

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ»

БИРСКИЙ ФИЛИАЛ УУН<sub>ИТ</sub>  
ФАКУЛЬТЕТ ФИЗИКИ И МАТЕМАТИКИ  
КАФЕДРА ИНФОРМАТИКИ И ЭКОНОМИКИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
ПО ПРОГРАММЕ БАКАЛАВРИАТА

ХУСАИНОВ РЕНАТ МАРАТОВИЧ

ЦИФРОВАЯ ГАЛЕРЕЯ

Выполнил:  
Студент 4 курса очной формы  
обучения  
Направление подготовки  
09.03.03 Прикладная информатика  
Направленность (профиль)  
Прикладная информатика в  
информационной сфере

Руководитель  
к.ф.-м.н., доцент кафедры ИиЭ  
Тазетдинова Ю.А.

БИРСК – 2023

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ .....	3
ГЛАВА 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ .....	4
1.1. Обзор деятельности организации.....	4
1.2. Структурно-функциональная диаграмма бизнес-процесса организации по типу «как есть» и ее описание .....	4
1.3. Анализ существующих разработок и применяемого программного обеспечения.....	9
1.4. Анализ затрат, выгод и рисков .....	12
Выводы по главе 1 .....	15
ГЛАВА 2. ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ .....	16
2.1. Разработка структурно-функциональной диаграммы по типу «как должно быть».....	16
2.2. Описание задач автоматизации информационных процессов поведенческими диаграммами в нотации UML .....	20
2.3. Формулирование технического задания .....	24
2.4. Описание информационной модели комплекса задач (схема потоков данных).....	25
2.5. Проектирование Базы Данных.....	26
2.6. Проектирование интерфейса .....	29
Выводы по главе 2 .....	30
ГЛАВА 3. РЕАЛИЗАЦИЯ КОМПОНЕНТОВ ИНФОРМАЦИОННОЙ СИСТЕМЫ.....	31
3.1. Реализация front-end части информационной системы .....	31
3.2. Реализация back-end части информационной системы.....	33
3.3. Реализация базы данных .....	37
3.4. Руководство использования .....	38
Выводы по главе 3 .....	43
ЗАКЛЮЧЕНИЕ .....	44
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ.....	45
ПРИЛОЖЕНИЕ .....	49

## ВВЕДЕНИЕ

Ещё 15–20 лет назад нельзя было предположить, что веб-приложения станут неотъемлемой частью жизни. Сегодня эта разновидность онлайн-инструментов используется для различных задач, включая оптимизацию бизнес-процессов, продажу товаров и услуг, распространение информации, общение пользователей друг с другом.

Веб-приложение представляет собой веб-сайт, на котором размещены страницы с частично либо полностью несформированным содержанием. Окончательное содержание формируется только после того, как посетитель сайта запросит страницу с веб-сервера. В связи с тем, что окончательное содержание страницы зависит от запроса, созданного на основе действий посетителя, такая страница называется динамической.

Любой информационный процесс может быть автоматизирован с помощью веб-приложения. Оно имеет неоспоримый ряд преимуществ: удобство развертывания, надежность, доступность, кроссплатформенность и функциональность. По этим причинам в данной работе оно было выбрано в качестве средства автоматизации.

Объект исследования – выставочная деятельность художественной галереи.

Предмет исследования – автоматизация информационных процессов сбора, хранения и обработки информации об объектах галереи.

Цель исследования – разработка прототипа информационной системы, осуществляющей автоматизацию процессов сбора, хранения и обработки информации об объектах галереи.

Задачи:

- Проведение анализа предметной области.
- Проектирование компонентов ИС.
- Разработка компонентов ИС.

# **ГЛАВА 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1. Обзор деятельности организации**

В качестве исследуемого учреждения для реализации проектного решения была выбрана галерея Бирского филиала Уфимского университета науки и технологий, которая ведет свою деятельность 52 года, начиная с 1971 года.

Основной вид деятельности – выставка произведений искусств студентов и художников из других регионов и стран.

Галерея располагается по адресу Республика Башкортостан, г. Уфа, ул. Давлетшиной, д. 18.

Основные цели художественной галереи – поддержание преемственности художественного творчества, культурно-просветительская работа среди населения, сохранение культурной среды города Бирска.

Непосредственно для данного данной галереи будет предназначена разрабатываемая информационная система.

## **1.2. Структурно-функциональная диаграмма бизнес-процесса организации по типу «КАК ЕСТЬ» и ее описание**

Исследуем информационные процессы предметной области в нотации IDEF0. Рассмотрим процесс "Организация выставки картин".

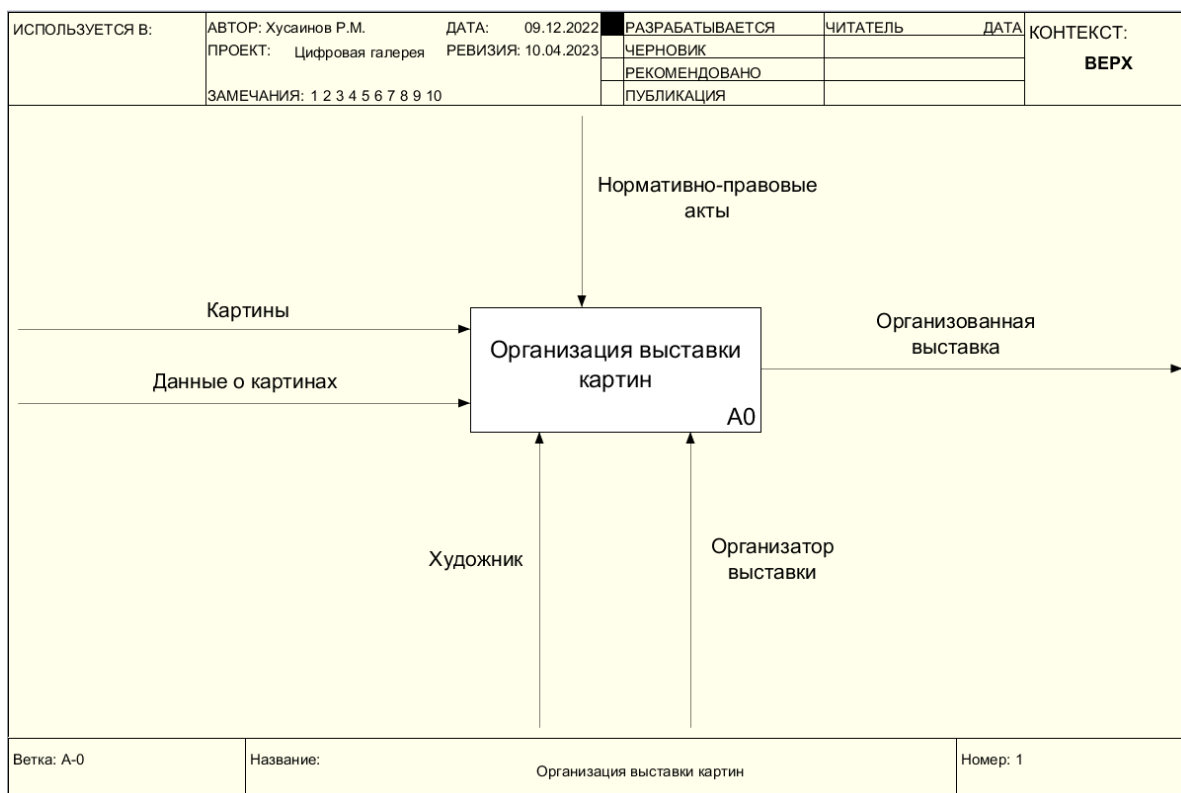


Рисунок 1.1 Контекстная диаграмма в нотации IDEF0, модель «Как есть» информационного процесса «Организация выставки картин»

Нотация IDEF0 предназначена для формализации и описания бизнес – процессов. С помощью средств данной нотации была создана контекстная диаграмма (Рисунок 1.1), направленная на отражение процесса создания картины и организации ее выставки.

На представленной контекстной диаграмме можно выявить следующие типы стрелок:

**Вход** – картины и данные о картинах. Они представляют собой объекты, которые будут использованы или преобразованы для получения результата (выхода);

**Выход** – организованная выставка, это объект, в который преобразуются входы;

Управление – нормативно-правовые акты, регулирующие выставочную деятельность. Они представляет собой правила и требования, регламентирующие правомерность организации выставки.

Механизм – художник, авторставляемых произведений, и организатор выставки, непосредственно участвующий в организации данной конкретной выставки.

В настоящее время создание и выставка картины происходит следующим образом:

1. Художник обращается в галерею для выставления своих картин. Назначается организатор выставки
2. Происходит планирование выставки, формируется план-конспект. План-конспект имеет свою структуру, вмещающую краткое описание всех составляющих данного процесса:
  - цели выставки;
  - впечатления, которые ожидается получить от посетителей;
  - пошаговое описание выставки с указанием коммуникативных задач и средств для их решения, направлений движения потока посетителей, эскизов экспозиции.
3. Выставка афишируется кругу лиц для привлечения посетителей.
4. Совершается подготовка к вернисажу, то есть торжественному открытию выставки. На данном этапе происходит монтаж выставки и оценка результатов проведенной работы, при необходимости вносятся коррективы.
5. Проведение выставки.

Данный процесс можно оформить следующей декомпозицией контекстной диаграммы, в которой были выявлены следующие функциональные блоки (Рисунок 1.2):

1. Планирование выставки
2. Афиширование выставки
3. Подготовка к вернисажу

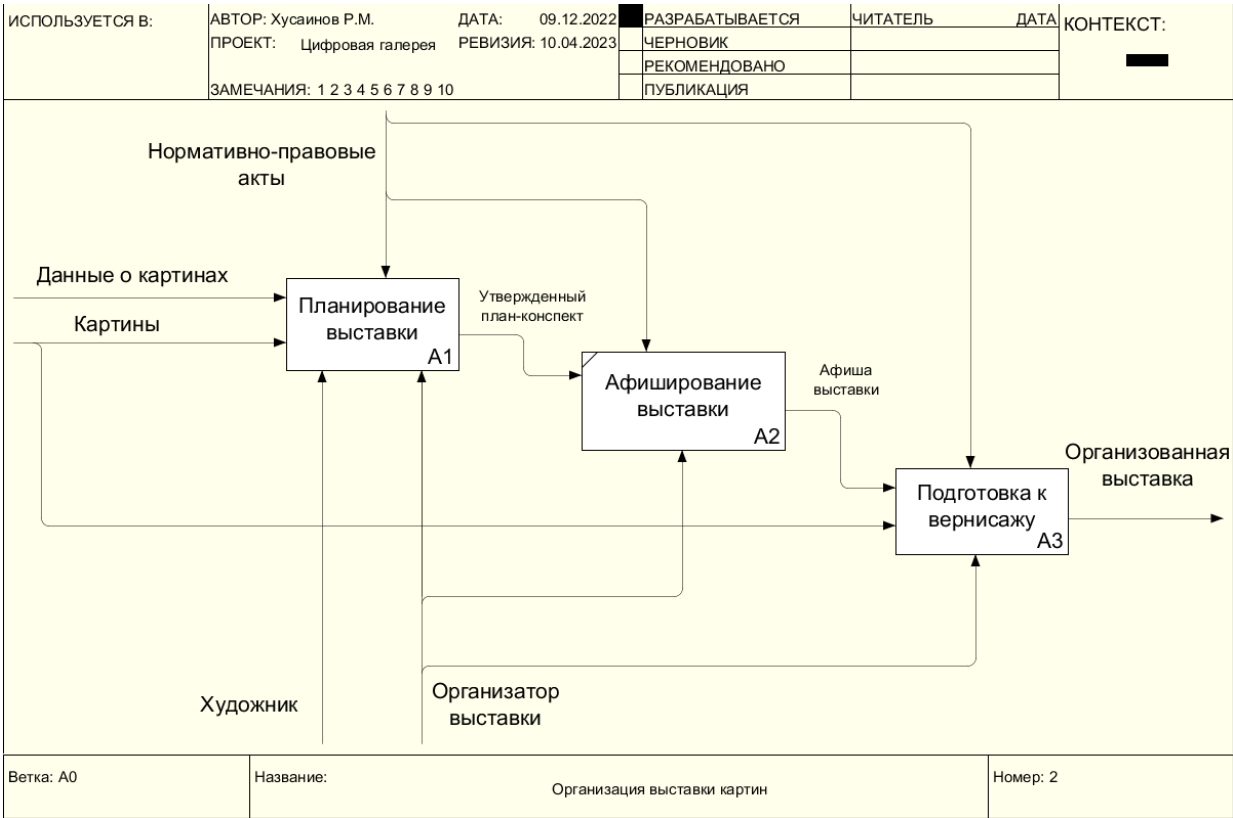


Рисунок 1.2 Результат декомпозиции контекстной диаграммы

Ниже на Рисунке 1.3 представлена иллюстрация процесса планирования выставки. Он происходит в 3 этапа: формирование концепции выставки, составление плана-конспекта и оценки формата выставки.

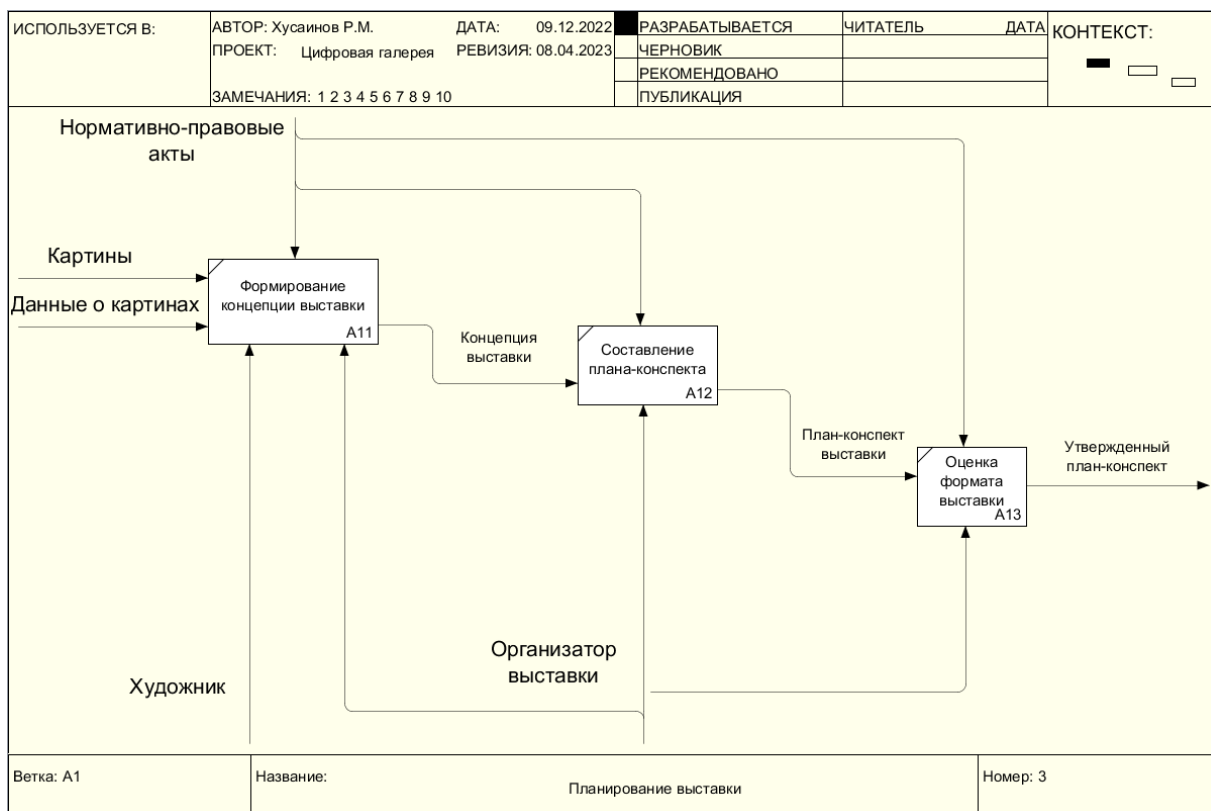


Рисунок 1.3 Диаграмма декомпозиции процесса "Планирование выставки"

В ходе разработки структурно-функциональной диаграммы по типу «Как есть» были выявлены следующие недостатки:

- выставляемые картины художника недоступны широкому кругу лиц;
- публикация картин только в офлайн галерее лишает художника шанса получить большее количество поклонников его творчества;
- отсутствует удобная централизованная площадка, где могли бы афишироваться выставки.



### **1.3. Анализ существующих разработок и применяемого программного обеспечения**

Галереи России преимущественно используют информационные системы, использующие web-технологии (например, web-сайт галереи <https://www.miras.ru/>). Web-сайт галереи "Мирас" был разработан с помощью конструктора сайтов Tilda, это влечет ряд весомых недостатков, свойственных конструкторам сайтов, таких как:

1. дороговизна тарифов,
2. отсутствие масштабируемости,
3. отсутствие изменения backend составляющей сайта,
4. размещение сайта исключительно на хостинге Tilda,
5. Относительно мало места на хостинге.

Вопрос о выборе инструментов разработки для создания информационной системы возникает всегда, т.к. от них зависит скорость и качество разработки. Вышеперечисленных недостатков конструкторов сайтов лишен самостоятельно разработанный продукт.

Для разработки frontend части приложения решено использовать технологии HTML, CSS и JavaScript, которые позволяют реализовать удобный и функциональный пользовательский интерфейс.

Одним из популярных библиотек для разработки интерфейса является React, который имеет неоспоримый ряд преимуществ:

1. построение интерфейса из отдельных компонентов, которые легко поддерживать;

2. добавление удобного слоя абстракции, который избавляет от необходимости работать с DOM (от англ. Document Object Model — «объектная модель документа») напрямую;
3. Благодаря сообществу, у React хорошо проработанная документация и большой опыт, накопленный в статьях, курсах и конференциях. Это значительно облегчает не только изучение библиотеки новичками, но и поиск ответов на всевозможные вопросы в процессе разработки;
4. React — это проект с открытым исходным кодом. Благодаря этому, его можно безопасно использовать даже в коммерческих приложениях.

Неотъемлемой частью информационной системы является ее серверная, backend составляющая, позволяющая реализовать логику работы приложения при взаимодействии пользователя с системой. Для ее реализации была выбрана программная платформа NodeJS, основанная на JavaScript-движке V8. V8 – это движок с открытым исходным кодом, который написан на C++. NodeJS превращает JavaScript из узкоспециализированного языка в язык общего назначения, позволяя реализовать сервер составляющую web-приложения.

Взаимодействие пользовательского интерфейса с серверной частью основано на протоколе HTTP. Используются конечные точки (endpoint), по которым клиентская часть приложения обращает свои запросы к программному интерфейсу приложения (API).

Для управления содержимым информационной системы используется система Strapi. Она представляет собой фреймворк для управления контентом, работающий на Node.js. Это полностью бесплатный проект с открытым исходным кодом. Система разворачивается локально на собственном сервере, что обеспечивает безопасность данных.

Главные особенности и преимущества Strapi:

- Открытый исходный код. Система разработана энтузиастами и поддерживается сотнями участников GitHub, которые развивают ее в соответствии с новыми требованиями и технологиями. Она всегда будет доступна и бесплатна.
- Широкие и гибкие настройки. Панель администратора, как и API, настраиваются достаточно гибко. Возможно расширение функционала за счет плагинов.
- RESTful или GraphQL. CMS поддерживает передачу данных посредством таких архитектурных решений как REST, так и GraphQL. Это расширяет возможности взаимодействия с разными клиентами, мобильными приложения, IoT-устройствами.
- Локальное размещение. Размещение на собственном сервере владельца системы гарантирует конфиденциальность и обеспечивает повышенный уровень защиты данных.
- Один язык. Система использует JavaScript, что позволяет работать с одним языком как в сервер составляющей, так и во фронтенде.

**Выбор СУБД.** В качестве СУБД была выбрана SQLite.

База данных — это набор структурированной информации. Для ее изменения требуются системы управления — СУБД. Как и любая СУБД, SQLite позволяет записывать новую и запрашивать существующую информацию, изменять ее, настраивать доступ.

Благодаря своим свойствам, SQLite применяется:

- на сайтах с низким и средним трафиком;
- в локальных однопользовательских, мобильных приложениях или играх, не предназначенных для масштабирования;

- в программах, которые часто выполняют прямые операции чтения/записи на диск;
- в приложениях для тестирования бизнес-логики.

#### 1.4. Анализ затрат, выгод и рисков

Каждый этап разработки требует времени. Для оценки трудоемкости разработки в Таблице 1.1 указаны ориентировочные затраты времени для каждого этапа. Общая трудоемкость складывается из суммарного количества часов каждого этапа.

Таблица 1.1  
Трудоемкость разработки по этапам

№	Название этапа	Количество часов, ч
1.	Анализ предметной области	32
2.	Проектирование веб-приложения	96
3.	Разработка веб-приложения	85
4.	Тестирование и отладка	32
	Общая трудоемкость:	245

В соответствии с данными таблицы, общая трудоемкость  $T = 245$  ч.

**Расчет заработной платы.** Заработная плата рассчитывается исходя из размера часовой заработной платы специалиста, которая в нашем случае составляет 200 руб/ч., тогда:

$$\text{Заработная плата} = 200 \cdot 245 = 49\,000 \text{ руб.}$$

На заработную плату делаются начисления в размере 30% от начисленной заработной платы.

$$\text{НЗП} = 49\,000 \cdot 0.3 = 14\,700 \text{ руб.}$$

Итоговая заработная плата разработчика: 49 000 руб. + 14 700 руб. = 63 700 руб.

**Расчет затрат на электроэнергию.** Потребление электричества компьютером составляет примерно 220 Ватт в час. Если компьютер работает сутки, то расход 5,28 кВт\*ч за сутки.

За год расход на электроэнергию на компьютер менеджера с округлением 1927 кВт\*ч в год.

Цена электроэнергии для юридических лиц 2,81 руб. за 1кВт\*ч.

Итого 5 415 руб. за год.

**Хостинг.** Для размещения проекта АИС в сети интернет необходимо приобретения хостинга. При низко нагруженном проекте подходящим вариантом будет виртуальный хостинг, стоимостью примерно 200 руб. в месяц.

Покупка имени домена 500 руб. / год.

**Цена оргтехники.** Цена компьютера, необходимого для работы администратора ИС, примерно равна 30 000 руб.

**Расчет итоговых затрат.** В Таблице 1.2 приводятся все затраты на разработку и эксплуатацию информационной системы «Цифровая галерея» за 1 год работы.

Расчеты являются примерными и не учитывают многих нюансов, вроде амортизационных отчислений, ремонта и обслуживание компьютера и др.

Таблица 1.2

Суммарные затраты на разработку и эксплуатацию информационной системы «Цифровая галерея» за 1 год работы.

Затраты на разработку и эксплуатацию ИС	Стоимость, руб.
Заработная плата разработчика	63 700
Электроэнергия	5 415
Домен	500
Хостинг	2 400
Оргтехника	30 000
<b>Общие затраты:</b>	<b>102 015</b>

В результате расчетов затраты на создание составили 102 015 рублей. Практически одну треть стоимости составляет стоимость компьютерной техники. Но она как правило уже есть в организациях, даже если они не используют ее для автоматизации бизнес-процессов.

**Риски.** Внедрение АИС несет в себе некоторые риски. В частности, риски потери данных при сбое в программном обеспечении (сбой хостинга, атака на сайт, вирус). Для минимизации данных рисков необходимо регулярное копирование данных для их резервного хранения.

## Выводы по главе 1

Проведен анализ предметной области – художественная галерея Бирского филиала УУНиТ. С помощью диаграмм IDEF0 описаны существующие бизнес-процессы. На основе анализа мы делаем вывод о необходимости автоматизации существующих процессов предметной области.

На данном этапе проводился анализ существующих средств, инструментов разработки ПО. Для разработки frontend части приложения сделан выбор в пользу технологии построения компонентов пользовательского интерфейса на основе JavaScript библиотеки React. Backend часть приложения было решено реализовать с использованием возможностей headless-CMS Strapi, в основе которой лежит платформа NodeJS, а в качестве базы данных использовать SQLite. Приведены примерные расчеты затрат и рисков разработки и внедрения ИС «Цифровая галерея».

## ГЛАВА 2. ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

### 2.1. Разработка структурно-функциональной диаграммы по типу «Как должно быть?»

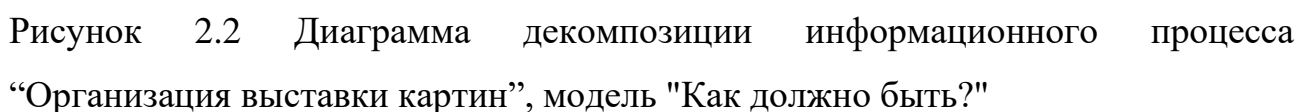
По ходу разработки бизнес-процесса была построена структурно-функциональная диаграмма «Как должно быть?». Был добавлен новый ресурс – информационная система (Рисунок 2.1). Добавлены новые ресурсы: механизм «Информационная система» и управление «Администратор информационной системы» (выделены красным), благодаря которым повысится эффективность организации.



Рисунок 2.1 Контекстная диаграмма информационного процесса «Организация выставки картин», модель «Как должно быть?»



1. Планирование выставки
2. Афиширование выставки
3. Подготовка к вернисажу



При анализе существующего процесса организации выставки была выявлена потребность заказчика в том, чтобы проведенные выставки были

доступны пользователям системы в качестве архивированных выставок, содержащих отчет о проведенном мероприятии.

Для этого необходимо рассмотреть процесс "Проведение выставки" (Рисунок 2.3).

Вход – "Организованная выставка". Это объект, который был получен в процессе "Организации выставки картин".

Выход – "Проведенная выставка", означает окончание проведения выставки. Информационная система в данном случае добавляет еще один объект – "Архивированная выставка". Администратор ИС по окончании выставки посредством панели управления контентом добавляет фотоотчет к существующей выставке, которая является уже архивированной в ИС по истечению времени проведения выставки.

Управление – нормативно-правовые акты, регулирующие выставочную деятельность. Они представляет собой правила и требования, регламентирующие правомерность организации выставки.

Механизм – художник, автор выставляемых произведений, организатор выставки, непосредственно участвующий в организации данной конкретной выставки, и информационная система.



Рисунок 2.3 Контекстная диаграмма информационного процесса “Проведение выставки”, модель "Как должно быть?"

## 2.2. Описание задач автоматизации информационных процессов поведенческими диаграммами в нотации UML

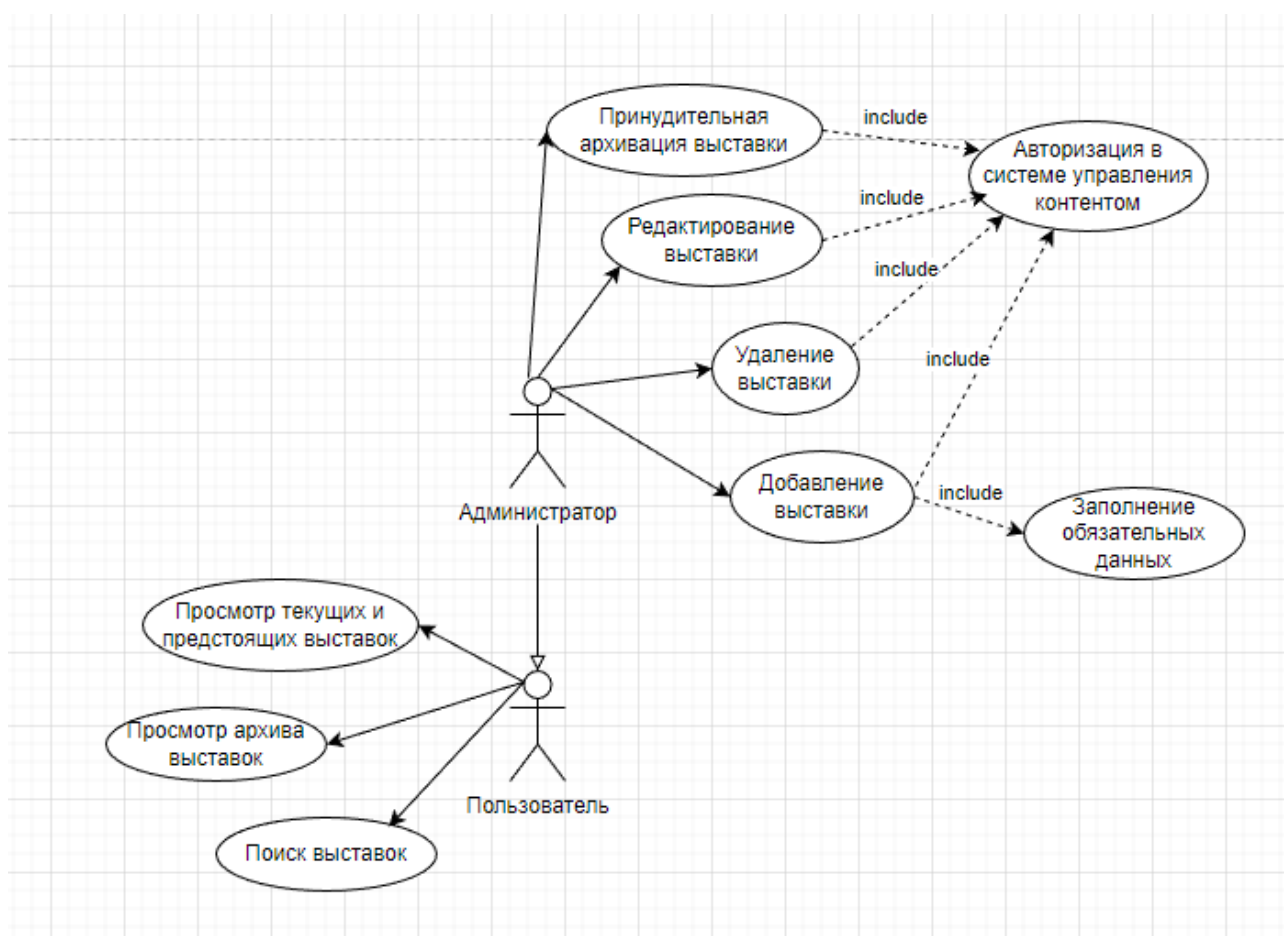


Рисунок 2.4 Диаграмма вариантов использования ИС «Цифровая галерея»

Диаграмма прецедентов или диаграмма вариантов использования – это диаграмма, на которой отражаются отношения между актерами и прецедентами системы, позволяющая описывать систему на концептуальном уровне.

Прецедент – это возможность системы, часть ее функциональности, с помощью которой актер может получить нужный ему результат. Прецедент является соответствием для отдельного сервиса системы и определяет варианты ее использования.

На рисунке 2.4 представлена диаграмма вариантов использования, где описаны действия, возможные для акторов «Администратор» и «Пользователь».

Наследование «Администратором» действий «Пользователя» показано через обобщение.

Рисунки 2.5 и 2.6 содержат диаграммы последовательностей для существующих прецедентов в информационной системе. Они уточняют диаграммы прецедентов, более детально описывают логику сценариев использования.

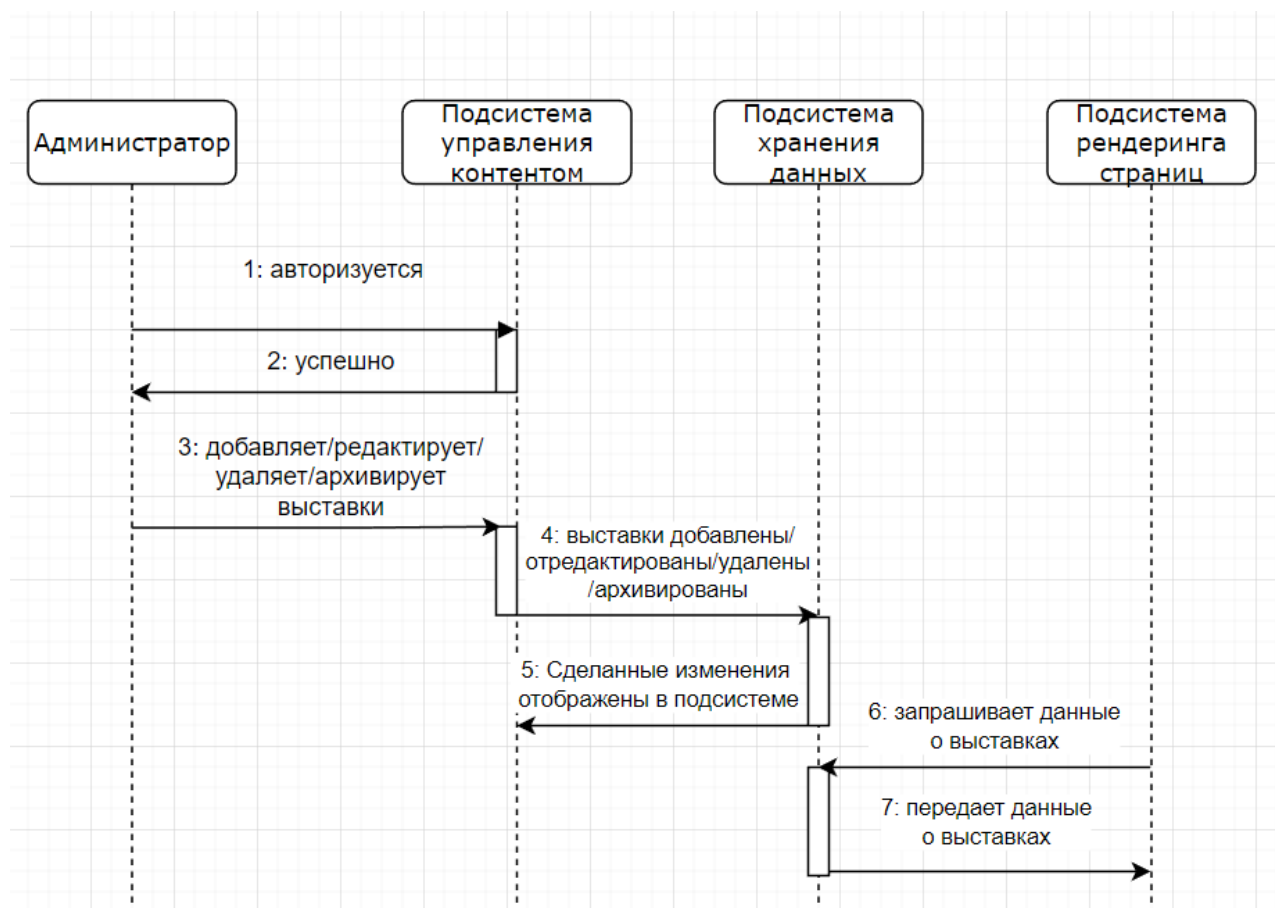


Рисунок 2.4 Диаграмма последовательностей для прецедентов актора «Администратор»



Рисунок 2.5 Диаграмма последовательностей для прецедента поиска актора «Пользователь»

Для моделирования динамических аспектов системы были использованы диаграммы состояний, представленные на рисунках 2.6 и 2.7. Данные диаграммы полезны при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

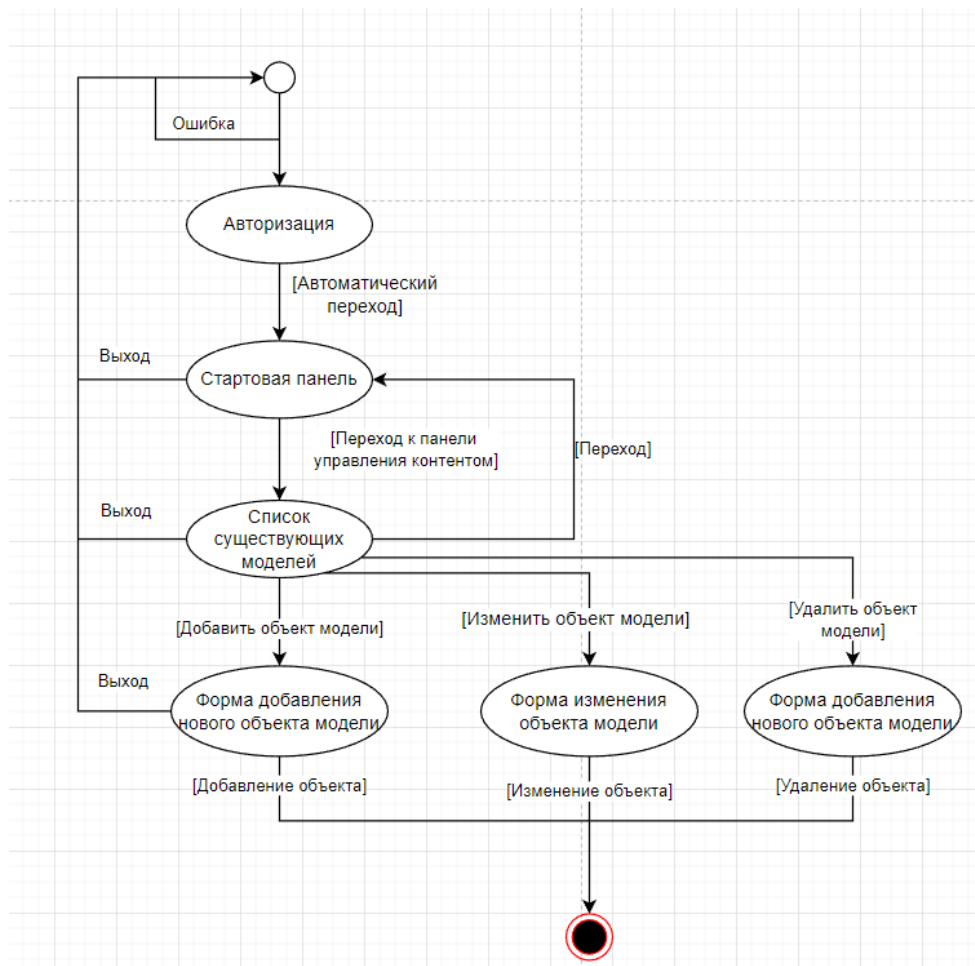


Рисунок 2.6 Диаграмма состояний подсистемы управления контентом для прецедентов Администратора.

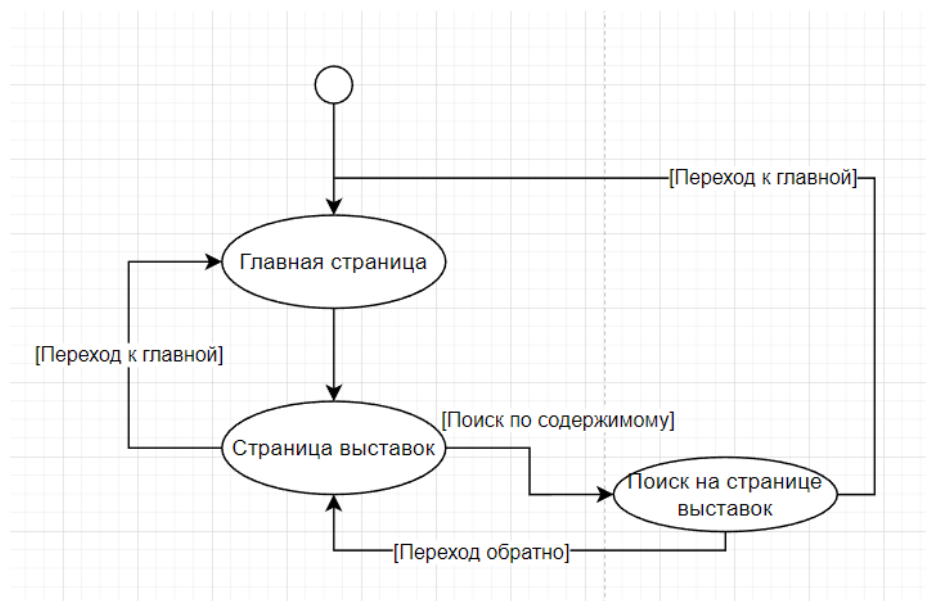


Рисунок 2.7 Диаграмма состояний системы для прецедента «Поиск» актора «Пользователь».

## **2.3. Формулирование технического задания**

**Общие сведения.** Полное наименование системы: Информационная система «Цифровая галерея». Краткое наименование: «Цифровая галерея».

**Назначение и цели создания системы.** Система предназначена для предоставления пользователю доступа к информации о выставках галереи посредством веб-сайта.

В рамках проекта автоматизируется информационно-аналитическая деятельность в следующих бизнес-процессах:

1. Хранение данных о текущих, прошедших и предстоящих выставках галереи;
2. Предоставление доступа к произведениям изобразительного искусства и выставкам, путем их публикации в информационной системе;
3. Афиширование выставок.

Предусматривается установка системы в пространствах, предназначенных для демонстрации изобразительного искусства.

ИС «Цифровая галерея» создается с целью:

- обеспечения удобного доступа к произведениям и выставкам галереи;
- создания единой системы хранения цифровых копий произведений изобразительного искусства, выставляющихся в галереях;
- предоставить возможность художникам осуществлять показ выполненных работ широкому кругу любителей искусства.

В результате создания информационной системы должны быть улучшены значения следующих показателей:

- посещаемость галереи;



- количество просмотров произведений изобразительного искусства галереи;
- доступность искусства и информации о выставках.

#### 2.4. Описание информационной модели комплекса задач (схема потоков данных)

Диаграмма DFD наглядно отображает течение информации в пределах процесса или системы. Для изображения входных и выходных данных, точек хранения информации и путей ее передвижения между источниками и пунктами доставки в таких диаграммах применяются стандартные фигуры, такие как прямоугольники и круги, а также стрелки и краткие текстовые метки. На рисунке 12 представлена контекстная диаграмма потоков данных в нотации Гейна-Сарсона. В ней «Пользователь» является внешней сущностью. При обращении к системе выступает как источник данных, а при получении данных как адресат.

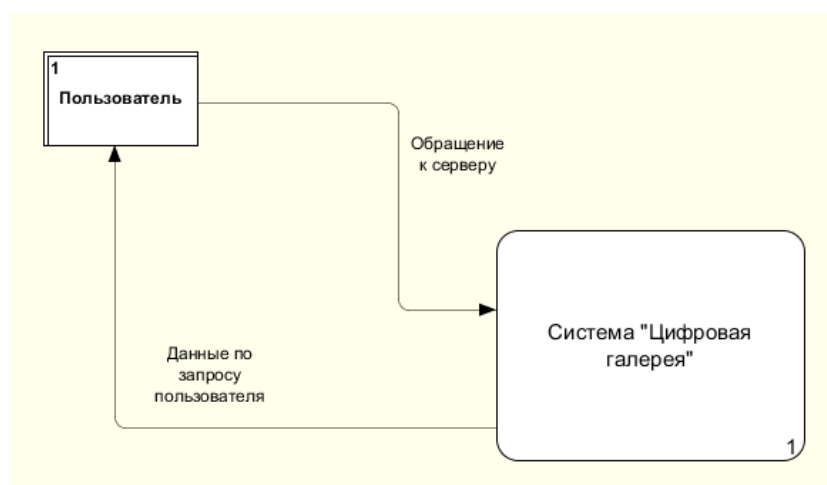


Рисунок 2.8 Контекстная диаграмма потоков данных в нотации Гейна-Сарсона

Декомпозиция контекстной диаграммы, что представлена на Рисунке 2.9, детализирует процессы, происходящие при взаимодействии «Пользователя» и «Системы». Наглядно демонстрируя поток данных в системе.

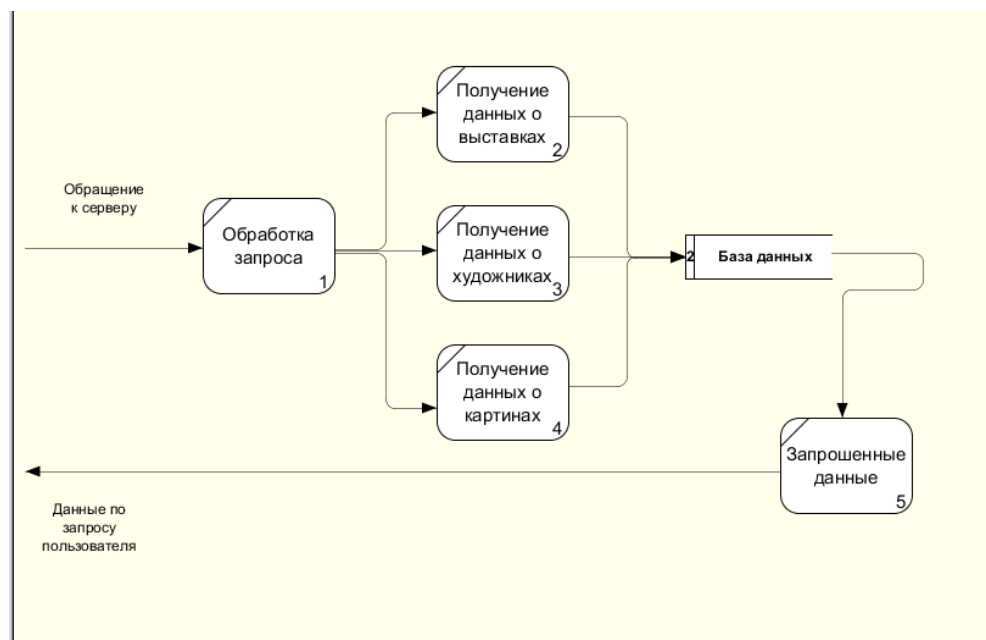


Рисунок 2.9 Декомпозиция контекстной диаграммы

## 2.5. Проектирование Базы Данных

Для создания UML-диаграмм использовалась программа в виде веб-приложения [app.diagrams.net](http://app.diagrams.net). На рисунке 2.10 изображена инфологическая ER-модель в нотации Питера Чена, включающая основные сущности и связи между ними.

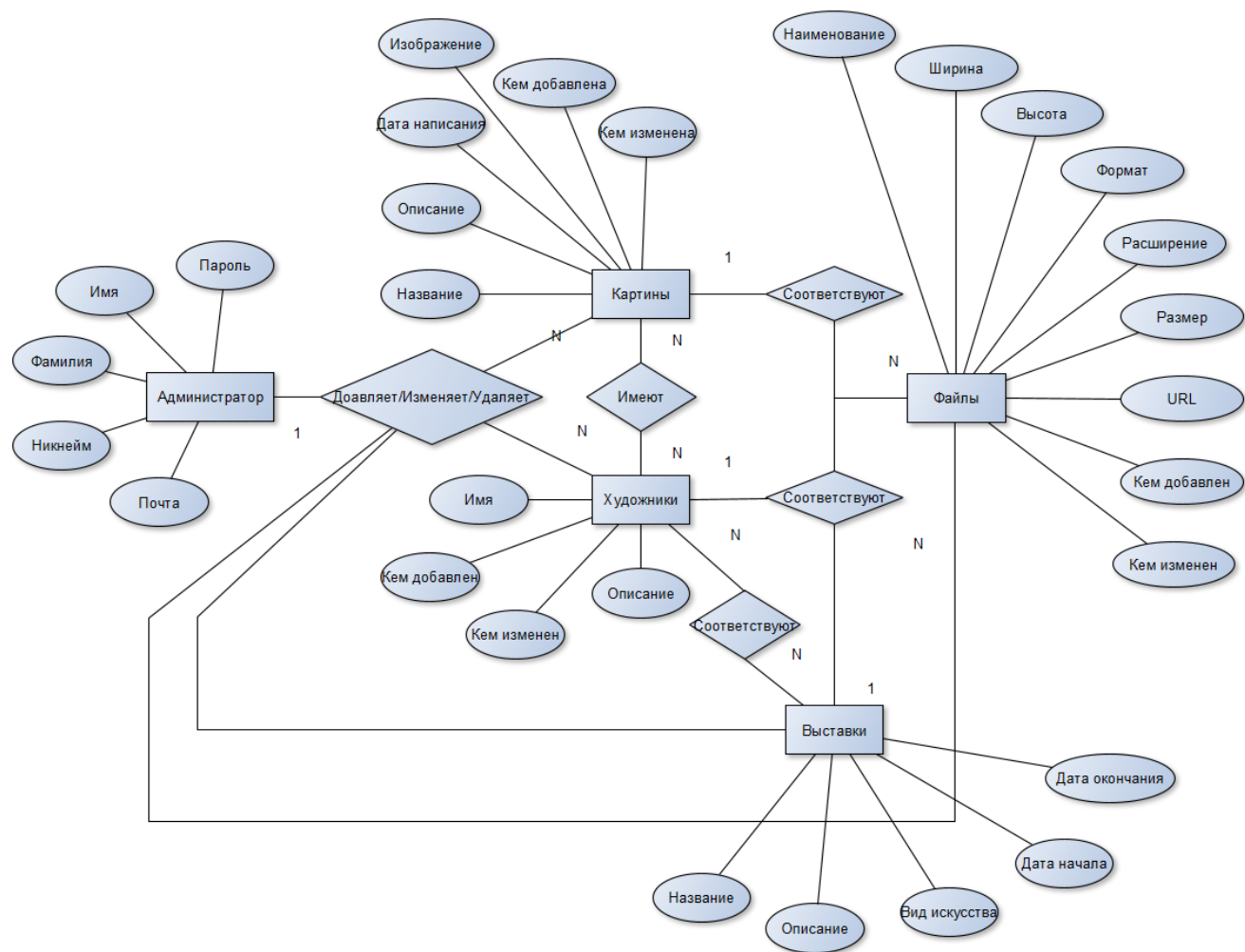


Рисунок 2.10 Инфологическая модель БД в нотации Питера Чена

На Рисунке 2.11 представлена даталогическая модель базы данных разрабатываемой системы.

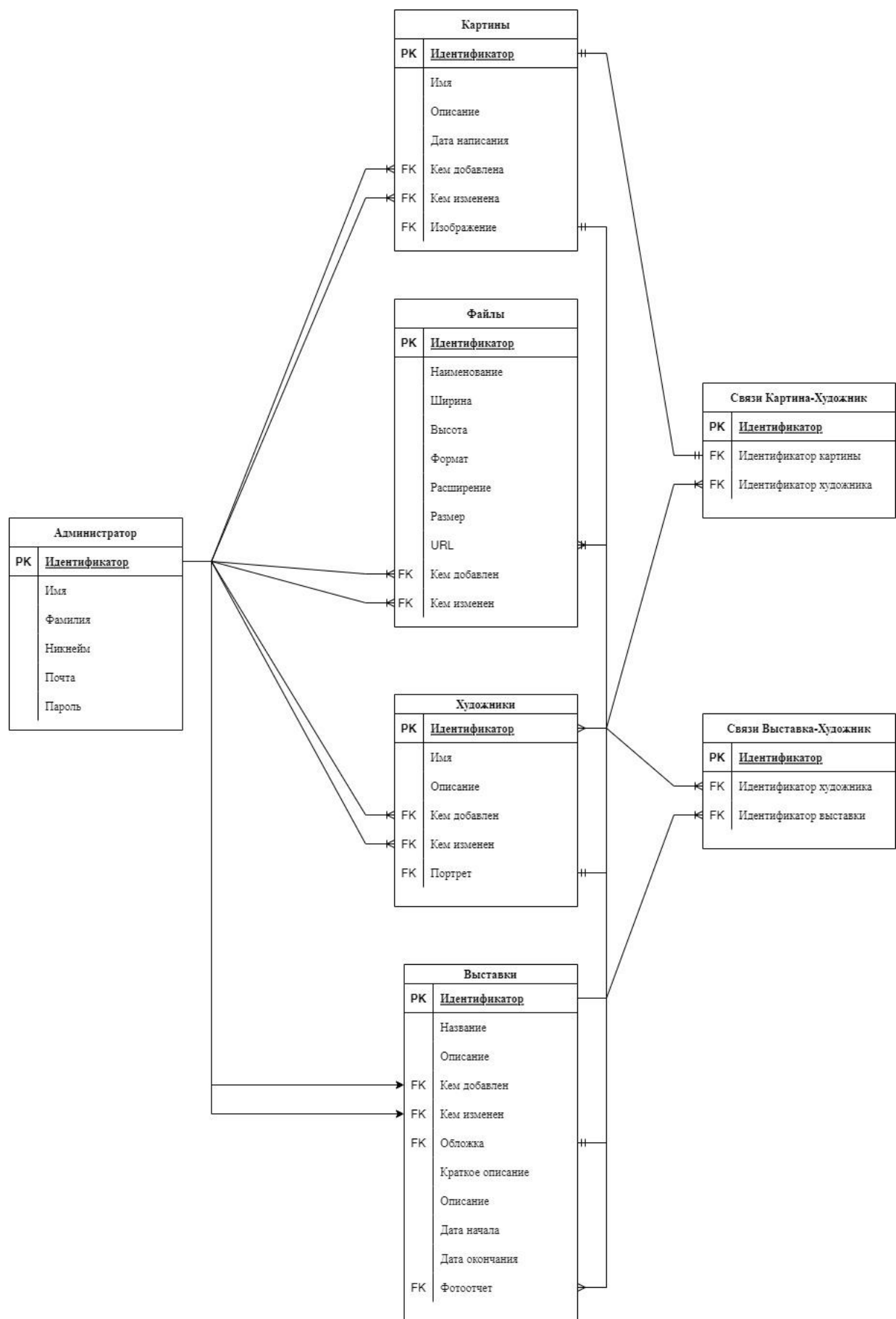


Рисунок 2.11 Даталогическая модель БД

## 2.6. Проектирование интерфейса

Интерфейс приложения является его ключевым звеном, так как он определяет, что увидит конечный пользователь приложения.

Веб-интерфейс – это веб-страница или совокупность веб-страниц, предоставляющая пользовательский интерфейс для взаимодействия с сервисом или устройством посредством протокола HTTP и веб-браузера. Веб-интерфейсы получили широкое распространение в связи с ростом популярности всемирной паутины (интернета), и соответственно повсеместного распространения веб-браузеров.

Одним из основных требований к веб-интерфейсам является их одинаковый внешний вид и одинаковая функциональность при работе в различных браузерах.

Согласованный с заказчиком веб-интерфейс пользователя должен содержать следующие элементы:

- Главный раздел, содержащую выдающиеся произведения галереи, карточки художников и последние выставки галереи;
- Раздел «О галерее» с информацией об истории галереи;
- Раздел «Ближайшие выставки»;
- Раздел «Архив выставок» с уже прошедшими выставками, которые попадают в него по истечении их срока проведения;
- Раздел «Коллекция» с подразделами категорий коллекций, включающих виды изобразительного искусства. Содержит карточки произведений искусства. Каждая карточка содержит соответствующую информацию о работе художника, внесенную при ее добавлении в систему или изменении;
- Раздел «Художники», в нем содержатся все художники, добавленные в систему. Аналогично «Коллекциям» карточка художника позволяет

перейти на его индивидуальную страницу. При этом возможно осуществление поиска по работам данного художника.

## **Выводы по главе 2**

В ходе работы над данной главой построены: диаграмма IDEF0 «Как должно быть», даталогическая и инфологическая модели базы данных, диаграмма прецедентов, диаграммы последовательностей, диаграммы состояний, а также, схема потоков данных. Сформулированы требования заказчика к информационной системе.

## ГЛАВА 3. РЕАЛИЗАЦИЯ КОМПОНЕНТОВ ИНФОРМАЦИОННОЙ СИСТЕМЫ

### 3.1. Реализация front-end части информационной системы

Для реализации пользовательского интерфейса была использована JavaScript библиотека React, что позволило разработать компоненты для их повторного использования и страницы, где эти компоненты используются. Такой подход к разработке называется SPA (Single Page Application). SPA – это одностраничное веб-приложение, которое загружается на одну HTML-страницу. Благодаря динамическому обновлению с помощью JavaScript, во время использования не нужно перезагружать или подгружать дополнительные страницы. На практике это означает, что пользователь видит в браузере весь основной контент, а при прокрутке или переходах на другие страницы, вместо полной перезагрузки нужные компоненты просто подгружаются.

Далее перечислены ключевые компоненты пользовательского интерфейса, листинг кода которых доступен в приложении 1:

1. Header. Отвечает за отрисовку «шапки» страницы приложения;
2. Footer. Ответственен за «подвал» страницы;
3. Artist, Picture, Exhibition. Используются для отображения миниатюр художников, коллекций и выставок;
4. SearchBarOnPage. Поисковая строка;
5. NavBar. Навигационное меню. Используется как в «шапке», так и в отдельных страницах приложения. В мобильной версии принимает вид меню, выпадающего сбоку;

Далее перечисляются страницы, использующие вышеперечисленные компоненты пользовательского интерфейса, листинг кода которых так же доступен в приложении 2:

1. Main;
2. Exhibitions, Paintings, Artists;
3. ExhibitionDetail, PictureDetail, ArtistDetail.

Точкой входа для выполнения веб-приложения является файл `index.js`. Данный файл отображает компонент `App.js`, который хранит в себе маршруты страниц приложения. Листинг кода представлен в разделе «Приложение 2».



### 3.2. Реализация back-end части информационной системы

Для реализации серверной составляющей приложения использовалась система управления контентом (headless-CMS) под названием Strapi, в которой создается API приложения. В качестве архитектурного стиля API был выбран REST.

Для url-адреса веб-приложения определены конечные точки (endpoint), к которым можно обращаться с помощью различных методов сетевых запросов. Данные запросы обрабатываются посредством методов контроллера конкретной модели объекта.

Для каждого контроллера модели определены следующие методы:

1. Find – получение всех объектов данной конечной точки:

```
async find(ctx) {  
    const { query } = ctx;  
  
    const { results, pagination } = await strapi.service(uid).find(query);  
  
    const sanitizedResults = await this.sanitizeOutput(results, ctx);  
  
    return this.transformResponse(sanitizedResults, { pagination });  
}
```

2. FindOne – получение одного объекта данной конечной точки:

```
async findOne(ctx) {  
    const { id } = ctx.params;  
  
    const { query } = ctx;  
  
    const entity = await strapi.service(uid).findOne(id, query);  
  
    const sanitizedEntity = await this.sanitizeOutput(entity, ctx);  
}
```

```

    return this.transformResponse(sanitizedEntity);
  }

```

### 3. Create – создание нового объекта:

```

async create(ctx) {
  const { query } = ctx.request;
  const { data, files } = parseBody(ctx);
  if (!isObject(data)) {
    throw new ValidationError('Missing "data" payload in the request body');
  }
  const sanitizedInputData = await this.sanitizeInput(data, ctx);
  const entity = await strapi
    .service(uid)
    .create({ ...query, data: sanitizedInputData, files });
  const sanitizedEntity = await this.sanitizeOutput(entity, ctx);
  return this.transformResponse(sanitizedEntity);
}

```

### 4. Update – обновление объекта.

```

async update(ctx) {
  const { id } = ctx.params;
  const { query } = ctx.request;
  const { data, files } = parseBody(ctx);
  if (!isObject(data)) {

```

```

    throw new ValidationError('Missing "data" payload in the request body');
  }

  const sanitizedInputData = await this.sanitizeInput(data, ctx);

  const entity = await strapi

    .service(uid)

    .update(id, { ...query, data: sanitizedInputData, files });

  const sanitizedEntity = await this.sanitizeOutput(entity, ctx);

  return this.transformResponse(sanitizedEntity);
}

```

#### 5. Delete –удаление объекта:

```

async delete(ctx) {

  const { id } = ctx.params;

  const { query } = ctx;

  const entity = await strapi.service(uid).delete(id, query);

  const sanitizedEntity = await this.sanitizeOutput(entity, ctx);

  return this.transformResponse(sanitizedEntity);

}

```

Для реализации архивации выставок по истечении их сроков проведения были использованы cron jobs. Это один из часто используемых инструментов для Unix-систем. Его используют для планирования выполнения команд на определённое время. Шаблон задания для cron job выглядит следующим образом:

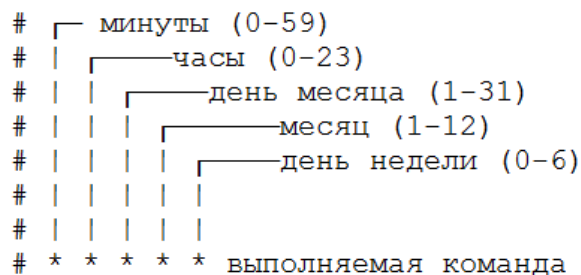


Рисунок 3.1 Шаблон cron задания

Для архивации выставок задано выражение: «00 00 00 \* \* \*». Это соответствует запуску задания ежедневно в полночь. Код самого cron job приведен в «Приложении 3». При выполнении задания проверяется, архивирована ли выставка, имеется ли у нее дата окончания, если она имеется, то проверяется, является ли она валидной. Таким образом, обрабатываются все возможные исключительные ситуации, которые могут возникнуть при эксплуатации информационной системы.

### 3.3. Реализация базы данных

Для хранения данных в системе используется встраиваемая СУБД SQLite. Подключение к БД происходит через конфигурационный файл database.js.

Фрагмент кода database.js:

```
module.exports = ({ env }) => ({  
  
  connection: {  
  
    client: 'sqlite',  
  
    connection: {  
  
      filename: path.join(__dirname, '..', env('DATABASE_FILENAME',  
'./tmp/data.db')),  
  
    },  
  
    useNullAsDefault: true,  
  
  },  
  
});
```

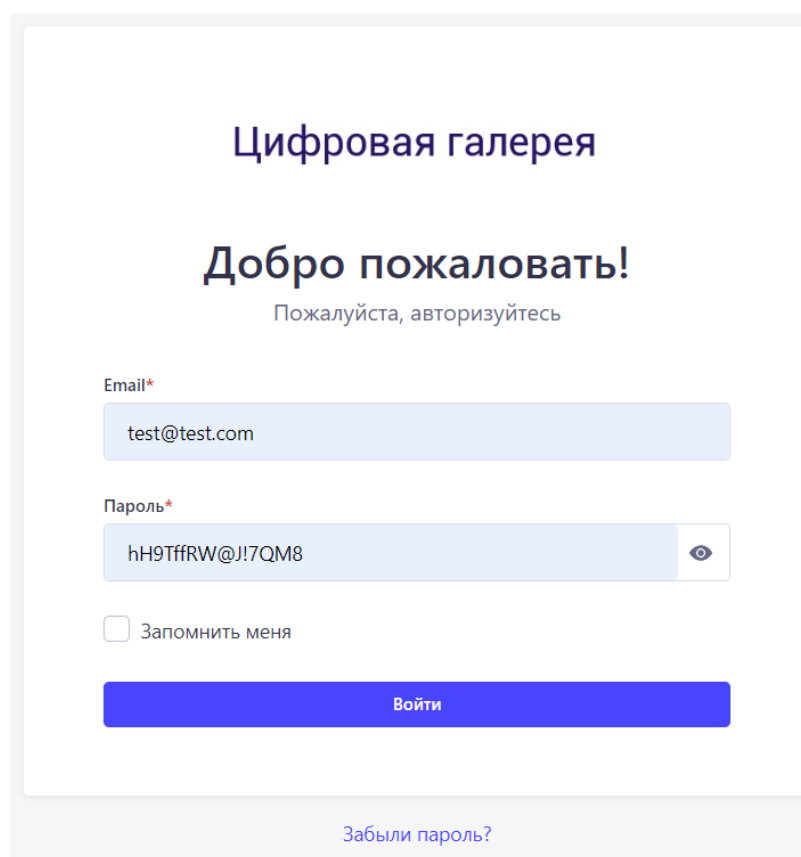
Запросы к БД формируются с помощью генератора SQL запросов Knex.js.

### 3.4. Руководство использования

После реализации всех компонентов информационной системы и развертывания ее на сервере пользователь может приступить к её эксплуатации. Для упрощения использования системы к программному обеспечению поставляется руководство использования. Оно подразделяется на:

- 1) Руководство администратора системы управления контентом
- 2) Руководство пользователя

**Руководство администратора.** Панель управления доступна по адресу <http://localhost:1337/>. После перехода по ссылке администратор направляется на страницу авторизации в системе.



The image shows a login interface for a system titled "Цифровая галерея" (Digital Gallery). The main heading is "Добро пожаловать!" (Welcome!). Below it, a subtitle says "Пожалуйста, авторизуйтесь" (Please, log in). The form contains two input fields: "Email\*" with the value "test@test.com" and "Пароль\*" (Password) with the value "hN9TffRW@J!7QM8". There is a checkbox labeled "Запомнить меня" (Remember me) which is unchecked. A blue button labeled "Войти" (Log in) is at the bottom of the form. At the very bottom of the page, there is a link "Забыли пароль?" (Forgot password?).

Рисунок 3.2 Страница авторизации администратора

Администратор может изменить свои данные для входа, перейдя к профилю через нижнюю кнопку в левой панели. Форма с данными администратора представлена на Рисунке 3.3.

Тест Тестов

Сохранить

Профиль

Имя\*

Тест

Фамилия

Тестов

Email\*

test@test.com

Имя пользователя

test@test.com

Сменить пароль

Нынешний пароль

Пароль

Подтверждение пароля

Рисунок 3.3. Форма изменения данных для авторизации администратора

Если администратор успешно авторизовался, то он получает доступ к типам коллекций, которыми являются Выставки, Картины, Художники, Статьи и Картины на главной (Рисунок 3.2). Каждый тип имеет свою коллекцию записей, соответствующих типу. Присутствует возможность фильтровать коллекцию по названию и другим полям записей посредством поиска, а также предусмотрена возможность сортировки путем клика на нужный столбец.

Контент

типы коллекций

Выставки

Картины

Картины на главной

Статьи

Художники

одиночные типы

Выставки

12 объектов найдено

Фильтры

5 выбрано

ID	Название	Дата начала	Дата завершения	Архивирована	Состояние
12	«Цвет странствий»: выставка художника Але...	10.06.2021	10.09.2021	true	Опубликован
11	Выставка памяти Усманова Салавата Мудр...	12.09.2022	-	-	Опубликован
10	Отчетная выставка ТСХ РБ	28.12.2022	-	-	Опубликован
9	Проверка на валидность даты	-	05.04.2023	true	Опубликован
8	Будет архивировано 03 апреля	-	02.04.2023	true	Опубликован
7	Тест	10.02.2023	25.03.2023	true	Опубликован
6	Тестовая выставка для проверки архивации	29.03.2023	01.04.2023	true	Опубликован

Рисунок 3.4 Страница с коллекциями

Далее администратор может осуществлять различные операции над данными коллекциями: добавлять записи в коллекции, просматривать их, изменять или удалять. На примере типа коллекции «Выставки» рассмотрим данные операции.

← Назад

### Создать запись

API ID : exhibition

✓ Опубликовать Сохранить

Название выставки\*

Обложка

Click to add an asset or drag and drop one in this area

Краткое описание

Например, время начала выставки

Отображается на миниатюре выставки

Вид выставленного искусства\*

Дата начала

Пример: 31.12.2022

Дата завершения

Пример: 31.12.2022

Художники

Выберите участвующих художников

Фотоотчет с мероприятия

Редистрирование Черновая версия

ИНФОРМАЦИЯ

Создано	сейчас
по	-
Последнее обновление	сейчас
по	-

Рисунок 3.5 Форма добавления записи в коллекцию «Выставки»

В форме представлены поля данных разного типа. При добавлении в поле типа «Изображение» используется подход drag&drop, позволяющий перетащить картинку на указанную область, и добавить ее в поле. Далее пользователю открывается модальное окно для подтверждения добавления изображения.

Поле «Архивирована ли выставка?» позволяет сделать выставку архивированной, по умолчанию она таковой не является. В системе присутствует алгоритм автоматической архивации выставки по мере истечения срока проведения выставки. Таким образом, администратору не нужно архивировать прошедшую выставку вручную. Однако, если выставка вновь стала актуальной, то уже архивированную выставку можно разархивировать, изменив значение данного поля с «Да» на «Нет». На стороне пользовательской части приложения архивированные выставки хранятся в отдельном разделе.

В приложении действует система публикаций. Запись может быть добавлена в коллекцию, но не опубликована. Это удобно, так как администратор



может заранее добавить запись, но опубликовать ее потом. Также, доступен обратный процесс снятия с публикации или же удаления записи навсегда. При изменении уже существующей записи, если запись уже опубликована, повторная публикация не требуется. После публикации выставки она будет отображена в коллекции «Выставки». И одновременно станет доступна пользователям клиентской части приложения «Цифровая галерея» в разделе «Выставки».

**Руководство пользователя.** Приложение для пользователя доступно по адресу <http://localhost:3000/>. После перехода по ссылке отображается главная страница веб-приложения «Цифровая галерея».

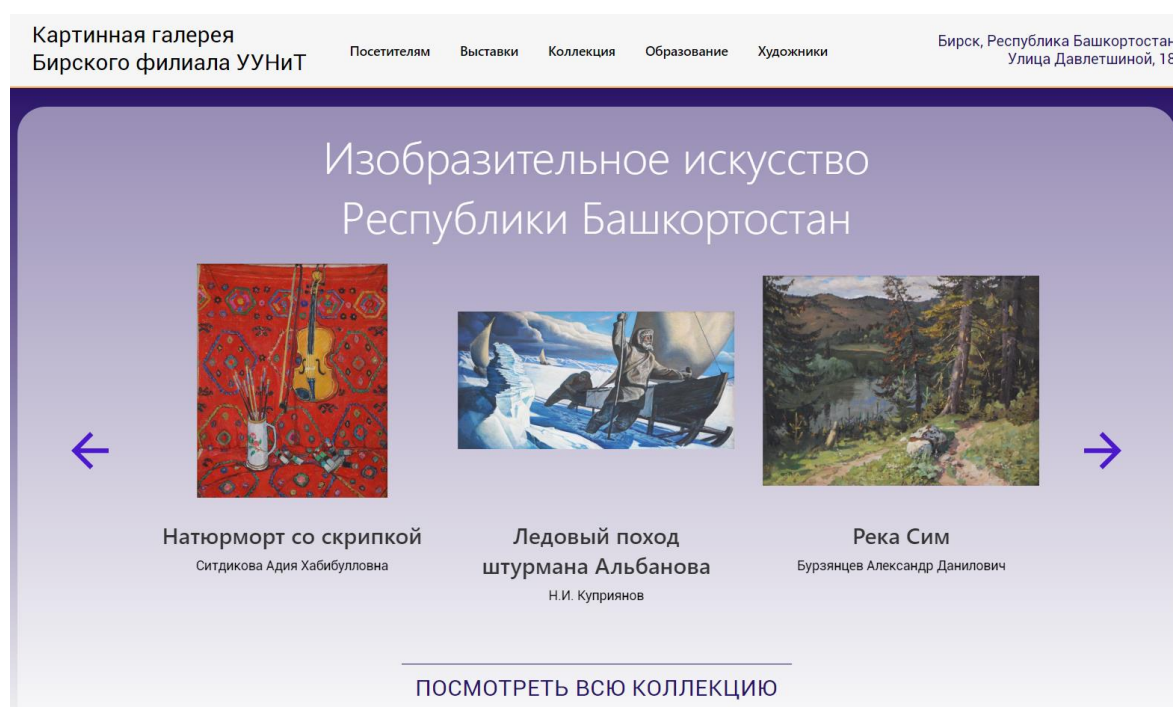


Рисунок 3.6 Главная страница веб-приложения «Цифровая галерея»

В верхней части страницы представлено меню, содержащее разделы приложения. Содержание разделов наполняется администратором. Пользователь имеет возможность только их просматривать. В некоторых из них предусмотрен поиск, так как записей может быть большое количество (Рисунок 3.7). Раздел «Коллекция» содержит в себе подразделы, позволяющие фильтровать произведения по видам изобразительного искусства.

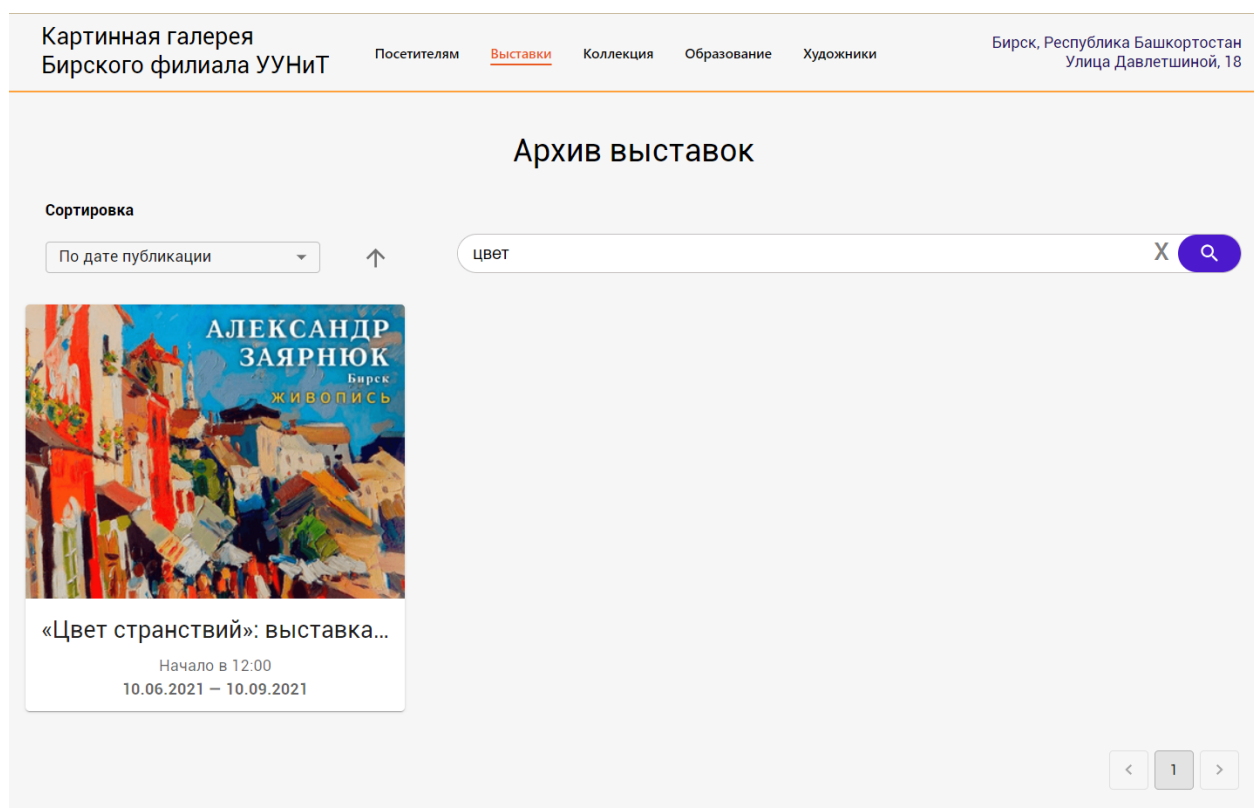


Рисунок 3.7 Поиск совпадений по слову «цвет» в разделе «Архив выставок»

Для каждой выставки/художника/картины реализована своя страница. К примеру, приводится страница выставки из поисковой выдачи на Рисунке 3.8.

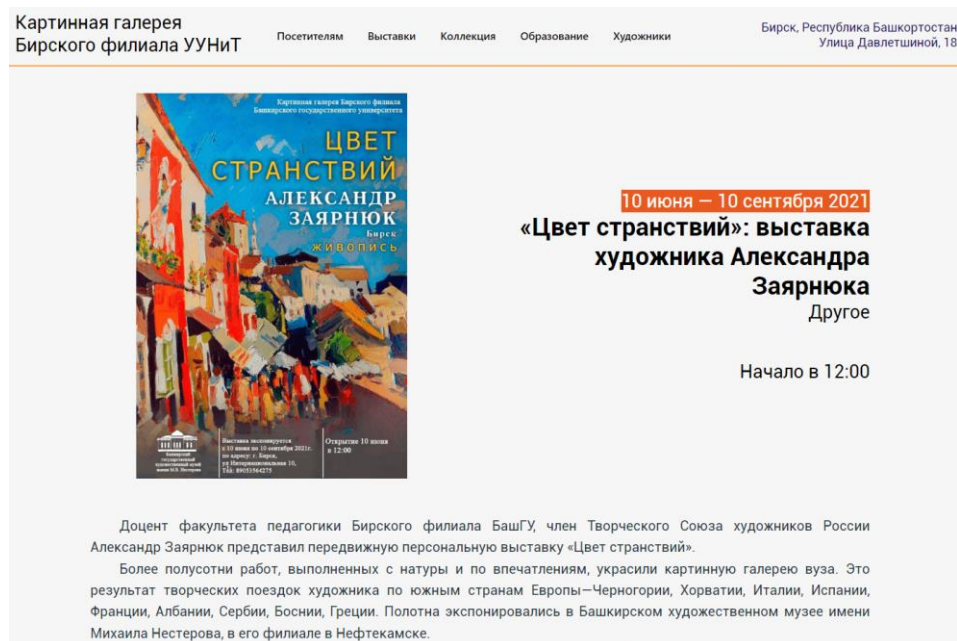


Рисунок 3.8 Страница выставки

Стоит отметить, что для веб-приложения «Цифровая галерея» предусмотрена мобильная версия приложения. Таким образом, страницы приложения будут одинаково удобными как для пользователей персональных компьютеров, так и мобильных устройств, таких как смартфоны и планшеты.

### **Выводы по главе 3**

В данной главе выпускной квалификационной работы описаны реализации клиентской и серверной составляющей информационной системы «Цифровая галерея» и приведена реализация подключения к базе данных.

Описана документация по использованию приложения для ролей администратора и пользователя.

## ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа посвящена разработке информационной системы для художественной галереи. Было проведено исследование бизнес-процессов предметной области, поставлены задачи автоматизации обработки и хранения информации об объектах галереи. Изучение объекта исследования отражено в виде спроектированных диаграмм в нотации IDEF0. Этап проектирования информационной системы сопровождался созданием диаграмм прецедентов, состояний, последовательностей, схем потоков данных в нотации DFD и моделей представления баз данных.

Для разработки были выбраны технологии, подходящие под специфику данного проекта. Использовался язык JavaScript. Реализация frontend части приложения выполнялось с использованием технологии построения компонентов пользовательского интерфейса на основе JavaScript библиотеки React. Предусмотрена мобильная версия приложения, поэтому интерфейс приложения является адаптивным. Backend часть приложения реализована с использованием возможностей технологии Strapi, функционирующей на платформе NodeJS. База данных функционирует посредством СУБД SQLite. Написано руководство по использованию информационной системы как для пользователя, так и для администратора системы.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Правовое регулирование отношений в сфере выставочно-ярмарочной деятельности [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/pravovoe-regulirovanie-otnosheniy-v-sfere-vystavочно-yarmarочноy-deyatelnosti>, свободный. – (Дата обращения: 11.01.2023).
2. Strapi v4 developer documentation [Электронный ресурс]. – Режим доступа: <https://docs.strapi.io/developer-docs/latest/getting-started/introduction.html>, свободный. – (Дата обращения: 11.02.2023).
3. React. JavaScript-библиотека для создания пользовательских интерфейсов [Электронный ресурс]. – Режим доступа: <https://ru.reactjs.org/docs/getting-started.html>, свободный. – (Дата обращения: 16.01.2023).
4. Бибо Бер , Кац Иегуда jQuery. Подробное руководство по продвинутому JavaScript; Символ-плюс - М., 2017. - 624 с.
5. IBM. Documentation. UNIX cron format [Электронный ресурс]. – Режим доступа <https://www.ibm.com/docs/en/db2/11.5?topic=task-unix-cron-format>, свободный. – (Дата обращения: 16.02.2023).
6. Дронов Владимир JavaScript и AJAX в Web-дизайне; БХВ-Петербург - М., 2015. - 736 с.
7. Дронов Владимир JavaScript. Народные советы; БХВ-Петербург - М., 2016. - 458 с.
8. Дунаев Вадим JavaScript. Самоучитель; Питер - М., 2015. - 400 с.
9. Дунаев Вадим HTML, скрипты и стили; БХВ-Петербург - М., 2015. - 816 с.
10. Изучаем Node.js; Питер - М., 2015. - 400 с.
11. Клименко Роман Веб-мастеринг на 100%; Питер - М., 2015. - 920 с.

12. Климов Александр JavaScript на примерах; БХВ-Петербург - М., 2017. - 812 с.
13. Крокфорд Д. JavaScript. Сильные стороны; Питер - М., 2016. - 262 с.
14. Лазаро Исси Коэн, Джозеф Исси Коэн Полный справочник по HTML, CSS и JavaScript; ЭКОМ Паблишерз - М., 2016. - 311 с.
15. Макфарланд Дэвид JavaScript. Подробное руководство; Эксмо - М., 2015. - 608 с.
16. Никсон Робин Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5; Питер - М., 2016. - 768 с.
17. IBM. What is a REST API? [Электронный ресурс]. – Режим доступа <https://www.ibm.com/docs/en/db2/11.5?topic=task-unix-cron-format>, свободный. – (Дата обращения: 16.02.2023).
18. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения; Библиотека программиста (Питер) - М., 2018. - 410 с.
19. Прохоренок Н. А. Python. Самое необходимое; БХВ-Петербург - М., 2015. - 416 с.
20. Резиг Джон , Бибо Беэр Секреты JavaScript ниндзя; Вильямс - М., 2015. - 416 с.
21. Роббинс Дженнифер HTML5, CSS3 и JavaScript. Исчерпывающее руководство (+ DVD-ROM); Эксмо - М., 2017. - 528 с.
22. Фримен Адам jQuery для профессионалов; Вильямс - М., 2015. - 960 с.
23. Херман Дэвид Сила JavaScript. 68 способов эффективного использования JS; Питер - М., 2015. - 952 с.
24. Чаффер Д. Изучаем jQuery 1.3. Эффективная веб-разработка на JavaScript; Символ-плюс - М., 2015. - 391 с.

25. Knex Query Builder [Электронный ресурс]. – Режим доступа: <https://knexjs.org/guide/query-builder.html>, свободный. – (Дата обращения: 24.02.2023).
26. Andy, Harris HTML, XHTML and CSS All-In-One For Dummies® / Andy Harris. - Москва: Наука, 2014. - 173 с.
27. Ben, Henick HTML & CSS – The Good Parts / Ben Henick. - Москва: СИНТЕГ, 2013. - 350 с.
28. Ed, Tittel HTML, XHTML & CSS For Dummies® / Ed Tittel. - Москва: Гостехиздат, 2012. - 416 с.
29. Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. - М.: Триумф, 2014. - 464 с.
30. Общие сведения о веб-приложениях [Электронный ресурс]. – Режим доступа: <https://helpx.adobe.com/ru/dreamweaver/using/web-applications.html>, свободный. – (Дата обращения: 25.01.2023).
31. Что такое веб-приложение простыми словами: виды и алгоритм разработки [Электронный ресурс]. – Режим доступа: <https://vc.ru/dev/397220-что-такое-veb-prilozhenie-prostymi-slovami-vidy-i-algoritm-razrabotki>, свободный. – (Дата обращения: 26.01.2023).
32. A Complete Guide to CSS Grid [Электронный ресурс]. – Режим доступа: <https://css-tricks.com/snippets/css/complete-guide-grid/>, свободный. – (Дата обращения: 26.03.2023).
33. Mdn web docs. JavaScript [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>, свободный. – (Дата обращения: 26.03.2023).

34. Mdn web docs. CSS: Cascading Style Sheets [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/CSS>, свободный. – (Дата обращения: 22.02.2023).
35. Mdn web docs. HTML: HyperText Markup Language [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/HTML>, свободный. – (Дата обращения: 24.03.2023).
36. Geeks for geeks. React Suite Dropdown Component [Электронный ресурс]. – Режим доступа: <https://www.geeksforgeeks.org/react-suite-dropdown-component/>, свободный. – (Дата обращения: 24.03.2023).
37. Strapi. Forum [Электронный ресурс]. – Режим доступа: <https://forum.strapi.io/>, свободный. – (Дата обращения: 27.03.2023).
38. Habr. React: лучшие практики [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/541320/>, свободный. – (Дата обращения: 17.03.2023).
39. Марк Майерс. A Smarter Way to Learn JavaScript: The New Tech-Assisted Approach that Requires Half the Effort JS; CreateSpace Independent Publishing Platform; F First Edition Used - М., 2014. - 254 с.
40. Хоган, Уоррен, Уэбер: Книга веб-программиста: секреты профессиональной разработки веб-сайтов; Питер – М., 2013. – 288с.
41. 10 правил проектирования интерфейсов, которые нельзя нарушать [Электронный ресурс]. – Режим доступа: <https://ux.pub/editorial/10-pravil-proiektirovaniia-intierfieisov-kotoryie-nielzia-narushat-21m6>, свободный. – (Дата обращения: 17.01.2023).



## ПРИЛОЖЕНИЕ

### Приложение 1

1. ...Components/Header.js:

```
import './header.css'
import SearchBar from '../components/SearchBar'
import {useNavigate} from 'react-router-dom'

function Header({searchValue, setSearchValue}) {

  const navigate = useNavigate()

  return (
    <header>
      <div className='wrapper'>
        <div className='logo' onClick={() => navigate('/')}>
          <h2>Цифровая галерея</h2>
        </div>
        <SearchBar searchValue={searchValue} setSearchValue={setSearchValue}/>
      </div>
    </header>
  )
}
```

export default Header

2. ...Components/Exhibition.js:

```
import './exhibition.css';
import { useNavigate } from 'react-router-dom';
import { Divider, Card, CardMedia, CardContent, Typography, CardActions, Button } from "@mui/material";
import { useEffect } from 'react';

function Exhibition({ cardTitle, about, image, id, startDate, finishDate, onClick = () => {} }) {

  const navigate = useNavigate();
  useEffect(() => {
    console.log(image.data);
  });
  return (
    <div className='exhibition' onClick={() => {
      navigate(`/exhibition/${id}`, {
        state: { id }
      }); onClick()
    }}>
      <Card className="exhibitionCard" sx={{ height: 430 }}>
        <CardMedia
          sx={{(image === '/noImage.svg') ? {height: 300,
            transform: 'scale(0.5)',
            backgroundSize: 'contain',
            WebkitBackgroundSize: 'none'
          } : {height: 300}}
          image={image}
          title={cardTitle}
        />
        <CardContent >
          <Typography noWrap gutterBottom variant="h5" component="div">
```

```

        {cardTitle}
      </Typography>
      <Typography variant="body1" color="text.secondary" noWrap>
        {about}
      </Typography>
      <Typography sx={{ fontWeight: 'bold' }} variant="body1" color="text.secondary">
        {startDate} {finishDate ? ` — ${finishDate}` : ""}
      </Typography>
    </CardContent>
    { /* <CardActions>
      <Button size="small">Узнать подробнее</Button>
    </CardActions> */ }
  </Card>

</div>
)
}

```

export default Exhibition;

3. ...Components/Footer.js:

```

import './footerStyles.js'
import React from "react";
import {
  Box,
  Container,
  Row,
  Column,
  FooterLink,
  Info,
  Heading,
  Img
} from "./footerStyles";
import LocalPhoneOutlinedIcon from '@mui/icons-material/LocalPhoneOutlined';
import PlaceOutlinedIcon from '@mui/icons-material/PlaceOutlined';

const Footer = () => {
  return (
    // <img src='/logo.svg' width='200px'></img>
    <Box>
      { /* <h1 style={{ color: "green",
        textAlign: "center",
        marginTop: "-50px" }}>

</h1> */ }
      <Container>
        <Row>
          { /* <Column>
            <img src='/logo.svg' width='200px'></img>
          </Column> */ }
          <Column>
            <Heading>Посетителям</Heading>
            <FooterLink to="/visit">Экскурсии</FooterLink>
            <FooterLink to="/about">О галерее</FooterLink>
          </Column>
          <Column>
            <Heading>Выставки</Heading>
            <FooterLink to="/exhibitions">Ближайшие выставки</FooterLink>
            <FooterLink to="/exhibitionsArchive">Архив выставок</FooterLink>
          </Column>
          <Column>

```

```

        <Heading>Коллекция</Heading>
        <FooterLink to="/paintings">Вся коллекция</FooterLink>
        <FooterLink to="#">Живопись</FooterLink>
        <FooterLink to="#">Графика</FooterLink>
        <FooterLink to="#">Декоративно-прикладное искусство</FooterLink>
      </Column>
    </Column>
    <Heading>Образование</Heading>
    <FooterLink to="/education">Образовательные услуги</FooterLink>
    <FooterLink to="#">Запись</FooterLink>
    <FooterLink to="#">Расписание</FooterLink>
  </Column>
</Row>
<Row>
  <Column>
    <Img src="/logo.jpg"></Img>
  </Column>
  <Column>
    <Heading>Контакты</Heading>
    <Info>
      Приемная галереи
    </Info>
    <FooterLink to="tel:+7(347)2999999">
      +7-(347)-2-99-99-99
    </FooterLink>
    <FooterLink to="https://yandex.ru/maps/org/birskiy_filial_uunit/1210611238/?ll=55.526310%2C55.411783&z=17.41" target="_blank">
      Бирск, Республика Башкортостан<br></br>Интернациональная улица, 10
    </FooterLink>
  </Column>
  <Column>
    <Heading>Социальные сети</Heading>
    <FooterLink to="https://vk.com/birskbgv" target="_blank">
      <i className="fab fa-facebook-f">
        <span style={{ marginLeft: "10px" }}>
          ВКонтакте
        </span>
      </i>
    </FooterLink>
    <FooterLink to="#">
      <i className="fab fa-youtube">
        <span style={{ marginLeft: "10px" }}>
          Youtube
        </span>
      </i>
    </FooterLink>
  </Column>
</Column>
</Row>
</Container>
</Box>
);
};
export default Footer;

```

```

4. App.js
import './App.css';
import { Routes, Route } from "react-router-dom"
import Home from './pages/Home';
import PictureDetail from './pages/PictureDetail';

```

```

import Header from './components/Header';
import Search from './pages/Search';
import { useState } from 'react';

function App() {
  const [searchValue, setSearchValue] = useState("")
  return (
    <div className="App">
      <Header searchValue={searchValue} setSearchValue={setSearchValue}/>
      <Routes>
        <Route path="/" exact element={<Home searchValue = {searchValue}
setSearchValue={setSearchValue}/>}/>
        <Route path="/detail/:id" element={<PictureDetail />}/>
        <Route path="/search/:query" element={<Search searchValue={searchValue}/>}/>
      </Routes>
    </div>
  );
}
export default App;

```

### 1. Pages/Main.js

```
import { useEffect, useState, useRef } from "react";
import "../main.css";
import "../ckeditor.css"

import axios from "axios";
import { Link } from 'react-router-dom';
import { Swiper, SwiperSlide } from "swiper/react";
import { Navigation, Pagination, Autoplay, A11y } from 'swiper';
import { useSwiper } from 'swiper/react';
import ArrowForwardOutlinedIcon from '@mui/icons-material/ArrowForwardOutlined';
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined';
import Artist from "../components/Artist";
import Exhibition from "../components/Exhibition";
import { textAlign } from "@mui/system";

function Main({ searchValue, setSearchValue }) {
  useEffect(() => {
    setSearchValue("");
    window.scrollTo(0, 0);
  }, []);

  const [slides, setSlides] = useState([]);
  const [articles, setArticles] = useState();
  const rootURL = "http://localhost:1337";
  const URL = "http://localhost:1337/api/slides/?populate[picture][populate]=*";
  const articlesURL = "http://localhost:1337/api/articles/3";

  const fetchData = async (setHook, URL) => {
    let data;
    const result = await axios.get(URL);
    data = result.data.data;
    setHook(data);
  };

  useEffect(() => {
    fetchData(setSlides, URL);
    fetchData(setArticles, articlesURL);
    // parseHtml(articles?.attributes?.Content);
    //console.log(typeof articles?.attributes?.Content);
  }, []);

  //console.log(JSON.stringify(slides[0]?.attributes?.picture?.data?.id));
  const ApiAddress = axios.create({
    baseURL: "http://localhost:1337/api/exhibitions"
  });

  const [exhibitionApiData, setExhibitionApiData] = useState([]);
  const imageURL = "http://localhost:1337";
  useEffect(() => {
    ApiAddress.get(`?pagination[page]=1&pagination[pageSize]=6&filters[${$or}][0][isArchived][${$eq}]=false&filters[${$or}][1][isArchived][${$null}]=true&populate=*&sort=createdAt:asc`)
      .then((response) => setExhibitionApiData(response.data.data))
      .catch((err) => console.log(err));
  }, []);

  const swiperRef = useRef();
  const swiperHeroRef = useRef();

  const [artistApiData, setArtistApiData] = useState([]);
```

```

const ArtistApiAddress = axios.create({
  baseURL: "http://localhost:1337/api/artists"
});
useEffect(() => {
  //ArtistApiAddress.get(`?pagination[page]=1&pagination[pageSize]=8&populate=*`)

  ArtistApiAddress.get(`?filters[onMainPage][$eq]=true&populate=*`)
    .then((response) => setArtistApiData(response.data.data))
    .catch((err) => console.log(err));
}, []);

return (
  <div className="main">
    <section className="hero">
      <div className="heroContentContainer wrapper">
        <h1>Изобразительное искусство <br></br>Республики Башкортостан</h1>
        <div className="sliderWrapper sliderWrapperHero wrapper">
          <button className="sliderBtn sliderBtnPrev" onClick={() =>
            swiperHeroRef.current.slidePrev()}><ArrowBackOutlinedIcon /></button>
          <Swiper className="slider sliderHero"
            modules={[Navigation, Pagination, A11y, Autoplay]}
            spaceBetween={30}
            centeredSlides
            loop
            slidesPerView={3}
            onSlideChange={() => console.log("slide change")}
            onSwiper={(swiper) => {
              swiperHeroRef.current = swiper;
            }}
            // pagination={{ clickable: true }}
            autoplay={{ delay: 5000 }}
            >
            </* <SwiperSlide className="SlideContainer" style={{ backgroundImage: `url(${slide})` }}>
            </SwiperSlide> */>
            </* достаём слайдеры из api strapi, созданные администратором */>
            {slides?.map((slide) => {
              let temp = slide.attributes?.picture?.data?.attributes?.image?.data[0]?.attributes?.formats?.large?.url ??
                slide.attributes?.picture?.data?.attributes?.image?.data[0]?.attributes?.formats?.medium?.url ??
                slide.attributes?.picture?.data?.attributes?.image?.data[0]?.attributes?.formats?.small?.url ??
                slide.attributes?.picture?.data?.attributes?.image?.data[0]?.attributes?.formats?.thumbnail?.url ??
                "/noImage.svg";
              return <SwiperSlide className="slideContainer">
                <Link to={` /painting/${slide.attributes?.picture?.data?.id}` }
                  state={slide.attributes?.picture?.data} className="slideLink">
                  <img src={temp === "/noImage.svg" ? "/noImage.svg" : (imageURL + temp)} alt="" />
                  <h1>{slide.attributes?.picture?.data?.attributes?.name}</h1>
                  <p>{slide.attributes?.picture?.data?.attributes?.artist?.data?.attributes?.name}</p>
                  </* <p>{slide.attributes?.picture?.data?.attributes?.description}</p> */>
                </Link>
              </SwiperSlide>
            })}
          </Swiper>
          <button className="sliderBtn sliderBtnNext" onClick={() =>
            swiperHeroRef.current.slideNext()}><ArrowForwardOutlinedIcon /></button>
        </div>
      </div>
    </section>
    <Divider></Divider>
    <section className="wrapper">
      <h1 className="sectionTitle">Выставки</h1>
      <div className="sliderWrapper">

```

```

        <button className="sliderBtn sliderBtnPrev" onClick={() =>
swiperRef.current.slidePrev()}><ArrowBackOutlinedIcon /></button>
    <Swiper className="sliderExhibition wrapper"
    slidesPerView={3}
    modules={[Navigation, Pagination, A11y, Autoplay]}
    spaceBetween={40}
    onSlideChange={() => console.log("slide change")}
    // pagination={{ clickable: true }}
    // autoplay={{ delay: 5000 }}
    onSwiper={(swiper) => {
        swiperRef.current = swiper;
    }}
    >

    <div className="exCardWrapper">
    {exhibitionApiData?.map((exhib) => {
    return (
    <SwiperSlide className="slideContainer">
    <Exhibition
    key={exhib.id}
    cardTitle={exhib.attributes.title}
    about={exhib.attributes.about}
    image={exhib.attributes.cover?.data ?
    (imageUrl + exhib.attributes.cover.data[0].attributes.formats.small.url)
    :
    "/noImage.svg"}
    id={exhib.id}
    startDate={exhib.attributes.startDate}
    finishDate={exhib.attributes.finishDate}
    />
    </SwiperSlide>
    )
    )}}
    <SwiperSlide className="slideContainer">
    <Card className="exhibitionCard lastCard" sx={{ height: 430 }}>
    <Link to={"/exhibitions"} className="linkToMore">
    <CardContent>
    <Typography gutterBottom variant="h5" component="div" sx={{ fontWeight: 400 }}>
    Смотреть еще...
    </Typography>
    </CardContent>
    </Link>
    </Card>
    </SwiperSlide>
    </div>
    </Swiper>
        <button className="sliderBtn sliderBtnNext" onClick={() =>
swiperRef.current.slideNext()}><ArrowForwardOutlinedIcon /></button>
    </div>
</section>
<Divider></Divider>
<section className="wrapper">
<h1 className="sectionTitle">Художники</h1>
<div className="pictures">
    {artistApiData?.map((art) => (
    <Artist
    key={art.id}
    name={art.attributes.name}
    image={imageUrl + art.attributes.avatar?.data.attributes.formats.thumbnail.url}
    id={art.id}
    />
    ))}

```

```

    </div>
    <div className="linkContainer">
      <Link to={"/artists"} className="linkToMore">
        Смотреть еще...
      </Link>
    </div>
  </section>
</Divider></Divider>
<section className="wrapper">
  <h1 className="sectionTitle">Образовательные услуги</h1>
  <div className="wrapperArticle">
    <div className="ck-content text-indent" dangerouslySetInnerHTML={{ __html:
articles?.attributes?.Content }}>
    </div>
  </div>
</section>

</div>
);
}

export default Main;

```

## 2. Pages/ExhibitionDetail.js

```

import './exhibitionDetail.css'
import './ckeditor.css'
import { useLocation, useParams } from 'react-router-dom'
import { useEffect, useState, useRef } from 'react'
import axios from 'axios'
import Zoom from 'react-medium-image-zoom'
import 'react-medium-image-zoom/dist/styles.css'

import { Swiper, SwiperSlide } from "swiper/react";
import { Navigation, Pagination, Autoplay, A11y } from 'swiper';
import ArrowForwardOutlinedIcon from '@mui/icons-material/ArrowForwardOutlined';
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined';

import ConvertDate from '../helpers/ConvertDate'

function ExhibitionDetail() {

  const [exhibition, setExhibition] = useState({});
  const [image, setImage] = useState("");
  const { id } = useParams();
  console.log('id = ' + id);

  const URL = `http://localhost:1337/api/exhibitions/${id}?populate=*`;
  console.log(URL);
  const imageURL = "http://localhost:1337";

  const swiperRef = useRef();

  const fetchData = async () => {
    let data;
    const result = await axios.get(URL).catch(
      (error) => {
        if (error.response) {
          // The request was made and the server responded with a status code
          // that falls out of the range of 2xx
          console.log(error.response.data);

```



```

        console.log(error.response.status);
        console.log(error.response.headers);
    } else if (error.request) {
        // The request was made but no response was received
        // `error.request` is an instance of XMLHttpRequest in the browser and an instance of
        // http.ClientRequest in node.js
        console.log(error.request);
    } else {
        // Something happened in setting up the request that triggered an Error
        console.log('Error', error.message);
    }
    console.log(error.config);
  }
);
data = result?.data?.data;
console.log('checking data ' + data);
setExhibition(data);
}

useEffect(() => {
  fetchData();
  window.scrollTo(0, 0);
}, []);

useEffect(() => {
  let temp = (exhibition?.attributes?.cover?.data[0]?.attributes.formats.large?.url ??
    exhibition?.attributes?.cover?.data[0]?.attributes.formats.medium?.url ??
    exhibition?.attributes?.cover?.data[0]?.attributes.formats.small?.url ??
    exhibition?.attributes?.cover?.data[0]?.attributes.formats.thumbnail.url ?? "");
  setImage((temp === "") ? "/noImage.svg" : imageURL + temp);
}, [Array.isArray(exhibition?.attributes?.cover?.data) && exhibition?.attributes?.cover?.data?.length]);

return (
  <div className='exhibitionDetail wrapper'>
    <div className='exhibitionDetail__topDec'>
      <div className='imgContainer'>
        <Zoom>
          <img src={image} alt='Выставка' />
        </Zoom>
      </div>

      <div className='exhibitionDetail__textBlock1'>
        <h2 className='exhibitionDates'>
          `{(exhibition.attributes?.startDate) === null ?
            "
            :
            exhibition.attributes?.finishDate ?
            ConvertDate(exhibition.attributes?.startDate, false)
            :
            ConvertDate(exhibition.attributes?.startDate, true)}

            ${(exhibition.attributes?.finishDate) === null ?
            "
            :
            `— ${ConvertDate(exhibition.attributes?.finishDate)}`}`
          `
        </h2>
        <h1>{exhibition.attributes?.title}</h1>
        <h2>{exhibition.attributes?.typeOfFineArt}</h2>
        <div className='exhibitionDetail__artists'>
          {exhibition.attributes?.artists?.data.map((name) => {
            return <h2>{name.attributes.name}</h2>

```

```

    }}
  </div>
  <h3 className='shortDescription'>{exhibition.attributes?.about}</h3>
</div>
</div>

<div className='exhibitionDetail__bottomDec'>
  <div className='ck-content text-indent' dangerouslySetInnerHTML={{ __html:
exhibition.attributes?.descriptionNew }}></div>
</div>

{exhibition.attributes?.pictureStory.data ?
  <
    <div className='titleContainer'>
      <h1 className="sectionTitle">Фотоотчет с мероприятия</h1>
    </div>
    <div className='sliderWrapper sliderWrapperExhib'>
      <button className="sliderBtnExhib sliderBtnPrev" onClick={() =>
swiperRef.current.slidePrev()}><ArrowBackOutlinedIcon /></button>
      <Swiper className="sliderExhibition wrapper"
        slidesPerView={1}
        modules={[Navigation, Pagination, A11y, Autoplay]}
        spaceBetween={40}
        loop
        onSlideChange={() => console.log("slide change")}
        pagination={{ clickable: true }}
        // autoplay={{ delay: 5000 }}
        onSwiper={(swiper) => {
          swiperRef.current = swiper;
        }}
      >
        <div >
          {exhibition.attributes?.pictureStory.data.map((pic) => {
            let temp = pic.attributes.formats?.large?.url ??
            pic.attributes.formats?.medium?.url ??
            pic.attributes.formats?.small?.url ??
            pic.attributes.formats?.thumbnail?.url ??
            "/noImage.svg";
            return (
              <SwiperSlide className="slideContainer exhibitionSlide">
                <Zoom>
                  <img className='pictureStoryImg' src={
                    temp === "/noImage.svg" ? "/noImage.svg" : (imageUrl + temp)
                  }></img>
                </Zoom>
              </SwiperSlide>
            )
          })}
        </div>
      </Swiper>
      <button className="sliderBtnExhib sliderBtnNext" onClick={() =>
swiperRef.current.slideNext()}><ArrowForwardOutlinedIcon /></button>
    </div>
  </>
  :
  <></>
}
</div>
)

```

```

    }

    export default ExhibitionDetail;

3.    Pages/Exhibitions.js

import { useEffect, useState } from "react";
import "../exhibitions.css";
import "../ckeditor.css"

import axios from "axios";
import { Link } from 'react-router-dom';
import Card from '@mui/material/Card';
import CardActions from '@mui/material/CardActions';
import CardContent from '@mui/material/CardContent';
import CardMedia from '@mui/material/CardMedia';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';
import { Pagination } from '@mui/material';
import Exhibition from "../components/Exhibition";

function Exhibitions({ searchValue, setSearchValue, type }) { //type - тип архива, используется для страницы
с архивом выставок
    useEffect(() => {
        setSearchValue("");
        window.scrollTo(0, 0);
    }, []);

    const imageURL = "http://localhost:1337"

    const [pageApi, setPageApi] = useState(1);
    const [apiData, setApiData] = useState([]);
    const [count, setCount] = useState(0);
    const PER_PAGE = 12; //отвечает за кол-во элементов на странице
    const [isArchived, setIsArchived] = useState((type === 'arch') ? true : false);

    const ApiAddress = axios.create({
        baseURL: "http://localhost:1337/api/exhibitions"
    });

    useEffect(() => {
        if (isArchived) {
            ApiAddress.get(`?pagination[page]=${pageApi}&pagination[pageSize]=${PER_PAGE}&filters[$and][0][isArchived][$eq]=true&populate=*)
                .then((response) => { setApiData(response.data.data);
                    console.log(response.data.data); })
                .catch((err) => console.log(err));
        }
        else {
            ApiAddress.get(`?pagination[page]=${pageApi}&pagination[pageSize]=${PER_PAGE}&filters[$or][0][isArchived][$eq]=false&filters[$or][1][isArchived][$null]=true&populate=*)
                .then((response) => { setApiData(response.data.data);
                    console.log(response.data.data); })
                .catch((err) => console.log(err));
        }
    }, [pageApi, type]);

    //забираем из api кол-во элементов для вычисления количества страниц
    useEffect(() => {
        ApiAddress.get()
            .then((response) => {

```

```

        setCount(Math.ceil(response.data.meta.pagination.total / PER_PAGE));
    })
    .catch((err) => console.log(err));
}, []);

return (
  <div className="main">
    <h1 className="title">{isArchived ? 'Архив выставок' : 'Выставки'}</h1>
    <div className="wrapper cardWrapper">

      {apiData?.map((exhib) => {
        return <Exhibition
          key={exhib.id}
          cardTitle={exhib.attributes.title}
          about={exhib.attributes.about}
          image={exhib.attributes.cover?.data ?
            (imageUrl + exhib.attributes.cover.data[0].attributes.formats.small.url)
            :
            "/noImage.svg"}
          id={exhib.id}
          startDate={exhib.attributes.startDate}
          finishDate={exhib.attributes.finishDate}
        />

      })}
    </div>
    <Pagination
      className='wrapper pagination'
      count={count} // передаем количество страниц
      size="large"
      variant="outlined"
      shape="rounded"
      onChange={(e, value) => {
        window.scrollTo(0, 0);
        setPageApi(value)
      }}
      sx={{
        display: 'flex',
        justifyContent: 'flex-end',
        marginBottom: '80px',
      }}
    />
  </div>
);
}

```

export default Exhibitions;

3./Index.js

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);

```

Листинг файла cron-tasks.js:

```
module.exports = {
  exhibitionsUpdate: {
    task: async ({ strapi }) => {
      try {
        let currentDate = new Date();
        currentDate.setHours(0,0,0,0);
        console.log(`${currentDate.getDate()}.${currentDate.getMonth()}.${currentDate.getFullYear()}`);
        const exhibitions = await strapi.entityService.findMany('api::exhibition.exhibition', {
          fields: ['id', 'isArchived', 'finishDate'],
        });
        exhibitions.map(async (exhib) => {
          try {
            if (exhib.finishDate === null) return;
            if (exhib.isArchived === true) return;
            const isValidDate = (Y, M, D) => {
              var d = new Date(Y, --M, D);
              return Y == d.getFullYear() && M == d.getMonth() && D == d.getDate();
            }
            let dateArr = exhib.finishDate.split('.');
            let day = parseInt(dateArr[0]);
            let month = parseInt(dateArr[1]);
            let year = parseInt(dateArr[2]);
            if (!isValidDate(year, month, day)) { //проверка даты на валидность
              console.log('!----- Выставка с id='+exhib.id+' имеет некорректную дату');
              return;
            }
            const finishDate = new Date(dateArr[2], dateArr[1] - 1, dateArr[0]);
            console.log('дата окончания ' + finishDate.toString());
            if (finishDate < currentDate) {
              let result = await strapi.entityService.update('api::exhibition.exhibition', exhib.id, {
                data: {
                  isArchived: true,
                }
              });
              console.log('Архивировано при сравнении месяца, id выставки:' + exhib.id);
            } else console.log('Архивации не произошло, id выставки:' + exhib.id);
          } catch (error) {
            console.log(error);
          }
        });
      } catch (error) {
        console.log(error)
      }
    },
    options: {
      rule: "00 00 00 * * *",
    },
  },
};
```

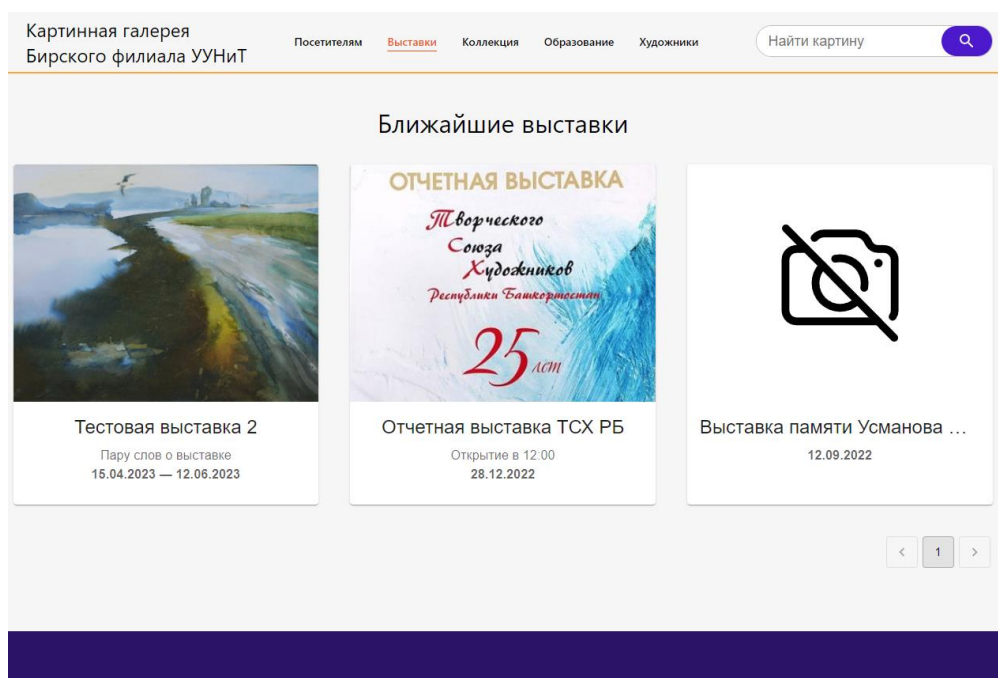


Рисунок 16. Страница с текущими и предстоящими выставками

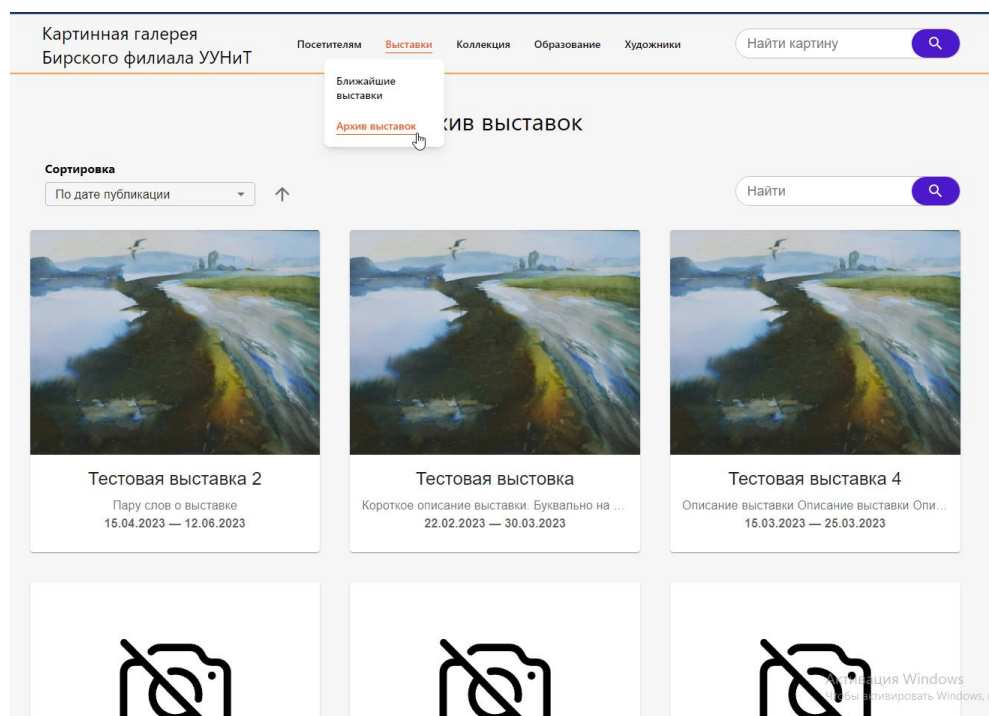


Рисунок 17. Страница с архивированными выставками



Рисунок 18. Страница выставки

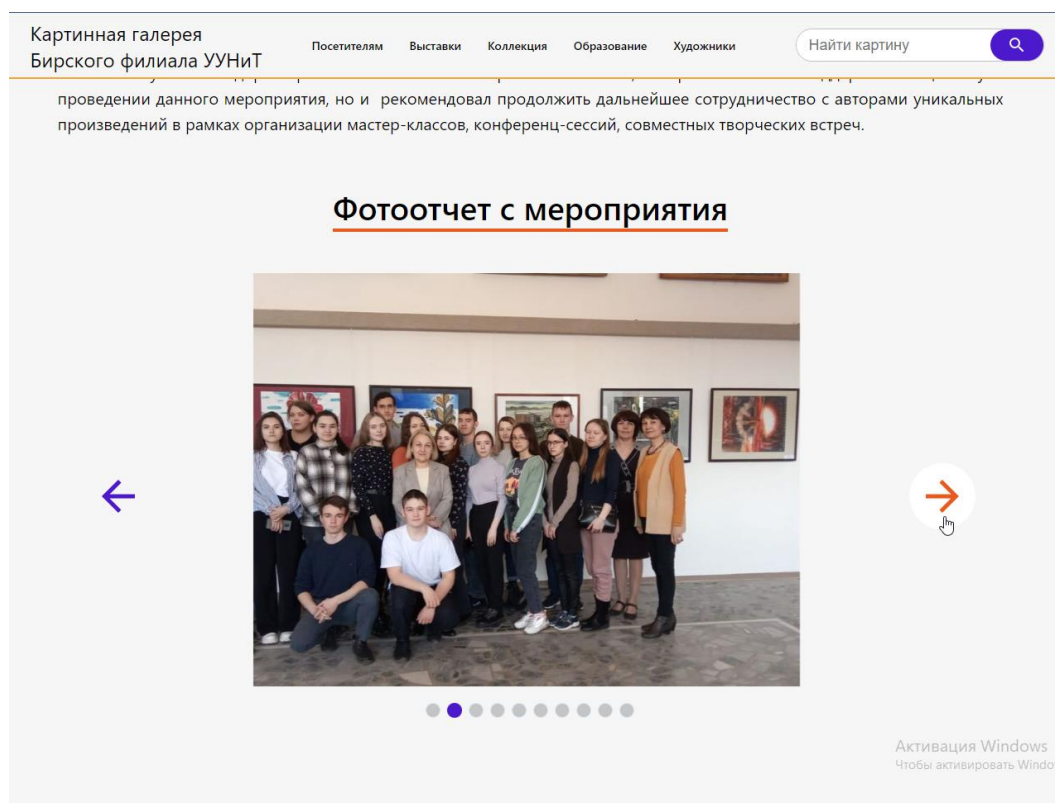


Рисунок 19. Страница выставки, фотоотчет с мероприятия

