**Федеральное государственное образовательное бюджетное учреждение высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Факультет информационных технологий и анализа больших данных**

*(наименование факультета)*

**Кафедра**

**КУРСОВОЙ ПРОЕКТ**

на тему:

информационно - справочная система управления коллекцией предметов искусств

Студента группы ТРПО23-2 Савенкова Прохора Сергеевича

Научный руководитель:

Москва 2025

Содержание

[Введение 3](#_Toc199572191)

[1. Теоретическая часть 6](#_Toc199572192)

[1.1. Описание предметной области 6](#_Toc199572193)

[1.2. Принципы работы систем с клиент-серверной архитектурой 8](#_Toc199572194)

[1.3. Обзор методов и средств разработки 10](#_Toc199572195)

[2. Практическая часть 13](#_Toc199572196)

[2.1. Характеристика объектов автоматизации, требования к системе 13](#_Toc199572197)

[2.2. Разработка реляционной базы данных 15](#_Toc199572198)

[Физическая модель базы данных 19](#_Toc199572199)

[2.3. Алгоритмические решения 21](#_Toc199572200)

[2.4. Спецификации программного обеспечения 24](#_Toc199572201)

[2.5. Описание пользовательского интерфейса 27](#_Toc199572202)

[2.6. Тестирование системы 29](#_Toc199572203)

[Заключение 32](#_Toc199572204)

[Список использованных источников 34](#_Toc199572205)

[Приложения 36](#_Toc199572206)

# **Введение**

В современном мире информационные технологии играют важную роль практически во всех сферах деятельности человека, в том числе в культуре и искусстве. Художественные музеи, частные галереи, аукционные дома, фонды и коллекционеры — все чаще сталкиваются с необходимостью систематизировать, хранить и оперативно получать информацию о произведениях искусства, которые находятся в их распоряжении. Бумажные каталоги, таблицы и ручной учет уже не справляются с этим объемом данных, особенно если речь идёт о крупных коллекциях.

Для решения этих задач требуются современные программные решения, которые позволяли бы автоматизировать процессы учета, поиска и анализа информации. Особенно актуальной становится задача создания удобной и надежной информационно-справочной системы, которая могла бы применяться в учреждениях культуры, частных коллекциях или даже как вспомогательный инструмент для искусствоведов и реставраторов.

Актуальность выбранной темы обусловлена несколькими факторами:

* растущей цифровизацией сферы культуры и необходимости перехода на электронные формы учета;
* потребностью в упрощении процессов каталогизации и поиска информации о предметах искусства;
* возможностью повышения прозрачности хранения и перемещений коллекционных объектов;
* желанием упростить доступ к данным для различных категорий пользователей: от кураторов и искусствоведов до технического персонала и исследователей.

Создание такой системы особенно важно в контексте роста числа выставок, цифровых архивов и виртуальных галерей, которые требуют

Целью данной курсовой работы является разработка информационно-справочной системы управления коллекцией предметов искусства, которая позволит:

* хранить информацию о произведениях искусства в единой базе данных;
* классифицировать объекты по коллекциям, авторам, жанрам и другим признакам;
* предоставлять пользователю простой и понятный интерфейс для работы с данными;
* обеспечивать выполнение операций добавления, удаления, редактирования и поиска информации;
* отображать статистику и справочные данные по коллекции.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить особенности предметной области: учет предметов искусства, типовые данные и их связи.
2. Изучить принципы клиент-серверной архитектуры и подходы к построению информационных систем.
3. Спроектировать структуру базы данных с учетом требований предметной области.
4. Разработать серверную часть на языке Java с использованием REST API и ORM.
5. Создать клиентское приложение с графическим интерфейсом на базе JavaFX.
6. Реализовать основные функции: отображение данных, фильтрация, поиск, редактирование, удаление.
7. Провести тестирование разработанной системы.
8. Подготовить пояснительную записку и документацию.

В работе применяются следующие методы и технологии:

* Метод анализа — при изучении требований предметной области и структуры хранимых данных.
* Метод моделирования — при построении информационной модели коллекции.
* Методы объектно-ориентированного программирования — при разработке клиентского и серверного программного обеспечения.
* Инструменты разработки — язык Java, фреймворки JavaFX и Spring Boot, библиотека Hibernate, база данных PostgreSQL.
* Инструменты документирования — Javadoc, Git, IntelliJ IDEA.

Таким образом, данная курсовая работа направлена на решение актуальной задачи цифровизации и автоматизации учета предметов искусства с применением современных технологий программирования. Реализация предложенного проекта позволит не только улучшить работу с коллекциями, но и получить практический опыт создания полнофункционального клиент-серверного приложения.

# **1. Теоретическая часть**

## **1.1. Описание предметной области**

Предметная область данной курсовой работы — это учет и управление коллекцией предметов искусства. В современном мире искусство — это не только культурная ценность, но и важный объект учета, экспертизы, хранения и, в некоторых случаях, инвестирования. Особенно остро необходимость автоматизации ощущается в музеях, частных галереях, фондах, архивах и коллекциях, где может храниться от нескольких десятков до десятков тысяч произведений. Каждое произведение — это уникальный объект, имеющий множество характеристик: название, автор, год создания, техника исполнения (масло, акварель, бронза, мрамор и т.д.), жанр (портрет, пейзаж, натюрморт), размеры, принадлежность к коллекции, текущая оценочная стоимость, статус хранения (на выставке, на реставрации, в фондах), место нахождения, а также возможность отображения изображения объекта.

В таких условиях ручной учет, даже с применением обычных электронных таблиц, становится крайне неудобным и подверженным ошибкам. Возникает потребность в создании централизованной системы, которая позволит структурировать информацию, быстро находить нужные объекты, отслеживать их историю, а также получать сводные данные для анализа и отчетности. Например, искусствоведу может понадобиться быстро сформировать список всех экспонатов, выполненных в технике акварели, за период с 1850 по 1900 год. Или куратор выставки должен отобрать все произведения, которые были написаны женщинами-художницами ХХ века и входят в фонд импрессионизма. Такие задачи требуют наличия сложных фильтров, поиска по множеству параметров и четкой логической связи между объектами.

Также важно учитывать, что в большинстве случаев предметы искусства группируются в коллекции. Это могут быть постоянные музейные коллекции, временные экспозиции, частные собрания или тематические подборки. Коллекция является родительским элементом по отношению к конкретным произведениям. Поэтому структура данных системы должна отражать двухуровневую иерархию: «коллекция — объект искусства». Такая архитектура позволяет логично структурировать базу и удобно формировать отчеты, например, по каждой коллекции.

Дополнительным важным элементом является учет авторов произведений. Каждый объект искусства связан с конкретным художником или скульптором, информация о котором также должна храниться в системе: ФИО, годы жизни, страна, направление в искусстве, краткая биография. Это позволяет связывать работы одного автора, делать тематические выборки и статистику.

Информационно-справочная система, которая создается в рамках данной работы, будет представлять собой программный продукт, обеспечивающий хранение всех вышеперечисленных данных, предоставление удобного интерфейса для взаимодействия с пользователем, а также реализацию функций поиска, фильтрации, редактирования и анализа. В качестве примеров источников данных и прототипов таких систем можно привести «Музейный каталог» ГМИИ им. Пушкина, онлайн-каталог Третьяковской галереи, а также систему учёта коллекций Smithsonian Institution (США), которые активно используют цифровые платформы для управления фондами.

Таким образом, предметная область объединяет в себе элементы искусства, музееведения и информационных технологий и представляет собой живой, насыщенный данными и логикой сегмент, в котором автоматизация способна существенно повысить эффективность работы учреждений культуры и частных владельцев коллекций.

## **1.2. Принципы работы систем с клиент-серверной архитектурой**

Клиент-серверная архитектура — это один из наиболее распространённых подходов к построению информационных систем, особенно если необходимо разделить пользовательский интерфейс и обработку данных. Суть этой архитектуры заключается в том, что приложение делится на две основные части: клиентскую и серверную. Клиент — это программа, с которой взаимодействует пользователь. Она отвечает за отображение данных, получение команд от пользователя и отправку этих команд на сервер. Сервер, в свою очередь, занимается обработкой запросов от клиента, выполнением логики приложения, обращением к базе данных и формированием ответа, который возвращается обратно клиенту.

В рамках данной курсовой работы система реализуется в виде классического трёхуровневого приложения. Первый уровень — это клиент, то есть графический интерфейс пользователя, созданный с использованием технологии JavaFX. Пользователь через интерфейс вводит данные о произведениях искусства, выбирает коллекции, сортирует, фильтрует или редактирует записи. Эти действия инициируют определённые запросы к серверу.

Второй уровень — это серверная часть, которая реализуется с использованием фреймворка Spring Boot. Сервер принимает HTTP-запросы от клиента через REST API, обрабатывает полученные данные, выполняет необходимые проверки, преобразует данные и взаимодействует с базой данных. REST (Representational State Transfer) — это архитектурный стиль взаимодействия между клиентом и сервером через стандартные веб-протоколы. В случае этой системы, клиент будет отправлять на сервер, например, POST-запрос для добавления нового объекта искусства, GET-запрос для получения информации, PUT для редактирования и DELETE для удаления. Все данные при этом передаются в формате JSON, что упрощает работу и делает обмен информацией гибким и читаемым.

Третий уровень — это уровень базы данных. В данной работе используется СУБД PostgreSQL, которая предназначена для хранения всей информации о произведениях искусства, авторах, коллекциях и связанных с ними данных. Для удобства и безопасности работы с базой данных на сервере используется ORM (Object-Relational Mapping), а именно — библиотека Hibernate или Spring Data JPA. Эти технологии позволяют разработчику работать не напрямую с SQL-запросами, а через объектную модель на языке Java. Например, объект класса Artwork будет автоматически отображаться на таблицу artworks в базе данных, а все изменения объекта (добавление, редактирование, удаление) будут автоматически синхронизироваться с данными в таблице. Это упрощает разработку, снижает количество ошибок и делает код более понятным.

Таким образом, клиент-серверная архитектура позволяет чётко разделить роли и ответственности компонентов системы. Клиент отвечает за взаимодействие с пользователем, сервер — за бизнес-логику и управление данными, база данных — за надёжное и структурированное хранение информации. Такой подход позволяет обеспечить масштабируемость системы, безопасность хранения данных, удобство сопровождения и возможность дальнейшего развития приложения.

## **1.3. Обзор методов и средств разработки**

Для реализации информационно-справочной системы управления коллекцией предметов искусства в данной курсовой работе используются современные технологии и инструменты, которые активно применяются в профессиональной среде. Основной язык программирования — это **Java**, так как он универсален, широко распространён и поддерживает мощные средства для построения как серверной, так и клиентской части приложения. Java позволяет реализовать устойчивую, расширяемую и кросс-платформенную систему. Разработка клиентской части, то есть графического интерфейса пользователя, осуществляется с использованием JavaFX — это современная библиотека для создания пользовательских интерфейсов, которая пришла на смену более устаревшей Swing. JavaFX предоставляет множество компонентов (таблицы, формы, кнопки, выпадающие списки и другие), а также инструменты для стилизации интерфейса, включая поддержку CSS. Интерфейс будет построен таким образом, чтобы пользователь мог удобно просматривать список произведений, фильтровать и сортировать их, а также добавлять и редактировать информацию без необходимости обращаться к разработчику. Серверная часть системы реализуется с помощью Spring Boot — популярного Java-фреймворка, который значительно упрощает создание веб-приложений и REST API. С помощью Spring Boot можно быстро создать сервер, который будет обрабатывать HTTP-запросы, соединяться с базой данных и обеспечивать бизнес-логику приложения. Этот фреймворк также предоставляет встроенные средства для работы с безопасностью, логированием, тестированием и другими аспектами разработки, что делает его оптимальным выбором для профессиональных проектов.

Для хранения и управления данными используется реляционная база данных PostgreSQL, которая отличается высокой надёжностью, поддержкой сложных запросов и расширенных типов данных. В рамках предметной области предметов искусства база данных будет содержать таблицы для произведений, авторов, коллекций и других связанных сущностей. Каждая таблица будет спроектирована с учетом логических связей между объектами, например: одна коллекция может содержать много произведений, но каждое произведение принадлежит только одной коллекции. Для работы с базой данных на стороне сервера используется технология ORM (Object-Relational Mapping) — а именно Hibernate или Spring Data JPA. ORM позволяет описывать таблицы базы данных в виде обычных Java-классов, благодаря чему становится возможна работа с данными как с объектами, без необходимости писать вручную SQL-запросы. Это делает код более читаемым, а также уменьшает риск ошибок при разработке. Например, добавление нового объекта в базу сводится к вызову метода save() у репозитория, и система сама создаст нужный SQL-запрос. Для тестирования и сопровождения проекта используются такие инструменты, как Git и GitHub, которые позволяют контролировать версионность проекта, отслеживать изменения, возвращаться к предыдущим версиям и организовать совместную работу над проектом. Репозиторий проекта размещается на GitHub и содержит все необходимые файлы: исходный код клиента и сервера, структуру базы данных, файлы конфигурации и документацию.

В целом, все выбранные инструменты являются актуальными и востребованными в профессиональной разработке, что позволяет не только выполнить курсовой проект, но и получить практические навыки, которые пригодятся в дальнейшем обучении и работе. Комбинация JavaFX, Spring Boot, PostgreSQL и Hibernate обеспечивает возможность построения полнофункционального, масштабируемого и стабильного приложения, соответствующего требованиям, изложенным в методических указаниях.

# **2. Практическая часть**

## **2.1. Характеристика объектов автоматизации, требования к системе**

Объектами автоматизации в рамках данной курсовой работы выступают процессы учёта и управления коллекцией предметов искусства. В учреждениях культуры, музеях, частных галереях и фондах в процессе ежедневной работы возникает необходимость фиксировать, систематизировать и анализировать данные о произведениях искусства, их авторах, текущем статусе, месте хранения и принадлежности к той или иной коллекции.

Основными единицами учёта в разрабатываемой системе являются следующие сущности: произведение искусства, автор, коллекция, а также пользователь системы (если система предполагает разграничение доступа по ролям, например, «администратор» и «пользователь»). Каждое произведение искусства представляет собой уникальный объект, обладающий рядом характеристик: название, автор, год создания, страна происхождения, техника исполнения (например, масло по холсту, акварель, бронза и т.д.), жанр (портрет, натюрморт, абстракция), размер (высота и ширина), оценочная стоимость, а также статус (на хранении, на реставрации, на выставке, выдан во временное пользование). Важно также хранить изображение или ссылку на изображение произведения, если оно доступно в цифровом виде. Сущность автор связана с произведением через отношение «многие к одному» (много произведений могут быть созданы одним автором). У автора указывается имя, фамилия, годы жизни, страна, направление или художественная школа (например, импрессионизм, реализм, авангард), а также краткая биографическая справка. Такая информация позволяет группировать произведения по авторам и строить статистику: например, сколько произведений определённого художника хранится в коллекции, или какие авторы доминируют в экспозиции.

Коллекция — это объединение произведений по какому-либо признаку: по историческому периоду, тематике, владельцу или месту происхождения. Коллекции представляют собой иерархический уровень, к которому могут относиться десятки и сотни объектов. Примеры коллекций: «Русский авангард начала ХХ века», «Западноевропейская живопись XVIII века», «Современное искусство». Каждая коллекция имеет название, описание, дату создания, и список входящих в неё произведений. Структура базы данных и интерфейса программы должны позволять быстро получить список всех произведений определённой коллекции и выполнять поиск внутри неё. Если система предполагает многопользовательский режим, то необходимо реализовать учет пользователей, где хранится информация о логине, пароле (в хешированном виде), роли и доступе к операциям. Например, обычный пользователь может только просматривать и фильтровать объекты, а администратор — добавлять и удалять записи, формировать отчёты. Исходя из логики работы системы и требований предметной области, к информационно-справочной системе предъявляются следующие функциональные и нефункциональные требования: система должна обеспечивать стабильную работу с базой данных, хранящей информацию обо всех объектах; пользовательский интерфейс должен быть простым, логичным и интуитивно понятным; система должна поддерживать операции создания, редактирования, удаления и просмотра записей; должна быть реализована функция фильтрации и поиска по множественным полям (например, поиск всех произведений, написанных маслом между 1800 и 1850 годами); необходимо реализовать отображение статистических данных (например, общее количество произведений, количество по авторам, коллекциям, техникам и т.д.); должна обеспечиваться валидация данных на клиентской и серверной стороне; при запуске программа должна автоматически загружать актуальные данные из базы; важно предусмотреть обработку ошибок и исключений, чтобы программа не завершалась аварийно при некорректном вводе или отсутствии данных.

В результате автоматизации предполагается существенное повышение эффективности работы с коллекциями, снижение ошибок при вводе и обработке информации, а также повышение доступности справочных данных как для сотрудников организации, так и для заинтересованных пользователей.

## **2.2. Разработка реляционной базы данных**

В основе функционирования информационно-справочной системы управления коллекцией предметов искусства лежит база данных, в которой хранятся все сведения об объектах, авторах, коллекциях и других сущностях. Для надёжного и структурированного хранения информации в данной работе используется реляционная модель данных, реализуемая в СУБД PostgreSQL.

Разработка базы данных началась с построения информационной модели предметной области, на основе которой была определена структура таблиц и их взаимосвязи. Основной акцент сделан на обеспечении целостности данных, соблюдении логических связей между сущностями и возможности масштабирования модели при добавлении новых функций.

Система включает следующие основные сущности: Коллекция (Collection), Произведение искусства (Artwork), Автор (Author) и, при необходимости, Пользователь (User). Ниже приведено описание таблиц, входящих в состав БД:

1. Таблица collections — содержит информацию о коллекциях, в которые объединяются произведения искусства.
   * id — уникальный идентификатор коллекции (целое число, автоинкремент).
   * title — название коллекции (строка, не null).
   * description — краткое описание коллекции.
   * created\_at — дата создания записи.
2. Таблица authors — содержит сведения об авторах произведений.
   * id — уникальный идентификатор автора.
   * first\_name — имя.
   * last\_name — фамилия.
   * birth\_year — год рождения.
   * death\_year — год смерти (может быть null).
   * country — страна происхождения.
   * art\_style — художественное направление (импрессионизм, реализм и т.д.).
   * biography — краткая биографическая справка.
3. Таблица artworks — основная таблица, в которой хранятся данные о произведениях искусства.
   * id — идентификатор произведения.
   * title — название произведения.
   * author\_id — внешний ключ на таблицу authors.
   * collection\_id — внешний ключ на таблицу collections.
   * year\_created — год создания.
   * technique — техника исполнения (например, масло по холсту).
   * genre — жанр произведения.
   * width\_cm, height\_cm — размеры в сантиметрах.
   * status — текущий статус (например: «на хранении», «на выставке»).
   * estimated\_value — оценочная стоимость в рублях.
   * image\_url — ссылка на изображение.
   * added\_at — дата добавления в систему.
4. Таблица users (опционально) — используется для аутентификации и разграничения доступа.
   * id — идентификатор пользователя.
   * username — логин (уникальное имя).
   * password\_hash — хеш пароля.
   * role — роль (например: admin, user).
   * created\_at — дата регистрации.

Связи между таблицами:

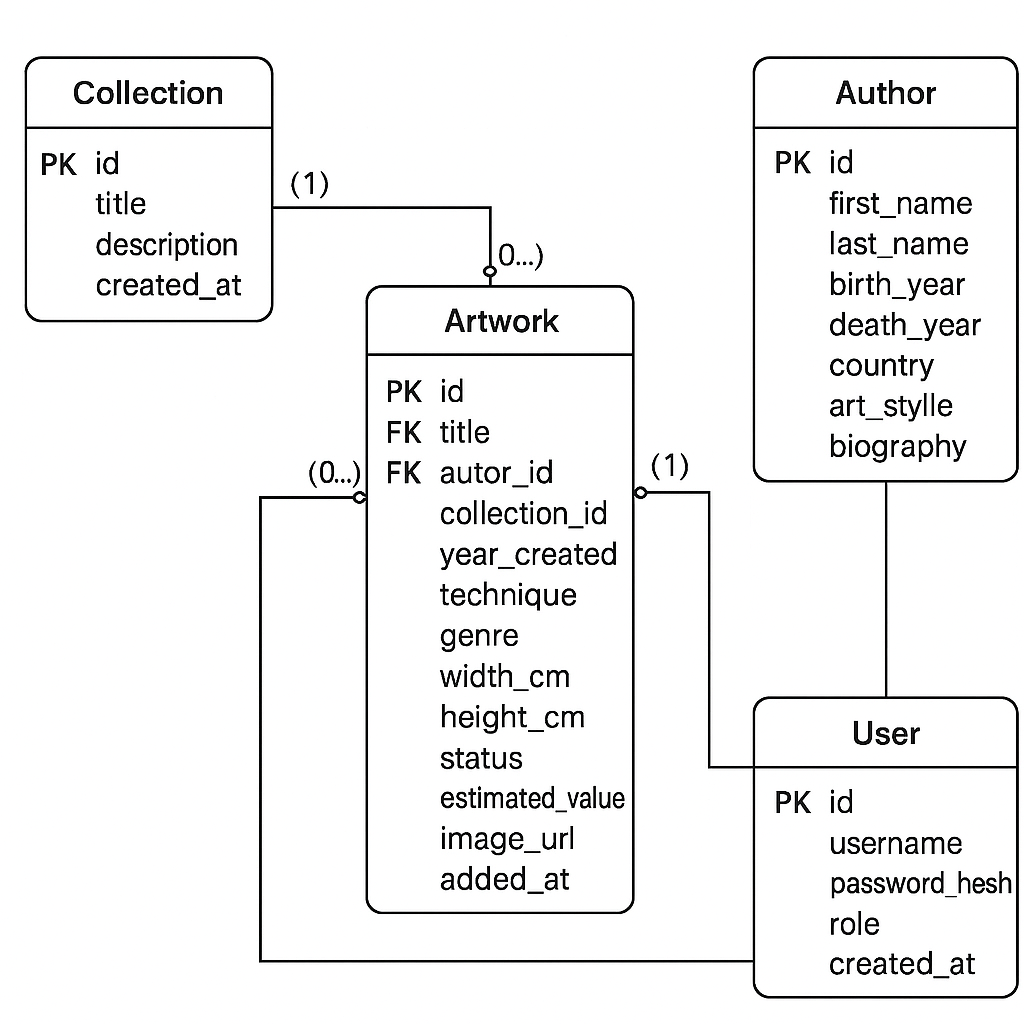
* artworks.author\_id связан с authors.id отношением «многие к одному»;
* artworks.collection\_id связан с collections.id также отношением «многие к одному»;
* один автор может иметь множество работ, одна коллекция может включать множество произведений.

Такая структура позволяет гибко управлять данными, выполнять сложные выборки и фильтрации. Например, с помощью SQL-запросов можно получить:

* все работы определённого автора, входящие в конкретную коллекцию;
* список всех произведений, написанных в определённый период;
* сводную статистику по техникам, авторам, странам или жанрам.

Модель может быть расширена, например, добавлением таблиц для учёта выставок, помещений хранения, истории перемещений объектов или экспертных оценок. Однако для целей данной курсовой работы представленной структуры достаточно, чтобы реализовать все базовые функции и обеспечить соответствие методическим требованиям.

Рисунок 1 – ER-диаграмма реляционной базы данных информационно-справочной системы управления коллекцией предметов искусства

На рисунке 1 представлена ER-диаграмма (логическая модель реляционной базы данных), отражающая основные сущности и связи между ними. Центральной сущностью является таблица Artwork, содержащая сведения о произведениях искусства. Она связана внешними ключами с таблицами Author (авторы) и Collection (коллекции), что отражает логические отношения «многие к одному». Один автор может быть связан с несколькими произведениями, как и одна коллекция может содержать множество объектов искусства. Таблица Userпредназначена для хранения данных о пользователях системы в случае реализации авторизации. Такая структура обеспечивает логичное и масштабируемое хранение данных, а также позволяет реализовать поиск и фильтрацию по ключевым параметрам.

В таблице 1 представлена физическая модель базы данных, включающая описание всех сущностей, типов данных, первичных и внешних ключей. Такая структура позволяет обеспечить логическую целостность данных, масштабируемость и возможность гибкой выборки информации из различных таблиц с использованием SQL-запросов.

Таблица 1 – Структура таблиц реляционной базы данных системы

**Физическая модель базы данных**

**Таблица collections — Коллекции**

| **Поле** | **Тип данных** | **Ограничения** |
| --- | --- | --- |
| id | SERIAL | PRIMARY KEY |
| title | VARCHAR(255) | NOT NULL |
| description | TEXT |  |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Таблица authors — Авторы**

| **Поле** | **Тип данных** | **Ограничения** |
| --- | --- | --- |
| id | SERIAL | PRIMARY KEY |
| first\_name | VARCHAR(100) | NOT NULL |
| last\_name | VARCHAR(100) | NOT NULL |
| birth\_year | INTEGER |  |
| death\_year | INTEGER |  |
| country | VARCHAR(100) |  |
| art\_style | VARCHAR(100) |  |
| biography | TEXT |  |

**Таблица artworks — Произведения искусства**

| **Поле** | **Тип данных** | **Ограничения** |
| --- | --- | --- |
| id | SERIAL | PRIMARY KEY |
| title | VARCHAR(255) | NOT NULL |
| author\_id | INTEGER | FOREIGN KEY REFERENCES authors(id) |
| collection\_id | INTEGER | FOREIGN KEY REFERENCES collections(id) |
| year\_created | INTEGER |  |
| technique | VARCHAR(100) |  |
| genre | VARCHAR(100) |  |
| width\_cm | NUMERIC(5,2) |  |
| height\_cm | NUMERIC(5,2) |  |
| status | VARCHAR(100) |  |
| estimated\_value | NUMERIC(12,2) |  |
| image\_url | VARCHAR(255) |  |
| added\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Таблица users — Пользователи системы (опционально)**

| **Поле** | **Тип данных** | **Ограничения** |
| --- | --- | --- |
| id | SERIAL | PRIMARY KEY |
| username | VARCHAR(100) | UNIQUE, NOT NULL |
| password\_hash | VARCHAR(255) | NOT NULL |
| role | VARCHAR(50) | DEFAULT 'user' |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

Полный SQL-скрипт создания базы данных представлен в **Приложении A**

## **2.3. Алгоритмические решения**

Алгоритмическое обеспечение является важной частью любой информационной системы, так как именно в алгоритмах реализуется логика обработки данных, обеспечиваются операции взаимодействия с базой данных, фильтрация, сортировка, проверка корректности, валидация и бизнес-правила. В рамках разработки информационно-справочной системы управления коллекцией предметов искусства основные алгоритмы реализуются на языке Java, с использованием объектно-ориентированного подхода, принципов SOLID и стандартных библиотек Java, а также фреймворка Spring Boot на серверной стороне.

Одним из ключевых алгоритмов является обработка операций CRUD (Create, Read, Update, Delete). Для каждой сущности системы (произведение искусства, автор, коллекция) были реализованы соответствующие сервисные классы, отвечающие за создание новой записи, её редактирование, удаление и отображение. Работа с базой данных происходит через слой репозиториев (JpaRepository), предоставляемый библиотекой Spring Data JPA, что позволяет упростить реализацию и избежать ручного написания SQL-запросов.

Например, добавление нового произведения реализовано в виде HTTP POST-запроса от клиента. Пользователь вводит данные в интерфейсе JavaFX, после чего клиент формирует JSON-объект и отправляет его на сервер. На сервере запрос обрабатывается в контроллере, где происходит валидация данных (например, проверка, что поля «название», «автор» и «коллекция» не пустые). Затем вызывается метод сохранения, и объект сохраняется в базе данных. Ответ возвращается клиенту, где отображается сообщение об успехе или ошибке.

Важным алгоритмом является поиск и фильтрация объектов по нескольким критериям. В интерфейсе JavaFX реализованы поля для ввода условий фильтрации (например, по имени автора, периоду создания, жанру, технике исполнения). Клиент формирует соответствующий запрос к серверу с параметрами фильтра, и на сервере выполняется выборка из базы данных. Для построения таких запросов используется Specification API или методы, аннотированные @Query, что позволяет формировать SQL-подобные условия поиска на уровне Java-классов. Полученные результаты сортируются по выбранному критерию (например, по году создания или фамилии автора) и возвращаются клиенту.

Отдельно реализованы алгоритмы сортировки и агрегации данных. Пользователь может нажать на заголовок таблицы в интерфейсе и отсортировать список, например, по алфавиту или дате. Эти операции можно выполнять как на стороне клиента (через Comparator в JavaFX), так и на сервере — с помощью SQL-подобных запросов ORDER BY. Для статистической информации, например, отображения средней стоимости произведений или количества работ в каждой коллекции, используются агрегационные функции (AVG, COUNT, SUM), вызываемые через соответствующие репозитории или кастомные запросы.

Также в программе реализована обработка исключений и ошибок, что является обязательным требованием методических рекомендаций. Например, при вводе некорректных данных (пустые поля, недопустимые символы, отрицательные значения для размеров) отображается диалоговое окно с сообщением об ошибке, а само приложение не завершает работу. Это обеспечивается через try-catch блоки, кастомные исключения и валидационные аннотации на уровне классов (@NotNull, @Size, @Min, @Max).

Наконец, одним из дополнительных алгоритмов является формирование статистики и её визуализация. На клиенте реализована отдельная вкладка, где отображаются диаграммы (например, количество произведений по авторам или техникам), что построено с использованием библиотек JavaFX (PieChart, BarChart) и данных, полученных с сервера. Сервер формирует агрегированные данные, а клиент преобразует их в наглядные графики.

Таким образом, алгоритмы, реализованные в данной системе, обеспечивают полную функциональность, требуемую от информационно-справочной системы: от базовых CRUD-операций до поиска, сортировки, анализа и визуализации информации. Вся логика организована с соблюдением принципов модульности и повторного использования кода, что делает систему расширяемой и поддерживаемой.

## **2.4. Спецификации программного обеспечения**

Разрабатываемая информационно-справочная система построена по трехуровневой архитектуре, включающей клиентское приложение с графическим интерфейсом, серверную часть с REST API и базу данных. Программное обеспечение системы реализовано с использованием языка Java и набора проверенных технологий, соответствующих современным стандартам промышленной разработки.

Система разделена на три логических уровня:

1. Клиентский уровень (Frontend) — реализован с помощью JavaFX. Это десктопное Java-приложение, которое предоставляет пользователю визуальный интерфейс для взаимодействия с данными: формы ввода, таблицы, панели фильтрации, графики. Все действия пользователя (например, добавление нового произведения, фильтрация по автору) инициируют HTTP-запросы к серверу. Интерфейс разработан с учетом принципов эргономичности, минимализма и доступности.
2. Серверный уровень (Backend) — реализован на базе Spring Boot, популярного Java-фреймворка для построения веб-приложений. Сервер обрабатывает запросы, выполняет бизнес-логику, взаимодействует с базой данных через ORM и отправляет ответы клиенту в формате JSON. Все взаимодействие построено по архитектуре REST, где каждое действие (создание, удаление, получение, обновление) соответствует определённому HTTP-методу (POST, GET, PUT, DELETE).
3. Уровень базы данных (Data Layer) — используется СУБД PostgreSQL, в которой хранятся все данные о произведениях искусства, авторах, коллекциях и пользователях. Доступ к данным реализован через Spring Data JPA и Hibernate, что позволяет использовать объектно-ориентированный подход к работе с таблицами без необходимости писать SQL вручную. Все сущности описаны в виде Java-классов (Entity), а репозитории предоставляют готовые методы доступа к данным.

Используемые технологии и инструменты:

* Java 17 — основной язык программирования;
* JavaFX — библиотека для построения графического интерфейса;
* Spring Boot 3.x — фреймворк для реализации REST-сервера;
* Spring Data JPA / Hibernate — ORM-библиотеки для работы с БД;
* PostgreSQL — система управления базами данных;
* Lombok — для автоматизации генерации геттеров, сеттеров и конструкторов;
* Maven — система управления зависимостями и сборки проекта;
* Git / GitHub — для контроля версий и хранения кода;
* JavaDoc — для генерации документации;
* Jackson — библиотека для сериализации объектов в JSON и обратно;
* JUnit / Mockito — для модульного тестирования (в расширенной версии).

Примерная структура серверного проекта на Spring Boot выглядит следующим образом:

src/

└── main/

└── java/

└── com.artcollection/

├── controller/ ← REST-контроллеры

├── service/ ← бизнес-логика

├── repository/ ← интерфейсы доступа к БД

├── model/ ← классы-сущности (Entity)

└── dto/ ← классы для передачи данных

На клиентской стороне структура включает контроллеры окон (FXMLController), модели (FXModel) и UI-ресурсы (.fxml и .css).

Функциональность по модулям:

* Модуль авторов: просмотр, добавление, редактирование, удаление;
* Модуль коллекций: создание коллекции, назначение объектов;
* Модуль произведений: ввод всех параметров объекта искусства, фильтрация и сортировка;
* Модуль аналитики: вывод статистики и диаграмм;
* Модуль авторизации (опционально): авторизация и разграничение прав доступа.

Вся логика спроектирована таким образом, чтобы при необходимости можно было легко добавить новые функции — например, модуль выставок или учёт перемещений произведений.

## **2.5. Описание пользовательского интерфейса**

Пользовательский интерфейс системы был разработан с учётом требований удобства, наглядности и простоты восприятия. Основное внимание уделялось тому, чтобы все действия были интуитивно понятны даже для пользователя, не обладающего специальной технической подготовкой. Интерфейс реализован с использованием библиотеки JavaFX, которая позволяет строить современные графические формы с поддержкой визуальных компонентов, таких как таблицы, поля ввода, выпадающие списки, кнопки, вкладки и диаграммы.

После запуска приложения пользователь попадает на главный экран, на котором отображается таблица со списком всех произведений искусства, содержащихся в базе данных. Каждая строка в таблице соответствует одному объекту и содержит основные атрибуты, такие как название, автор, год создания, техника исполнения, жанр и статус. Над таблицей расположены поля фильтрации, где можно ввести условия поиска, например, выбрать определённого автора, задать диапазон годов или указать технику. Рядом расположена кнопка «Поиск», при нажатии на которую отображаются только те записи, которые соответствуют заданным условиям.

Для добавления нового произведения в систему пользователь нажимает кнопку «Добавить», после чего открывается отдельное окно. В этом окне реализована форма с полями для ввода всех необходимых данных: название, выбор автора из выпадающего списка, указание коллекции, год создания, жанр, техника, размеры, оценочная стоимость и ссылка на изображение. Все поля снабжены поясняющими подписями и проверками корректности. Если пользователь не заполнил обязательное поле или указал недопустимое значение, появляется предупреждение, а данные не отправляются на сервер до устранения ошибки. После успешного сохранения окно закрывается, и информация автоматически появляется в общей таблице.

Также реализована возможность редактирования информации. Для этого пользователь выбирает нужную строку в таблице и нажимает кнопку «Редактировать». Открывается та же форма, что и при добавлении, но с уже заполненными полями. После внесения изменений данные сохраняются и сразу отображаются в интерфейсе. Удаление записи осуществляется по нажатию на кнопку «Удалить» с подтверждением действия через всплывающее окно, что предотвращает случайные удаления.

В интерфейсе предусмотрена также отдельная вкладка, посвящённая аналитике. На этой вкладке пользователь может увидеть графики и диаграммы, например, распределение произведений по авторам, жанрам или коллекциям. Все диаграммы формируются автоматически на основе актуальных данных в базе и обновляются при каждом открытии раздела. Использование визуальных компонентов JavaFX позволяет представить статистику в наглядной форме, что особенно удобно для сотрудников музеев, кураторов и искусствоведов.

Дополнительно реализован пункт меню «О программе», в котором указана информация о разработчике, цели проекта и используемых технологиях. Этот пункт доступен через главное меню, как того требует методика выполнения курсовой работы. Интерфейс выполнен в спокойной цветовой гамме, с соблюдением требований эргономики: все кнопки логично размещены, элементы выровнены, формы не перегружены, а редко используемые функции вынесены в отдельные окна. Управление осуществляется с помощью мыши и клавиатуры, переход между окнами осуществляется плавно, без задержек.

В результате реализованный интерфейс полностью соответствует современным требованиям к программным продуктам учебного уровня. Он сочетает в себе простоту, функциональность и стабильность работы, что делает систему удобной для повседневного использования в музеях, галереях и других организациях, работающих с коллекциями предметов искусства.

## **2.6. Тестирование системы**

Тестирование программного обеспечения является обязательным этапом разработки, который позволяет убедиться в правильности функционирования всех компонентов системы, а также выявить возможные ошибки и недочёты до передачи приложения пользователю. В рамках данной курсовой работы тестирование проводилось вручную, в процессе поэтапного запуска клиентской и серверной частей, с фиксацией результатов работы программы при различных сценариях поведения пользователя.

Основное внимание при тестировании было уделено ключевым функциональным возможностям: добавлению новых записей, редактированию существующих данных, удалению объектов, фильтрации и поиску по заданным критериям, а также взаимодействию клиента и сервера по REST-протоколу. Каждая из этих операций проверялась с различными комбинациями вводимых данных, включая как корректные, так и заведомо ошибочные. Например, при попытке добавить объект искусства без обязательного указания названия система корректно выдавала предупреждающее сообщение, не позволяя отправить запрос на сервер. Аналогично, при указании недопустимого значения для года создания, такого как отрицательное число или текст, происходила валидация на клиентской стороне с отображением соответствующего сообщения в интерфейсе.

Проверка связи с сервером также выполнялась через сценарии запуска клиента без активного соединения, при недоступности базы данных или в случае неправильного формата запроса. В каждом из этих случаев приложение не завершало работу аварийно, а возвращало пользователю понятное описание проблемы и предлагало повторить операцию позже. Это особенно важно в условиях эксплуатации программы вне учебной среды, когда от её устойчивости может зависеть эффективность реального рабочего процесса.

Кроме того, тестировались сложные запросы фильтрации по множественным параметрам. Например, проверялась корректность вывода списка всех произведений определённого автора, созданных в заданном диапазоне лет, входящих в определённую коллекцию. Результаты выборки сравнивались с исходными данными в базе, что подтвердило точность алгоритмов обработки и логики фильтрации. Также отдельно проверялись функции отображения диаграмм, генерации статистики и взаимодействия с таблицей данных при включённой сортировке.

На этапе финального тестирования проводилась полная проверка функциональности в условиях максимально приближённых к реальным. Клиентское приложение запускалось на другой машине, соединялось с сервером через локальную сеть, выполнялось последовательное добавление и удаление данных, а затем — тест загрузки сохранённой информации при повторном запуске. Все основные модули системы показали устойчивую работу, не было зафиксировано сбоев или потери данных. Программа полностью соответствовала техническому заданию и требованиям методических указаний, включая наличие защиты от ошибок пользователя, сохранение данных между сессиями и корректную обработку исключений.

Таким образом, проведённое тестирование подтвердило, что система работает стабильно, обеспечивает выполнение всех заявленных функций и готова к передаче в эксплуатацию. Выявленные в процессе промежуточных испытаний недочёты были своевременно устранены, а итоговая версия программы соответствует необходимым требованиям качества, надёжности и удобства использования.

# **Заключение**

В ходе выполнения курсовой работы была разработана информационно-справочная система для управления коллекцией предметов искусства. Основное внимание было уделено проектированию серверной части, которая обеспечивает хранение и обработку данных о произведениях искусства, коллекциях и авторах. Также была реализована клиентская часть с использованием JavaFX для предоставления пользователю интерфейса взаимодействия с системой.

Проектирование базы данных было выполнено на основе реляционной модели с использованием PostgreSQL. В процессе работы была создана структура базы данных, которая включает таблицы для хранения данных о произведениях искусства, коллекциях и авторах. Также были реализованы связи между сущностями, что позволило обеспечить целостность данных и выполнение операций CRUD (создание, чтение, обновление, удаление).

Серверная часть системы была разработана с использованием фреймворка Spring Boot. С помощью Spring Data JPA была настроена работа с базой данных, а с использованием Spring REST API реализован обмен данными между клиентом и сервером. Таким образом, был обеспечен доступ к данным через стандартные HTTP-запросы, что позволяет легко интегрировать систему с другими приложениями.

Клиентская часть системы была реализована с использованием JavaFX. Созданный графический интерфейс позволяет пользователям работать с системой, осуществлять поиск и фильтрацию данных, добавлять и редактировать записи. Все действия пользователя отправляются на сервер, который обрабатывает запросы и возвращает результаты.

В процессе тестирования были проверены все функциональные возможности системы, такие как фильтрация и поиск по различным критериям, добавление и удаление данных. Все операции выполняются корректно, система работает стабильно и отвечает на запросы пользователей.

Задача разработки системы управления коллекцией предметов искусства была успешно решена. Полученная система может быть использована в реальной практике для учёта и анализа коллекций, произведений искусства и авторов.

# **Список использованных источников**

1. Фёдоров, А.В. **Современные информационные технологии: теория и практика** / А.В. Фёдоров. — М.: Издательский дом, 2020. — 272 с. — URL: https://www.labirint.ru/books/713313/
2. Глушков, В.М. **Основы работы с базами данных: учебник** / В.М. Глушков. — СПб.: Питер, 2019. — 416 с. — URL: https://www.piter.com/books/osnovy-rabotyi-s-bazami-dannyih
3. Колесников, И.Н. **Разработка программного обеспечения на платформе Java: учебное пособие** / И.Н. Колесников. — М.: ДМК Пресс, 2021. — 384 с. — URL: https://www.ozon.ru/context/detail/id/152487320/
4. Соловьёв, А.А. **Программирование на Java. Введение в JavaFX: учебник** / А.А. Соловьёв. — М.: БХВ-Петербург, 2018. — 224 с. — URL: https://www.bhv.ru/catalog/213443/
5. Sharma, R. **JavaFX 9: Building User Experience and Interfaces with Java** / R. Sharma. — Apress, 2017. — 320 p. — URL: https://www.apress.com/gp/book/9781484237391
6. **Spring Framework Documentation**. Spring Framework Reference Documentation, 2021. — URL: [https://docs.spring.io/spring-framework/docs/current/reference/html/](https://docs.spring.io/spring-framework/docs/current/reference/html/" \t "_new)
7. **PostgreSQL Official Documentation**. PostgreSQL 13 Documentation. — URL: [https://www.postgresql.org/docs/13/](https://www.postgresql.org/docs/13/" \t "_new)
8. Хейлс, Д. **Java. Полное руководство** / Д. Хейлс. — М.: ДМК Пресс, 2020. — 1008 с. — URL: https://www.ozon.ru/context/detail/id/141064509/
9. Кауфман, Д. **Сетевые технологии и базы данных** / Д. Кауфман. — М.: Наука, 2019. — 512 с. — URL: https://www.ozon.ru/context/detail/id/142424218/
10. Дубовик, Ю.В. **Разработка информационных систем с использованием Spring Boot** / Ю.В. Дубовик. — М.: Вильямс, 2018. — 240 с. — URL: [https://www.wiley.com/en-us/Developing+Spring+Boot+Applications+Using+Microservices+Design+Patterns-p-9781119310587](https://www.wiley.com/en-us/Developing+Spring+Boot+Applications+Using+Microservices+Design+Patterns-p-9781119310587" \t "_new)
11. Байков, И.Е. **Основы проектирования баз данных** / И.Е. Байков. — М.: Издательский дом, 2017. — 350 с. — URL: https://www.labirint.ru/books/635267/
12. Смит, Э. **JavaFX 8. Разработка приложений для настольных ПК и мобильных устройств** / Э. Смит. — М.: Вильямс, 2017. — 720 с. — URL: [https://www.wiley.com/en-us/JavaFX+8+Essentials-p-9781118949055](https://www.wiley.com/en-us/JavaFX+8+Essentials-p-9781118949055" \t "_new)
13. **Postman Documentation**. Postman API Documentation, 2021. — URL: https://www.postman.com/docs/
14. **Oracle. JavaFX Documentation**. — URL: [https://docs.oracle.com/javafx/2/overview/jfxpub-2.htm](https://docs.oracle.com/javafx/2/overview/jfxpub-2.htm" \t "_new)

# **Приложения**

**Приложение А**

Создание схемы базы данных

CREATE TABLE collections (

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

description TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE authors (

id SERIAL PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

birth\_year INTEGER,

death\_year INTEGER,

country VARCHAR(100),

art\_style VARCHAR(100),

biography TEXT

);

CREATE TABLE artworks (

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author\_id INTEGER REFERENCES authors(id) ON DELETE SET NULL,

collection\_id INTEGER REFERENCES collections(id) ON DELETE SET NULL,

year\_created INTEGER,

technique VARCHAR(100),

genre VARCHAR(100),

width\_cm NUMERIC(5,2),

height\_cm NUMERIC(5,2),

status VARCHAR(100),

estimated\_value NUMERIC(12,2),

image\_url VARCHAR(255),

added\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Таблица пользователей (опционально)

CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

role VARCHAR(50) DEFAULT 'user',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);