**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**Пояснительная записка**

**по дисциплине «Введение в нереляционные базы данных»**

**Тема: Сервис для дизайна мозаик**

| Студенты гр. 0383 |  | Зенин П.А. |
| --- | --- | --- |
|  |  | Ханина М.И. |
|  |  | Орлов Д.С. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2023

**СОДЕРЖАНИЕ**

|  | Введение | 3 |
| --- | --- | --- |
| 1. | Сценарий использования | 5 |
| 1.1. | Макет UI | 5 |
| 1.2. | Сценарий использования | 5 |
| 2. | Модель данных | 8 |
| 2.1. | Нереляционнная модель | 8 |
| 2.2. | Оценка удельного объема информации в модели и скорость её роста | 10 |
| 2.3 | Направление роста модели | 11 |
| 2.4 | Запросы к модели | 11 |
| 3. | Разработанное приложение | 12 |
| 3.1. | Краткое описание | 12 |
| 3.2. | Использованные технологии | 12 |
| 3.3 | Снимки экрана приложения | 13 |
| 4. | Выводы | 15 |
| 4.1. | Достигнутые результаты | 15 |
|  | Недостатки и пути для улучшения | 15 |
|  | Будущее развитие решения | 15 |
| 5. | Приложения | 17 |
| 6. | Список использованных источников | 18 |

**ВВЕДЕНИЕ**

1. Актуальность решаемой проблемы:

Сервис для дизайна мозаик находит актуальность в современном мире, отвечая на ряд потребностей и тенденций. В эпоху стремления к индивидуальности в дизайне интерьера, мозаика предоставляет возможность создания уникальных узоров и рисунков, подчеркивая потребность в персонализации пространства. Тренд на ручную работу и уникальные элементы в дизайне усиливает интерес к мозаике, которая становится средством творческого самовыражения.

1. Постановка задачи:

Необходимо создать веб-приложение для создания мозаики из

загруженного пользователем изображения. Пользователь должен иметь возможность зарегистрировать аккаунт, сделать мозаику, сохранить результат локально, посмотреть инструкцию по сборке и все созданные ранее проекты.

1. Предлагаемое решение:

Для решения поставленных задач предлагается использовать базу данных Neo4j. Neo4j - база данных, предназначенная специально для работы с графовыми структурами данных и подходит для сервиса дизайна мозаик из-за его способности эффективно управлять сложными графовыми структурами, типичными для мозаичного искусства. База данных обеспечивает гибкость в моделировании отношений между цветами и формами, а также позволяет анализировать пути в графе, что полезно для оптимизации композиций.

1. Качественные требования к решению:

* UX и UI:
  1. Интуитивно понятный и легко осваиваемый пользовательский интерфейс.
* Безопасность и конфиденциальность:
  1. Гарантии безопасности и конфиденциальности данных пользователей;
  2. Возможность сохранения проектов в защищенных личных профилях.
* Удобство использования:

1. Возможность создания мозаичных композиций без необходимости специализированных навыков;
2. Возможность редактирования исходного изображения;
3. Режим с пошаговой инструкцией сборки готовой мозаики;
4. Доступ к подробной информации о характеристиках мозаики.

**1. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ**

**1.1. Макет UI**

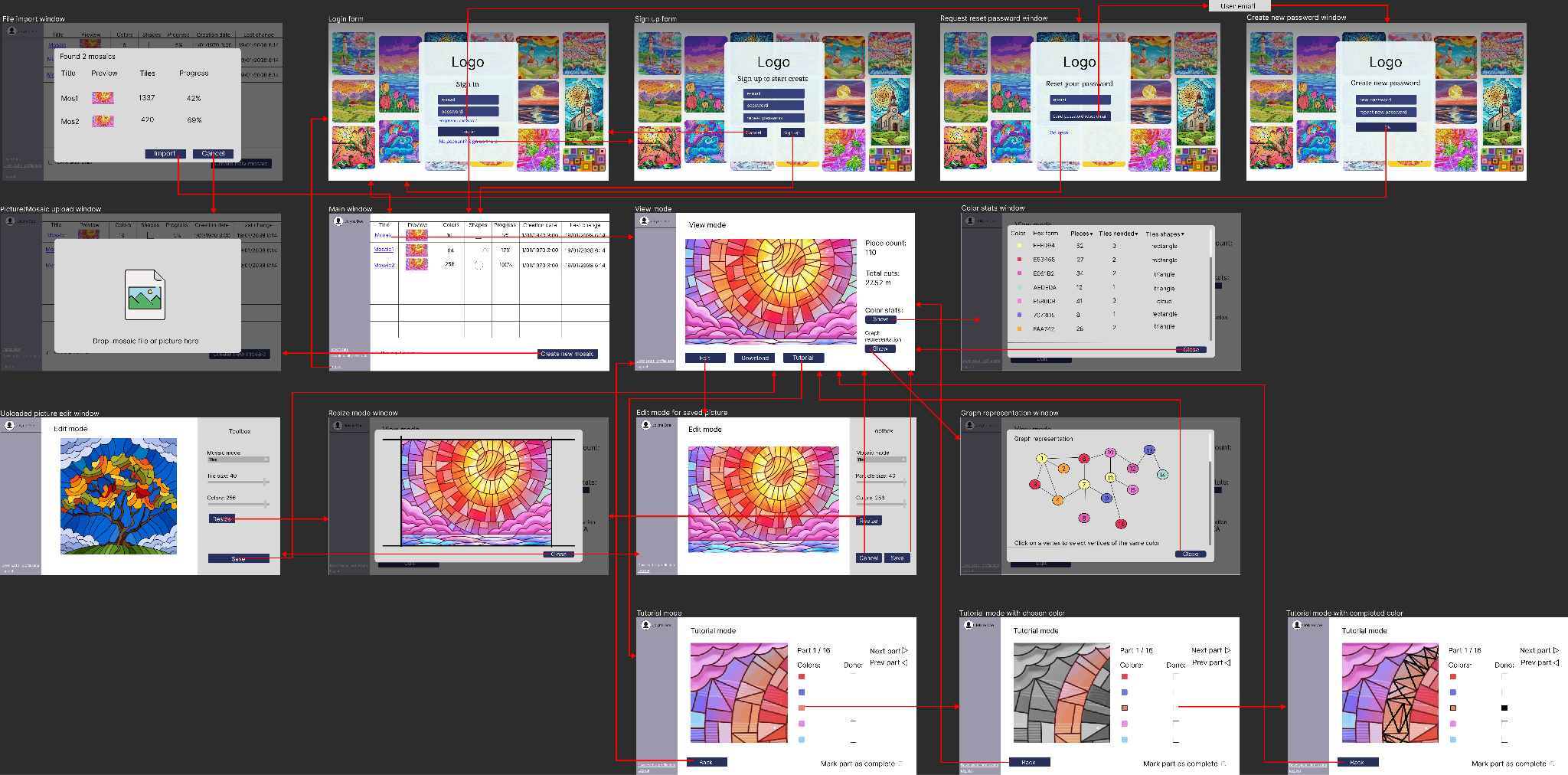


Рис. 1 – Скриншот макета UI

**1.2. Сценарий использования**

Было выделено несколько сценариев использования:

1. Основной сценарий: создание мозаики:

* Пользователь вводит свой логин и пароль в форму авторизации
* Пользователь на главной странице выбирает действие "создать новую мозаику"
* Пользователь перетаскивает изображение в поле для загрузки
* Пользователь редактирует изображение по своему желанию и нажимает кнопку далее
* Пользователь видит результат (статистика, мозаика) и может его сохранить на свой компьютер (по желанию нажатием кнопки "скачать").

1. Сценарий использования – просмотр готовой мозаики:

* Пользователь вводит свой логин и пароль в форму авторизации
* Пользователь на главной странице нажимает на thumbnail готовой мозаики
* Перед пользователем появляется статистика и выбранная мозаика, которую он может сохранить на свой компьютер (по желанию нажатием кнопки "скачать").

1. Сценарий использования – редактирование готовой мозаики:

* Пользователь вводит свой логин и пароль в форму авторизации
* Пользователь на главной странице нажимает на thumbnail готовой мозаики
* Пользователь нажимает на кнопку "редактировать", после чего открывается окно с возможностью редактировать параметры (цвет, мелкость разбиения)
* Пользователь нажимает кнопку "готово"
* Перед пользователем появляется статистика и выбранная (изменённая) мозаика, которую он может сохранить на свой компьютер (по желанию нажатием кнопки "скачать").

Альтернативные сценарии:

Логин или пароль неправильный:

* Пользователь вводит неправильные данные в форму авторизации, высвечивается окно с ошибкой

Пользователь забыл пароль:

* Пользователь нажимает кнопку "forgot your password?"
* Пользователь вводит e-mail в форму
* После перехода по ссылке из письма, пользователь задаёт новый пароль

Неправильный формат изображения (формат сжатия или размер):

* Пользователь перетаскивает в форму некорректное изображение, высвечивается окно с ошибкой
* Пользователь закрывает окно с ошибкой и видит то же самое состояние приложения (форма для загрузки изображения)

Случайное закрытие вкладки во время редактирования изображения:

* Пользователь закрывает вкладку с редактором изображения
* Высвечивается окно с подтверждением "Хотите ли покинуть эту страницу?", где пользователь может выбрать "остаться" или "покинуть"

Пользователь нажимает кнопку "отмена" при создании мозаики:

* Пользователь возвращается на главную страницу

Пользователь нажимает кнопку "отмена" при редактировании готовой мозаики:

* Пользователь возвращается на страницу с просмотром выбранной мозаики

**2. МОДЕЛЬ ДАННЫХ**

**2.1. Нереляционнная модель**

Для разработки приложения необходимо было использовать нереляционную базу данных. В качестве таковой использовалась графовая СУБД neo4j.

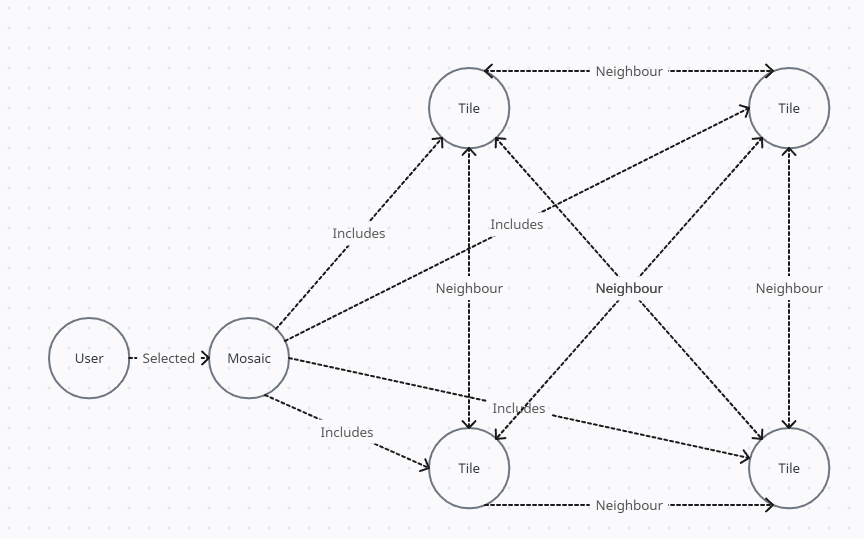


Рис. 2 – Графовая схема нереляционной БД

В ходе анализа макета приложения, были выделены следующие сущности и отношения между ними:

Сущности:

User - предназначен для хранения каждого пользователя. Узел имеет метки:

* eMail: String - хранит адрес электронной почты.
* password: String - хранит пароль пользователя.

Tile - предназначен для хранения информации о плитке мозаики. Узел имеет метки:

* shape: String - хранит информацию о форме плитки.
* colour: String - хранит цвет плитки.
* position: String - хранит информации о позиции плитки.

Mosaic - предназначен для хранения мозаик. Имеет метку name: String, которая хранит название мозаики

Отношения:

Selected - взаимосвязь, предназначена для описания мозаик, которые видны пользователю - то есть те, которые он выбрал для воспроизведения. Данное отношение имеет атрибуты:

* dateCreated: String - для хранения момента создания мозаики;
* dateEdited: String - для хранения момента последнего редактирования;
* progress: Integer - для отслеживания прогресса воспроизведения мозаик.

Includes - взаимосвязь, показывающая плитки, принадлежащие определённой мозаике.

Neighbour - взаимосвязь, показывающая плитки, являющиеся соседями.

**2.2. Оценка удельного объема информации в модели и скорость её роста**

Neo4j для хранения данных использует фиксированное количество памяти для хранения каждой сущности:

* 15 байт для хранения каждого узла;
* 34 байта для хранения каждого отношения;
* 41 байт для хранения атрибутов узлов и отношений;
* 128 байт для хранения строчных атрибутов.

Следовательно, для хранения:

* узла User с его атрибутами требуется - 15 + 128 \* 2 = 271 байт;
* узла Mosaic с его атрибутами требуется - 15 + 128 = 143 байта;
* узла Tile с его атрибутами требуется - 15 + 128 \* 3 = 399 байт;
* отношения Selected с его атрибутами требуется - 34 + 41 + 128 \* 2 = 331 байт;
* отношения Includes с его атрибутами требуется - 34 байта;
* отношения Neighbour с его атрибутами требуется - 34 байта.

Пусть количество пользователей - U, количество мозаик пользователя - uM, среднее число плиток в мозаике - T, количество соседей плитки - tN. Тогда чистый объём базы данных можно оценить следующим образом:

271\*U+331\*uM+143\*M+339\*T\*M+34\*T\*M+34\*tN,

где M - общее число мозаик.

Рост будет происходить линейным образом.

Если принять среднее значения uM = 5 мозаик, U = 50 пользователей, T = 2500 плиток, tN = 8 соседей, то в получившейся базе данных будет зависимость:

size = 1082643\*M + 15477 байт

Если принять, что в системе будет 250 уникальных мозаик, то общий объём получится:

1082643 \* 250 + 15477 = 258,1 мегабайт.

**2.3. Направление роста модели**

Исходя из предыдущего пункта, можно сделать вывод, что модель обладает линейным ростом, так как при создании узла или отношения выделяется константное количество памяти.

**2.4. Запросы к модели**

Запросы к СУБД neo4j осуществляются при помощи специальных команд под названием “Cypher”. Были использованы следующие команды:

Создание узла User:

CREATE (:User {email: $email, password: $password})

Создание узла Mosaic и отношения к пользователю:

MATCH (u:User) WHERE u.email = $email CREATE (m:Mosaic {picture: $picture, title: $title, creation\_timestamp: $timest}) CREATE (m)-[:OWNED\_BY]->(u)

Поиск всех активных мозаик пользователя:

MATCH (n:User) WHERE n.email = $email MATCH (m:Mosaic)-[:OWNED\_BY]->(n) return m

Удаление мозаики:

MATCH (m:Mosaic) WHERE m.title = $title DETACH DELETE (m)

**3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ**

**3.1. Краткое описание**

Было разработано приложение, осуществляющее авторизацию пользователя и добавление мозаики. Модули приложения, отвечающие за выполнение задач других сценариев разработаны не были.

**3.2. Использованные технологии**

1. Frontend:

* HTML, CSS, SASS - для верстки;
* Vue.js 3 - JavaScript-фреймворк для построения интерфейсов;
* TableLite.vue - компонент Vue.js для отображения табличных данных;
* Axios - библиотека для выполнения HTTP-запросов из браузера.

1. Backend :

* Flask - Python библиотека для создания HTTP-сервера.
* Neo4j for python - библиотека для обращения к серверу СУБД neo4j

1. База данных:

* Neo4j - графовая база данных.

**3.3 Снимки экрана приложения**

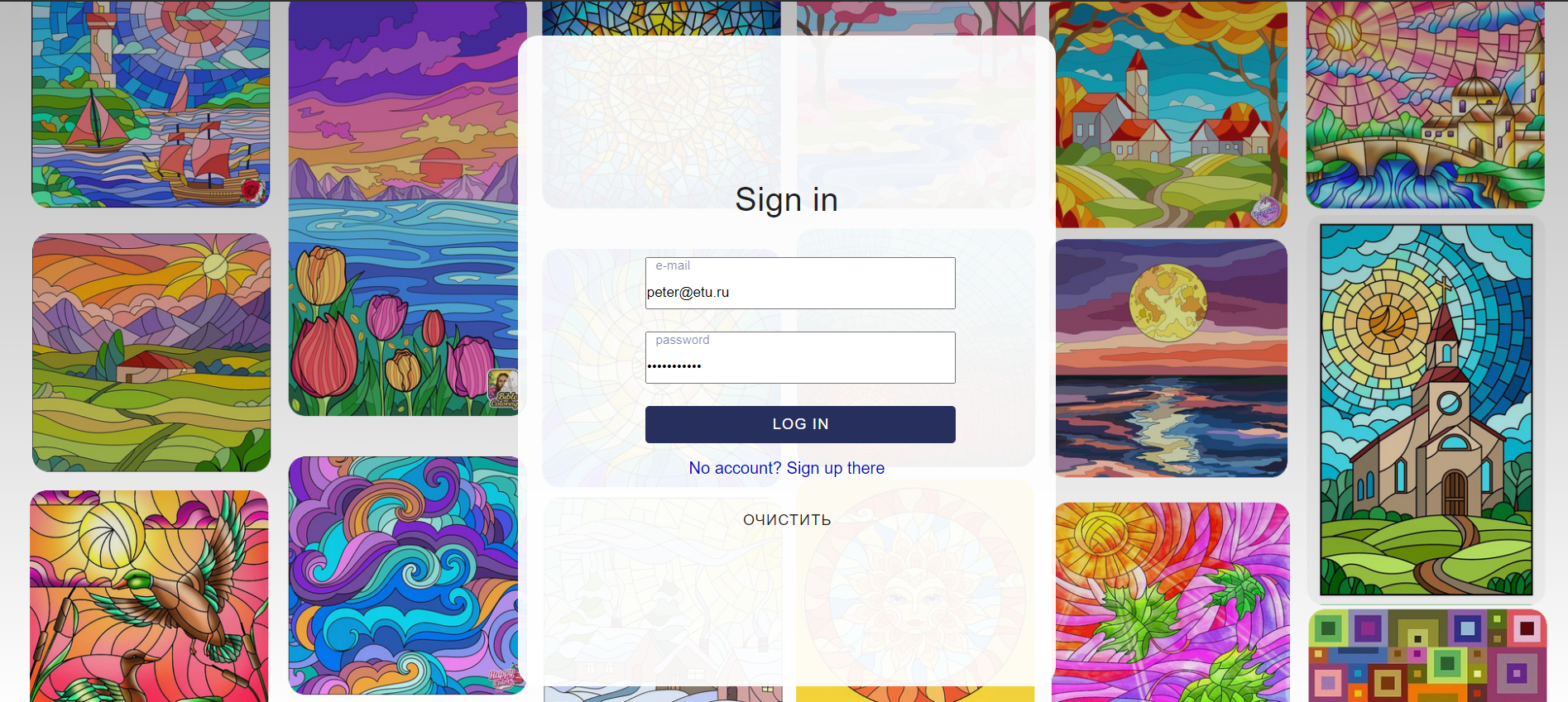


Рис. 3 – Форма авторизации

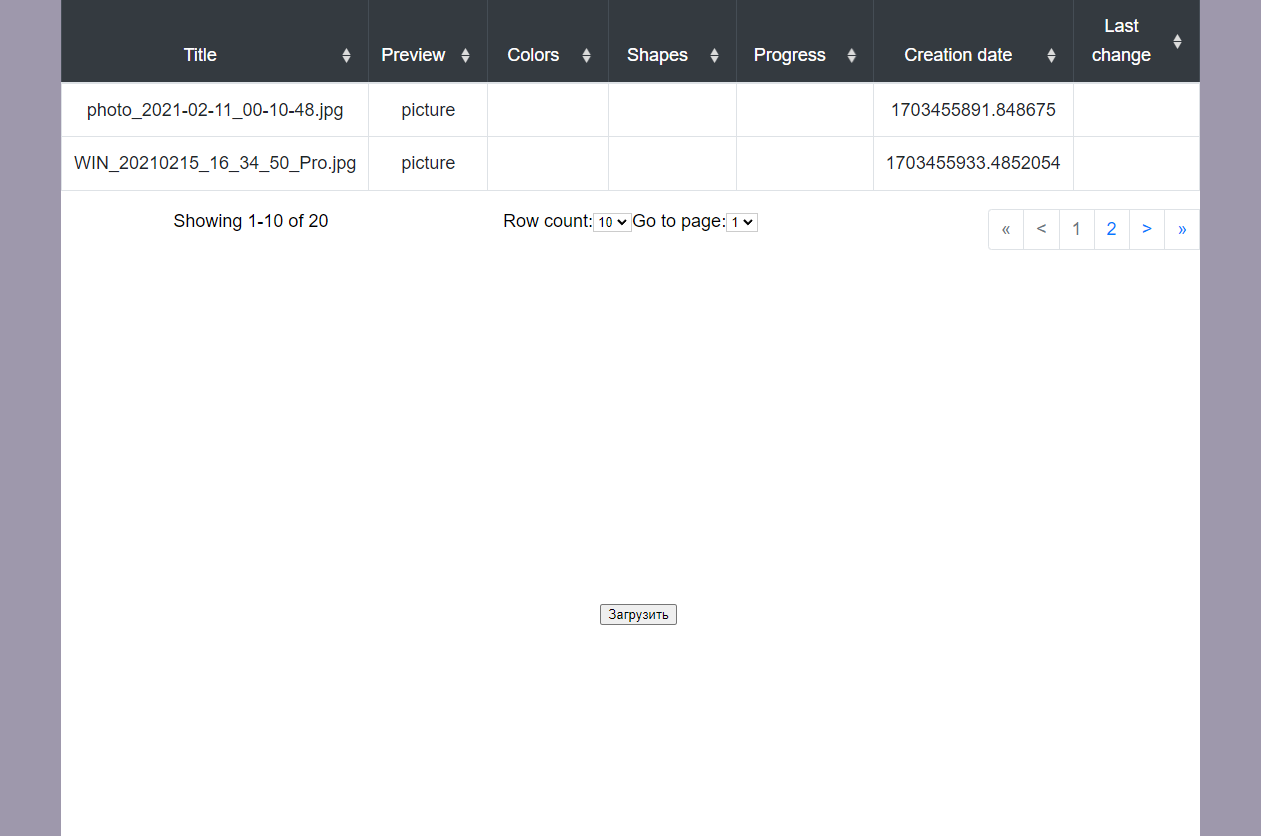


Рис. 4 – Главное окно со списком мозаик пользователя

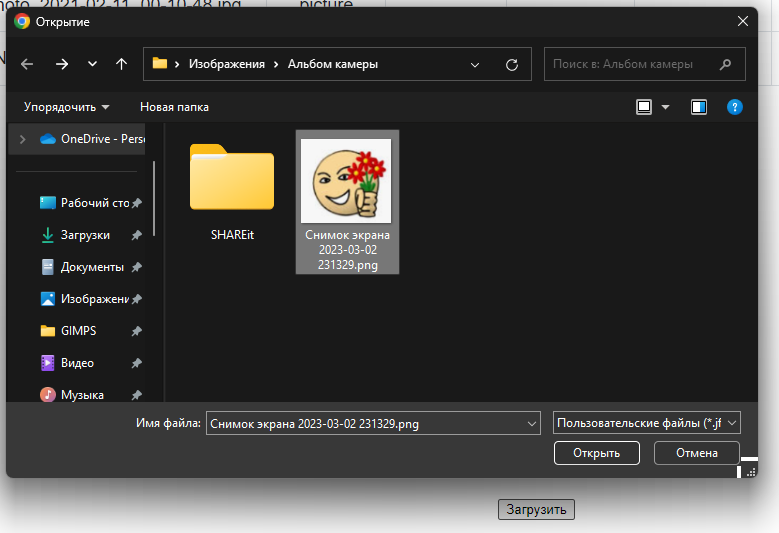


Рис. 5 – Окно загрузки изображения

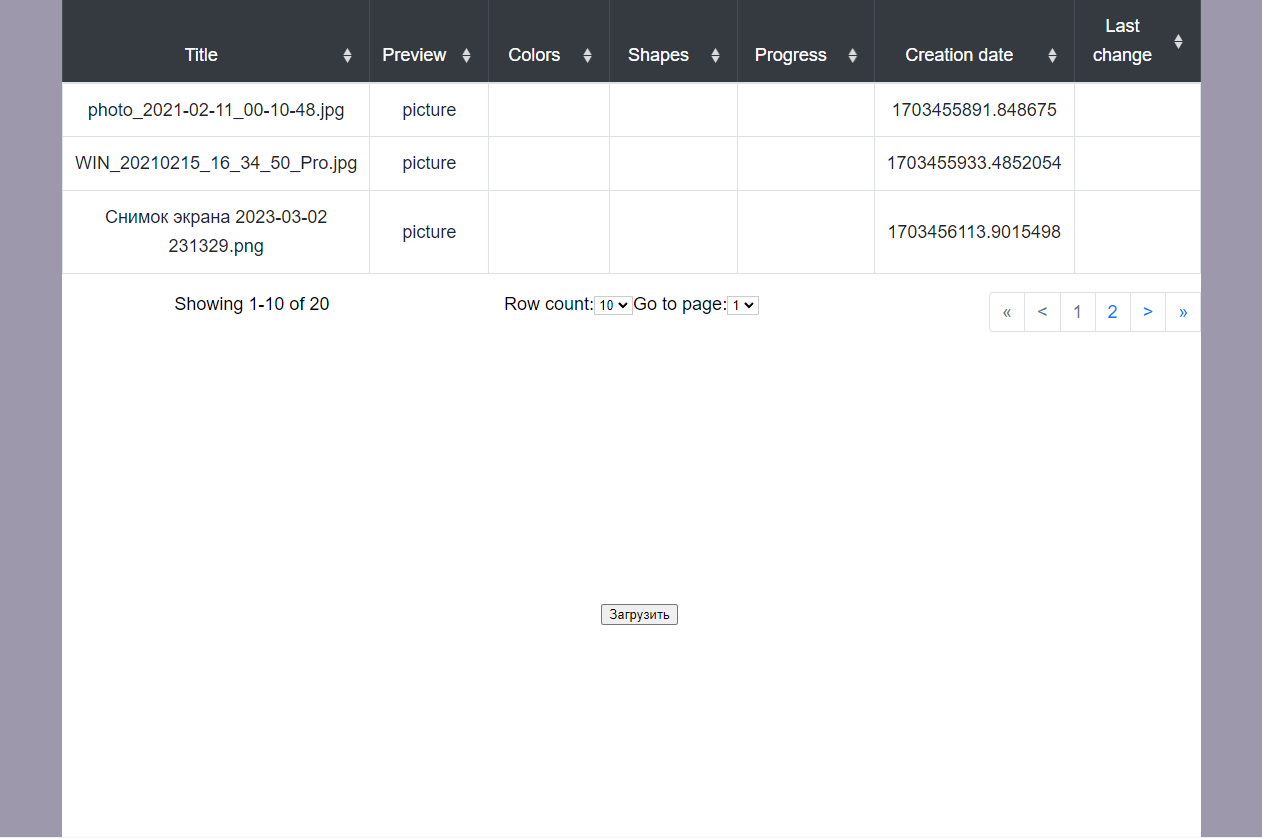


Рис. 6 – Обновлённый список мозаик пользователя после добавления нового изображения**ВЫВОДЫ**

**Достигнутые результаты:**

Было написано приложение, реализующее основной сценарий использования сервиса. При создании сервиса использовались WEB-технологии и нереляционная БД.

**Недостатки и пути для улучшения:**

1. Нереализованные альтернативные сценарии работы сервиса:
2. Завершение разработки онлайн-редактора мозаики, где пользователи могут не только загружать изображения, но и редактировать их, добавлять и перемещать элементы, изменять цвета и формы;
3. Оптимизация запросов:
4. Выполнять кэширование мозаик на стороне клиента для уменьшения количества запросов на получение изображений с сервера;

3. Интерфейс:  
 a. Доработать внешний вид сервиса для улучшения пользовательского

опыта.

**Будущее развитие решения:**

1. Внедрение функционала совместной работы, позволяющего пользователям создавать мозаики в режиме реального времени, а также обмениваться идеями и отзывами;
2. Добавление возможности автоматической загрузки изображений из профилей социальных сетей, а также обмена проектами между пользователями через социальные платформы;
3. Разработка пользовательских профилей с возможностью создания портфолио проектов мозаики, что может стимулировать творческую активность и конкуренцию между пользователями;
4. Внедрение кроссплатформенности, разработка мобильного приложения.

**ПРИЛОЖЕНИЯ**

Документация по сборке и развертыванию приложения:

Запуск базы данных: ./neo4j.bat console

Запуск backend: python3 ./backend.py

Запуск клиента: vite dev

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

<https://github.com/moevm/nosql2h23-mosaic> **-** репозиторий приложения

<https://neo4j.com/docs/> - документация по работе с Neo4j

<https://flask.palletsprojects.com/en/3.0.x/> - документация по работе с фреймвором Flask

<https://vuejs.org/guide/introduction.html> - документация по работе с фреймворком Vue

<https://developer.mozilla.org/ru/docs/Web/JavaScript> - документация по работе с JavaScript

<https://github.com/nuno-faria/tiler> - репозиторий конвертера изображений в мозаику