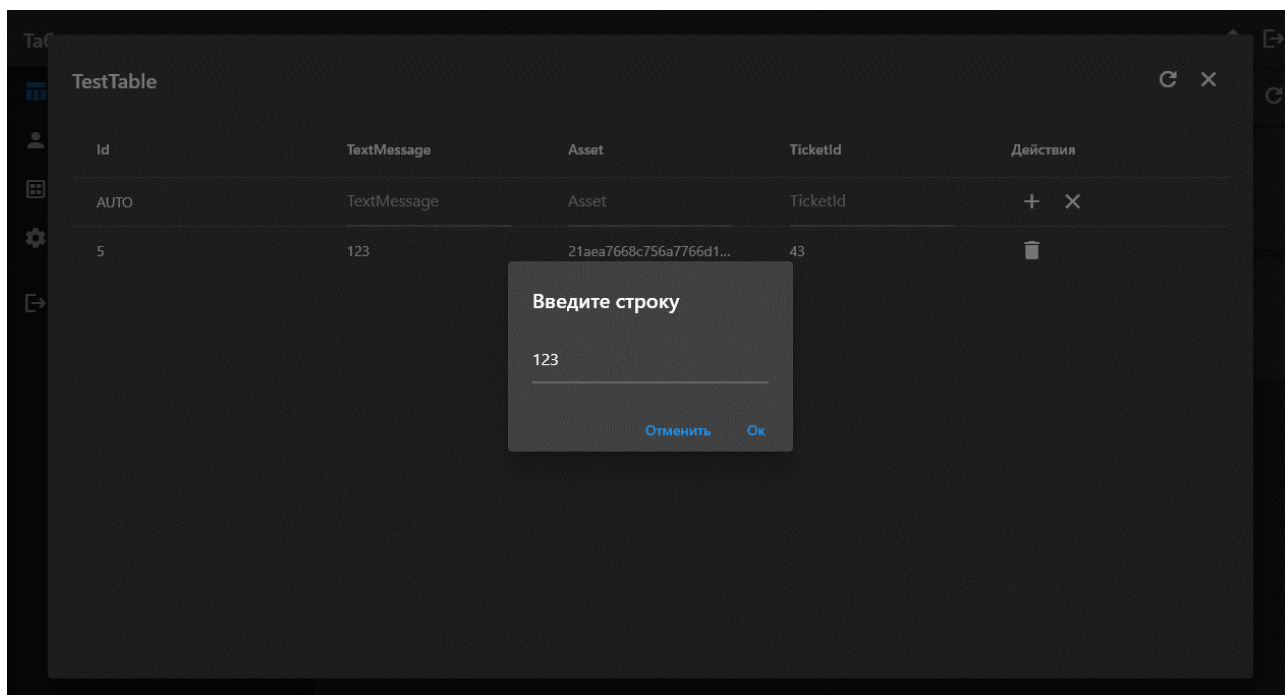


Рисунок 14 – Диалог изменения поля строковых данных

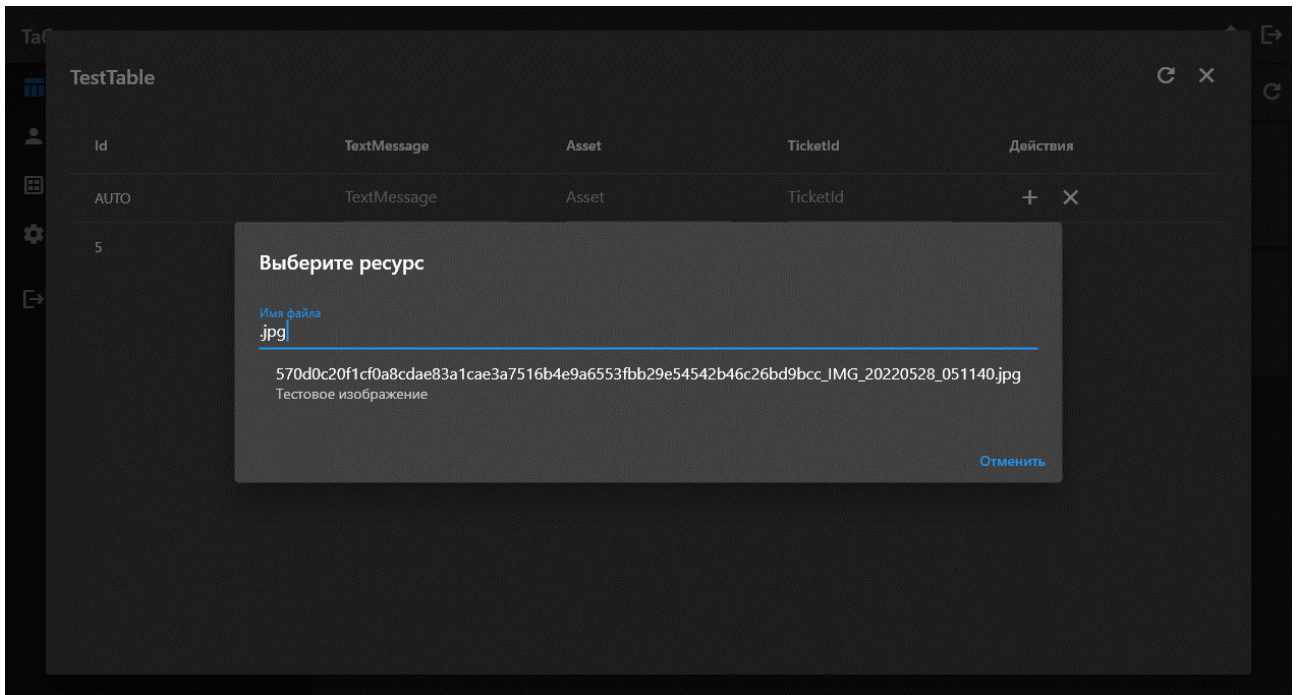


Рисунок 15 – Диалог изменения поля ресурса

Название прецедента: удаление элемента из таблицы.

Цель сценария: удалить существующий элемент из таблицы.

Предусловия: открыт диалог таблицы (рисунок 13), необходимый к редактированию элемент существует, пользователь состоит в группе с разрешением на редактирование данной контента.

Основной сценарий:

1) пользователь нажимает на кнопку корзины на строке элемента таблицы.

Постусловия: из таблицы базы данных удаляется запись.

Название прецедента: создание нового пользователя.

Цель сценария: создать нового пользователя системы.

Предусловия: открыта панель пользователей системы (рисунок 16), выбран модуль «Пользователи», пользователь состоит в группе администраторов

Основной сценарий:

1) пользователь нажимает кнопку со знаком плюс;

2) в открывшемся диалоговом окне (рисунок 17) пользователь вводит логин и пароль нового пользователя;

3) пользователь нажимает кнопку «Создать».

Постусловия: диалог закрывается, в базу данных системы добавлена запись о новом пользователе.

Альтернативный сценарий:

1) пользователь нажимает кнопку со знаком плюс;

2) пользователь нажимает кнопку «Отменить».

Альтернативные постусловия: диалог закрывается, в базу данных системы ничего не добавляется.

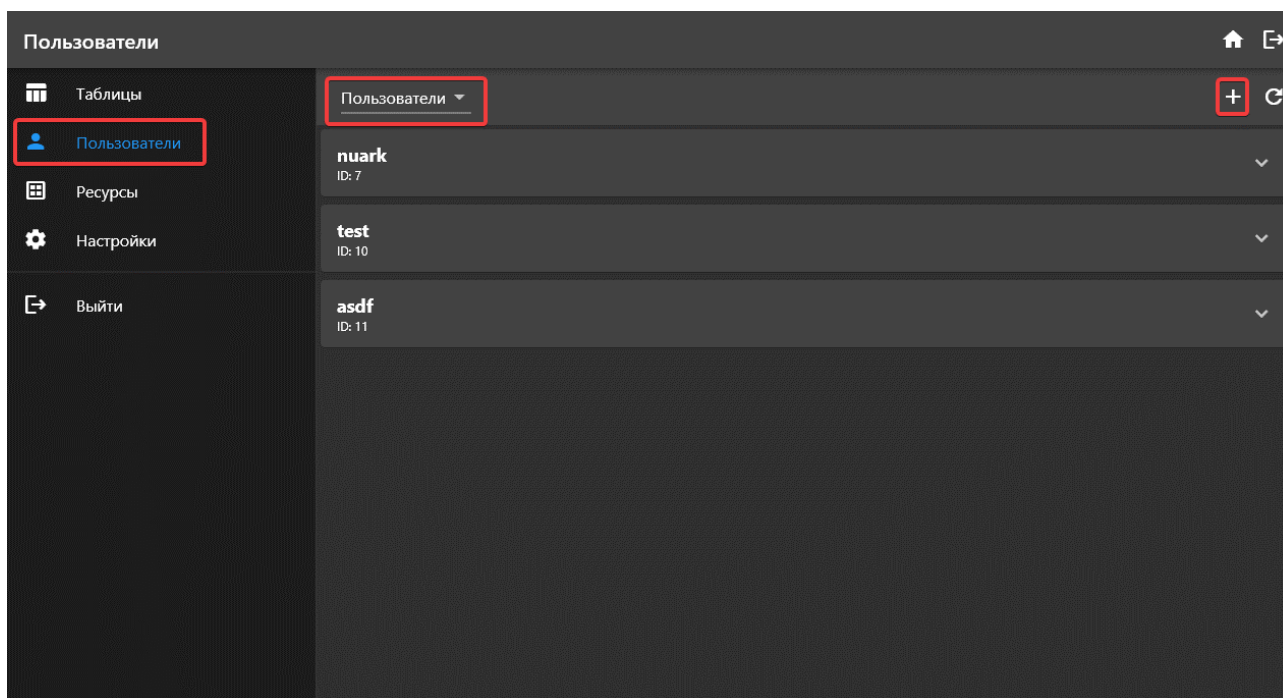


Рисунок 16 – Панель пользователей с выбранным модулем «Users»

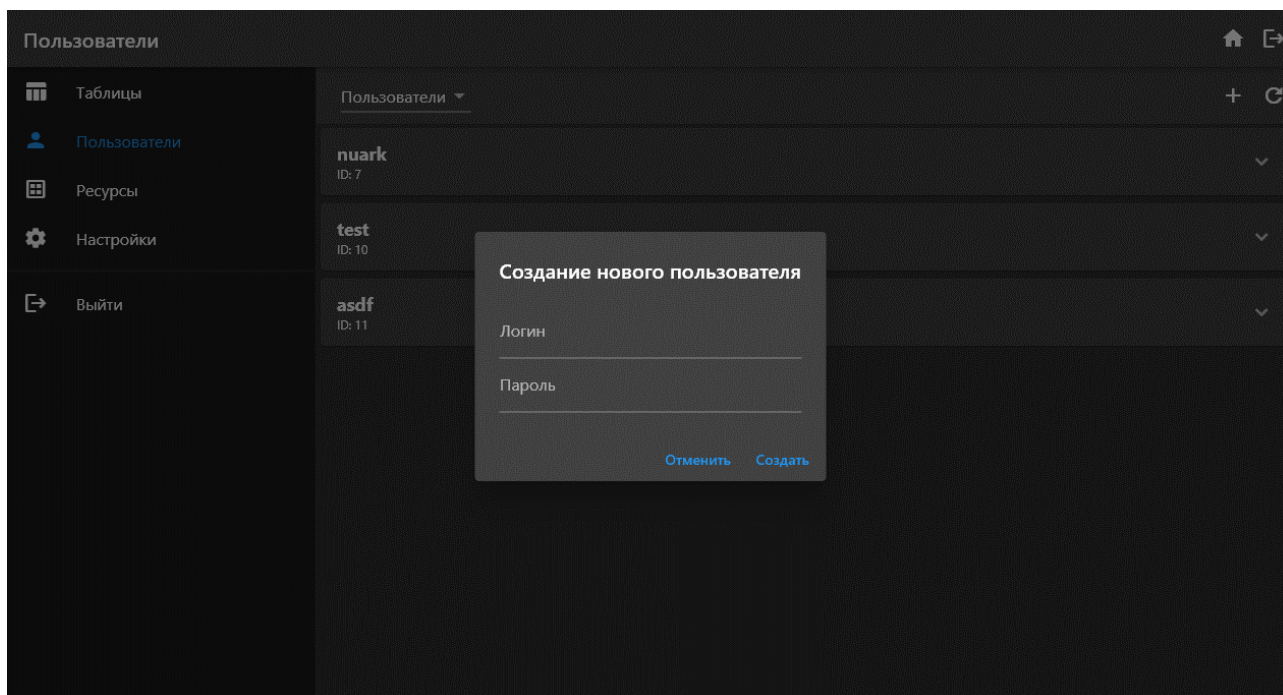


Рисунок 17 – Диалог создания нового пользователя

Название прецедента: удаление пользователя.

Цель сценария: удаление аккаунта пользователя.

Предусловия: открыта панель пользователей системы, выбран модуль «Пользователи», пользователь не пытается удалить свой аккаунт, пользователь состоит в группе администраторов.

Основной сценарий:

- 1) пользователь нажимает на шеврон на карточке пользователя;
- 2) пользователь нажимает кнопку «Удалить» на развернувшейся карточке;
- 3) в открывшемся диалоге пользователь подтверждает своё намерение нажатием кнопки «Удалить».

Постусловия: диалог закрывается, данные пользователя удалены из базы данных.

Альтернативный сценарий:

- 1) пользователь нажимает на шеврон на карточке пользователя;
- 2) пользователь нажимает кнопку «Удалить» на развернувшейся карточке;
- 3) в открывшемся диалоге пользователь отменяет своё намерение нажатием кнопки «Отменить».

Альтернативные постусловия: диалог закрывается, данные пользователя удалены из базы данных.

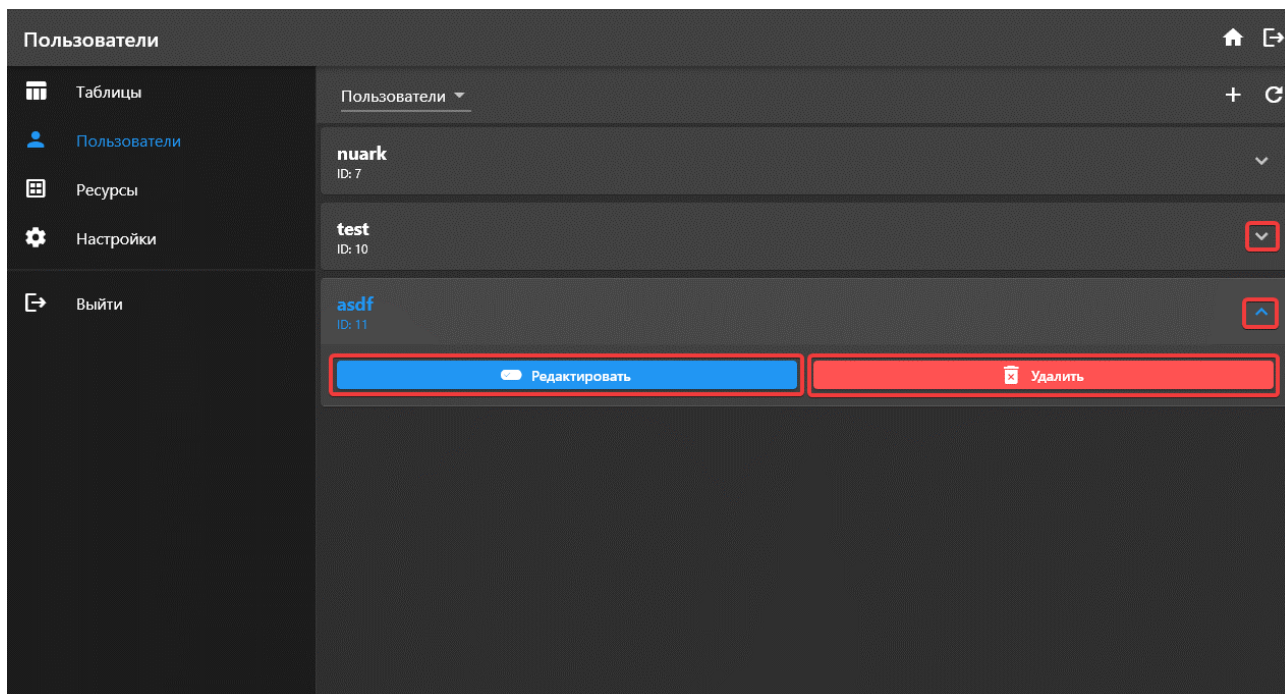


Рисунок 18 – Панель пользователей с развёрнутой карточкой пользователя

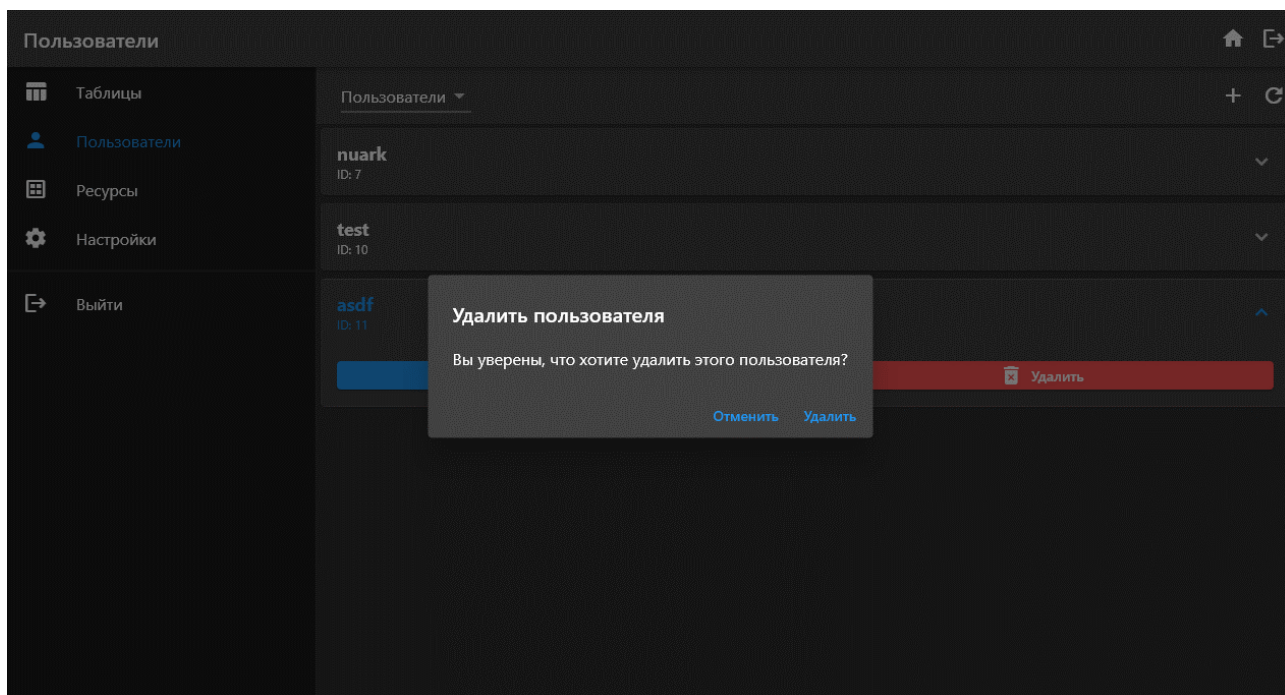


Рисунок 19 – Диалог подтверждения намерения удалить пользователя

Название прецедента: изменение пользователя.

Цель сценария: изменить пароль пользователя.

Предусловия: открыта панель пользователей системы, выбран модуль «Пользователи», пользователь состоит в группе администраторов.

Основной сценарий:

- 1) пользователь нажимает на шеврон на карточке пользователя;
- 2) пользователь нажимает кнопку «Редактировать» на развернувшейся карточке;
- 3) в открывшемся диалоговом окне (рисунок 20) пользователь вводит в поле «Пароль» новый пароль;
- 4) пользователь нажимает кнопку «Сохранить».

Постусловия: в базе данных обновлена запись пароля пользователя.

Альтернативный сценарий:

- 1) пользователь нажимает на шеврон на карточке пользователя;
- 2) пользователь нажимает кнопку «Edit user» на развернувшейся карточке;
- 3) в открывшемся диалоговом окне пользователь нажимает кнопку «Отменить».

Альтернативные постусловия: в базе данных ничего не изменяется.

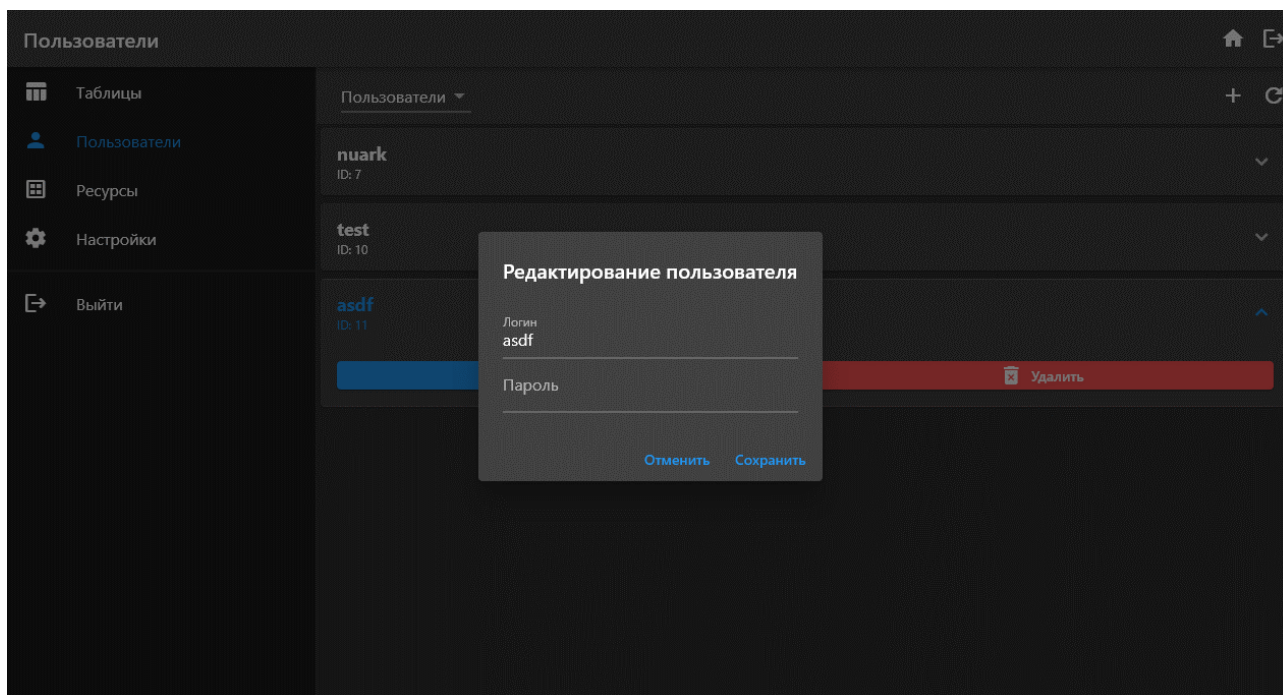


Рисунок 20 – Диалог изменения пароля пользователя

Название прецедента: добавление ресурса.

Цель сценария: добавить новый ресурс в систему.

Предусловия: открыта панель ресурсов (рисунок 21), пользователь состоит в группе с разрешением на изменение ресурсов.

Основной сценарий:

- 1) пользователь нажимает кнопку со знаком плюс;
- 2) на открывшееся диалоговое окно (рисунок 22) пользователь переносит необходимый для загрузки файл;
- 3) диалог закрывается, открывается диалог с прогрессом загрузки (рисунок 23).

Постусловия: диалог прогресса закрывается, файл добавляется в объектное хранилище системы.

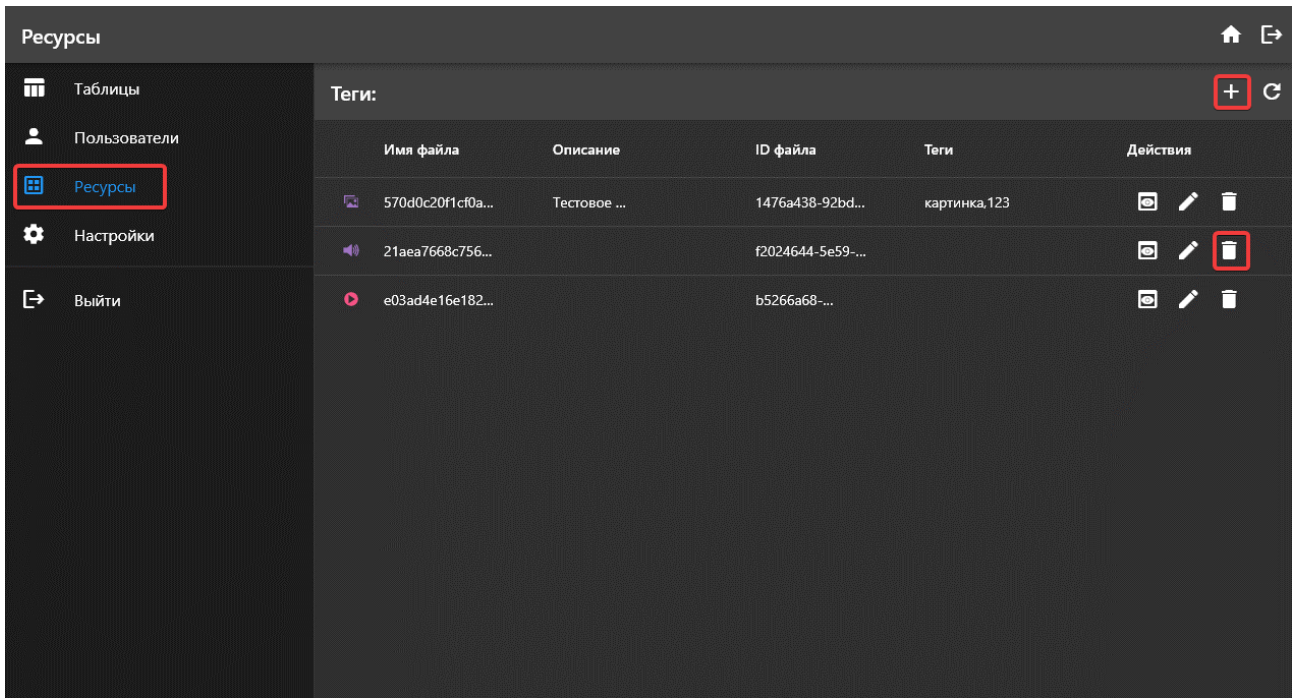


Рисунок 21 – Панель ресурсов

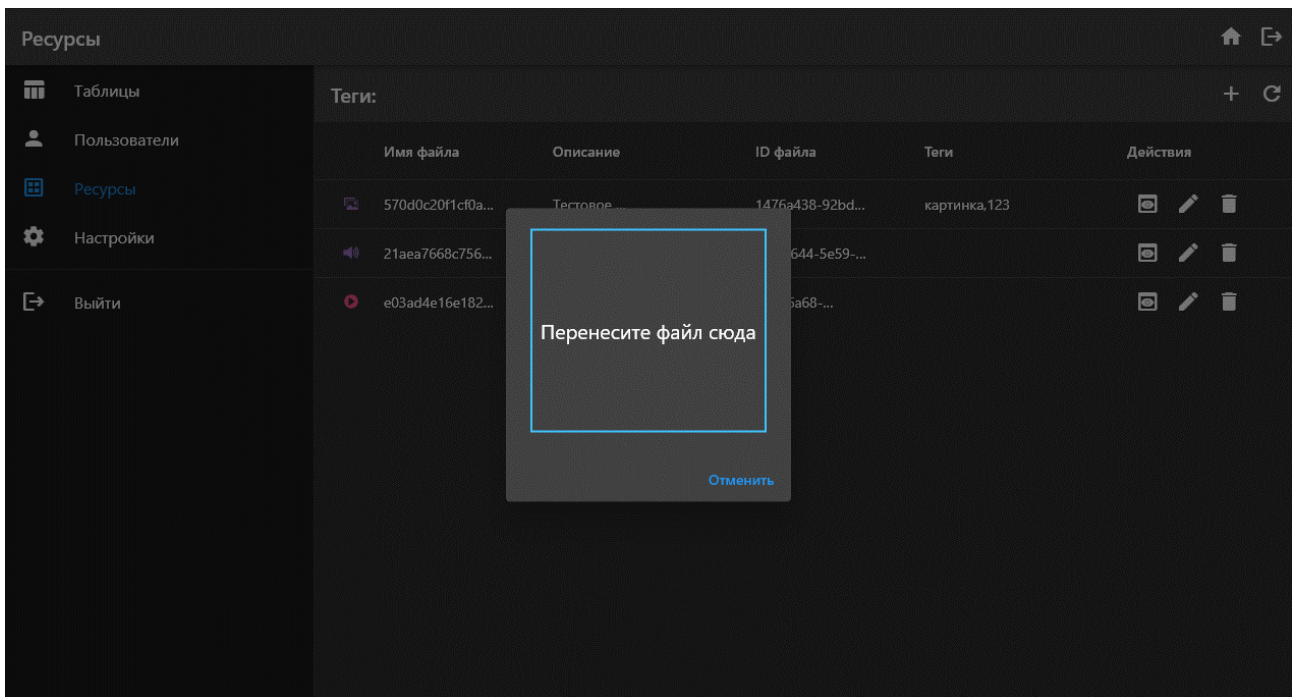


Рисунок 22 – Диалог загрузки файла

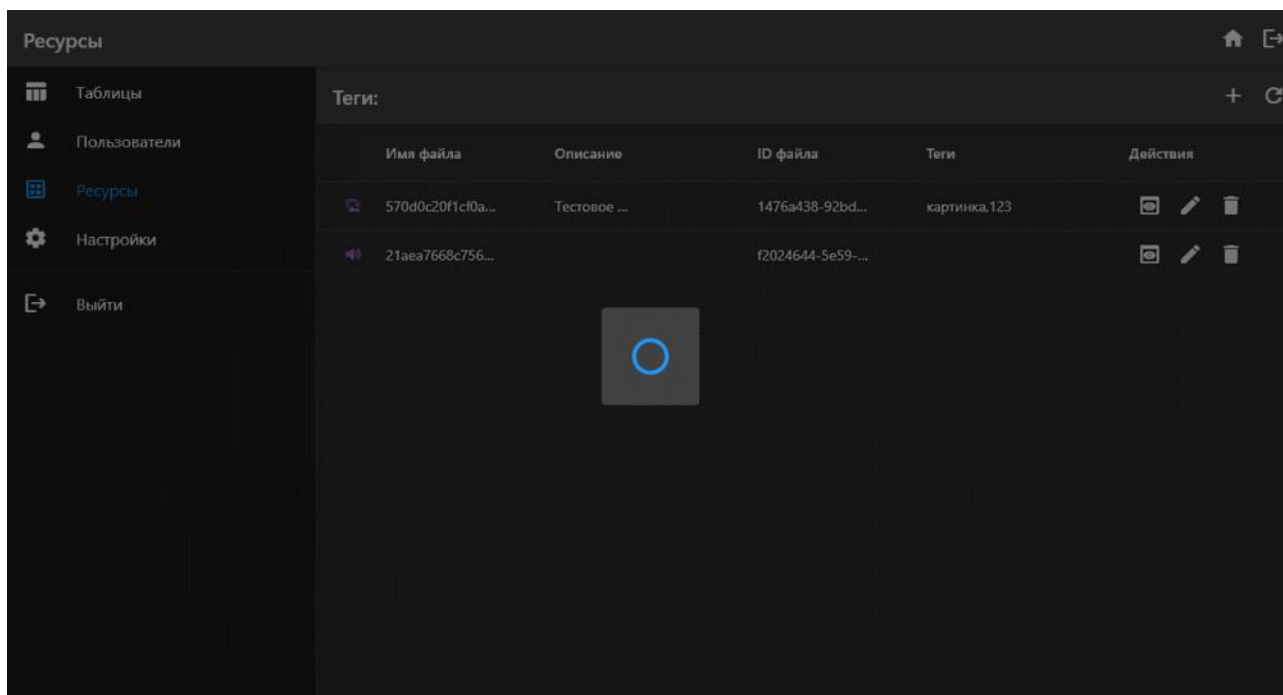


Рисунок 23 – Диалог прогресса загрузки

Название прецедента: удаление ресурса.

Цель сценария: удалить существующий ресурс.

Предусловия: открыта панель ресурсов (рисунок 21), пользователь состоит в группе с разрешением на изменение ресурсов.

Основной сценарий:

1) пользователь нажимает кнопку со значком корзины на строке таблицы необходимого ресурса;

2) в открывшемся диалоге пользователь выбирает опции удаления и подтверждает своё намерение нажатием кнопки.

Постусловия: ресурс пропадает из хранилища системы, запись о нём пропадает из базы данных.

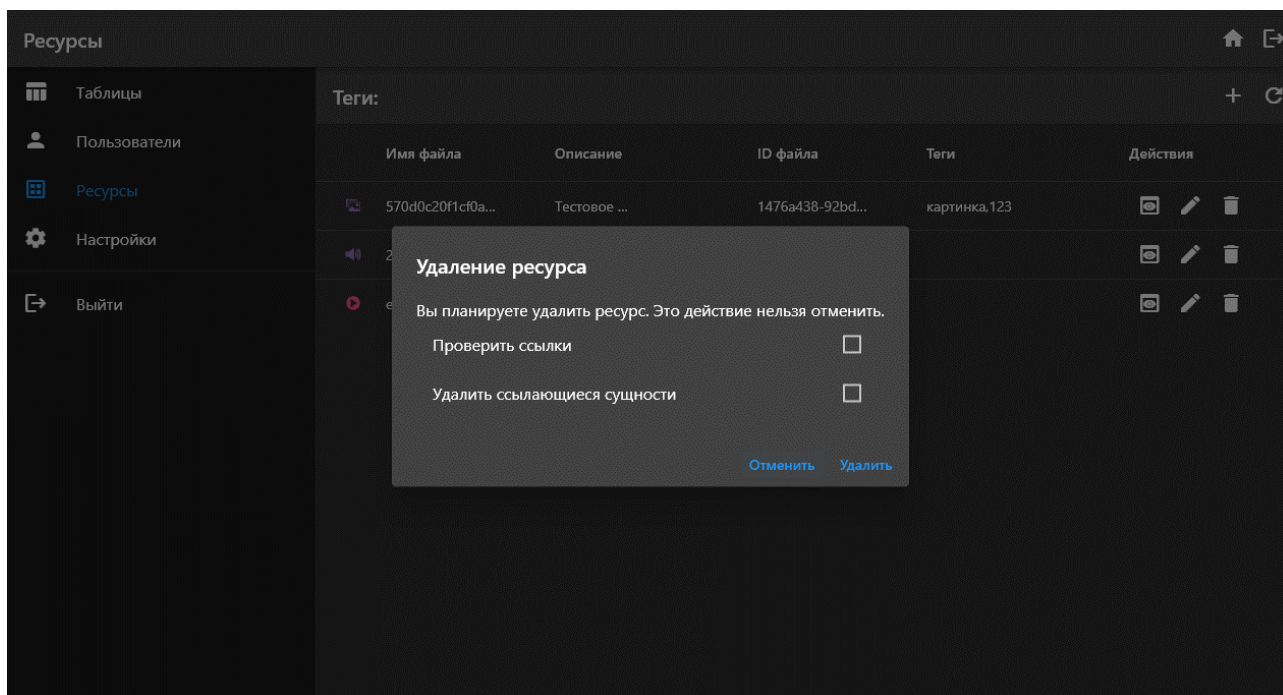


Рисунок 24 – Диалог удаления ресурса

Название прецедента: изменение ресурса.

Цель сценария: изменение информации существующего ресурса.

Предусловия: открыта панель ресурсов, пользователь состоит в группе с разрешением на изменение ресурсов.

Основной сценарий:

- 1) пользователь нажимает на кнопку со значком карандаша на строке таблицы необходимого ресурса;
- 2) в открывшемся диалоге (рисунок 25) пользователь изменяет описание ресурса или его теги;
- 3) пользователь нажимает кнопку «Подтвердить» для подтверждения изменений.

Постусловия: диалог закрывается, в базе данных изменяется информация о ресурсе.

Альтернативный сценарий:

- 1) пользователь нажимает на кнопку со значком карандаша на строке таблицы необходимого ресурса;
- 2) в открывшемся диалоге пользователь жмёт кнопку «Отменить».

Альтернативные постуловия: диалог закрывается, в базе данных изменяется информация о ресурсе.

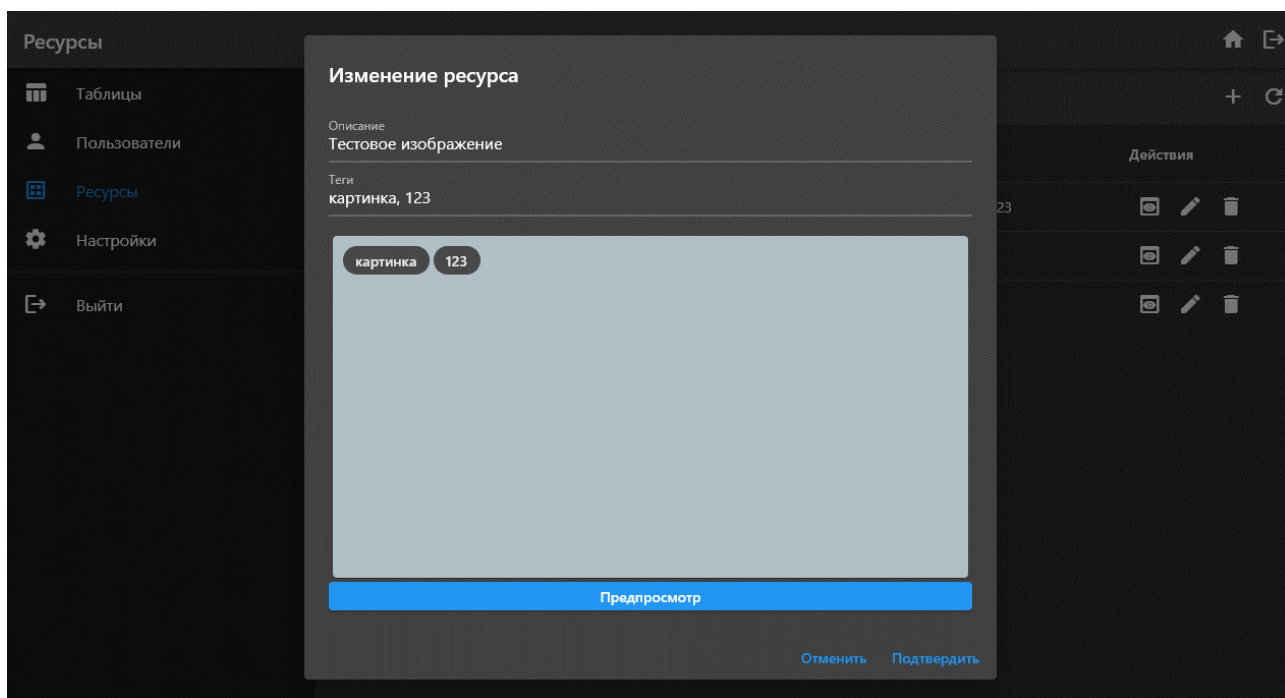


Рисунок 25 – Диалог редактирования ресурса

Название прецедента: создание новой группы пользователей.

Цель сценария: создать новую группу пользователей.

Предусловия: открыта панель пользователей, выбран модуль «Группы» (рисунок 26), пользователь состоит в группе администраторов.

Основной сценарий:

- 1) пользователь нажимает на кнопку со знаком плюс;
- 2) в открывшемся диалоге (рисунок 27) пользователь вводит название группы и, опционально, её описание;
- 3) пользователь подтверждает создание группы нажатием кнопки «Создать».

Постуловия: диалог закрывается, в базе данных появляется запись о новой группе пользователей.

Альтернативный сценарий:

- 1) пользователь нажимает на кнопку со знаком плюс;

2) пользователь отменяет своё намерение нажатием кнопки «Отменить».

Альтернативные постусловия: диалог закрывается, в базе данных не появляется новой записи.

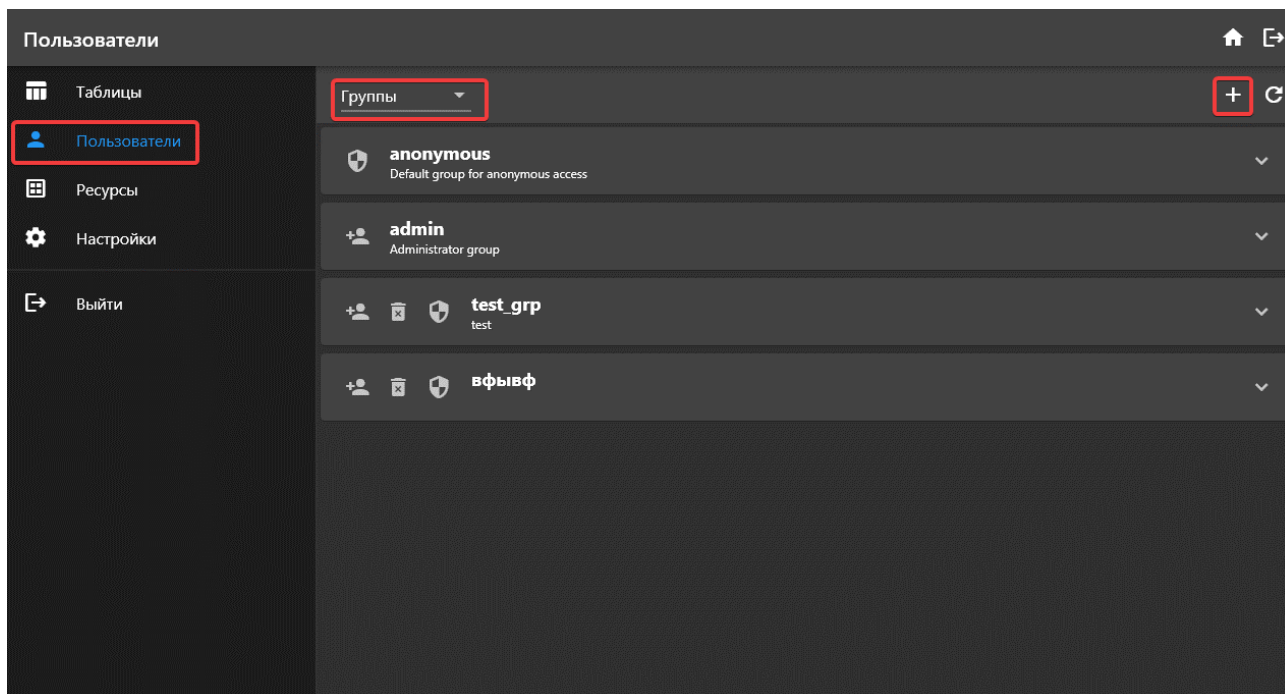


Рисунок 26 – Панель пользователей с выбранным модулем «Groups»

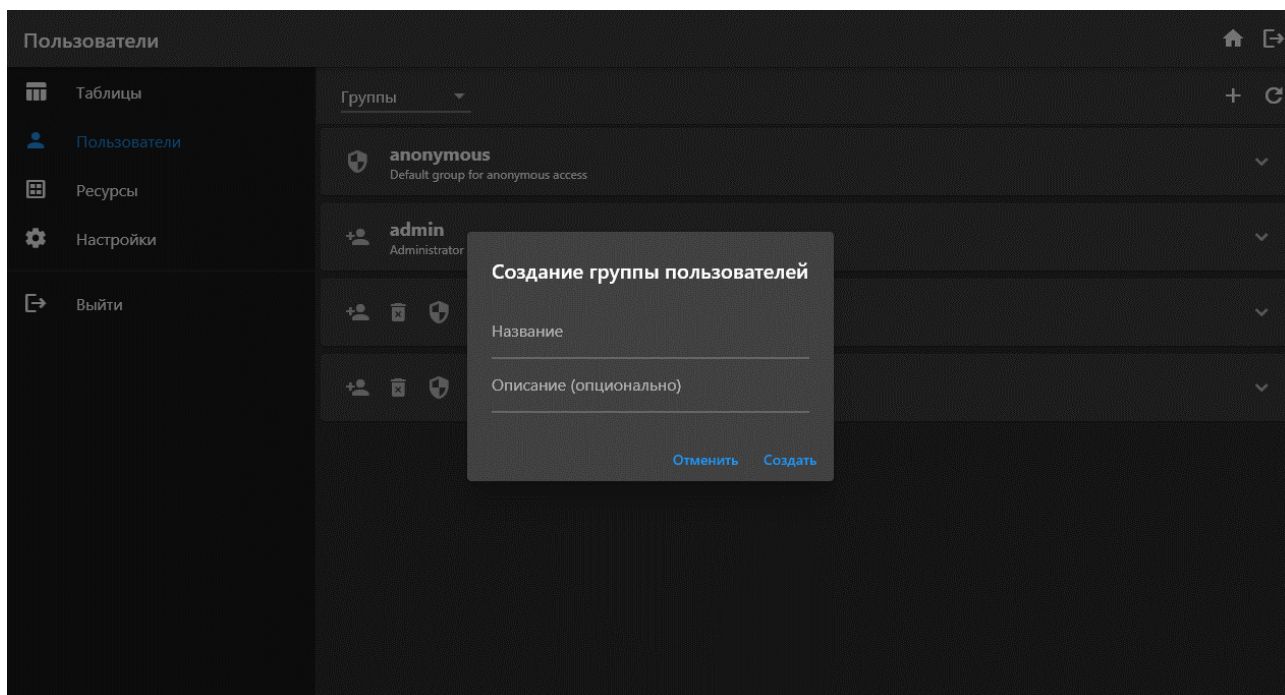


Рисунок 27 – Диалог создания новой группы

Название прецедента: удаление группы пользователей.

Цель сценария: удалить группу пользователей.

Предусловия: открыта панель пользователей, выбран модуль «Группы», пользователь не пытается удалить группу администраторов или анонимную группу, пользователь состоит в группе администраторов.

Основной сценарий:

1) пользователь нажимает на кнопку со знаком корзины на карточке необходимой к удалению группы;

2) в открывшемся диалоге пользователь подтверждает намерение нажатием кнопки «Удалить»;

3) пользователь подтверждает создание группы нажатием кнопки «Create».

Постусловия: диалог закрывается, из базы данных удаляется запись о группе пользователей.

Альтернативный сценарий:

1) пользователь нажимает на кнопку со знаком корзины на карточке необходимой к удалению группы;

2) пользователь отменяет своё намерение нажатием кнопки «Отменить».

Альтернативные постусловия: диалог закрывается, из базы данных ничего не удаляется.

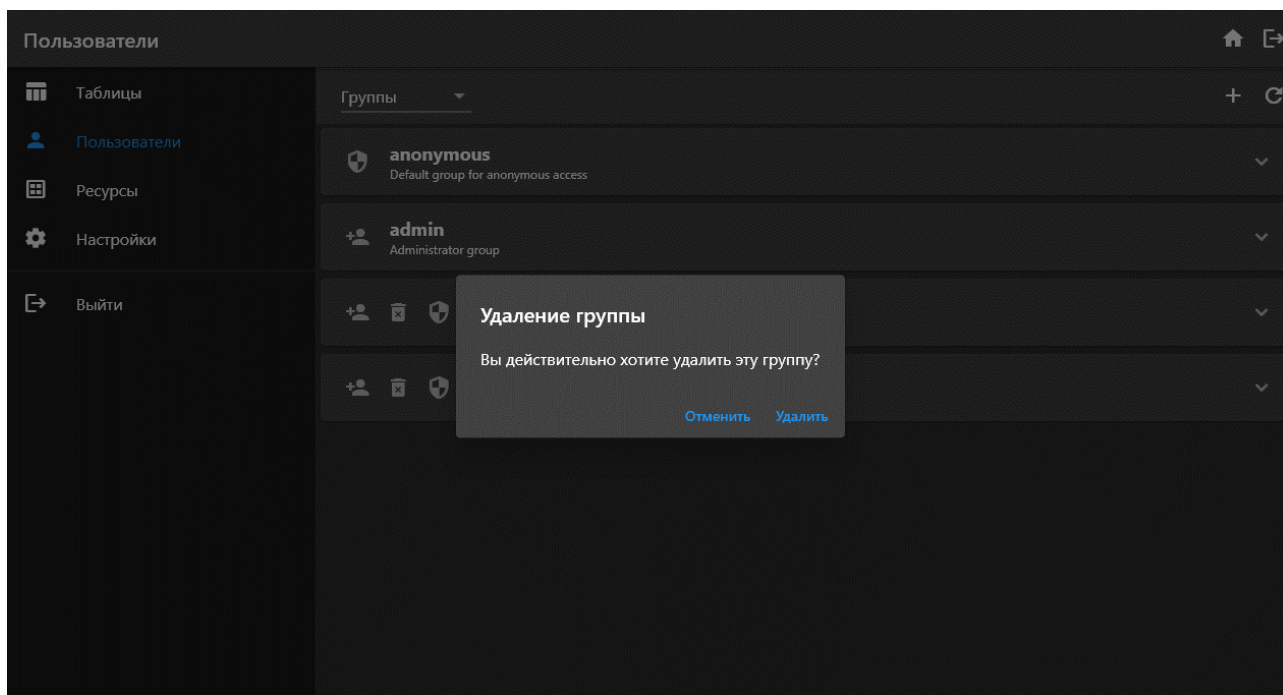


Рисунок 28 – Диалог подтверждения удаления группы

Название прецедента: изменение списка пользователей группы.

Цель сценария: изменить список пользователей группы.

Предусловия: открыта панель пользователей, выбран модуль «Группы», пользователь не пытается изменить анонимную группу, пользователь состоит в группе администраторов.

Основной сценарий:

- 1) пользователь нажимает кнопку со знаком «плюс человек»;
- 2) в открывшемся диалоге (рисунок 29) пользователь выбирает нужного пользователя из списка.

Постусловия: диалог закрывается, в базе данных добавляется информация о нахождении пользователя в группе.

Альтернативный сценарий:

- 1) пользователь нажимает кнопку со знаком «плюс человек»;
- 2) пользователь отменяет своё намерение нажатием кнопки «Отменить».

Альтернативные постусловия: диалог закрывается, в базе данных не появляется новой информации.

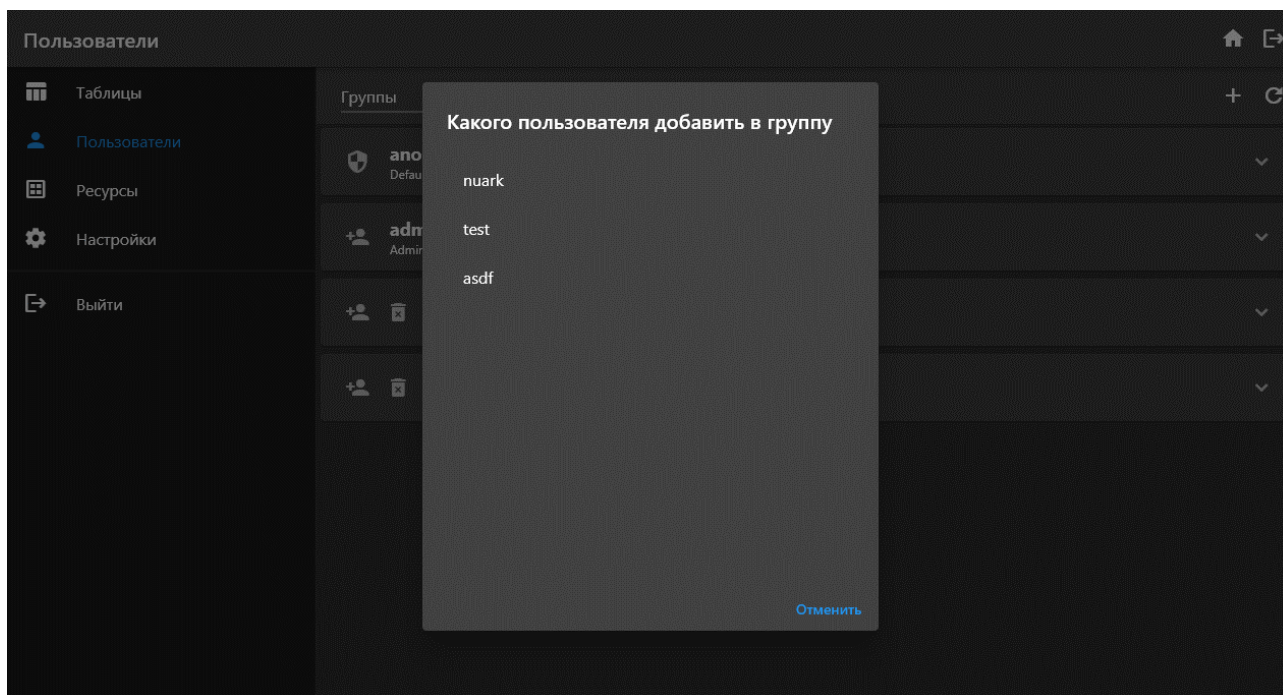


Рисунок 29 – Диалог выбора нового пользователя группы

Название прецедента: удаление пользователя из списка группы.

Цель сценария: удалить пользователя из списка группы.

Предусловия: открыта панель пользователей, выбран модуль «Группы», пользователь не пытается редактировать анонимную группу, пользователь не пытается удалить самого себя из группы администраторов, пользователь состоит в группе администраторов.

Основной сценарий:

- 1) пользователь нажимает на шеврон карточки необходимой группы;
- 2) в развернувшейся части карточки пользователь нажимает знак корзины напротив необходимого к удалению пользователя.

Постусловия: диалог закрывается, из базы данных удаляется запись о нахождении пользователя в группе пользователей.

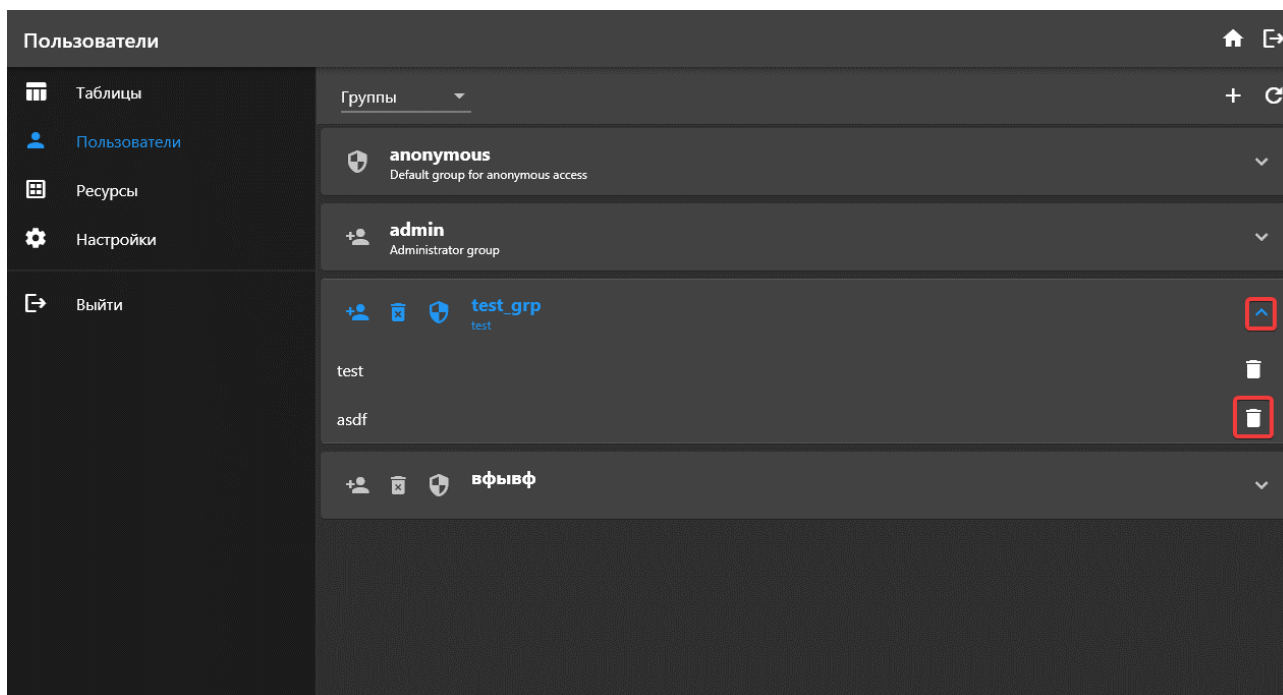


Рисунок 30 – Развёрнутая карточка группы

Название прецедента: редактирование разрешения группы на таблицы.

Цель сценария: отредактировать разрешения группы пользователей на таблицы системы.

Предусловия: открыта панель пользователей, выбран модуль «Группы», пользователь не пытается редактировать группу администраторов, пользователь состоит в группе администраторов.

Основной сценарий:

- 1) пользователь нажимает на кнопку со знаком щита на карточке группы;
- 2) в открывшемся диалоге (рисунок 31) пользователь выбирает необходимые разрешения для таблиц.

Постусловия: в базе данных обновляется информация о доступе группы к таблицам.

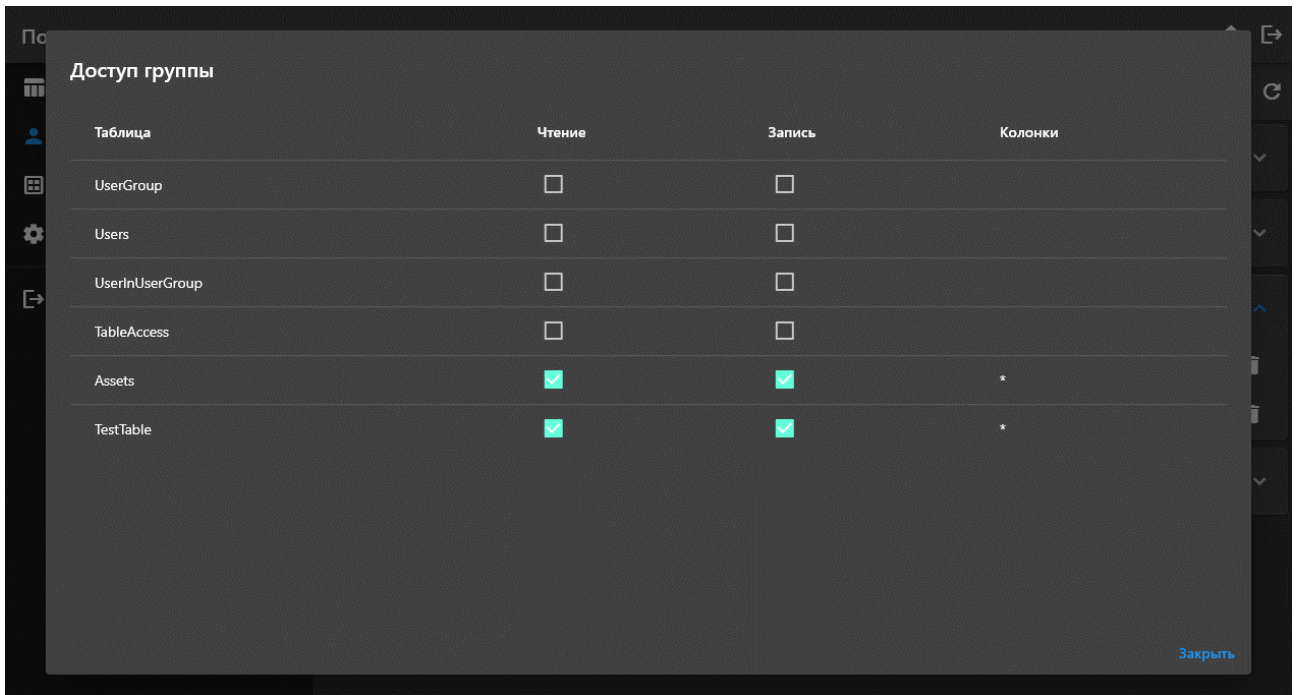


Рисунок 31 – Диалог редактирования прав группы на таблицы

Название прецедента: изменение адреса бэкенда.

Цель сценария: удалить изменить адрес бэкенда, к которому подключается клиентское приложение.

Предусловия: открыта панель настроек (рисунок 32).

Основной сценарий:

- 1) пользователь нажимает на кнопку «Выйти и изменить»;
- 2) происходит выход из аккаунта пользователя.

Постусловия: происходит переход на страницу входа, где пользователь может ввести новый адрес бэкенда.

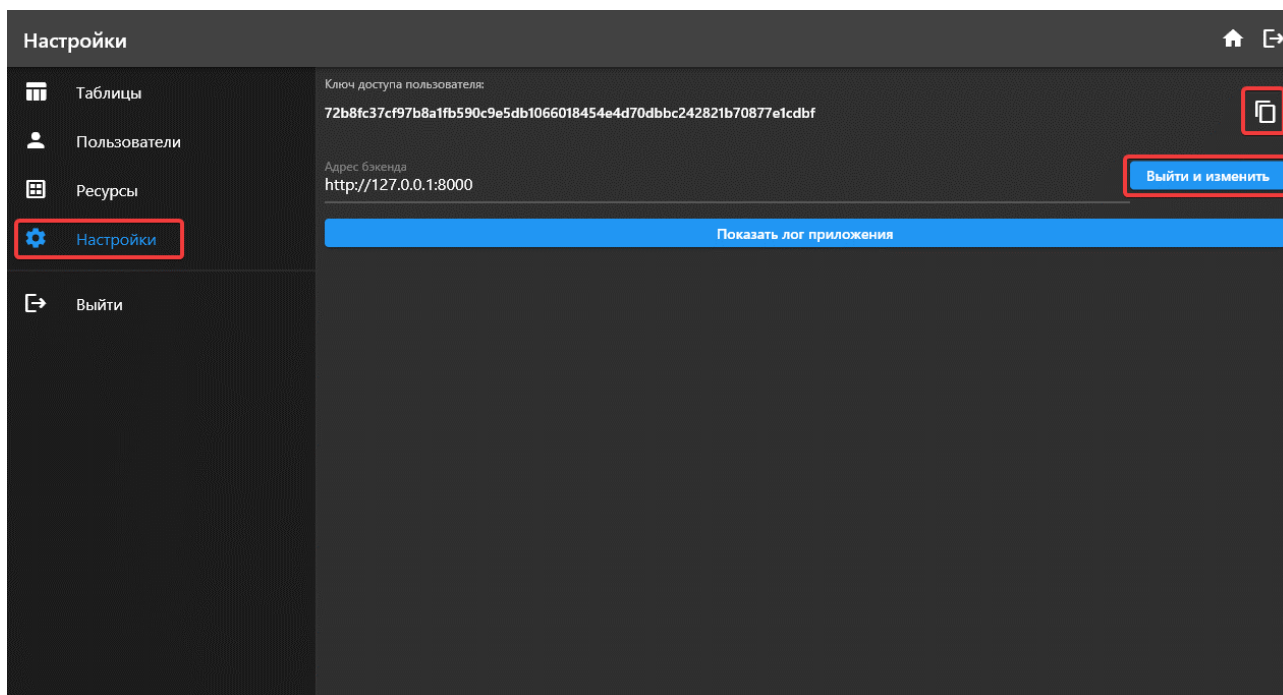


Рисунок 32 – Панель настроек

Название прецедента: скопировать ключ доступа текущего пользователя.

Цель сценария: скопировать ключ доступа текущего пользователя.

Предусловия: открыта панель настроек.

Основной сценарий:

1) пользователь нажимает на кнопку со знаком копирования напротив ключа доступа пользователя (рисунок 32).

Постусловия: в буфере обмена пользователя появляется ключ доступа пользователя.

Название прецедента: просмотреть лог приложения.

Цель сценария: просмотреть лог приложения для выявления возможных проблем.

Предусловия: открыта панель настроек.

Основной сценарий:

1) пользователь нажимает на кнопку «Показать лог приложения».

Постусловия: открывается диалог с логом приложения (рисунок 33).

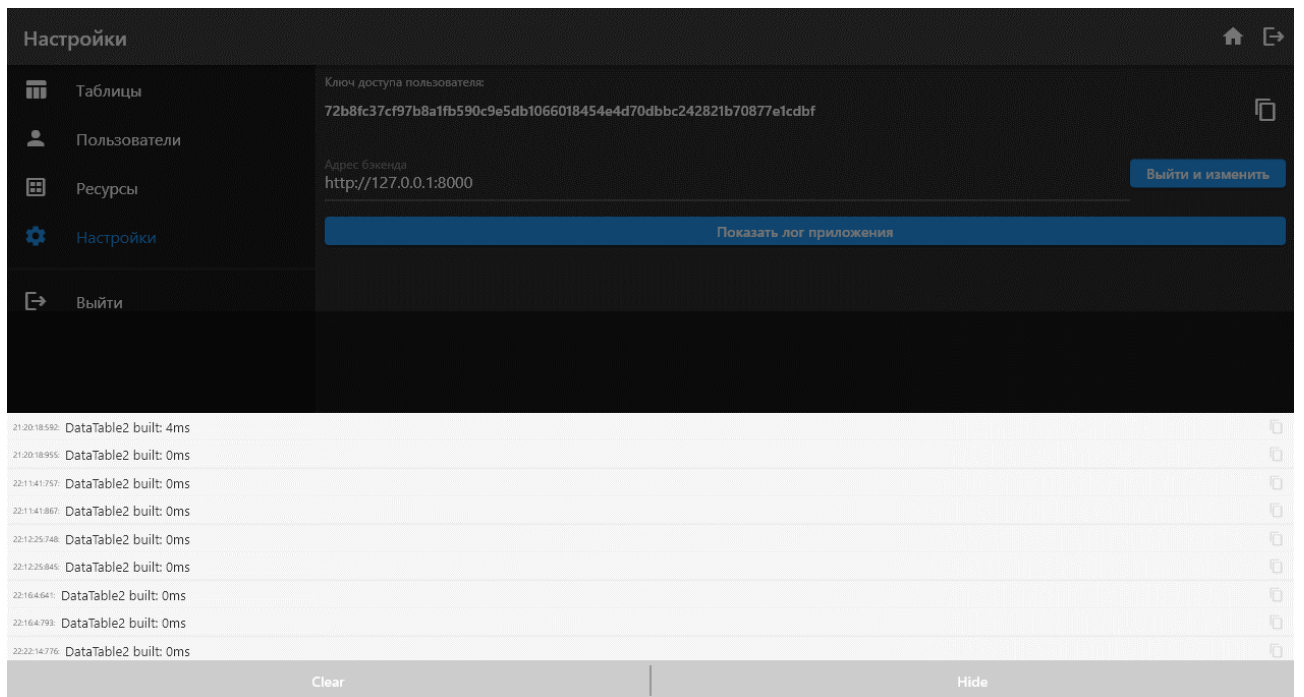


Рисунок 33 – Диалог лога приложения

1.4 Модель предметной области

В ходе формирования требований к системе была построена модель предметной области, изображённая в виде диаграммы классов UML на рисунке 7.

Модель предметной области

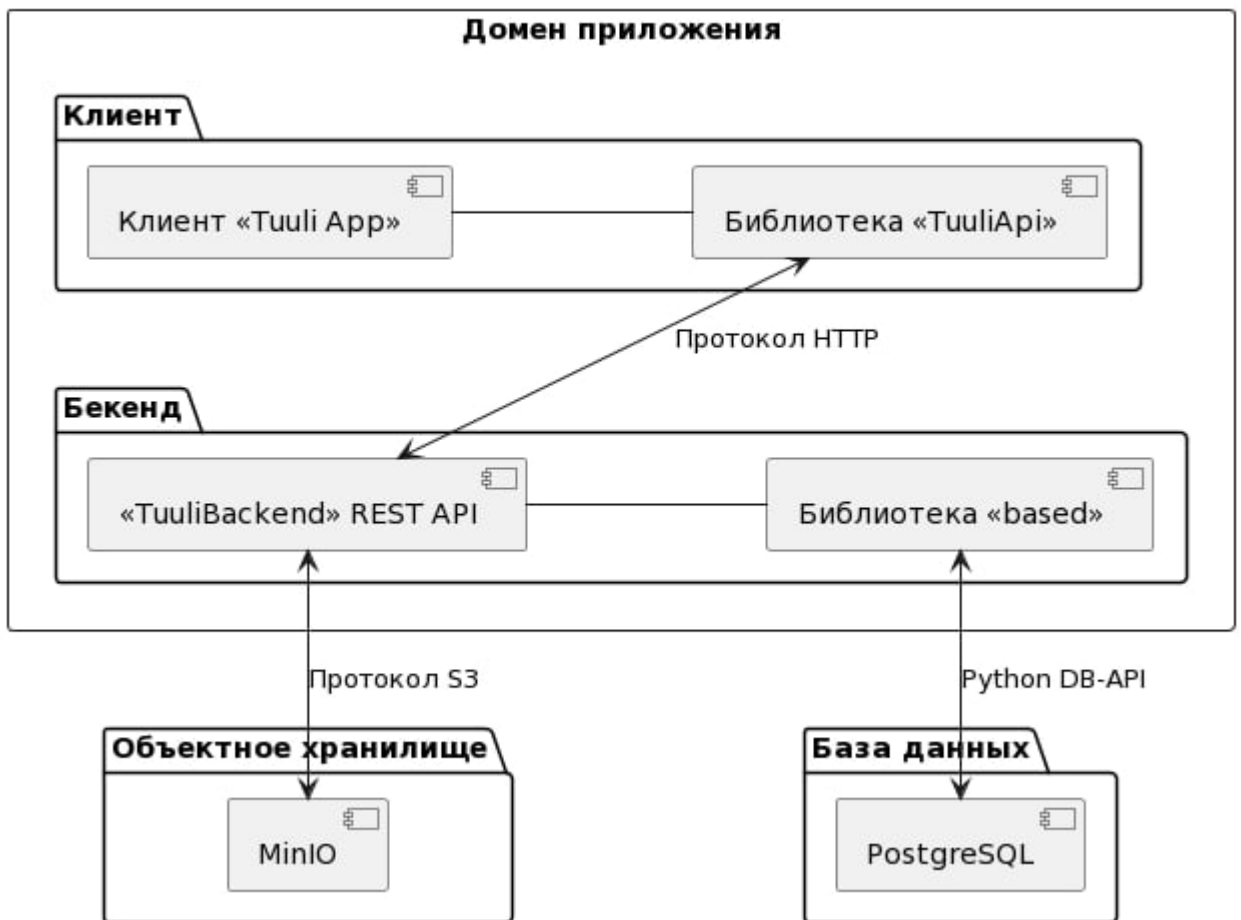


Рисунок 34 – UML-диаграмма предметной области

1.5 Выводы по главе

1. Проведён анализ открытых реализаций систем подобного типа с целью возможного заимствования удачных решений;
2. Сформулированы функциональные и нефункциональные требования к системе.

2 Проектирование

В качестве основного инструмента проектирования был выбран процесс проектирования ICONIX, как один из более распространённых фреймворков проектирования ПО. Данный процесс представляет систему как набор прецедентов, что отлично вписывается в архитектуру моей системы – в основе своей она реализует шаблон CRUD (Create – Создать, Read – Прочитать, Update – Изменить, Delete – Удалить). Мною были выделены ключевые прецеденты (не являющиеся тривиальными) и построены соответствующие диаграммы пригодности и последовательности, из которых затем синтезированы начальные варианты классов, представленных в диаграммах классов.

2.1 Диаграммы пригодности и последовательности

Диаграммы пригодности позволяют выявить ключевые моменты прецедентов, а диаграммы последовательности дают возможность глубже рассмотреть взаимодействие подсистем. Поэтому проектирование системы начинается именно с них.

2.1.1 Прецедент «Вход в систему»

Пользователь вводит адрес бэкенда, свои данные для входа (логин и пароль), система проверяет их на корректность, затем на соответствие с базой данных.

В случае успешного входа пользователь перенаправляется на главную страницу клиента системы.

В случае неуспешной попытки входа пользователь будет уведомлён о соответствующих проблемах: неверная связка логин/пароль, неправильный адрес бэкенда, проблемы с сетью и другие.

На рисунке 35 представлена диаграмма пригодности, на рисунке 36 – диаграмма последовательности.

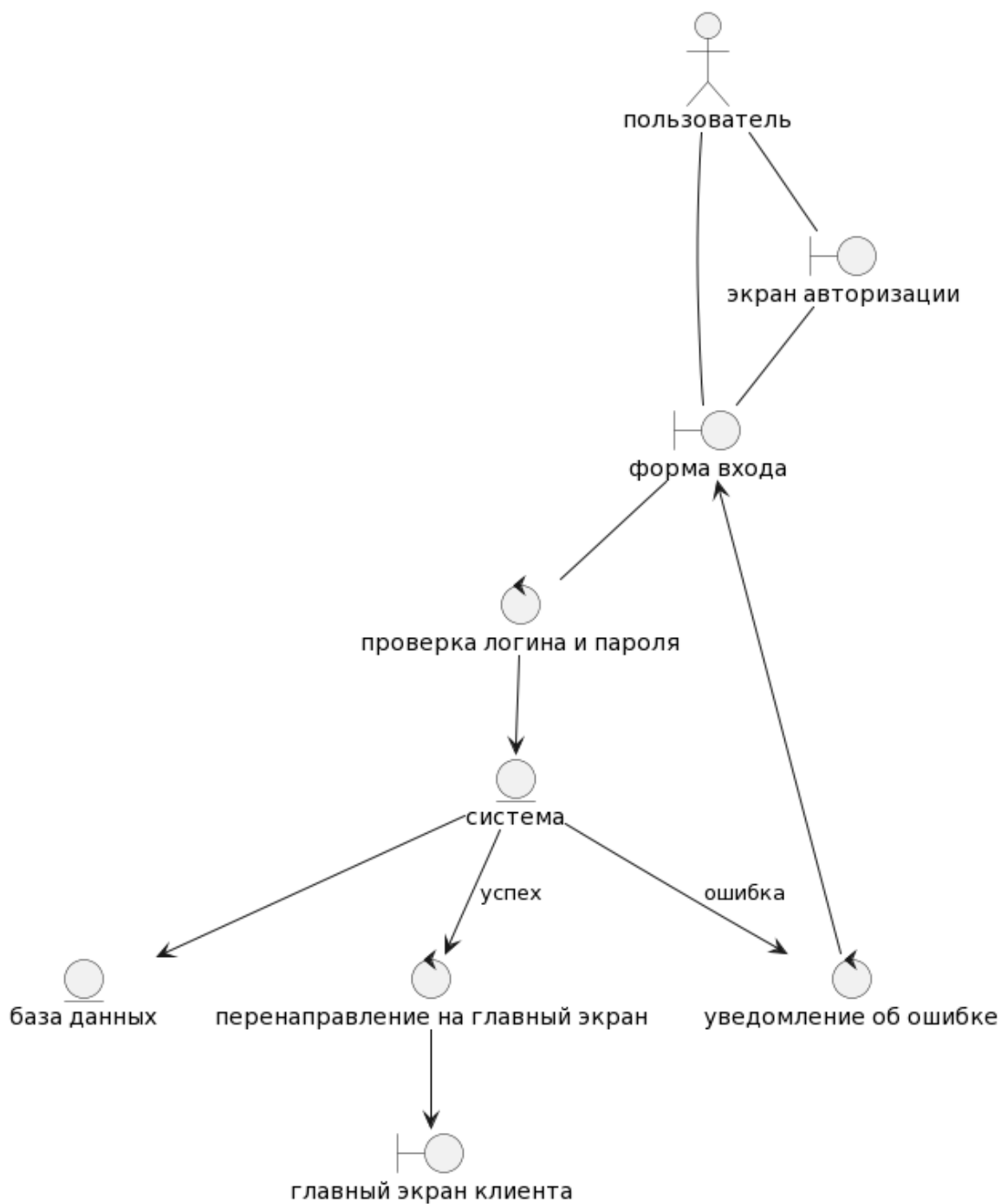


Рисунок 35 – Диаграмма пригодности прецедента «Вход в систему»

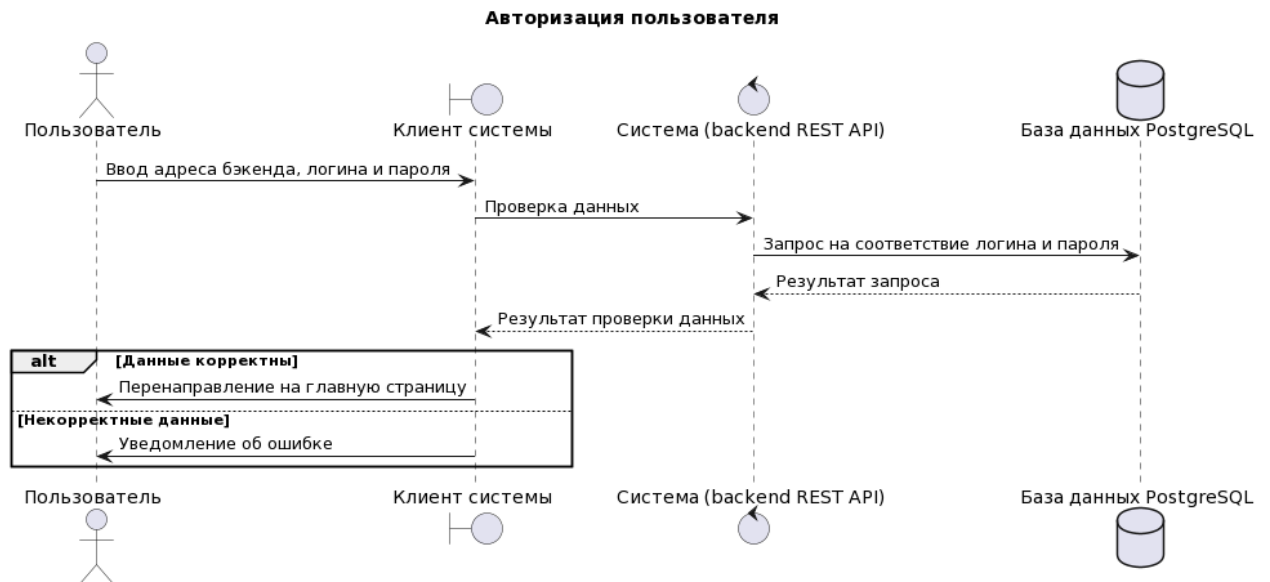


Рисунок 36 – Диаграмма последовательности прецедента «Вход в систему»

2.1.2 Прецедент «Создать новую таблицу»

На панели таблиц пользователь нажимает на кнопку «плюс», ему открывается диалог создания новой таблицы. В этом диалоге пользователь вводит название таблицы и с помощью кнопки «Создать колонку» и появляющегося затем диалога выбирает тип новой колонки. Когда пользователь удовлетворён схемой таблицы он может сохранить её или, если он решил не создавать таблицу, может закрыть диалог. В первом варианте новая таблица и метаданные будут созданы в базе данных.

Если создание прошло успешно или возникла ошибка – пользователь получит уведомление.

На рисунке 37 представлена диаграмма пригодности, на рисунке 38 – диаграмма последовательности.

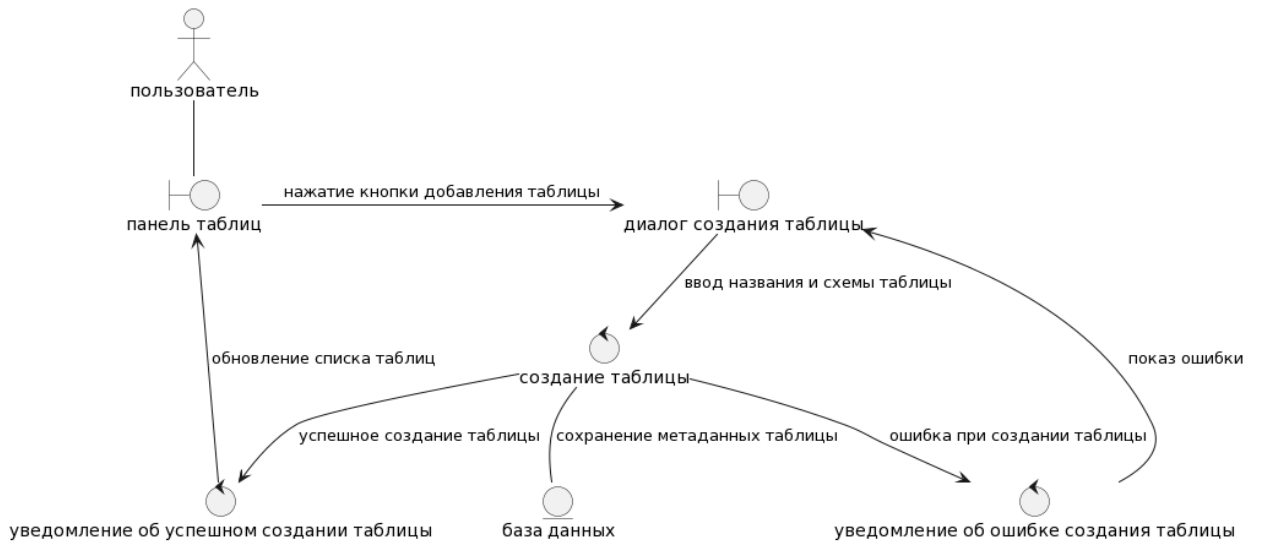


Рисунок 37 – Диаграмма пригодности прецедента «Создать новую таблицу»

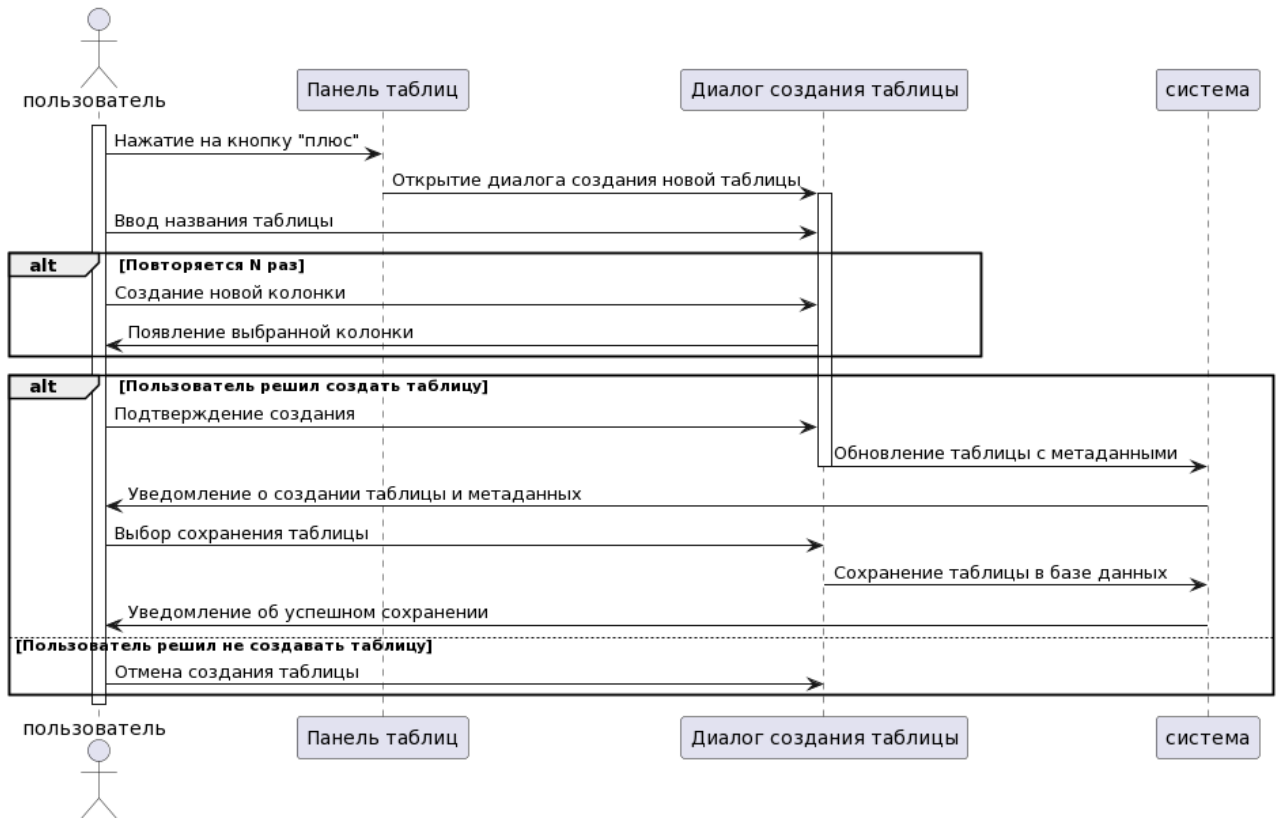


Рисунок 38 – Диаграмма последовательности прецедента «Создать новую таблицу»

2.1.3 Прецедент «Создание пользователя»

На панели пользователей пользователь выбирает модуль «Пользователи», затем нажимает на кнопку «плюс». Это действие открывает диалог создания нового пользователя, в котором необходимо ввести логин и пароль нового пользователя. Далее пользователь может нажать кнопку «Создать» для подтверждения своего действия или «Отменить» – для отмены действия.

При успешном создании пользователя выводится уведомление, как и при возникновении ошибки.

На рисунке 39 представлена диаграмма пригодности, на рисунке 40 – диаграмма последовательности.

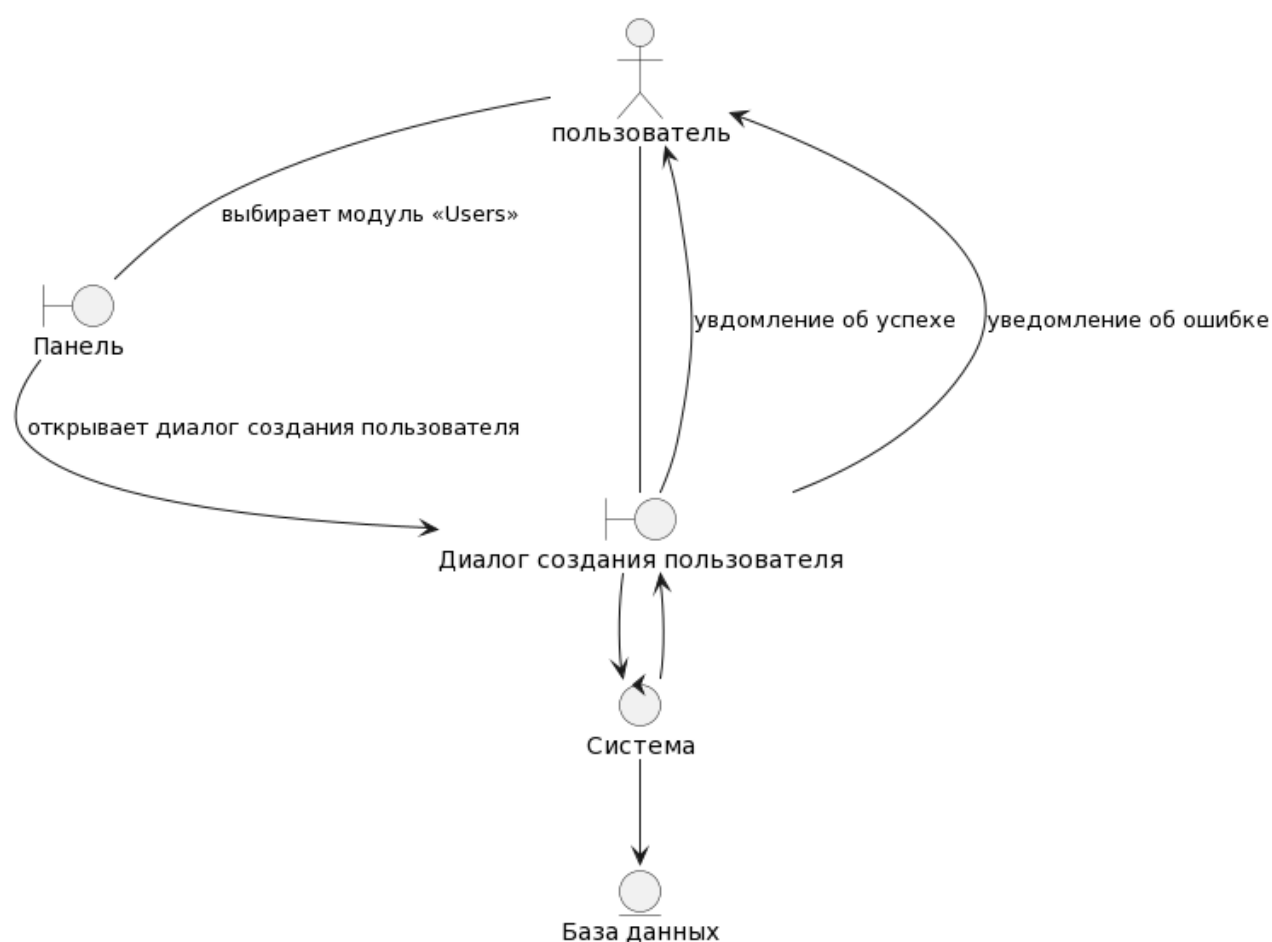


Рисунок 39 – Диаграмма пригодности прецедента «Создание пользователя»

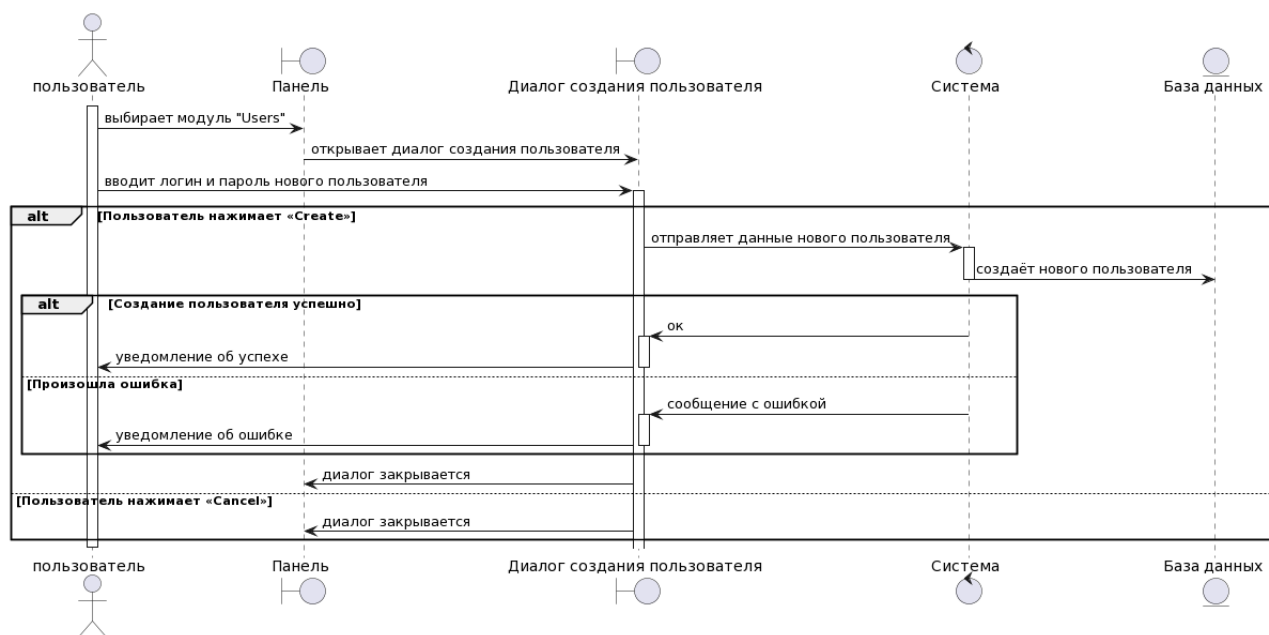


Рисунок 40 – Диаграмма последовательности прецедента «Создание пользователя»

2.1.4 Прецедент «Создание группы пользователей»

На панели пользователей пользователь выбирает модуль «Groups», затем нажимает на кнопку «плюс». Открывается диалог создания группы, в котором пользователь вводит название новой группы и, опционально, её описание. Для завершения создания пользователь нажимает кнопку «Создать» или «Отменить» – для отмены создания.

При успешном создании группы пользователей выводится уведомление, как и если в ходе создания возникнет ошибка.

На рисунке 41 представлена диаграмма пригодности, на рисунке 42 – диаграмма последовательности.

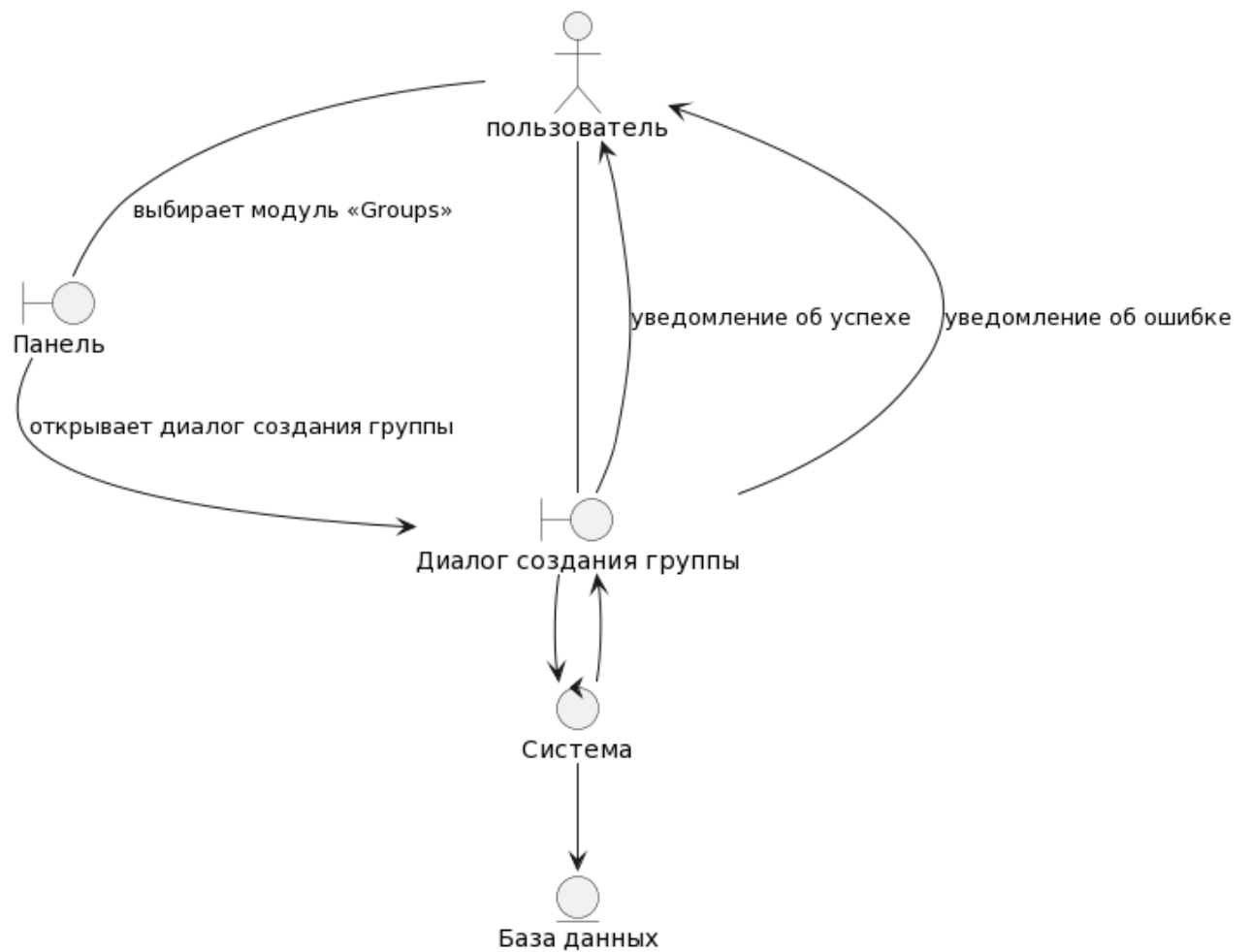


Рисунок 41 – Диаграмма пригодности прецедента «Создание группы пользователей»

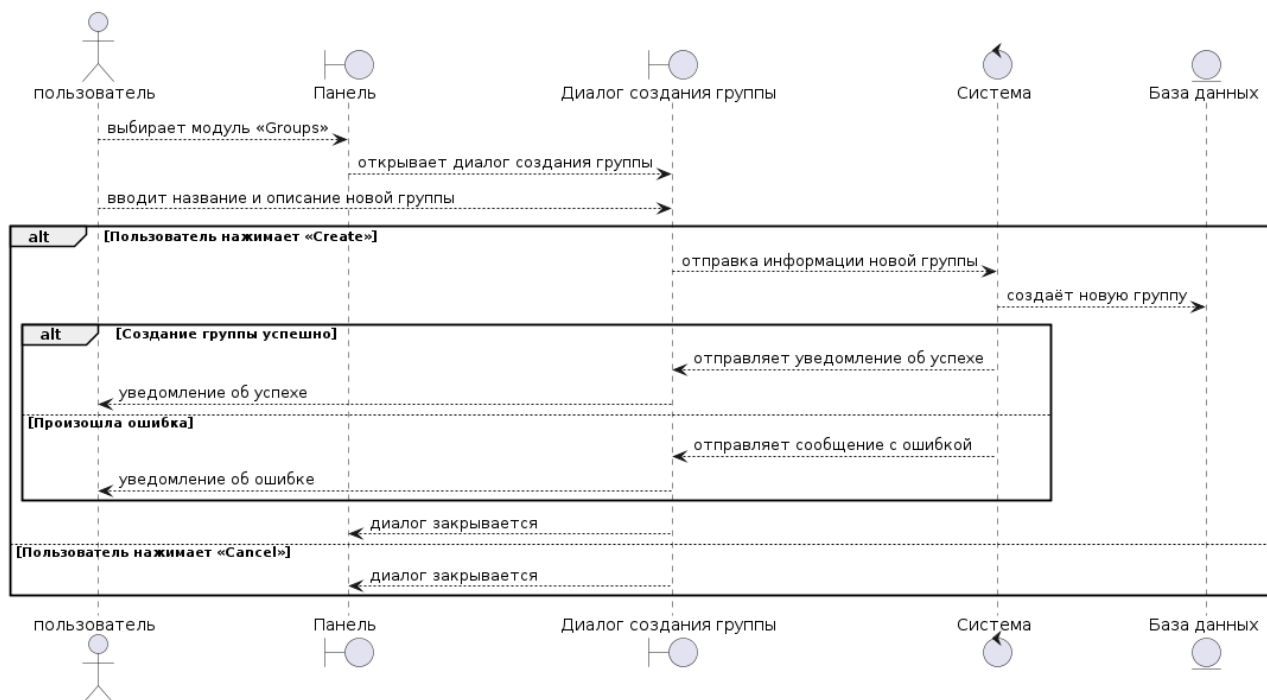


Рисунок 42 – Диаграмма последовательности прецедента «Создание группы пользователей»

2.1.5 Прецедент «Редактирование разрешений группы на таблицы»

На панели пользователей пользователь выбирает модуль «Группы», затем нажимает на кнопку со значком «щит» на карточке необходимой к изменению группы пользователей. В открывшемся диалоговом окне представлена таблица групп, на каждой строке которой находится название таблицы, состояние разрешения записи, состояние разрешения чтения и разрешённые группе колонки. Все изменения сохраняются в автоматическом режиме.

На рисунке 43 представлена диаграмма пригодности, на рисунке 44 – диаграмма последовательности.

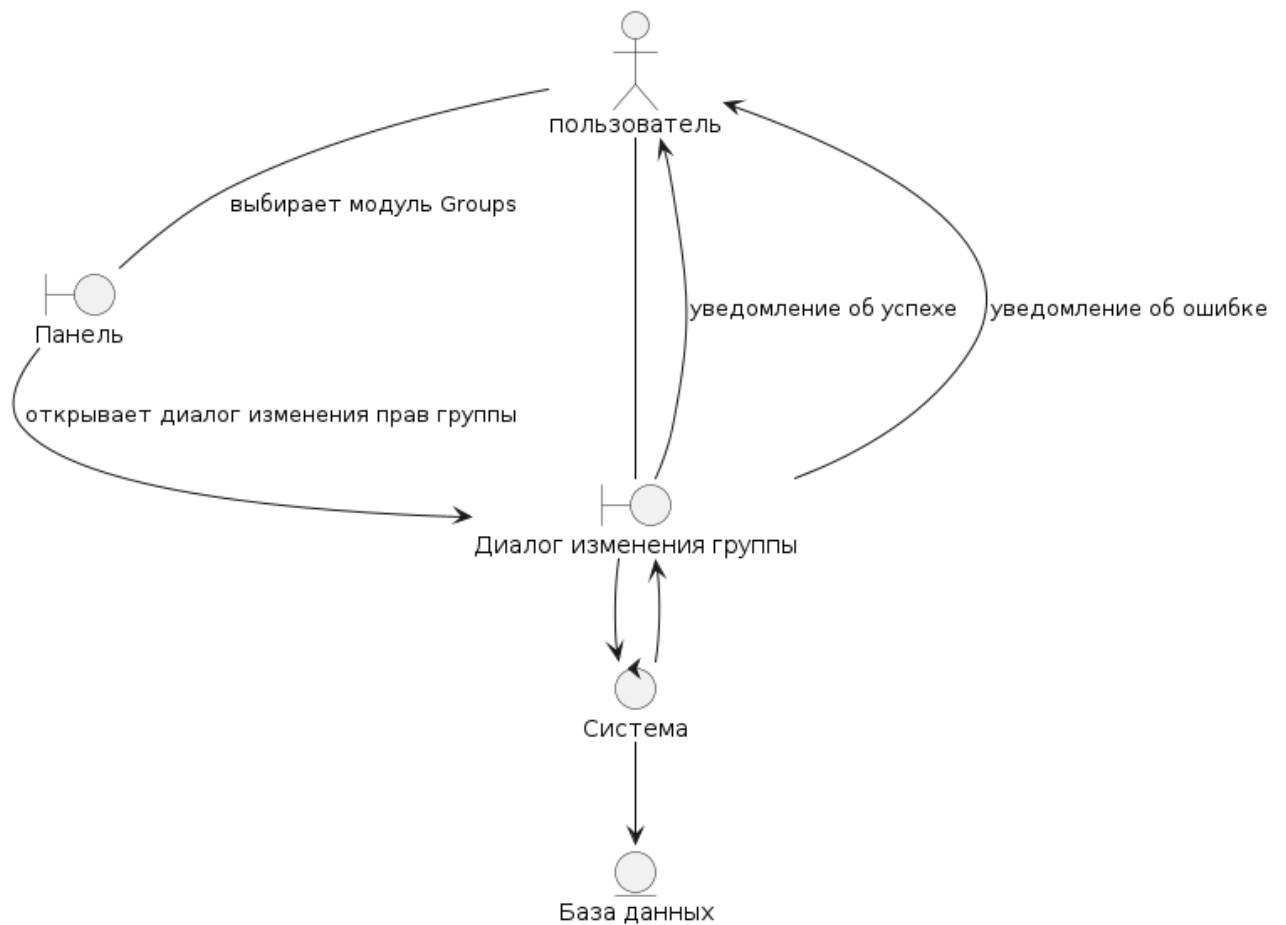


Рисунок 43 – Диаграмма пригодности прецедента «Редактирование разрешений группы на таблицы»

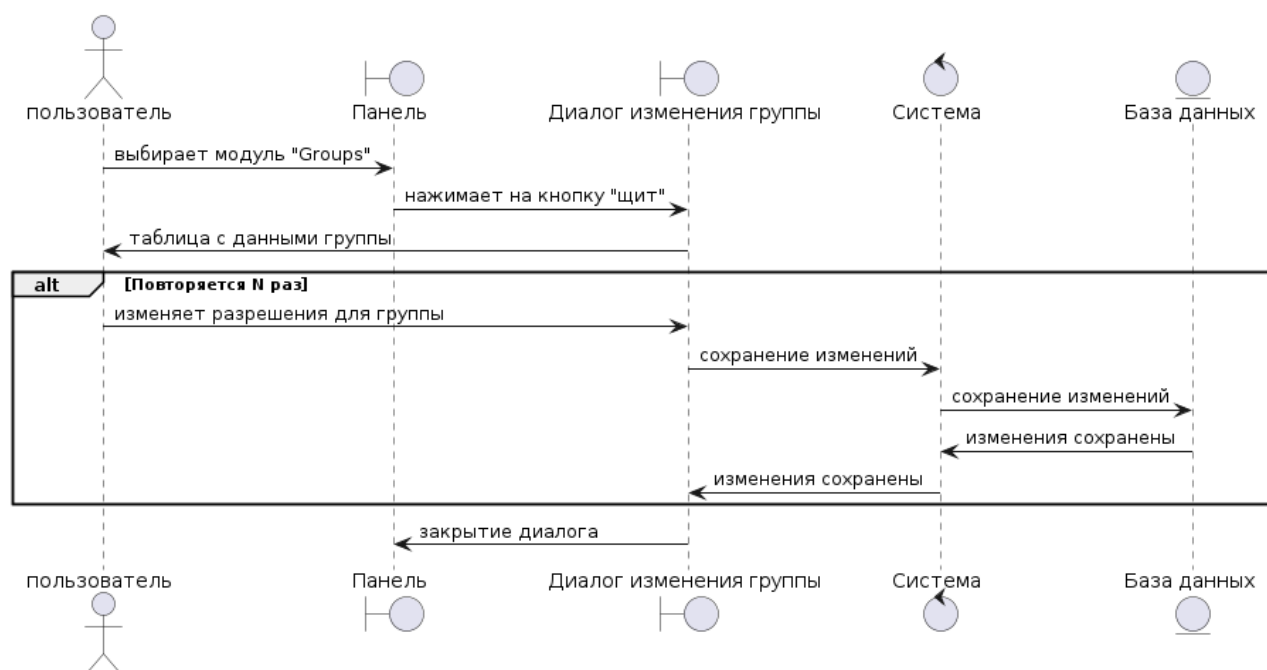


Рисунок 44 – Диаграмма последовательности прецедента «Редактирование разрешений группы на таблицы»

2.1.6 Прецедент «Редактирование существующего ресурса»

На панели ресурсов пользователь нажимает кнопку с изображением карандаша на строке того ресурса, который он хочет отредактировать. В результате нажатия открывается диалог, в котором пользователь может изменить описание ресурса, его теги. Для сохранения изменений пользователь должен нажать кнопку «Подтвердить», для отмены изменений – на кнопку «Отменить».

На рисунке 45 представлена диаграмма пригодности, на рисунке 46 – диаграмма последовательности.



Рисунок 45 – Диаграмма пригодности прецедента «Редактирование существующего ресурса»

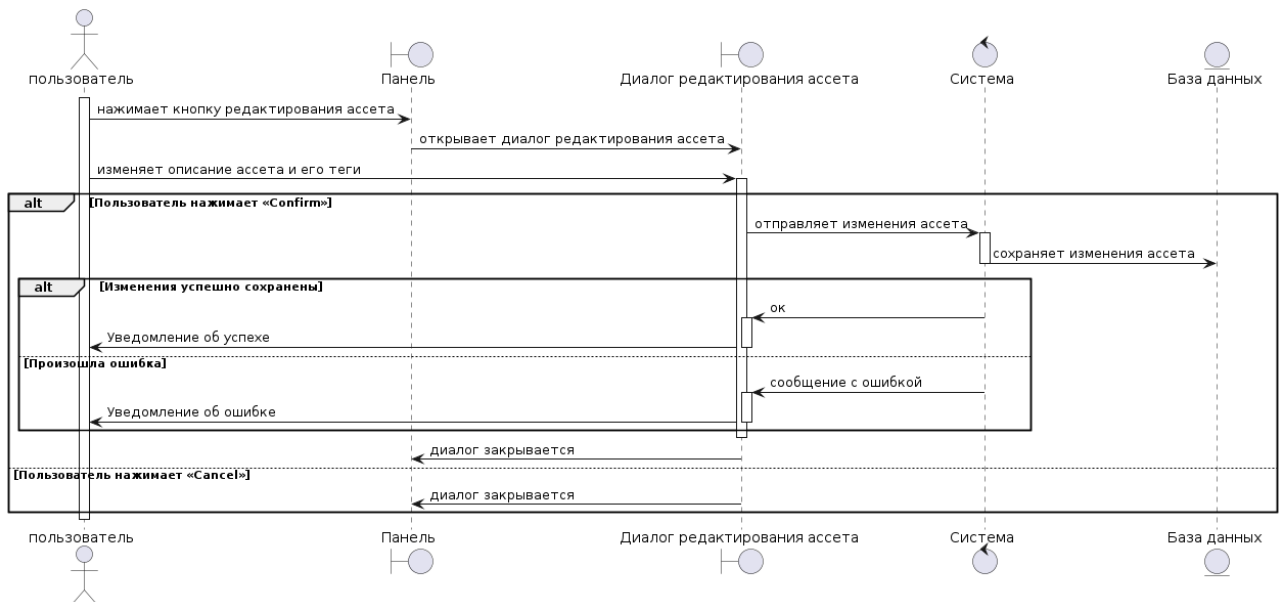


Рисунок 46 – Диаграмма последовательности прецедента «Редактирование существующего ресурса»

2.2 Выводы по главе

В соответствии со спецификацией требований были построены диаграммы пригодностей и последовательностей, в соответствии с которыми будет происходить разработка системы.

3 Реализация и тестирование

3.1 Выбор инструментов

Разработка системы разделена на три части: разработка библиотеки динамической работы с базой данных, разработка программного интерфейса приложения и разработка графического интерфейса пользователя. Все этапы разработки проводятся в VSCode – Visual Studio Code.

VSCode является одним из самых популярных инструментов для разработки, и он прекрасно подходит для написания курсового проекта на Python и Flutter. Преимущества использования VSCode для такого проекта включают в себя:

1. Он имеет множество полезных функций для поддержки программирования на Python и Flutter, включая автоматическое дополнение кода, подсказки по использованию API и подсветку синтаксиса.
2. Встроенные инструменты для отладки позволяют проще искать и исправлять ошибки в коде.
3. Интеграция с Git позволяет легко отслеживать изменения в коде и управлять историей коммитов.
4. Встроенные темы и иконки позволяют настроить внешний вид редактора по своему вкусу.

Все эти функции делают Visual Studio Code идеальным инструментом для разработки проекта на Python и Flutter.

3.2 Реализация

3.2.1 Библиотека динамического взаимодействия с базой данных Based

Библиотека разрабатывается на языке Python с утилизацией сторонней библиотеки `psycopg3` [8], реализующей интерфейс DBAPI, описанный в PEP 249 [9]. Данная библиотека даёт нам унифицированный интерфейс доступа к базе данных, а также включает в себя множество классов, помогающих в разработке моей библиотеки динамического доступа к БД.

Моя библиотека предоставляет пользователю возможность в рантайме определять новые таблицы в базе данных, их поля, а также полный доступ CRUD созданных таблиц.

Для сборки библиотеки использовались возможности собственного GitlabCI [10] с последующей публикацией артефактов.

Исходный код библиотеки доступен на Gitlab [11].

3.2.2 Программный интерфейс приложения (API) Tuuli Backend

Для создания API использовался Python-фреймворк FastAPI [12], так как этот фреймворк позволяет писать быстрые сервисы и имеет широкую поддержку сообщества. Немаловажной причиной выбора стало также то, что данный фреймворк используется такими большими компаниями, как Microsoft, Uber, Netflix и многими другими IT-гигантами, что говорит о высокой надёжности.

Большим плюсом стало использование библиотеки Pydantic, что чётко и типобезопасно определять модели ответов сервера и принимаемых сервером данных. Также данный фреймворк в автоматическом режиме генерирует документацию API в формате OpenAPI [13], что позволяет легко интегрировать как внешние сервисы, так и интерфейс пользователя, для работы с системой.

Обмен данными между сервисной и клиентской происходит посредством передачи данных в формате JSON.

Общение с базой данных сервис производит посредством библиотеки, описанной в предыдущем пункте. Чтобы хранить информацию о существующих таблицах, их полях, пользователях и их уровнях доступа, а также о ресурсах и их уровнях доступа используются мета-таблицы. Возможности представить полную модель ER-диаграммы базы данных отсутствует ввиду динамической сущности системы, однако часть её представлена на рисунке 47.

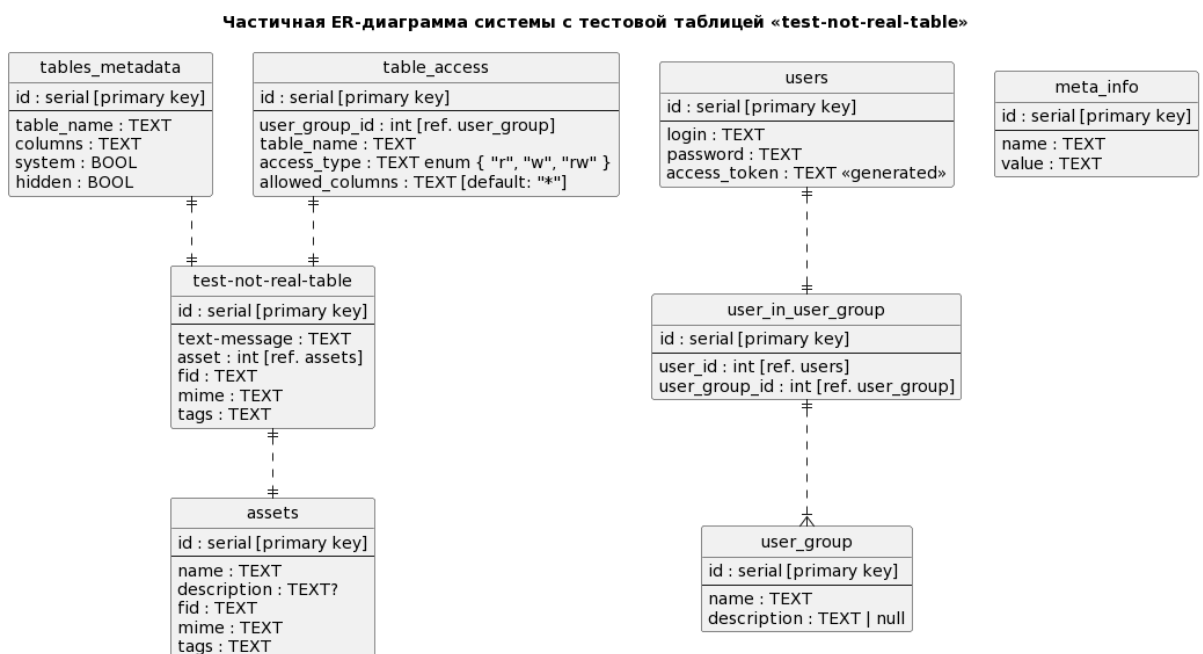


Рисунок 47 – Часть ER-диаграммы системы

3.2.3 Библиотека TuuliApi для языка программирования Dart

Для упрощения поддержки клиентского приложения TuuliApp взаимодействие с Tuuli Backend было вынесено в отдельную библиотеку. Так для разработки TuuliApp используются Dart и Flutter – целевым языком разработки библиотеки выбран Dart.

Данная библиотека реализует взаимодействие по HTTP, сериализацию и десериализацию в требуемый сервером формат входных и выходных данных.

Данная библиотека требует некоторых дополнительных доработок – автоматический проверка качества кода и CI/CD, поэтому на данный момент она не доступна публично.

3.2.4 Графический интерфейс пользователя TuuliApp

Разработка интерфейса пользователя отделена от разработки серверной части. Данный подход был выбран, так как для разработки пользовательского интерфейса используется другой стек, нежели Python и FastAPI, а именно – Dart и Flutter.

Flutter – это комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных, веб- и настольных приложений с использованием языка программирования Dart [14]. Flutter позволяет создавать красивые, нативно скомпилированные, мультиплатформенные приложения из одной кодовой базы [15]. Flutter разработан и поддерживается компанией Google [14]. Целевой платформой для данной работы был выбран десктоп.

В основу архитектуры приложения легла библиотека GetX [16], с помощью которой была реализована архитектура MVC. Выбор данной архитектуры облегчил разработку, позволив отделить отображение от бизнес-логики, а бизнес-логику от информационной модели, что в будущем сделает расширение функционала более простым.

3.3 Документация

3.3.1 Инструкция пользователя

Система состоит из двух частей: бэкенд (серверная часть, **Tuuli Backend**) и клиент (пользовательская часть, **TuuliApp**).

Для использования серверной части необходимо:

- 1) удостовериться, что установлен Python минимум версии 3.10;

- 2) скачать репозиторий приложения с GitLab;
- 3) перейти в директорию приложения и создать файл «.env», в котором необходимо прописать параметры DATABASE_URL со ссылкой подключения к базе данных PostgreSQL, «MINIO_HOST» с адресом хоста, на котором запущен экземпляр MinIO, «MINIO_ACCESS» и «MINIO_SECRET» с ключами доступа и секрета MinIO;
- 4) затем необходимо создать виртуальное окружение Python с помощью команды «python -m venv venv», которое затем необходимо активировать (способ зависит от целевой платформы запуска);
- 5) после активации необходимо установить зависимости с помощью команды «pip install -r requirements.txt»;
- 6) приложение готово к запуску с помощью команды «python -m uvicorn app:app», его автоматический запуск можно настроить соответствующими методами целевой платформы запуска;
- 7) после запуска приложения в автоматическом режиме создаётся пользователь «admin» с паролем «admin», под которым можно авторизоваться (рекомендуется создать нового пользователя-администратора, а этого удалить или сменить его пароль)

В качестве документации пользователя клиентского приложения могут использоваться разработанные в первом разделе прецеденты.

3.3.2 Инструкция разработчика

Настройка среды разработки VSCode для Python производится установкой двух дополнений: PyLance и Python, показаны на рисунке 48.

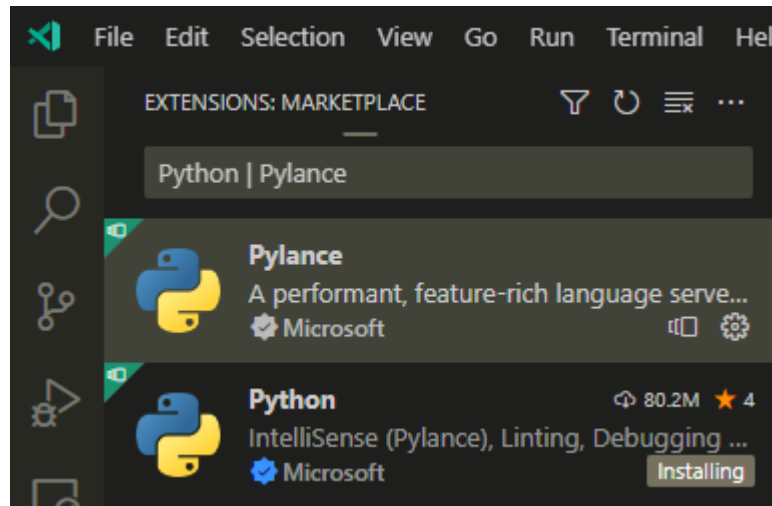


Рисунок 48 – Дополнения, необходимые для разработки на Python

Для разработки на Dart с использованием Flutter понадобится ещё два дополнения, показаны на рисунках 49 и 50.

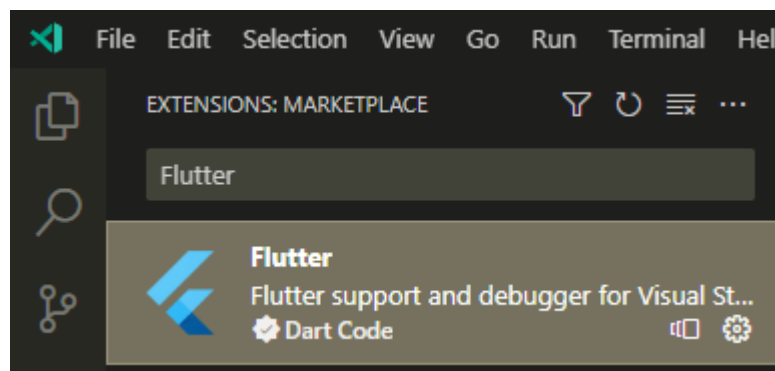


Рисунок 49 – Дополнение для разработки с Flutter

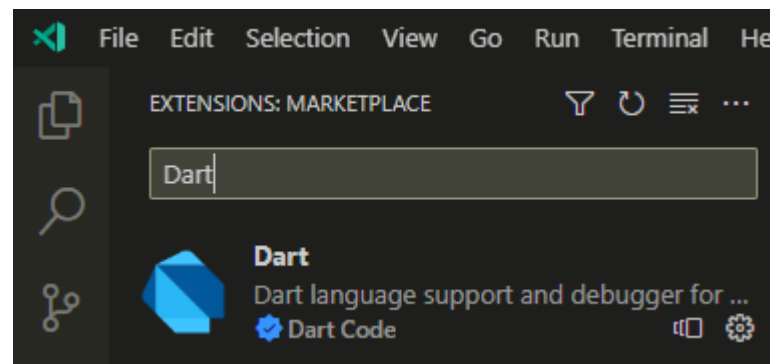


Рисунок 50 – Дополнение для разработки на Dart

Далее разработчику следует скачать себе репозиторий библиотеки, API и клиентского интерфейса и проводить разработку согласно потребностям.

3.4 Тестирование

Тестирование библиотеки использовались встроенные средства Python, в частности пакет unittest [17]. Для корректного тестирования следует озаботиться созданием базы данных и поменять строку подключения в главном файле db_test.py, затем запустить тест и убедиться, что все тесты прошли успешно.

Тестирование API и клиентского интерфейса проводилось совместно, по мере интеграции, вручную. Например, вход пользователя в систему тестировался следующим образом:

- открыто клиентское приложение «TuuliApp»;
- введена строка «admin» в качестве логина и пароля, «http://localhost:9000» в качестве адреса бэкенда;
- нажата кнопка «Войти».

Пользователь успешно авторизовался и перед ним открылась главная страница с меню переходов слева.

Таким образом были протестированы все прецеденты, в результате чего никаких дефектов обнаружено не было.

3.5 Выводы по главе

1. Реализована библиотека динамического взаимодействия с базой данных;
2. Реализовано пользовательское API;
3. Реализован пользовательский интерфейс;
4. Составлена инструкция разработчика по настройке VSCode для разработки частей системы;
5. Проведено тестирование, не выявившее дефектов.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы были проанализированы проблемы предметной области; имеющиеся системы управления репозиторием контента, их сильные и слабые стороны.

Реализована собственная система управления контентом и библиотека динамического взаимодействия с базой данных, которая может использоваться также вне проекта.

Все поставленные задачи реализованы, однако имеется возможность расширения: в библиотеку взаимодействия с базой данных «Based» можно добавить больше абстракции, отвязавшись таким образом от PostgreSQL, интерфейс клиентского приложения «TuuliApp» может быть доработан и сделан более удобным для, например, людей с ограниченными возможностями.

Код библиотеки, API и пользовательского интерфейса доступен для скачивания из Git репозитория, расположенного на Gitlab [11] [17] [18].

СПИСОК СОКРАЩЕНИЙ

БД – база данных

API (Application Programming Interface) – программный интерфейс приложения

CRUD (Create Read Update Delete) – операции создания, чтения, обновления и записи

JSON (JavaScript Object Notation) – объектная нотация JavaScript

PEP (Python Enhancement Proposals) – предложения по улучшению Python

MVC (Model-View-Controller) – модель, отображение, контроллер

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2001. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.
2. ГОСТ 7.9-95 (ИСО 214-76). Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования.
3. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.
4. СТО 4.2-07-2014. Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности.
5. directus/directus: The Modern Data Stack — Directus is an instant REST+GraphQL API and intuitive no-code data collaboration app for any SQL database. : сайт. – URL: <https://github.com/directus/directus> (дата обращения: 26.12.2022).
6. keystonejs/keystone: The most powerful headless CMS for Node.js — built with GraphQL and React : сайт. – URL: <https://github.com/keystonejs/keystone> (дата обращения: 26.12.2022).
7. appwrite/appwrite: Secure Backend Server for Web, Mobile & Flutter Developers AKA the 100% open-source Firebase alternative. : сайт. – URL: <https://github.com/appwrite/appwrite> (дата обращения: 26.12.2022).
8. The Psycorg 3 project. — Текст : электронный // Psycorg : [сайт]. — URL: <https://www.psycorg.org/psycorg3/> (дата обращения: 20.03.2023).
9. 1. PEP 249 – Python Database API Specification v2.0 | peps.python.org [Электронный ресурс]. URL: <https://peps.python.org/pep-0249/> (дата обращения: 20.03.2023).
10. GitLab CI/CD | GitLab [Электронный ресурс]. URL: <https://docs.gitlab.com/ee/ci/> (дата обращения: 20.03.2023).

11. nuark / based · GitLab // GitLab [Электронный ресурс]. URL: <https://glab.nuark.xyz/nuark/based> (дата обращения: 20.03.2023).
12. FastAPI [Электронный ресурс]. URL: <https://fastapi.tiangolo.com/> (дата обращения: 20.03.2023).
13. Home // OpenAPI Initiative [Электронный ресурс]. URL: <https://www.openapis.org/> (дата обращения: 20.03.2023).
14. Dart programming language [Электронный ресурс]. URL: <https://dart.dev/> (дата обращения: 20.03.2023).
15. Flutter - Build apps for any screen [Электронный ресурс]. URL: <https://flutter.dev/> (дата обращения: 20.03.2023).
16. getx | Dart Package // Dart packages [Электронный ресурс]. URL: <https://pub.dev/packages/getx> (дата обращения: 20.03.2023).
17. nuark / tuuli_backend · GitLab // GitLab [Электронный ресурс]. URL: https://glab.nuark.xyz/nuark/tuuli_backend (дата обращения: 20.03.2023).
18. nuark / tuuli_frontend · GitLab // GitLab [Электронный ресурс]. URL: https://glab.nuark.xyz/nuark/tuuli_frontend (дата обращения: 20.03.2023).

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра вычислительной техники

кафедра

УТВЕРЖДАЮ

Заведующий кафедрой



О.В. Непомнящий

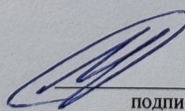
«12» 06 2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 — Информатика и вычислительная техника

Разработка системы управления репозиторием контента

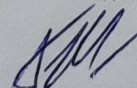
Руководитель



16.06.23
подпись, дата

доцент, канд. техн. наук М. С. Медведев

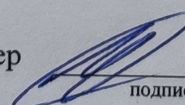
Выпускник



16.06.23
подпись, дата

А. А. Горбацевич

Нормконтролер



16.06.23
подпись, дата

доцент, канд. техн. наук М. С. Медведев

Красноярск 2023