Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых» (ВлГУ)

Кафедра информационных систем и программной инженерии

**КУРСОВОЙ ПРОЕКТ**

по дисциплине

«Распределённые программные системы»

на тему:

Программная система «Виртуальная арт-галерея». Модуль «Эксплуатация»

Выполнил: студент гр. ПРИ-122

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шутова Т.Е.

(подпись)

Принял: доц. каф. ИСПИ

\_\_\_\_\_\_\_\_\_\_\_\_\_ Проскурина Г.В.

(подпись)

Дата сдачи: \_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_

Владимир, 2025

**АННОТАЦИЯ**

В данном курсовом проекте производилась разработка приложения для программной системы «Виртуальная арт-галерея». Модуль «Эксплуатация». Пояснительная записка содержит разделы: введение, описание предметной области, проектирование приложение для ПС, реализация приложения для ПС, нагрузочное тестирование и оптимизация, заключение, список используемой литературы, приложение А, приложение Б, приложение В.

Пояснительная записка содержит 44 листа, в том числе 28 рисунков, 4 использованных источника, 3 приложения.

**ANNOTATION**

In this course project, an application was developed for the Virtual Art Gallery software system. The "Operation" module. The explanatory note contains the following sections: introduction, description of the subject area, design of the PS application, implementation of the PS application, load testing and optimization, conclusion, list of references, Appendix A, Appendix B, Appendix B.

The explanatory note contains 44 sheets, including 28 figures, 4 sources used, 3 appendices.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc198501360)

[ЭТАП 9. НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ 5](#_Toc198501361)

[9.1. Нагрузочное тестирование и оптимизация 5](#_Toc198501362)

1 ВВЕДЕНИЕ

С развитием цифровых технологий и их активным внедрением в различные сферы жизни, искусство претерпевает значительные изменения. Одной из ключевых тенденций является переход к виртуальным форматам взаимодействия с произведениями искусства, что особенно актуально в условиях глобализации и развития удаленных коммуникаций. Создание виртуальных арт-галерей становится важным инструментом для популяризации искусства, расширения аудитории и предоставления пользователям доступа к культурному наследию, независимо от их географического положения.

В данной работе рассматривается процесс разработки виртуальной арт-галереи, представляющей собой программный продукт, созданный с использованием современных технологий. Главной целью проекта является обеспечение удобного и интерактивного доступа к публикациям. Разработка приложения будет осуществляться с использованием стека технологий, обеспечивающего высокую производительность, надежность и масштабируемость.

В рамках проекта будут проведены анализ требований к системе, проектирование архитектуры приложения, а также реализация его функционала, включающего управление публикациями, выставками и профилями художников. Приложение будет ориентировано на создание уникального пользовательского опыта, что сделает виртуальную арт-галерею важным инструментом для популяризации искусства в цифровую эпоху.

2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

# 2.1 Цель и задачи

Цель работы: разработать программную информационную систему «Виртуальная арт-галерея» с функционалом, связанным преимущественно с областью эксплуатации.

Для достижения поставленной цели необходимо реализовать следующие функциональные требования:

* Авторизация и регистрация (администратор, художник, посетитель);
* Система просмотра публикаций, выставок, художников (администратор, художник, посетитель);
* Отображение публикаций и выставок в профиле художника (администратор, художник);
* Публикация работ (администратор, художник);
* Создание приватных и открытых выставок и их редактирование (администратор, художник);
* Участие в публичных и приватных выставках (администратор, художник).

А также необходимо решить следующие задачи:

* Анализ предметной области;
* Проектирование системы;
* Проектирование базы данных;
* Реализация системы и интеграция базы данных;
* Тестирование и доработка системы;
* Составление пояснительной записки.

# 2.2 Описание предметной области

Для художников необходимо иметь платформу, на которой можно размещать собственные работы с целью получения известности или поиска работодателя. Представителям компаний, в чьи ряды требуется графический художник той или иной специальности, важно иметь на руках портфолио кандидата, и именно платформы с графическими публикациями не только упрощают поиск, но и дают доступ к работам и ценам на услуги художника.

Виртуальная арт-галерея – это как раз платформа для художников и их нанимателей. Яркий пример – зарубежный сайт artstation.com. Российского аналога нет, следовательно, требуется заняться его разработкой.

# 2.3 Словарь предметной области

*Посетитель* – незарегистрированный или неавторизованный пользователь. Имеет минимальные права, может просматривать работы, художников и выставки и пользоваться поиском. Также он может авторизоваться или зарегистрироваться.

Атрибуты: нет.

*Художник* – зарегистрированный и прошедший авторизацию пользователь. Художнику не обязательно иметь в своём профиле работы, это только роль, присваиваемая при регистрации. Художник имеет те же возможности, что и посетитель, а также он может публиковать работы, устраивать выставки, создавать прайс-листы, заказывать услуги, участвовать в выставках, распоряжаться своим профилем и управлять собственными записями.

Атрибуты: ID, никнейм, пароль, почта.

*Администратор* – зарегистрированный пользователь, имеющий доступ к редактированию содержимого чужих публикаций, чужих выставок и художников. Администратор следит за соблюдением правил платформы, удаляет нарушения, принимает или отклоняет публикации, а также может позиционировать себя как художник и обладает всеми теми же функциями.

Атрибуты: ID, никнейм, пароль, почта.

*Публикация, или Работа, или Картина* – художественная работа определённого автора, располагающаяся либо внутри, либо вне выставки.

Атрибуты: ID, название, описание, изображение, статус публикации, дата создания, категория, автор, привязка к выставке, количество лайков.

*Выставка* – собранная по определенному критерию коллекция работ художника или художников, доступная для редактирования заранее указанному кругу лиц.

Атрибуты: ID, название, описание, приватность, публикации, создатель.

*Профиль художника или Профиль пользователя* – страница отображения информации о художнике, его публикациях и выставках.

Атрибуты: ID, никнейм, почта, публикации, выставки.

*Галерея* – главная страница системы, на которой отображается самое необходимое для среднестатистического пользователя: публикации, художники и выставки.

Атрибуты: публикации, художники, выставки.

3 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ ДЛЯ ИС

# 3.1 Диаграмма прецедентов

Диаграмма прецедентов [1] представлена на рисунке 1.

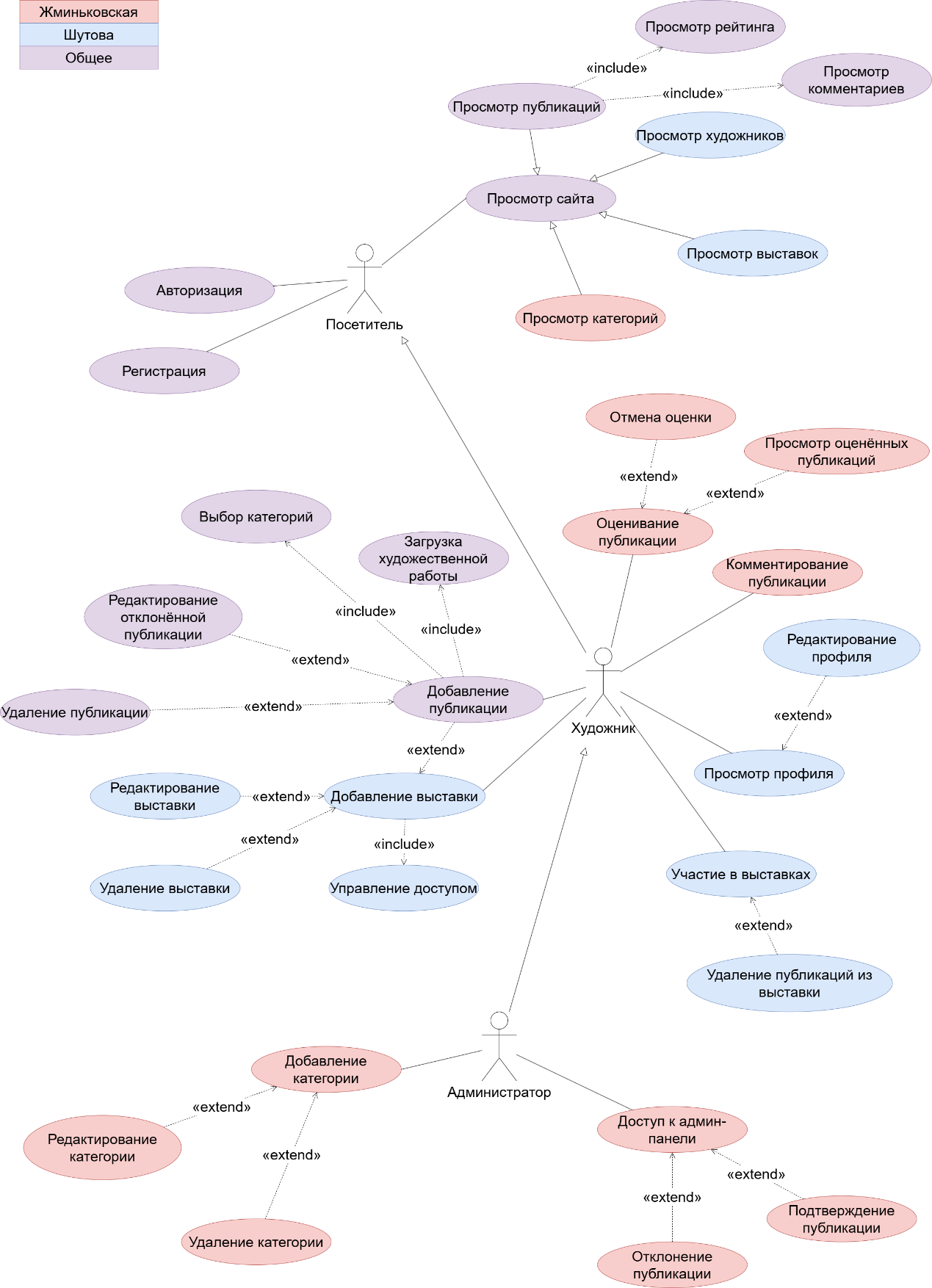


Рисунок 1. Диаграмма прецедентов

# 3.2 Описание прецедентов

|  |
| --- |
| Расширенное описание прецедента «Добавление публикации» |
| **Название:** «Добавление публикации»  **Предусловие:** нет  **Действующее лицо:** художник, администратор  **Основной поток:** *Добавление публикации*  Пользователь нажимает на кнопку «Добавить публикацию» и переходит в форму для публикации.  Обязательно нужно загрузить изображение, выбрать категории, ввести название и описание.  Работе присваивается статус «Отправлена на модерацию», она становится доступной для просмотра только создателю публикации и администраторам в админ-панели.  **Альтернативный поток:** *нет*  **Постусловие:** пользователя автоматически перебрасывает на страницу со всеми публикациями. |

|  |
| --- |
| Расширенное описание прецедента «Добавление выставки» |
| **Название:** Добавление выставки  **Предусловие:** нет  **Действующее лицо:** художник, администратор  **Основной поток:** *Добавление выставки*  Пользователь нажимает на кнопку «Создать выставку», после чего его перебрасывает на страницу с заполняемой формой.  В выставке заполняются все необходимые поля и присваивается приватность или публичность.  Пустая выставка автоматически добавляется на сайт и в профиль художника.  **Альтернативный поток:** *нет*  **Постусловие:** пользователя автоматически перебрасывает на страницу со всеми выставками. |

# 3.3 Диаграмма классов

Диаграмма классов [2] представлена на рисунке 2.

Рисунок 2. Диаграмма классов

# 3.4 Диаграмма состояний

Диаграмма состояний [3] для объекта «Выставка» представлена на рисунке 3.

**Описание диаграммы состояний для объекта «Выставка»**

**1) Выставка создана/обновлена**

Как для создания выставки, так и для редактирования, пользователю открывается форма заполнения. После введения всех нужных данных выставка создаётся/обновляется после нажатия на соответствующую кнопку

**2) Редактирование выставки**

Если необходимо внести изменения в названии, описании или приватности выставки, создатель выставки может сделать это в любой момент.

**3) Выставка обновлена с учётом новых/удалённых публикаций**

Если выставка приватна, публикации в неё может добавлять только создатель выставки. Если публична – любой авторизованный пользователь. При добавлении новой или существующей публикации к выставке она прикрепляется к выставке вне зависимости от её статуса. При удалении публикации из выставки удаляется только запись из связующей таблицы БД.

**4) Выставка функционирует (цикл)**

Пока выставка не удалена, её можно редактировать и в неё можно добавлять сколько угодно новых публикаций.

**5) Подтверждение удаления**

Когда создатель выставки или администратор решает, что выставка должна быть удалена, после подтверждения о своём решении выставка удаляется, однако работы остаются в профиле пользователей, но те уже не привязаны к выставке.

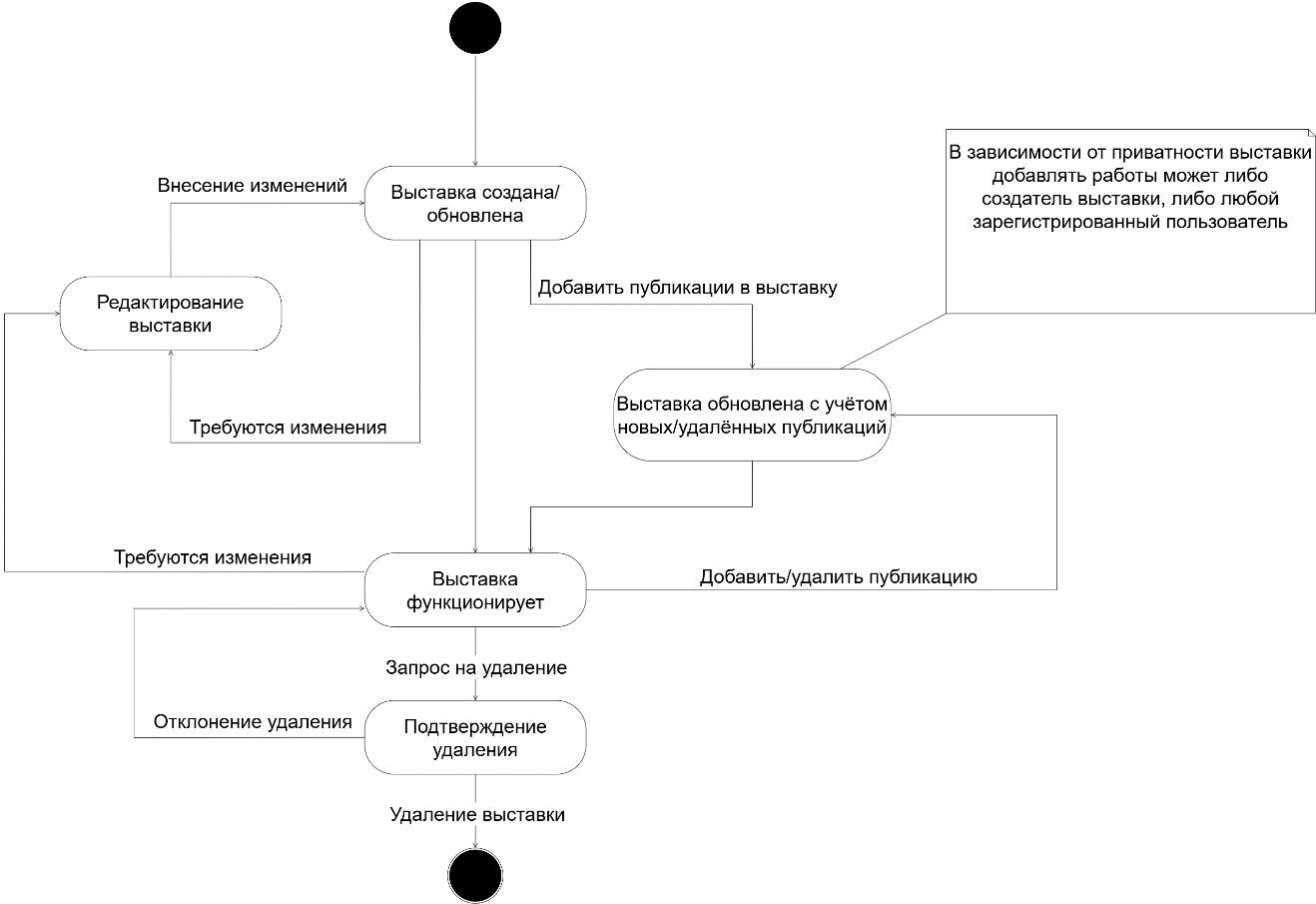


Рисунок 3. Диаграмма состояний

# 3.5 Диаграмма последовательностей

Диаграмма последовательностей [4] для объекта «Выставка» представлена на рисунке 4.

**Описание диаграммы последовательностей объекта «Выставка».**

1. Художник создаёт выставку на форме создания выставки.

2. Запрос передаётся в контроллер.

3. Объект выставки создаётся.

4. Данные выставки заполнены.

5. Выставка отображается в веб-приложении.

6. Художник заходит в детали выставки и хочет добавить выставку.

7. В таблицу связей публикации и выставки добавляется новая запись.

8. Публикация добавляется в выставку.

9. Выставка обновлена.

10. Те же операции по добавлению публикаций могут проделывать другие пользователи, если выставка публична.

11. Если художнику нужно удалить выставку, он делает запрос на удаление.

12. Публикации в выставке безопасно открепляются от неё.

13. Выставка удаляется.

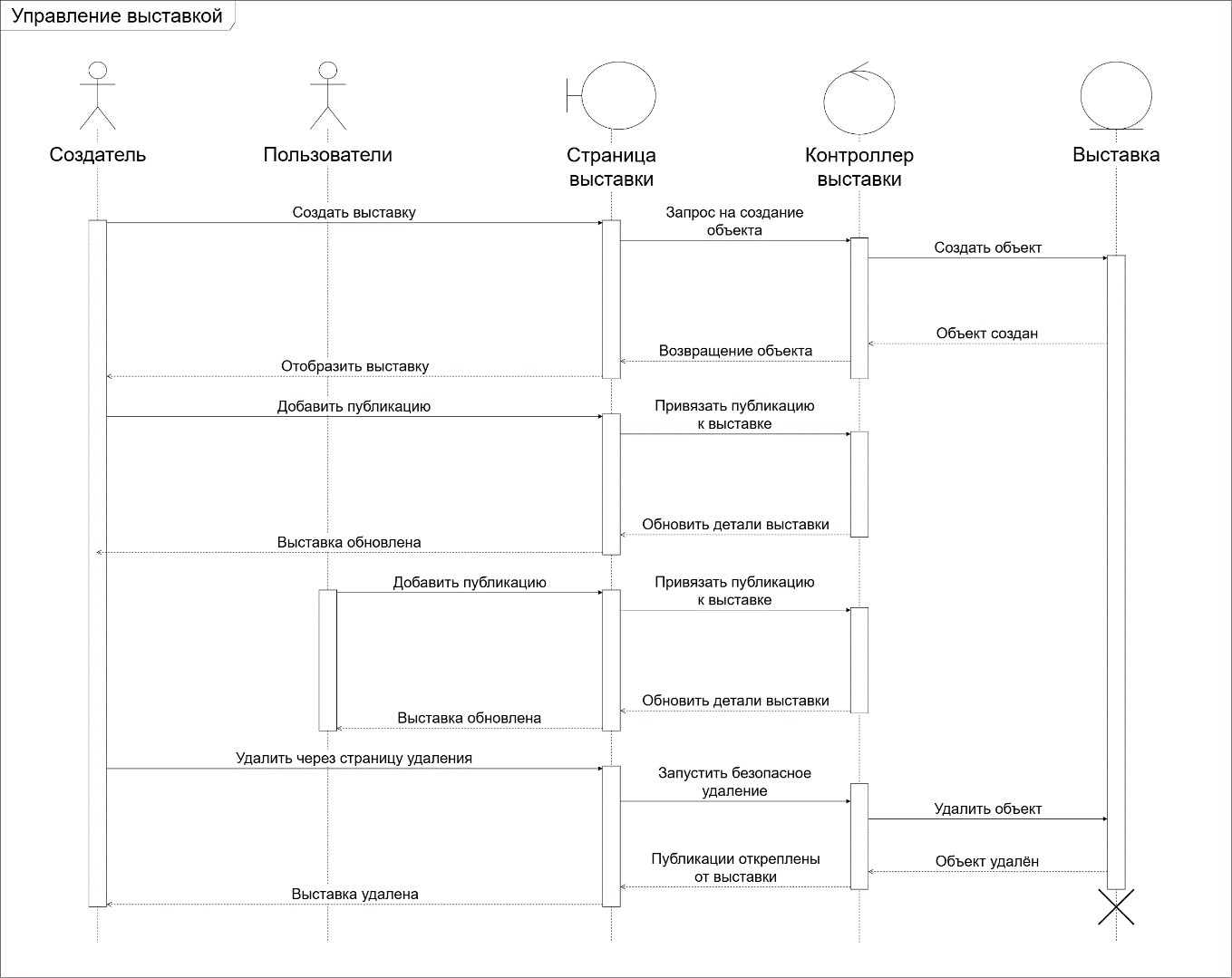


Рисунок 4. Диаграмма последовательностей

# 3.6 Схема бизнес-процесса (BPMN)

Схема бизнес-процесса [5] «Управление выставкой» представлена на рисунке 5.

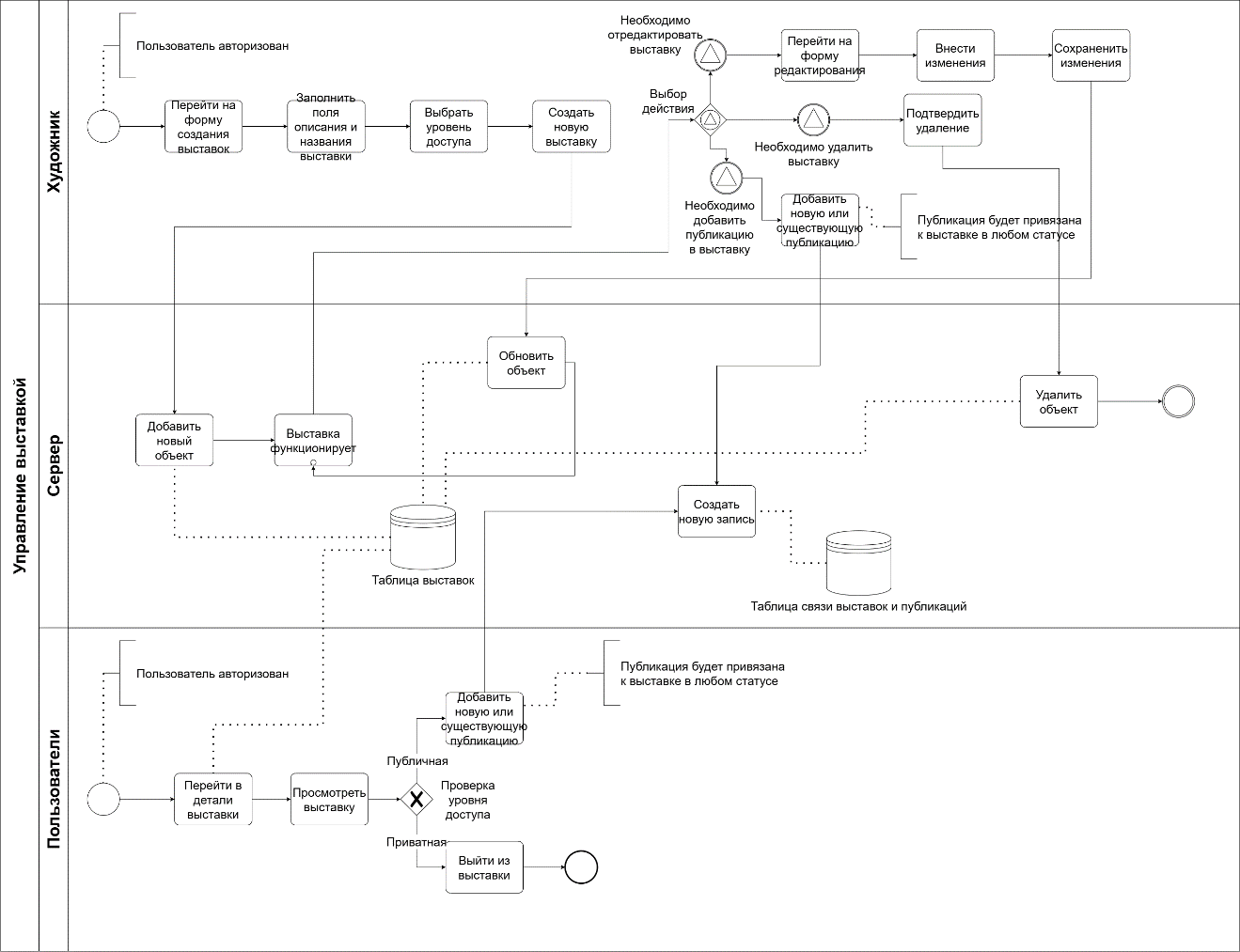


Рисунок 5. Схема бизнес-процесса

4 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ ДЛЯ ИС

# 4.1 Разработка базы данных

Для разработки системой управления базами данных была выбрана технология MySQL [6]. На рисунке 6 представлена ER-диаграмма [7].

На диаграмме представлено 9 таблиц:

* artworks – для записей публикаций;
* exhibitions – для записей выставок;
* exhibition\_artworks – для записей публикаций, прикреплённых к выставкам;
* comments – для записей комментариев;
* users – для записей пользователей;
* roles – для записей ролей пользователей;
* likes – для записей лайков на публикациях;
* categories – для записей категорий;
* artwork\_category – для записей категорий, указанных в публикациях.

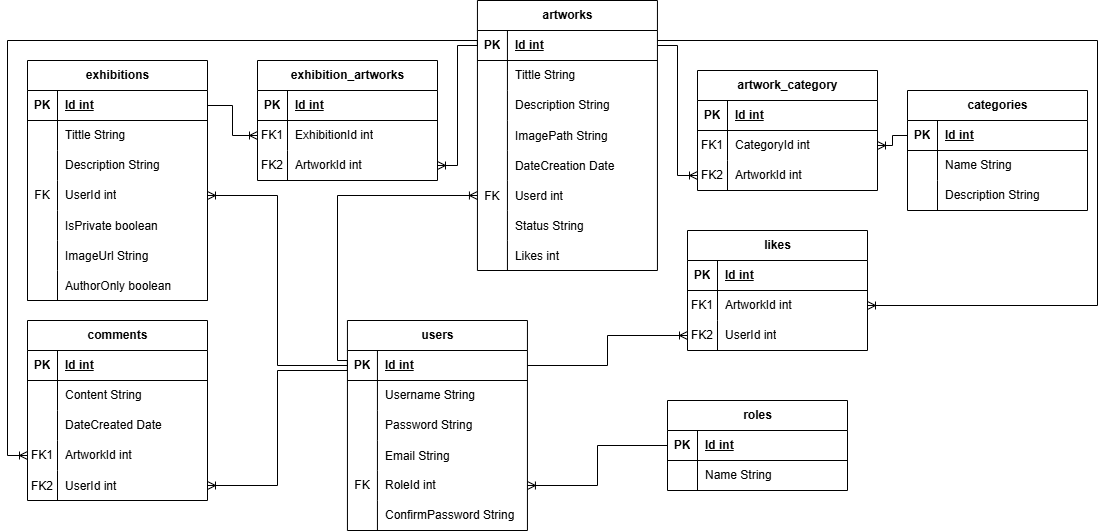


Рисунок 6. ER-диаграмма

# 4.2 Разработка веб-приложения

Для реализации веб-приложения были использованы зависимости.

**Spring Framework [8]**

* **spring-web**: Обеспечивает базовую интеграцию Spring с веб-приложениями (HTTP-клиенты, REST-поддержка).
* **spring-webmvc**: Реализует MVC-паттерн для веб-приложений (контроллеры, представления).
* **spring-orm**: Интеграция с ORM-библиотеками (Hibernate, JPA).
* **spring-tx**: Управление транзакциями (аннотации @Transactional).
* **spring-context**: Ядро Spring (DI, IoC-контейнер, аннотации @Component, @Autowired).
* **spring-data-jpa**: Упрощает работу с JPA (репозитории, автоматические запросы).
* **spring-data-commons**: Общие компоненты Spring Data (Pagination, CrudRepository).

**Spring Security [9]**

* **spring-security-web**: Защита веб-приложений (фильтры, аутентификация, авторизация).
* **spring-security-config**: Конфигурация через аннотации и XML (класс WebSecurityConfigurerAdapter).
* **spring-security-core**: Базовые компоненты безопасности (роли, права, шифрование).

**Thymeleaf [10]**

* **thymeleaf**: Шаблонизатор для HTML (динамические шаблоны на стороне сервера).
* **thymeleaf-spring5**: Интеграция с Spring 5 (поддержка Spring-выражений, MVC).
* **thymeleaf-extras-springsecurity5**: Дополнения для Spring Security (отображение данных пользователя).
* **thymeleaf-extras-java8time**: Поддержка Java 8 Time API в шаблонах (форматирование дат).

**Базы данных**

* **hibernate-core**: ORM-фреймворк (маппинг сущностей, HQL).
* **hibernate-validator**: Валидация данных через аннотации (например, @NotNull).
* **mysql-connector-java**: Драйвер для подключения к MySQL.
* **HikariCP**: Пул соединений для БД (высокая производительность).

**JPA [11]**

* **javax.persistence-api**: Спецификация JPA (аннотации @Entity, @Table).

**Веб и Servlet**

* **javax.servlet-api**: API для работы с сервлетами (классы HttpServlet, Filter).
* **javax.servlet.jsp-api**: API для JSP-страниц.
* **jstl**: Теги для JSP (упрощение логики в шаблонах).

**Утилиты**

* **commons-fileupload**: Загрузка файлов на сервер (обработка multipart/form-data).
* **lombok**: Генерация кода (геттеры, сеттеры через аннотации @Data).
* **javax.annotation-api**: Аннотации общего назначения (например, @PostConstruct).

**Логирование**

* **slf4j-api**: Интерфейс для логирования (абстракция над реализациями).
* **slf4j-simple**: Простая реализация SLF4J (логи в консоль).

**Tomcat**

* **tomcat-jdbc**: Реализация DataSource для Tomcat (пул соединений).

**Прочее**

* **maven-antrun-plugin**: Плагин Maven для выполнения Ant-задач (например, копирование WAR-файла).
* **maven-war-plugin**: Сборка WAR-архива.
* **maven-compiler-plugin**: Настройка компиляции (Java 16).

# 4.3 Разработка приложения для информационной системы

Для разработки использовался язык программирования Java. Были разработаны модели, контролеры, репозитории, сервисы и представления.

**Модели:**

1. Artwork – модель для хранения публикаций.
2. Exhibition – модель для хранения выставок.
3. Role – модель для хранения ролей пользователей.
4. User – модель профилей пользователей.

**Представления:**

1. index – просмотр галереи.
2. artwork
   1. create – создание публикации;
   2. details – детали публикации;
   3. edit – редактирование публикации со статусом «REJECTED»;
   4. list – просмотр всех публикаций.
3. exhibition
   1. add-exist-artwork – добавление существующей публикации к выставке;
   2. create – создание выставки;
   3. details – детали выставки с возможностью добавить в неё публикацию;
   4. edit – редактирование названия, описания и приватности выставки;
   5. list – просмотр всех выставок.
4. user
   1. liked – просмотр понравившихся публикаций;
   2. list – просмотр всех пользователей;
   3. profile – просмотр профиля пользователя;
   4. settings – редактирование профиля пользователя.
5. auth
   1. access-denied – страница запрещённого доступа;
   2. login – страница авторизации;
   3. register – страница регистрации.

**Контроллеры:**

1. ArtworkController – контроллер для работы с публикациями;
2. AuthController – контроллер аутентификации пользователей;
3. ExhibitionController – контроллер для работы с выставками;
4. HomeController – контроллер с реализацией отображения публикаций, художников и выставок;
5. ProfileController – контроллер для работы с профилями пользователей.

**Репозитории:**

1. ArtworkRepository – работа с публикациями;
2. ExhibitionRepository – работа с выставками;
3. RoleRepository – работа с ролями;
4. UserRepository – работа с пользователями.

**Сервисы:**

1. ArtworkService – бизнес-логика для публикаций;
2. CategoryService – бизнес-логика для категорий;
3. CustomUserDetailsService – бизнес-логика профилей пользователей;
4. ExhibitionService – бизнес-логика для выставок;
5. UserService – бизнес-логика для управления пользователями.

# 4.4 Диаграмма компонентов

Диаграмма компонентов [12] представлена на рисунке 7, на которой показана связь всех компонентов системы.

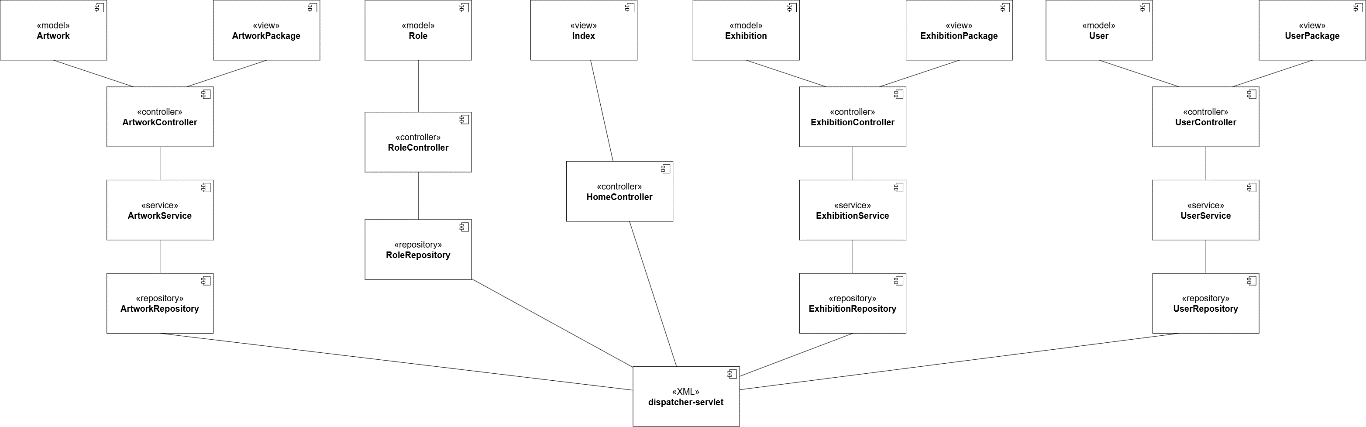


Рисунок 7. Диаграмма компонентов

# 4.5 Диаграмма развёртывания

Диаграмма развёртывания [12] представлена на рисунке 8.

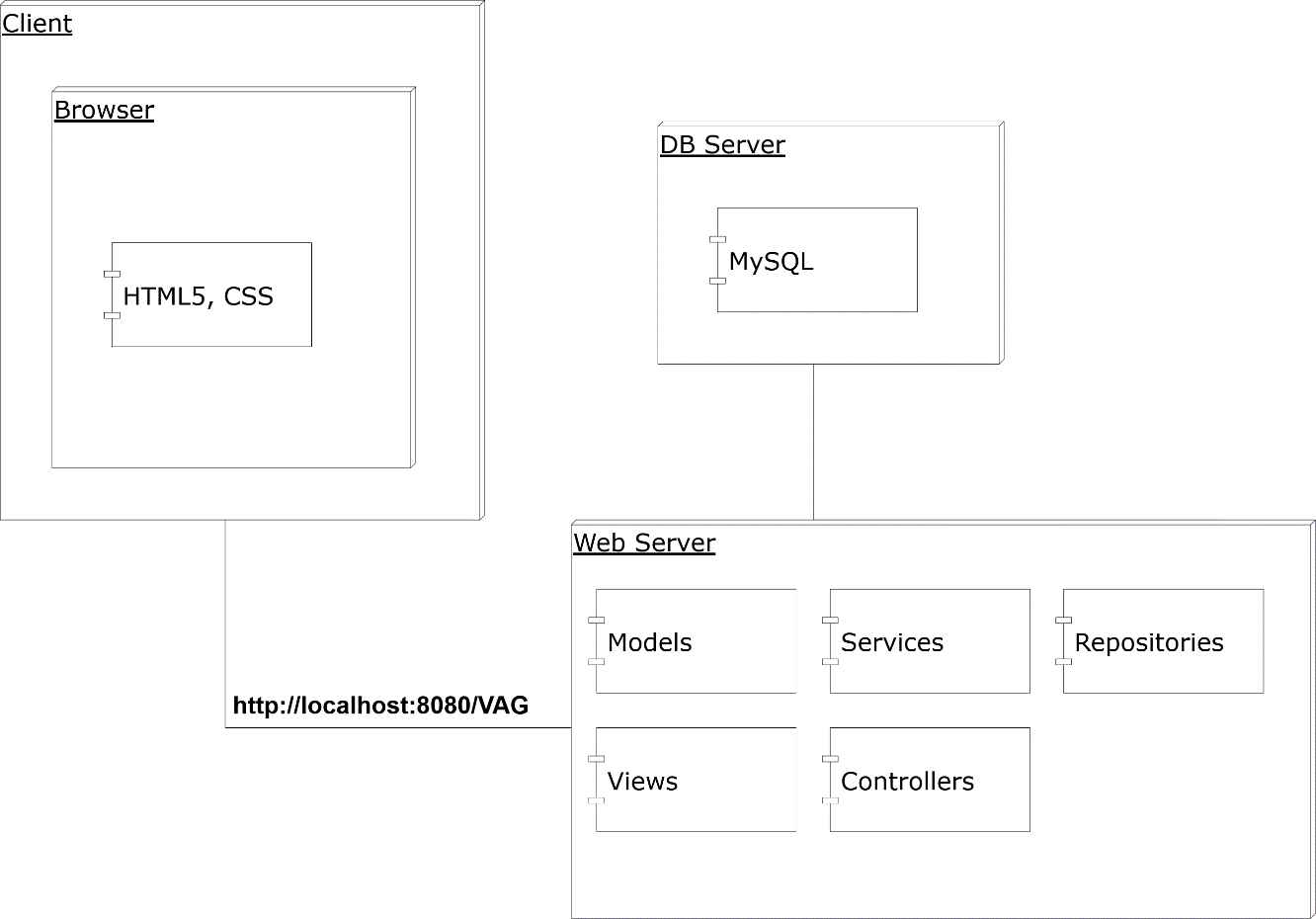


Рисунок 8. Диаграмма развёртывания

# 4.6 Репозиторий проекта

Ссылка на репозиторий: <https://github.com/ShutovaTE/VAG-Java>

# 4.7 Реализация

Основной код публикаций, выставок и пользователей представлен в приложениях А, Б и В соответственно.

5 НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ

Для того чтобы определить производительность разрабатываемой системы необходимо провести нагрузочное тестирование с постепенным увеличением нагрузки. В качестве объектов тестирования были выбраны объекты «Выставка» и «Комментарий». Нагрузочное тестирование позволит выявить уязвимые места в быстром добавлении выставок и комментариев различными пользователями, выполняющими действия одновременно.

Нагрузочное тестирование проводится с помощью инструмента Apache JMeter. Были определены следующие параметры потока:

* количество пользователей в потоке – 10;
* период запуска потоков – 10 секунд;
* количество итераций – 100;
* единственный пользователь для всех итераций.

Объекты «Выставка» и «Комментарий» будут подвержены нагрузочному тестированию согласно последовательности действий, описанной ниже.

1. GET-запрос на страницу авторизации. Необходимо для того, чтобы авторизовать пользователя.
2. POST-запрос на авторизацию. Авторизуются 10 различных пользователей, описанных в CSV-файле, созданном заранее.
3. GET-запрос на переход к деталям публикации с ID 8.
4. POST-запрос на добавление комментария. В параметрах комментария вводится текст комментария.
5. GET-запрос на страницу отображения всех существующих выставок. Один из вариантов перехода к функции «Создать выставку».
6. GET-запрос на переход на форму создания выставки.
7. POST-запрос на создание выставки. В параметрах выставки указывается приватность выставки (приватная), описание выставки и название выставки.
8. GET-запрос на страницу отображения всех существующих выставок. Необходимо для проверки быстрой загрузки данных.
9. GET-запрос на детали конкретной выставки. Необходимо для проверки быстрой загрузки всех данных выставки.

Нагрузочное тестирование с указанными выше условиями и параметрами было успешно проведено без перебоев, ошибок и провальных попыток исполнения запроса (рис. 1, 2).

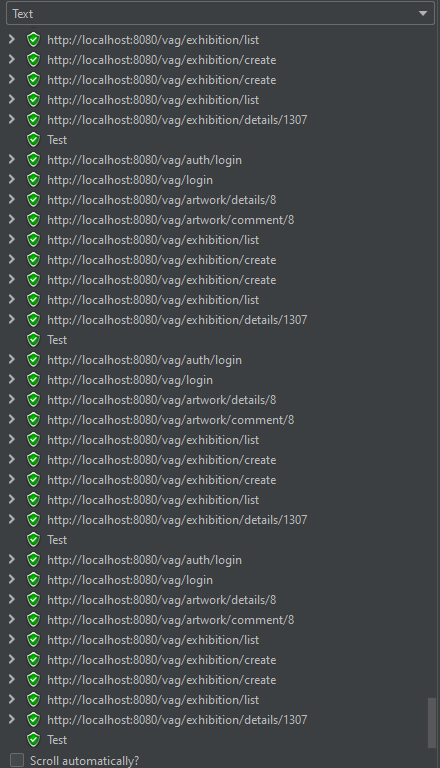


Рисунок 1 – Нагрузочное тестирование выставок и комментариев

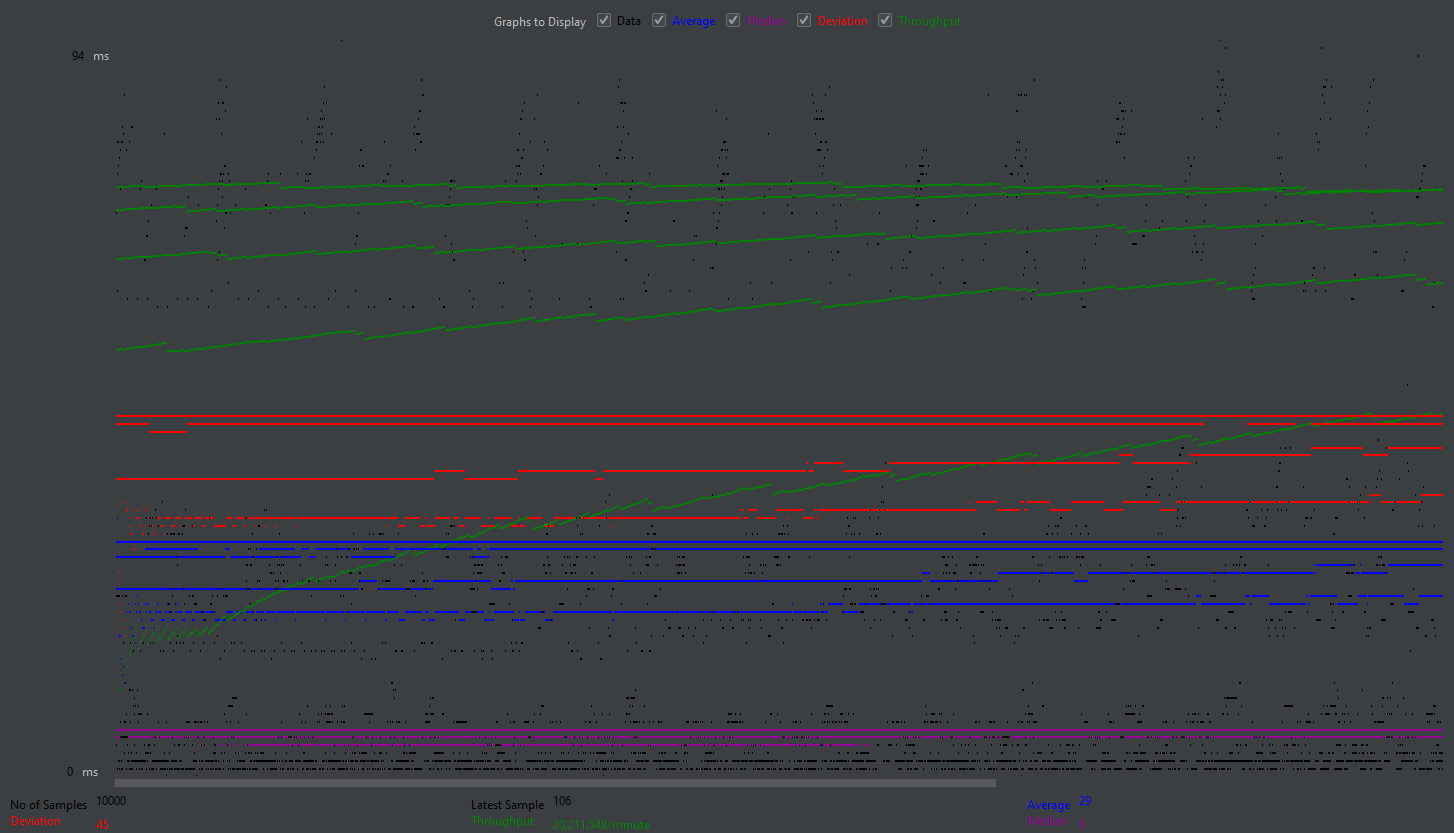


Рисунок 2. Графический результат тестирования

Нагрузочное тестирование показало, что система достаточно оптимизирована для текущих требований. Проект не требует доработки для оптимизации. Для успешного прохождения тестирования в проекте имеются следующие аспекты:

* Эффективная работа с комментариями.
* Оптимизация управления выставками.
* Параллельная обработка запросов (независимая обработка запросов).
* Оптимизированная загрузка списков с помощью EntityGraph и пагинации.
* Эффективное управление транзакциями.
* Оптимизированное отображение деталей с помощью Join Fetch.

6 ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А. ПУБЛИКАЦИИ

ПРИЛОЖЕНИЕ Б. ВЫСТАВКИ

ПРИЛОЖЕНИЕ В. ПОЛЬЗОВАТЕЛИ

ПРИЛОЖЕНИЕ Г.