

**Министерство образования Иркутской области**  
Государственное бюджетное профессиональное образовательное учреждение  
Иркутской области  
«Иркутский авиационный техникум»  
(ГБПОУИО «ИАТ»)

**КП.09.02.07.23.221.19 ПЗ**

**ВЕБ-ПРИЛОЖЕНИЕ «КИНОТЕАТР»**

Председатель ВЦК:	_____	(Н.Р. Огородникова)
	(подпись, дата)	
Руководитель:	_____	(Е.С. Кубата)
	(подпись, дата)	
Студент:	_____	(В.М. Седых)
	(подпись, дата)	

Иркутск 2025

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Описание предметной области.....	5
1.1 Описание предметной области.....	5
2 Обзор инструментальных средств для разработки.....	8
3 Техническое задание .....	14
4 Проектирование веб-приложения .....	15
4.1 Структурная схема веб-приложения.....	15
4.2 Функциональная схема веб-приложения.....	19
4.3 Проектирование базы данных .....	26
4.4 Проектирование интерфейса .....	33
5 Разработка веб-приложения.....	40
5.1 Разработка интерфейса веб-приложения.....	40
5.2 Разработка базы данных веб-приложения.....	42
5.3 Разработка веб-приложения.....	46
6 Тестирование веб-приложения .....	49
6.1 Тестовые сценарии .....	49
6.2 Методы тестирования.....	53
7 Документирование веб-приложения.....	56
7.1 Руководство по установке веб-приложения.....	56
7.2 Руководство гостя.....	57
7.3 Руководство авторизованного пользователя.....	60
7.4 Руководство администратора .....	63
ЗАКЛЮЧЕНИЕ .....	72
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	75
Приложение А – Техническое задание .....	77
Приложение Б – Листинг Urls.....	87

					КП.09.02.07.23.201.09.ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	<div>ВЕБ-ПРИЛОЖЕНИЕ «КИНОТЕАТР» пояснительная записка</div>						
Разраб.		Седых В.М.									
Провер.		Кубата Е.С.									
Реценз.											
Н. Контр.											
Утверд.											
					Лит.		Лист		Листов		
							2		88		
					ГБПОУИО «ИАТ» БД-22-1						

## ВВЕДЕНИЕ

В условиях активной цифровой трансформации сферы услуг и развлечений автоматизация ключевых бизнес-процессов перестала быть опцией и стала критически важным элементом конкурентоспособности. Особенно это касается индустрии кинопоказа, где удобство, скорость и безопасность обслуживания напрямую влияют на общую эффективность работы предприятия. Разрабатываемое в рамках данного курсового проекта веб-приложение «Кинотеатр» призвано устранить очереди в кассу и оптимизировать внутренние операционные процессы за счет внедрения комплексной системы онлайн-бронирования и покупки билетов.

Актуальность проекта обусловлена комплексом факторов, определяющих развитие рынка сегодня:

- Повышенный спрос на бесконтактные и дистанционные сервисы. Потребители хотят планировать посещение и оплачивать услуги онлайн;
- Необходимость снижения операционных издержек. Сокращение ошибок, связанных с человеческим фактором;
- Потребность в аналитике для управленческих решений. Руководство кинотеатра получает в руки инструмент для мониторинга ключевых показателей.

Целью курсового проекта является разработка полнофункционального, безопасного и удобного веб-приложения для автоматизации процесса бронирования и продажи билетов в кинотеатре.

Для достижения поставленной цели в проекте решен следующий комплекс задач:

1. Проведение предпроектного исследования, включающего анализ предметной области и сравнительный анализ современных инструментальных средств разработки для обоснованного выбора технологического стека.
2. Проектирование архитектуры веб-приложения на основе стандартов UML и IDEF0: разработка диаграмм вариантов использования, деятельности, компонентов, развертывания, а также функциональных моделей и инфологической модели данных.

					КП.09.02.07.23.221.19.ПЗ	Лист 3
Изм.	Лист	№ докум.	Подпись	Дата		

3. Разработка технического задания в соответствии со стандартом ISO/IEC/IEEE 29148:2011, четко формализующего все требования к функционалу, интерфейсу и условиям эксплуатации программного продукта.

4. Реализация логики серверной части веб-приложения, включая:

- Создание моделей данных для основных сущностей;
- Разработку системы аутентификации и авторизации пользователей;
- Реализацию модуля управления для администратора;
- Создание интерактивного механизма выбора мест на схеме зала с отображением их статуса;
- Организацию процесса формирования заказа, интеграцию с платежной системой и генерацию электронных билетов с QR-кодами.

5. Разработка пользовательского интерфейса, обеспечивающего интуитивно понятное взаимодействие для клиента и администратора.

6. Проектирование и реализация реляционной базы данных, обеспечивающей целостность, надежность хранения и эффективность выполнения запросов.

7. Тестирование и отладка веб-приложения, документирование проекта в виде руководства пользователя и настоящего пояснительной записки.

Технологический стек проекта, выбранный по результатам сравнительного анализа, включает:

- Серверная часть: Python, фреймворк Django;
- База данных: PostgreSQL;
- Клиентская часть: HTML5, CSS3;
- Среда разработки: PyCharm Professional;
- Средства проектирования: Draw.io, MySQL Workbench.

# 1 Описание предметной области

## 1.1 Описание предметной области

Предметной областью курсового проекта является современный многозальный кинотеатр. Кинотеатр – это комплексное общественное заведение, основная деятельность которого заключается в организации публичных кинопоказов для широкой аудитории. Деятельность кинотеатра включает в себя не только демонстрацию фильмов, но и управление коммерческими процессами, взаимодействие с дистрибьюторами, планирование работы, маркетинг и непосредственное обслуживание зрителей.

Данное веб-приложение предназначено для автоматизации ключевых бизнес-процессов кинотеатра, сфокусированных на клиентской части. Её цель – предоставить пользователям удобный и современный инструмент для выбора и приобретения билетов онлайн, а также предоставить администрации базовые средства для управления контентом и анализа продаж.

Веб-приложение оперирует следующими основными сущностями:

1. Фильм:
  - основной продукт. Характеризуется названием, описанием, постером, жанром и хронометражем.
2. Кинозал:
  - физическое помещение для показа фильмов. Имеет уникальный номер, название, описание и схему расположения мест.
3. Сеанс:
  - событие, определяющее показ конкретного фильма в конкретном кинозале в определенное время. Является центральной сущностью для бронирования. Содержит ссылку на фильм, ссылку на зал, дату и время начала.
4. Место:

– физическое место в кинозале на конкретном сеансе. Имеет идентификатор и данные о расположении.

5. Билет:

– электронный документ, подтверждающий право зрителя на посещение определенного сеанса и занятие конкретного места. Билет имеет уникальный номер, QR-код для контроля на входе, стоимость и дату приобретения.

6. Пользователь:

– клиент кинотеатра. Может быть незарегистрированным или зарегистрированным.

В контексте клиентской части веб-приложения основными участниками и процессами являются:

1. Гость:

– просмотр афиши: ознакомление с списком фильмов, идущих в прокате, просмотр их описаний;

– выбор сеанса: фильтрация сеансов по дате, времени, жанру и формату кинозала. Просмотр доступных мест на схеме зала;

– Просмотр информации о кинотеатре: ознакомление с кинотеатром, просмотр возможностей гостя и авторизованного пользователя.

2. Авторизованный пользователь:

– все возможности гостя;

– бронирование и покупка: выбор одного или нескольких мест, формирование корзины, оформление заказа с выбором способа оплаты;

– получение билета: после успешной оплаты билеты в электронной форме сохраняются в его личном кабинете, а также билет можно получить через telegram.

3. Администратор:

– все возможности авторизованного пользователя;

					КП.09.02.07.23.221.19.ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

- управление каталогом фильмов: добавление новых фильмов, редактирование информации о них;
- формирование расписания: создание сеансов путем привязки Фильма к Кинозалу, указание даты и времени начала;
- мониторинг продаж: просмотр статистики по проданным билетам, заполняемости залов и популярности фильмов.

Таким образом, разрабатываемое веб-приложение «Кинотеатр» предназначено для автоматизации процесса взаимодействия кинотеатра с конечными потребителями. Оно предоставляет пользователям удобный, прозрачный и современный сервис для выбора и покупки билетов онлайн, что повышает лояльность клиентов и общую эффективность работы кинотеатра за счет снижения нагрузки на кассовые узлы и предоставления инструментов для базового анализа спроса.

					КП.09.02.07.23.221.19.ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

## 2 Обзор инструментальных средств для разработки

Для успешной реализации проекта веб-приложения «Кинотеатр» был проведен анализ и выбор современных инструментальных средств. Выбор конкретных технологий обусловлен такими факторами, как производительность, простота разработки, надежность, сообщество и поддержка, а также соответствие требованиям проекта.

Draw.io: бесплатное кроссплатформенное средство для построения диаграмм. Было использовано для создания UML-диаграмм и схем взаимодействия компонентов веб-приложения для лучшего понимания архитектуры проекта, а также для создания прототипов страниц.

MySQL Workbench: визуальный инструмент для проектирования баз данных. MySQL Workbench был применен на этапе проектирования для создания ER-модели предметной области из-за удобного и наглядного интерфейса.

Для хранения данных требовалась надежная реляционная СУБД. Были рассмотрены три наиболее популярных варианта.

Таблица 1 – Сравнение СУБД

Критерий	PostgreSQL	MySQL	SQLite
Тип	Объектно-реляционная СУБД (ORDBMS)	Реляционная СУБД (RDBMS)	Встраиваемая реляционная СУБД (на основе файлов)
Производительность	Высокая, особенно на сложных запросах и больших объемах данных.	Высокая на операциях чтения.	Очень высокая для однопользовательского доступа, не предназначена для высоких нагрузок.
Масштабируемость	Отличная поддержка сложных структур данных, транзакций.	Некоторые механизмы могут уступать PostgreSQL.	Отсутствует, так как это встраиваемая БД.
Функции и стандарты	Наиболее строгое соответствие стандарту SQL, поддержка JSON.	Хорошая поддержка SQL.	Поддерживает большинство SQL-запросов.
Использование	Крупные и сложные проекты, где важна целостность данных и богатый функционал.	Веб-приложения, где преобладают операции чтения.	Мобильные приложения, простые десктопные приложения, прототипирование.



Для данного веб-приложения была выбрана PostgreSQL. Основными причинами стали:

- строгое соответствие стандартам SQL и поддержка сложных запросов, что важно для аналитики и отчетности;
- надежность и целостность данных, обеспечиваемые строгой системой транзакций;
- масштабируемость. База данных проекта потенциально может расти, и PostgreSQL предоставляет все необходимые инструменты для этого;
- богатый набор функций, таких как работа с JSON, полнотекстовый поиск, которые могут пригодиться для будущего расширения функционала.

Серверная часть веб-приложения требует языка с богатой экосистемой для веб-разработки, простотой поддержки и надежностью.

Таблица 2 – Сравнение языков программирования

Критерий	Python (Django)	PHP (Laravel)	Java (Spring Boot)
Синтаксис	Простой и читаемый синтаксис, низкий порог входа.	Синтаксис прост, но может быть непоследовательным.	Строгий, вербальный синтаксис
Фреймворки	Мощные и фреймворки (Django), множество библиотек для любых задач.	Огромное количество готовых решений и фреймворков (Laravel, Symfony).	Огромная enterprise-экосистема (Spring).
Безопасность	Django предоставляет «из коробки» многие механизмы защиты.	Требует большей внимательности к безопасности.	Высокий уровень безопасности
Производительность	Средняя. Вполне достаточна для данного проекта.	Ниже, чем у Python и Java.	Очень высокая.
Использование	Быстрая разработка complex-приложений, веб-сервисы.	Традиционно веб-сайты	Крупные корпоративные, высоконагруженные и распределенные системы.

В качестве основного языка программирования был выбран Python вместе с фреймворком Django. Это решение обусловлено следующими факторами:

- скорость разработки. Принцип «батарейки в комплекте» фреймворка Django позволяет быстро создать каркас приложения;

- читаемость кода и простота. Четкий синтаксис Python облегчает командную работу и будущую поддержку проекта;
- мощный ORM. Django ORM позволяет работать с базой данных на высоком уровне абстракции;
- безопасность. Django предоставляет встроенные и надежные механизмы защиты от распространенных уязвимостей.

Интегрированная среда разработки – ключевой инструмент, влияющий на продуктивность программиста.

Таблица 3 – Сравнение IDE

Критерий	PyCharm (Professional)	Visual Studio Code	Sublime Text
Тип	Полнофункциональная IDE	Редактор кода с возможностями IDE	Высокопроизводительный текстовый редактор с плагинами
Поддержка Python/Django	Нативная, глубокая «из коробки».	Требуется установка расширений.	Требуется установка множества плагинов для достижения функционала IDE.
Производительность	Высокая, оптимизирована для больших проектов.	Очень высокая, быстрый запуск.	Чрезвычайно высокая, самый быстрый запуск и отклик.
Отладка	Мощный встроенный дебаггер.	Дебаггер через расширение.	Только через сторонние плагины.
Стоимость	Платная, есть бесплатный Community-вариант.	Полностью бесплатная.	Условно-бесплатная.

В качестве основной IDE был выбран PyCharm Professional. Ключевые причины:

- исключительная поддержка Django. PyCharm предлагает лучшую на рынке интеграцию с Django: понимание структуры проекта, автодополнение для шаблонов, моделей и представлений, удобная навигация между файлами;
- встроенный набор инструментов. Наличие встроенного терминала и инструментов для работы с Git значительно ускоряет процесс разработки и отладки;

– умный код-ассистент. Высококачественное автодополнение кода и анализ кода помогают избегать ошибок и писать код в соответствии со стандартами.

Обеспечение качества и надежности программного продукта является неотъемлемой частью процесса разработки.

Таблица 4 – Сравнение инструментов тестирования

Критерий	pytest	unittest (стандартный )	Django Test Case (на базе unittest)
Синтаксис и простота	Очень простой и лаконичный. Используются обычные функции и assert.	Более многословный, требует наследования от TestCase и использования специальных методов.	Аналогичен unittest, но предоставляет дополнительные assertion-методы для Django (например, assertContains, assertTemplateUsed).
Фикстуры (Fixtures)	Мощная и гибкая система фикстур с широкими возможностями.	Базовые возможности setup/teardown методов.	Использует механизм setUp/tearDown, а также предоставляет готовые фикстуры (тестовый клиент, тестовая БД).
Интеграция с Django	Требуется установка дополнительных плагинов (например, pytest-django) для полноценной работы.	Интегрируется на базовом уровне.	Прямая и нативная интеграция. Создан специально для тестирования Django-приложений.
Использование	Универсальные проекты на Python, где важна простота и мощь.	Проекты, где достаточно стандартной библиотеки или требуется строгая структура.	Проекты на Django. Наиболее предпочтителен для тестирования моделей, представлений, форм и шаблонов.

Для тестирования веб-приложения «Кинотеатр» был выбран инструмент Django Test Case. Основными причинами стали:

– нативная интеграция с Django. Это главное преимущество. Фреймворк автоматически создает тестовую базу данных, предоставляет готовый тестовый

клиент для эмуляции запросов к приложению и специализированные методы для проверки ответов, что критически важно для тестирования представлений и шаблонов;

- скорость написания тестов. Для разработчика, работающего в среде Django, использование встроенного инструмента является наиболее прямым и понятным путем, не требующим изучения дополнительного синтаксиса или настройки плагинов;

- надежность. Как часть самого фреймворка Django, этот инструмент гарантированно совместим со всеми его версиями и правильно обрабатывает все специфичные для Django аспекты.

Для создания программного продукта было решено использовать следующие инструментальные средства:

1. Для проектирования архитектуры и баз данных использовались CASE-средства:

- Draw.io – для создания структурных схем, диаграмм потоков данных (DFD), диаграмм вариантов использования и других UML-диаграмм;
- MySQL Workbench – для наглядного проектирования структуры реляционной базы данных и построения ER-диаграммы.

2. Для реализации серверной части использовался следующий стек технологий:

- язык программирования Python – в качестве основного языка разработки благодаря его чистоте синтаксиса, богатой экосистеме и высокой скорости создания приложений;
- Веб-фреймворк Django – в качестве основы серверной части для быстрой разработки безопасного и поддерживаемого кода;
- СУБД PostgreSQL – в качестве надежной и мощной системы управления базами данных для хранения всей информации.

3. Для реализации клиентской части применялись стандартные веб-технологии:

					КП.09.02.07.23.221.19.ПЗ	Лист 12
Изм.	Лист	№ докум.	Подпись	Дата		

– HTML5, CSS3 – для создания семантической разметки и стилизации веб-страниц.

4. В качестве основной среды разработки (IDE) был выбран:

– PyCharm Professional – как наиболее мощная и удобная среда, обеспечивающая глубочайшую поддержку как Python, так и фреймворка Django, включая отладку, рефакторинг и работу с базой данных прямо из IDE.

5. Для тестирования веб-приложения был выбран:

– Django Test Case, так как он максимально ориентирован на предметную область и технологический стек разработки.

					КП.09.02.07.23.221.19.ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

### 3 Техническое задание

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602–2020 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы».

Согласно ГОСТ 34.602–2020 техническое задание должно включать следующие разделы:

- 1 Введение.
- 2 Основания для разработки.
- 3 Назначение приложения.
- 4 Требования к веб-приложению.
- 5 Требования к техническому обеспечению.
- 6 Требования к программному обеспечению.
- 7 Требования к тестированию.
- 8 Организационно - технические требования.

Техническое задание на разработку веб-приложения представлено в приложении А.

					КП.09.02.07.23.221.19.ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

## 4 Проектирование веб-приложения

### 4.1 Структурная схема веб-приложения

Диаграмма прецедентов описывает высокоуровневые функциональные возможности веб-приложения с точки зрения пользователей. Она изображена на рисунке 1.

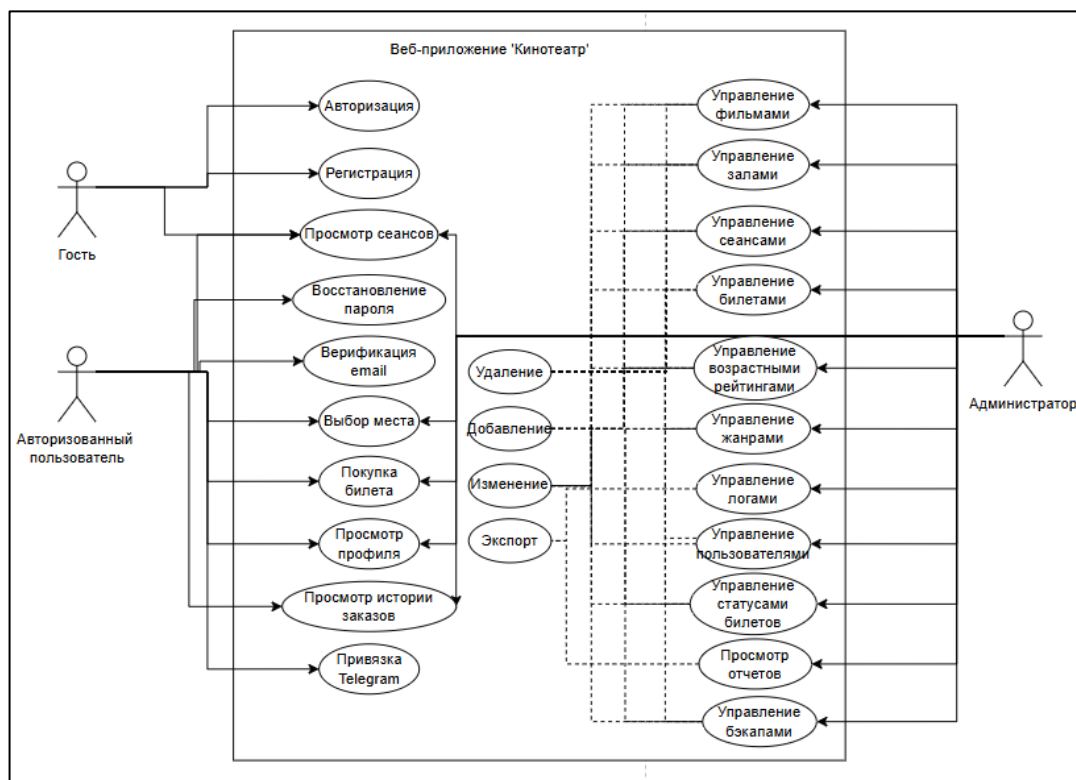


Рисунок 1 – Диаграмма прецедентов

На диаграмме представлены три основных прецедента:

- Гость: может зарегистрироваться или авторизоваться, может просматривать афишу;
- Пользователь: может просматривать афишу, может выполнять ключевые бизнес-процессы: выбирать места, покупать билеты и просматривать свою историю заказов.
- Администратор: имеет полный доступ, включая управление контентом и просмотр статистики.

Диаграмма показывает, что веб-приложение разделено на клиентскую часть и административную часть.

Диаграмма деятельности показывает поток выполнения операций в веб-приложении, в данном случае – процесс покупки билета. Она изображена на рисунке 2.

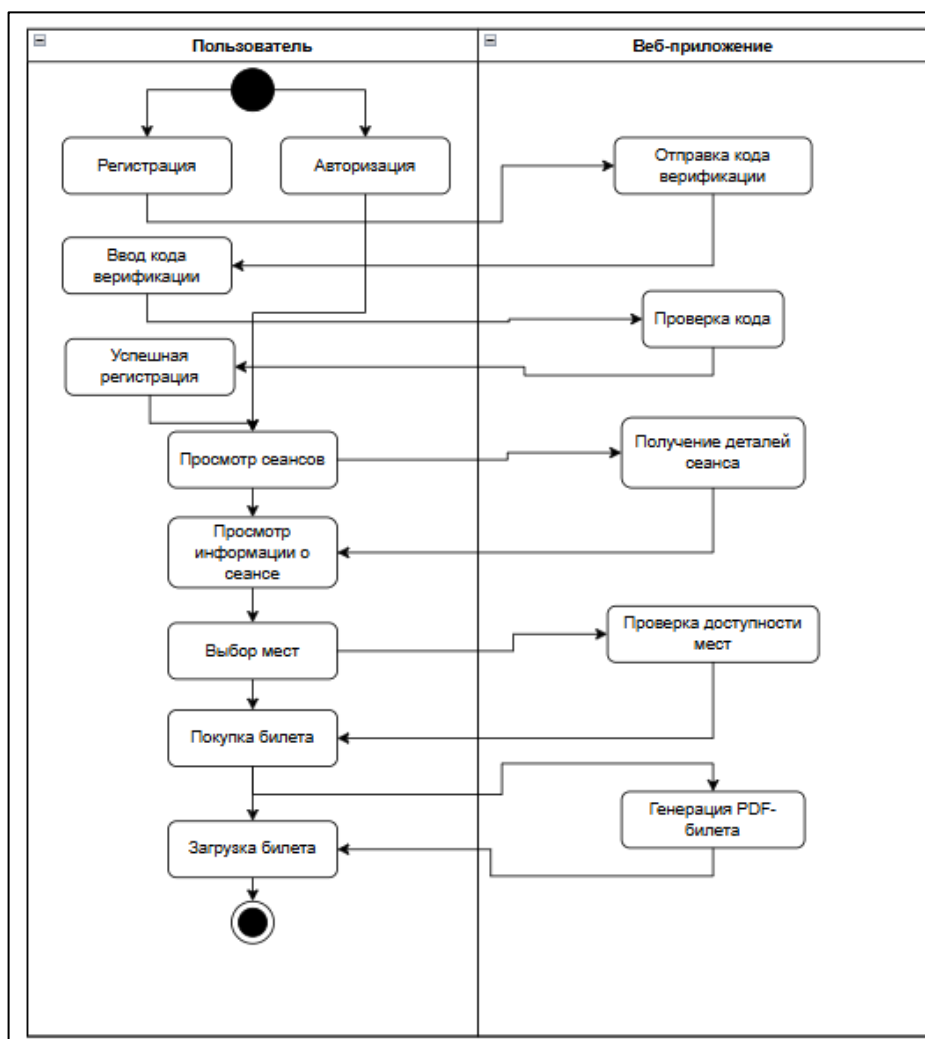


Рисунок 2 – Диаграмма деятельности

Процесс начинается с регистрации или авторизации. После успешной аутентификации пользователь просматривает сеансы, выбирает сеанс и место. Если место свободно, формируется заказ, происходит процесс оплаты. После подтверждения оплаты фиксируется бронирование, генерируется электронный билет с QR-кодом и даёт пользователю возможность загрузить PDF-файл билета.



Диаграмма детализирует основной бизнес-процесс приложения – онлайн-покупку билета. Она демонстрирует последовательность шагов, точки принятия решений и ключевые операции, такие как интеграция с платежной системой и генерация билета, что соответствует реализованному функционалу.

Диаграмма показывает структурные компоненты веб-приложения и связи между ними в виде информационных объектов: файлов, модулей, библиотек, пакетов. Она изображена на рисунке 3.

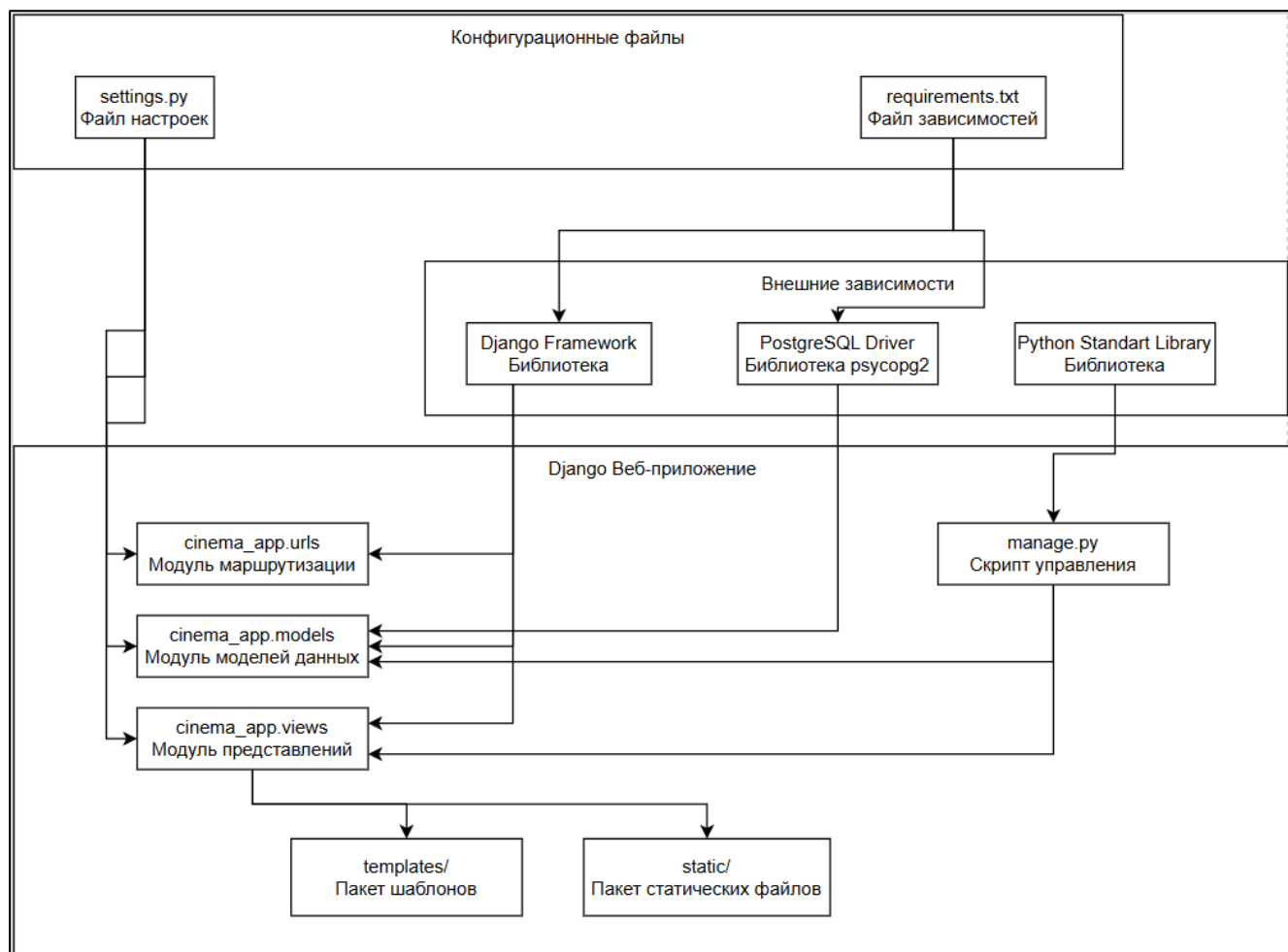


Рисунок 3 – Диаграмма компонентов

Описание компонентов:

- cinema\_app.models – модуль, содержащий модели данных;
- cinema\_app.views – модуль бизнес-логики, обработчики запросов;
- cinema\_app.urls – модуль маршрутизации URL;

- templates/ – пакет HTML-шаблонов;
- static/ – пакет статических файлов (CSS, JS, изображения);
- manage.py – скрипт управления Django-приложением;
- settings.py – конфигурационный файл настроек приложения;
- requirements.txt – файл зависимостей проекта;
- Django Framework – основная библиотека веб-фреймворка;
- PostgreSQL Driver – библиотека для работы с PostgreSQL;
- Python Standard Library – стандартная библиотека Python.

Листинг модуля маршрутизации URL представлен в приложении Б.

Диаграмма компонентов демонстрирует модульную архитектуру веб-приложения, где каждый компонент выполняет строго определенную функцию. Четкое разделение на модели, представления, шаблоны и статические файлы соответствует принципам MVC/MVT. Зависимости от внешних библиотек явно указаны, что облегчает развертывание и сопровождение.

Диаграмма развертывания показывает аппаратные компоненты («узлы») и программные компоненты («артефакты»), работающие на каждом узле. Она изображена на рисунке 4.

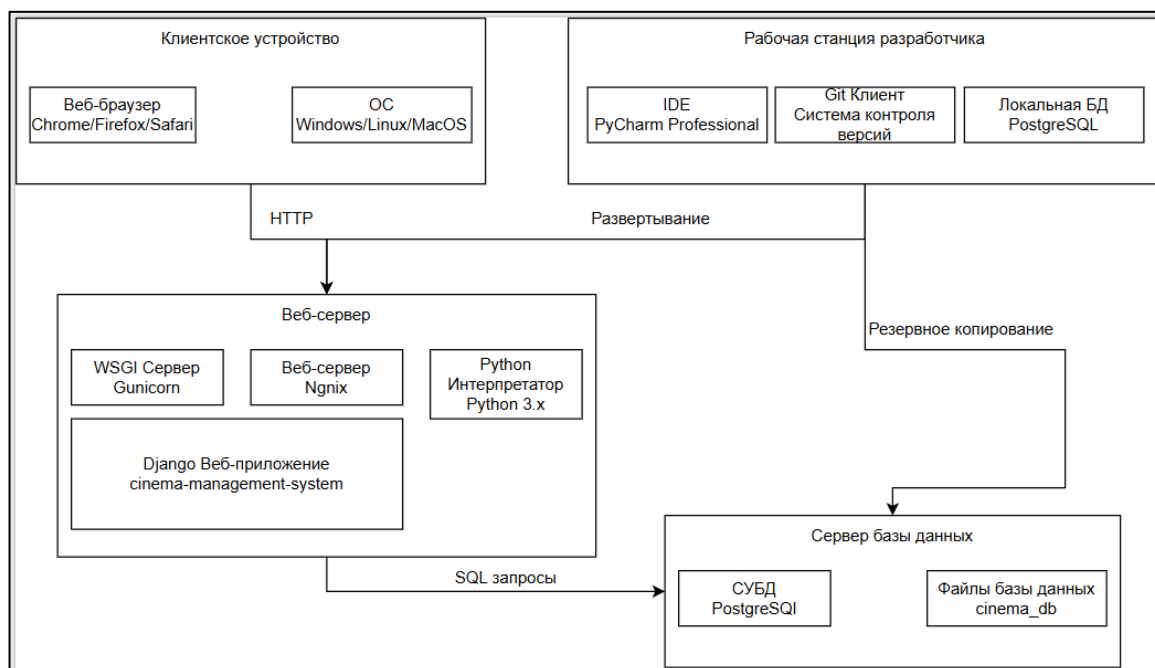


Рисунок 4 – Диаграмма развертывания

Описание узлов и артефактов:

Узел: Клиентское устройство:

- Артефакт: Веб-браузер – пользовательский интерфейс;
- Артефакт: Операционная система – платформа для работы браузера.

Узел: Веб-сервер:

- Артефакт: WSGI Сервер – обработчик Python-приложений;
- Артефакт: Веб-сервер – статические файлы и прокси;
- Артефакт: Django Приложение – основная бизнес-логика;
- Артефакт: Python Интерпретатор – среда выполнения приложения.

Узел: Сервер базы данных:

- Артефакт: СУБД PostgreSQL – система управления базами данных;
- Артефакт: Файлы базы данных – физическое хранилище данных.

Узел: Рабочая станция разработчика:

- Артефакт: IDE PyCharm – среда разработки;
- Артефакт: Git Клиент – контроль версий кода;
- Артефакт: Локальная БД – база данных для разработки.

Диаграмма развертывания показывает трехзвенную архитектуру веб-приложения (клиент-сервер-БД), что обеспечивает масштабируемость и надежность. Разделение веб-сервера и сервера базы данных на разные узлы позволяет независимо масштабировать вычислительную мощность и ресурсы хранения данных. Использование Nginx в качестве фронтенд-сервера и Gunicorn в качестве бэкенд-сервера соответствует лучшим практикам развертывания Django-приложений.

## 4.2 Функциональная схема веб-приложения

Контекстная диаграмма – это высокоуровневое визуальное представление, которое показывает взаимодействия между разрабатываемым веб-приложением и его внешними объектами. Она изображена на рисунке 5.

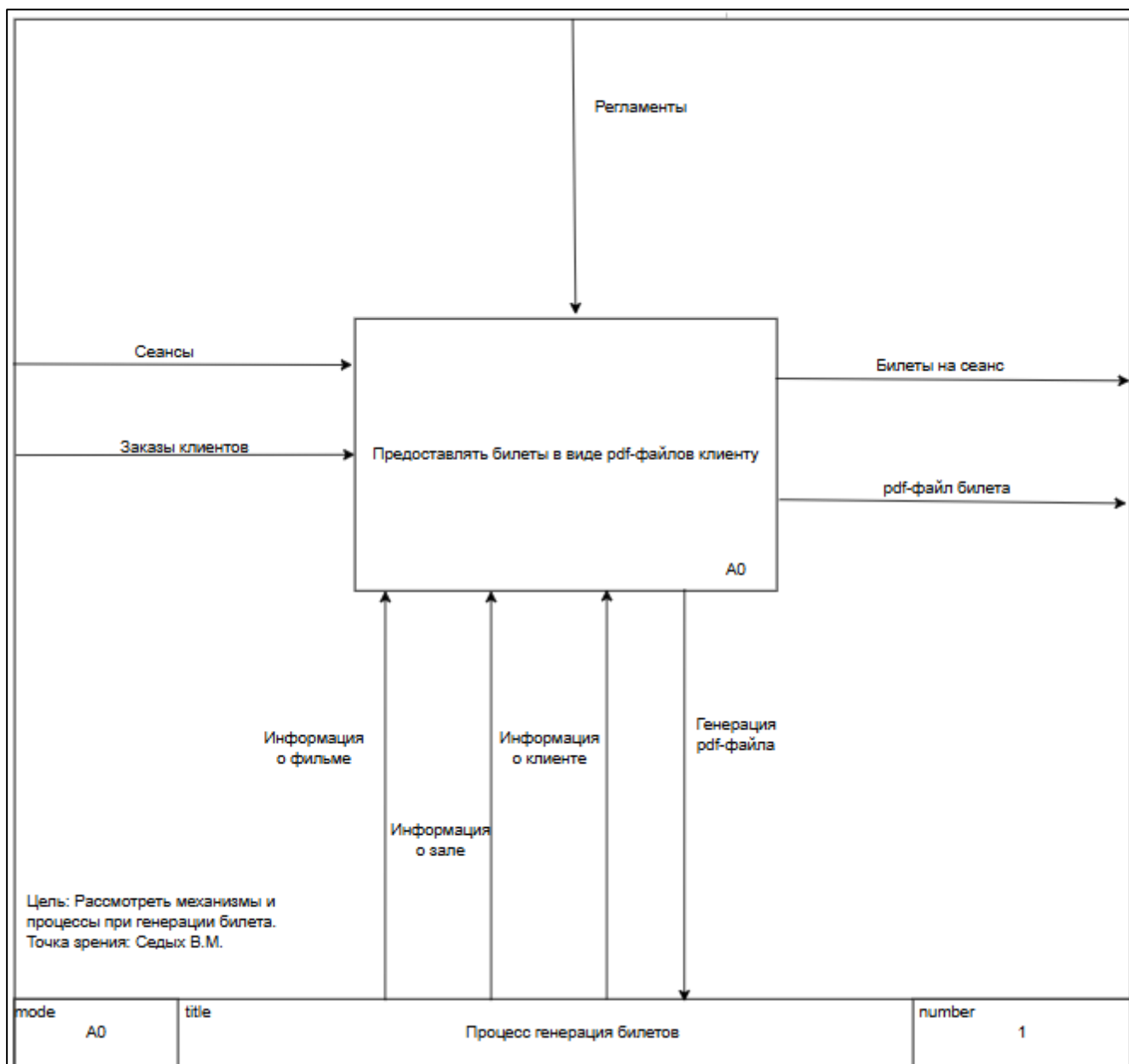


Рисунок 5 – Контекстная диаграмма

Диаграмма показывает, что веб-приложение получает информацию о билетах, обрабатывает ее согласно установленным регламентам с использованием механизмов генерации PDF и выдает готовый электронный билет.

Диаграмма декомпозиций (A1) предназначена для детализации работы, то есть декомпозицией называется разделение бизнес-процессов на более мелкие составляющие. Она изображена на рисунке 6.

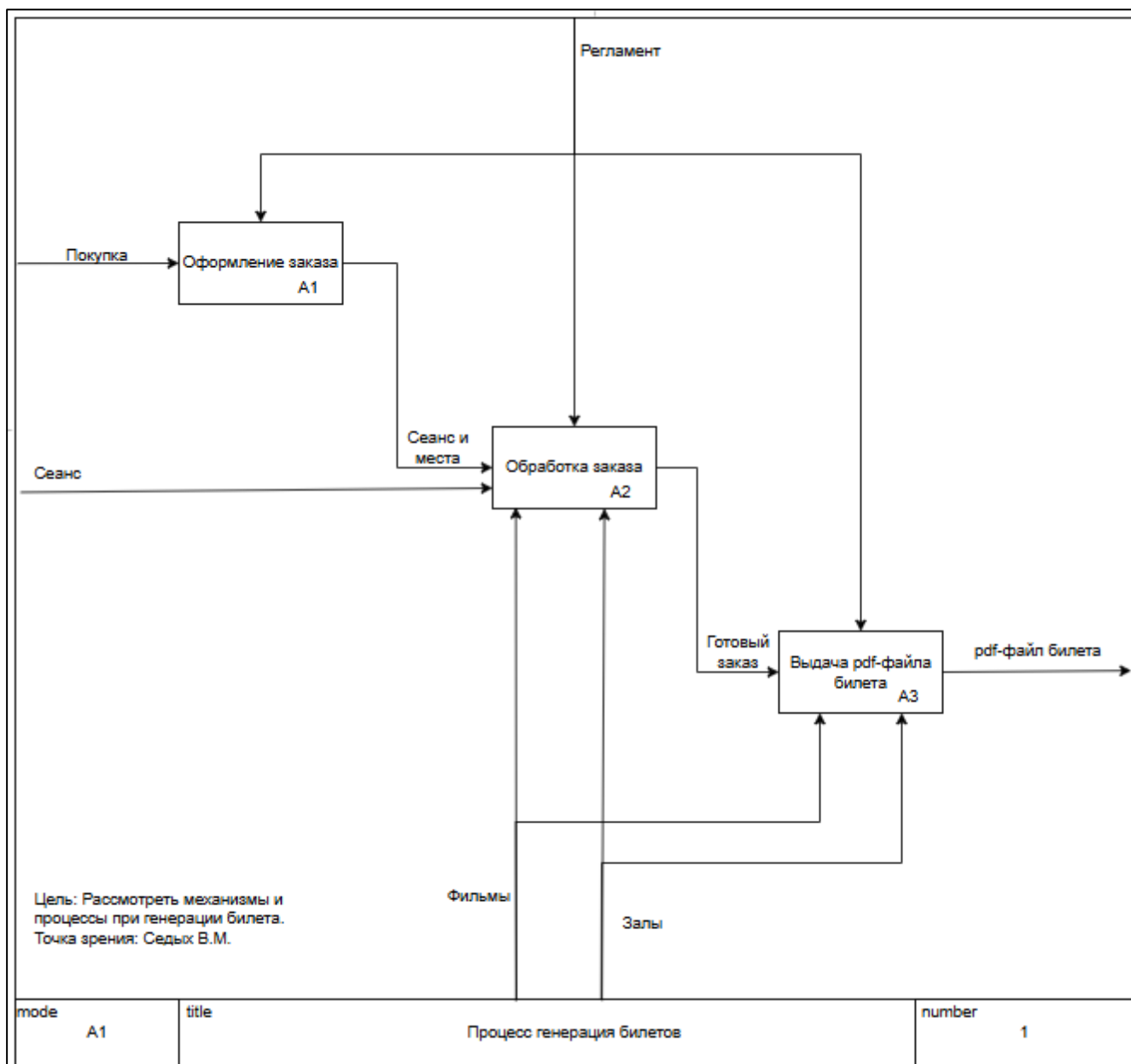


Рисунок 6 – Диаграмма декомпозиций A1

Процесс использует информацию из справочников «Фильмы» и «Залы», обрабатывает входные данные «Сеанс и места» и выдает «pdf-файл билета». Управление осуществляется через «Регламенты», а механизмами выступают «Сеансы» и «Заказы клиентов».

Диаграмма декомпозиции A2 (IDEF0) – это диаграмма, которая описывает детализацию функций веб-приложения на втором уровне в нотации IDEF0. Она изображена на рисунке 7.

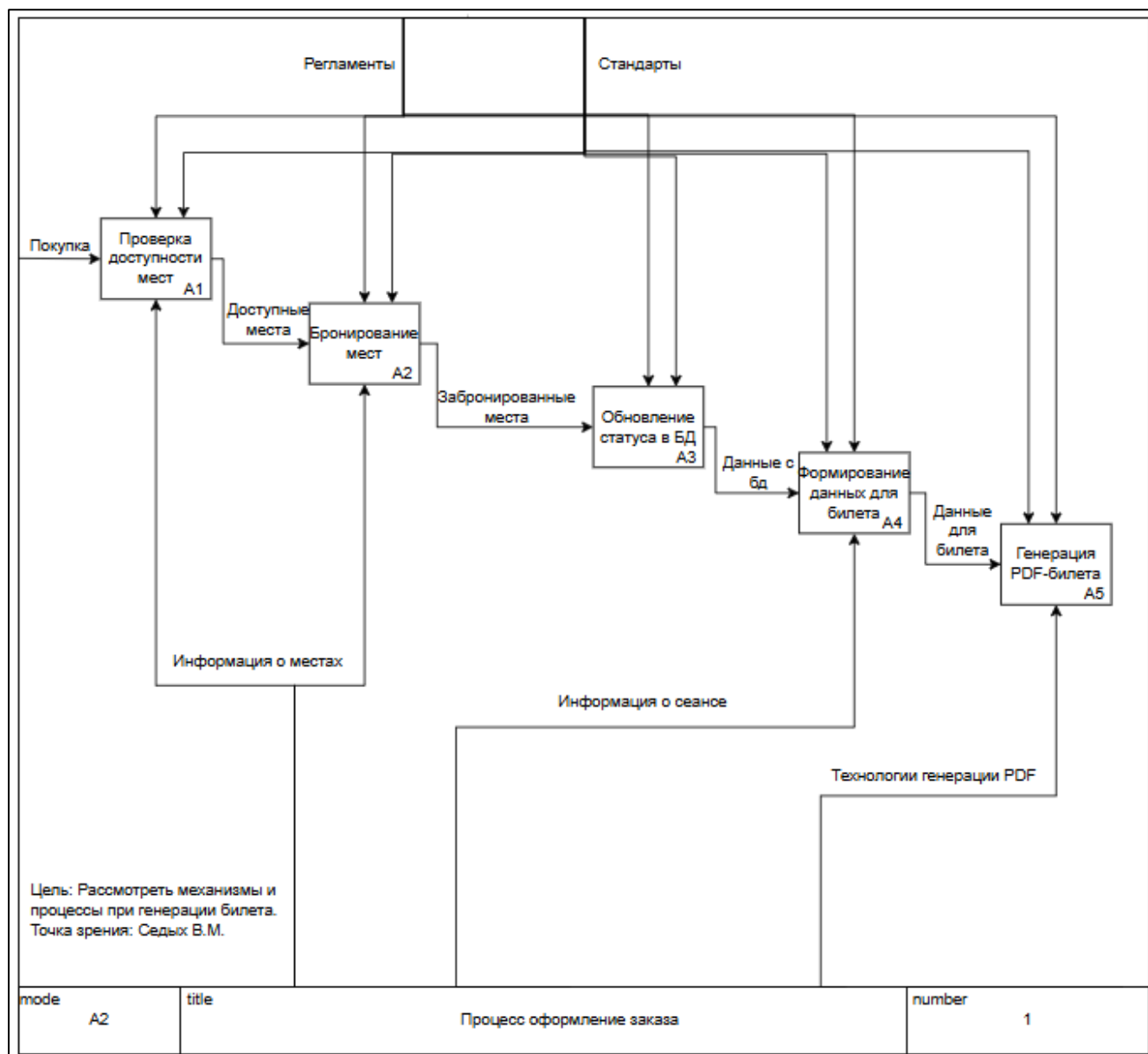


Рисунок 7 – Диаграмма декомпозиций A2

Диаграмма A2 демонстрирует детальную последовательность операций при оформлении заказа. Четко прослеживается разделение на логические этапы: проверка данных, бронирование, формирование билета и его доставка. Такая декомпозиция позволяет эффективно распределить ответственность между модулями и обеспечивает прозрачность бизнес-процесса.

Диаграмма классов показывает статичную структуру системы, демонстрируя классы, их атрибуты, поведение и связь друг с другом. Она изображена на рисунке 8.

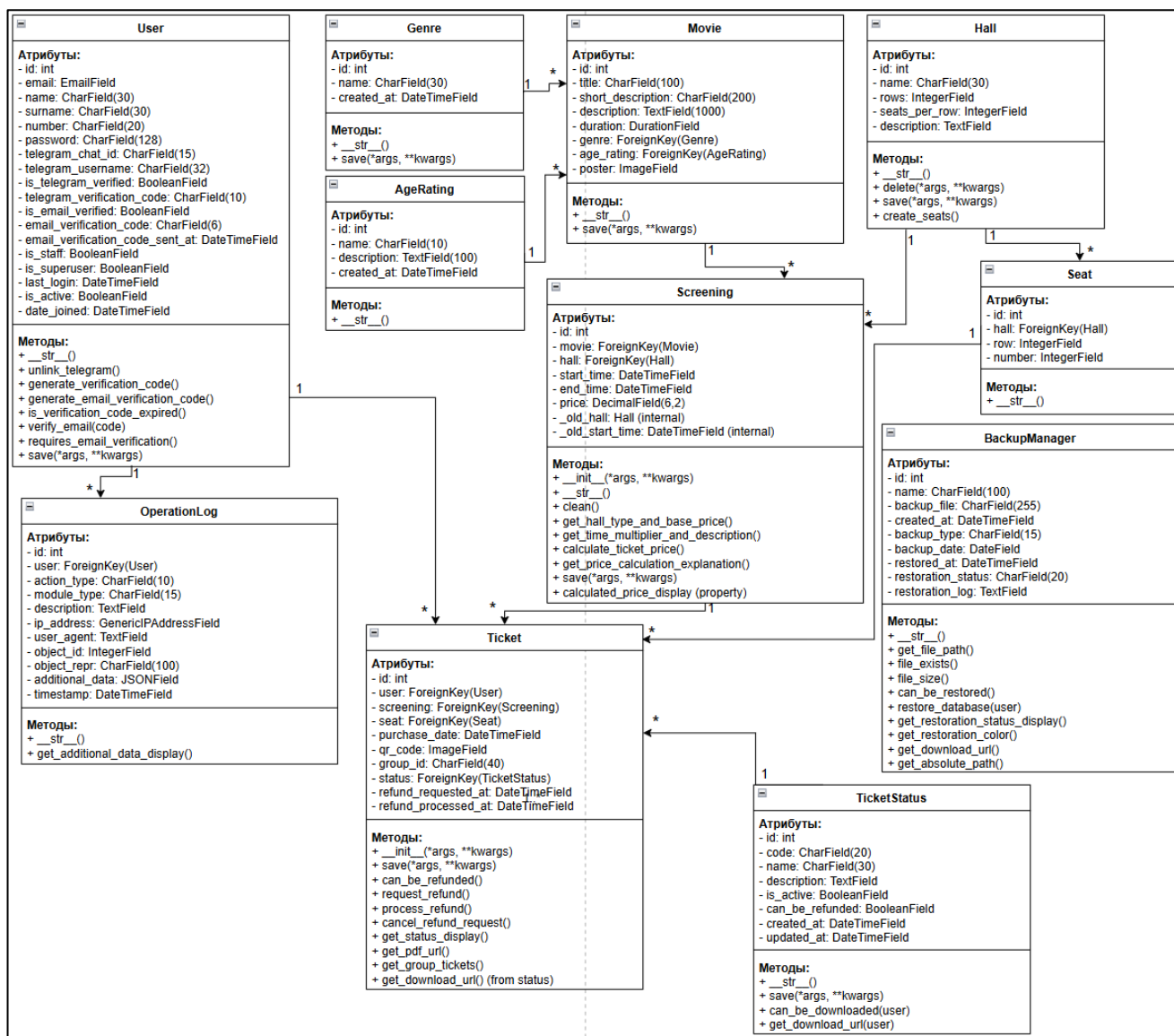


Рисунок 8 – Диаграмма классов

Диаграмма классов представляет статическую структуру данных веб-приложения:

- User – модель пользователя с аутентификацией и управлением заказами;
- Genre – сущность жанра фильма с уникальным названием;
- AgeRating – сущность возрастного рейтинга фильма;
- Movie – сущность фильма с метаданными и медиа-контентом;
- Hall – кинозал с параметрами вместимости и схемой расположения мест;

- Screening – сеанс как связующее звено между фильмом и залом;
- Seat – конкретное место в зале с возможностью бронирования;
- TicketStatus – статус билета;
- Ticket – билет с функционалом генерации PDF и отправки;
- OperationLog – система логирования операций;
- BackupManager – система резервного копирования.

Диаграмма классов демонстрирует хорошо продуманную объектную модель, соответствующую требованиям предметной области. Четкое разделение ответственности между классами, наличие необходимых атрибутов и методов, а также логичные связи между сущностями обеспечивают основу для устойчивой и расширяемой архитектуры системы.

Диаграмма потоков данных показывает движение данных в информационной системе: откуда они поступают, как обрабатываются, где хранятся и кому передаются. Она изображена на рисунке 9.

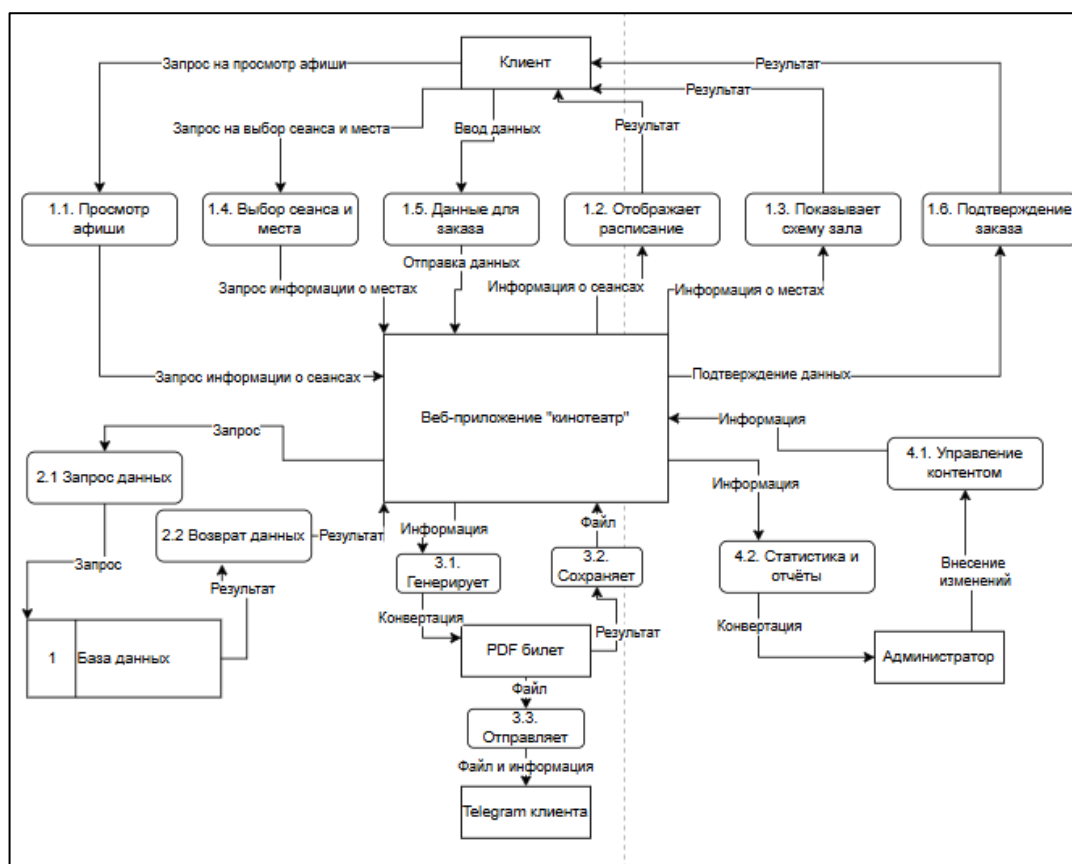


Рисунок 9 – Диаграмма потоков данных DFD



Диаграмма потоков данных (DFD) показывает движение информации в веб-приложении:

- Внешние сущности: Клиент, Администратор, Email система;
- Процессы: Веб-приложение Кинотеатр;
- Потоки данных: запросы расписания, выбор сеансов, данные заказов, PDF билеты;
- Хранилища данных: база данных PostgreSQL.

Диаграмма иллюстрирует как информация поступает от клиента, обрабатывается, сохраняется в базе данных и возвращается в виде билетов и подтверждений.

Разработанный комплект диаграмм полностью охватывает функциональную схему веб-приложения:

- IDEF0 диаграммы (A0, A1, A2) эффективно отображают иерархию бизнес-процессов, начиная от контекстного представления и заканчивая детализированными операциями генерации билетов;
- Диаграмма классов четко определяет структуру данных, показывая сущности, их атрибуты, методы и взаимосвязи, что соответствует реализованным в проекте моделям Django;
- DFD диаграмма наглядно демонстрирует потоки информации между внешними сущностями и веб-приложением.

Все диаграммы взаимосвязаны и непротиворечивы: сущности из диаграммы классов соответствуют данным, обрабатываемым в DFD, а бизнес-процессы IDEF0 отражают функциональность, реализованную в методах классов. Такой комплексный подход к проектированию обеспечивает полное понимание архитектуры веб-приложения и облегчает дальнейшую разработку и сопровождение.

### 4.3 Проектирование базы данных

Перед началом разработки программного обеспечения важно спроектировать базу данных, определив, с какими данными будут работать пользователи системы и как эти данные взаимосвязаны. Инфологическая модель базы данных изображена на рисунке 10.

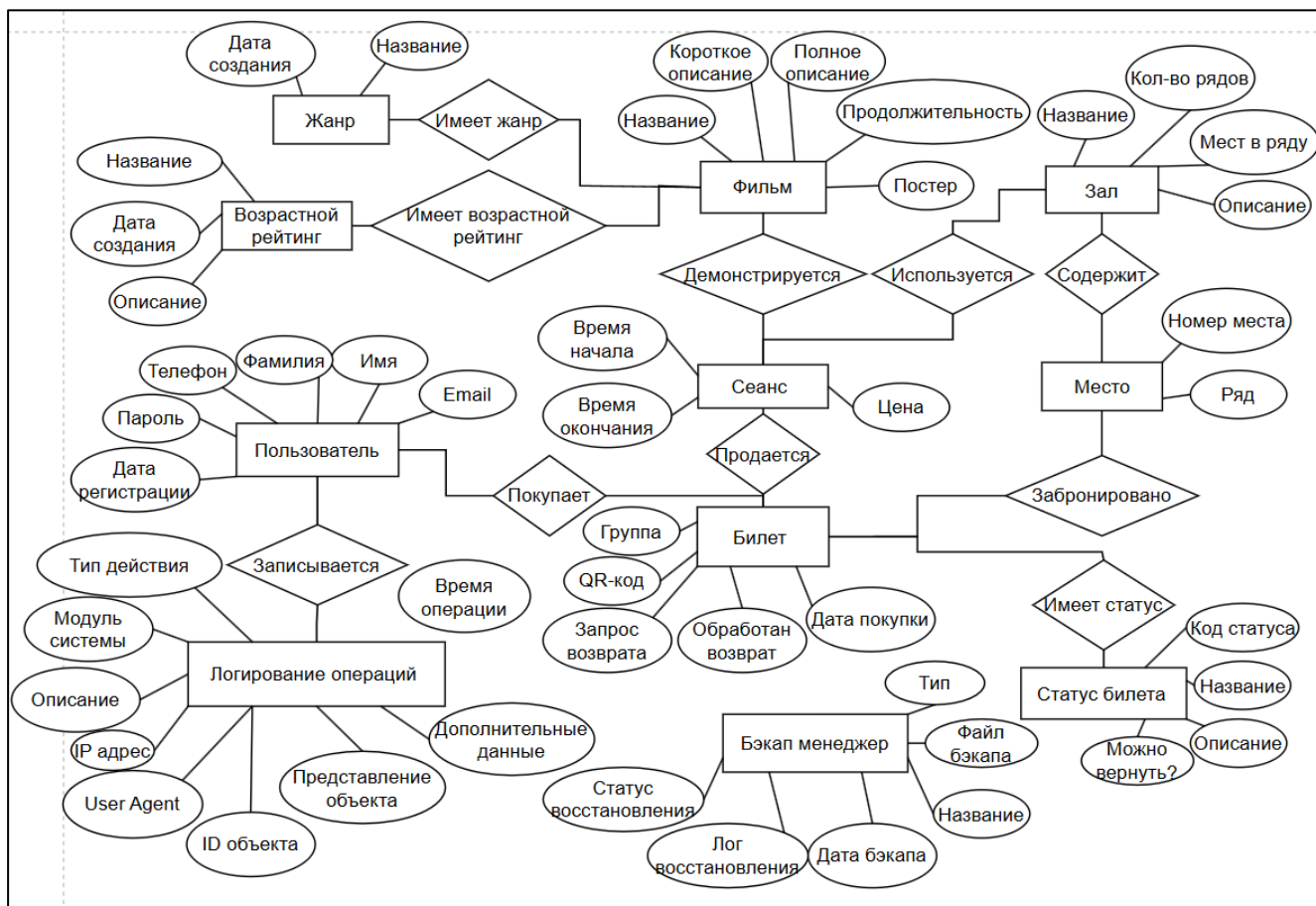


Рисунок 10 – Инфологическая модель базы данных

Инфологическая модель представляет собой семантическое описание предметной области в терминах сущностей и отношений:

- USER – сущность «Пользователь» с персональными данными;
- GENRE – сущность «Жанр» с уникальными названиями;
- AGE\_RATING – сущность «Возрастной рейтинг» с возрастными ограничениями;
- MOVIE – сущность «Фильм» с характеристиками кинопродукции;

- HALL – сущность «Зал» с параметрами размещения зрителей;
- SCREENING – сущность «Сеанс» как событие показа фильма в зале;
- SEAT – сущность «Место» с координатами расположения в зале;
- TICKET\_STATUS – сущность «Статус билета» с информацией о возврате;
- TICKET – сущность «Билет» как документ, подтверждающий право посещения;
- OPERATION\_LOG – система логирования операций;
- BACKUP\_MANAGER – система резервного копирования.

Связи между сущностями:

- Пользователь покупает несколько билетов (1:M);
- Жанр связан с несколькими фильмами (1:M);
- Возрастной рейтинг связан с фильмами (1:M);
- Фильм демонстрируется на нескольких сеансах (1:M);
- Зал используется для нескольких сеансов (1:M);
- Зал содержит несколько мест (1:M);
- Сеанс продается через несколько билетов (1:M);
- Место бронируется через билет (1:M);
- Статус билета отображает статус возврата у билета (1:M);
- Система записывает действия пользователя (1:M).

На рисунке 11 изображена датологическая модель базы данных.

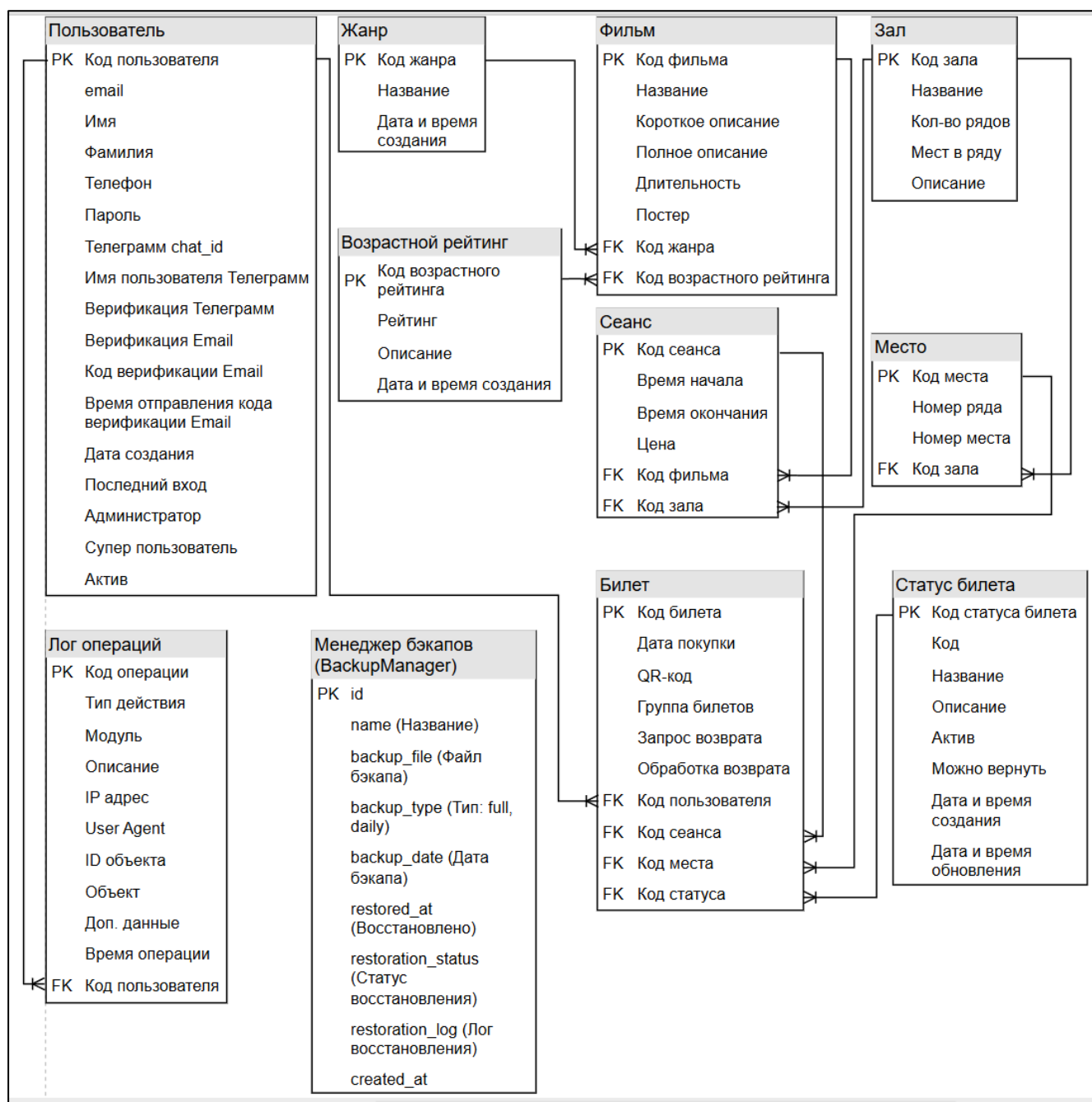


Рисунок 11 – Датологическая модель базы данных

Датологическая модель представляет физическую реализацию БД с учетом конкретной СУБД:

- Индексы: определены уникальные индексы и первичные ключи;
- Внешние ключи: явно указаны FOREIGN KEY ограничения.

На рисунке 12 изображена ER–модель базы данных.

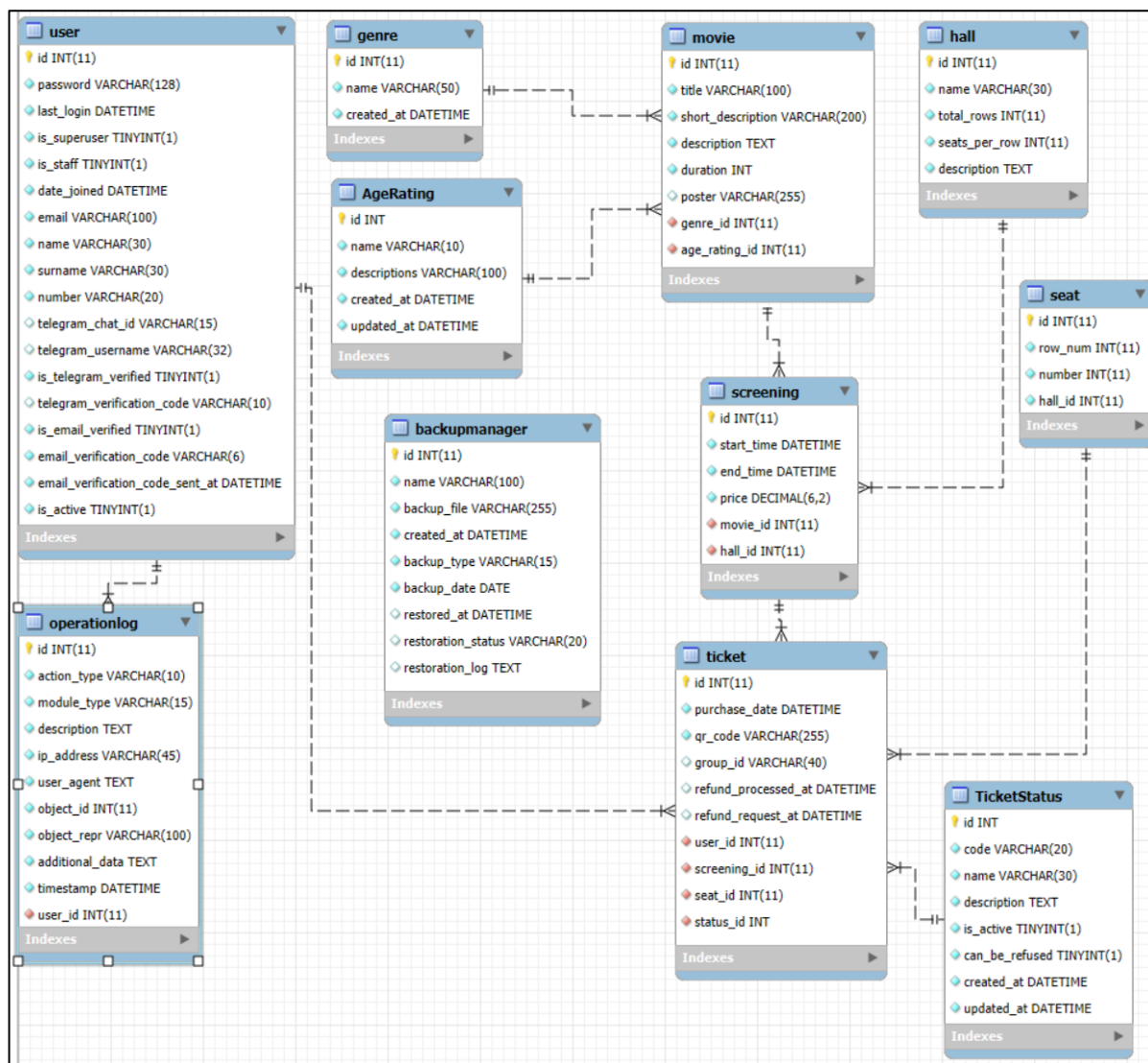


Рисунок 12 – ER-модель базы данных

Представленная ER-модель соответствует третьей нормальной форме:

1. Первая нормальная форма: все атрибуты атомарны.
2. Вторая нормальная форма: нет частичных зависимостей от составного ключа.
3. Третья нормальная форма: нет транзитивных зависимостей - все неключевые атрибуты зависят только от первичного ключа.

Обоснование выбора 3NF:

- Устраняет избыточность данных;
- Обеспечивает целостность данных;
- Упрощает поддержку и модификацию БД;

Изм.	Лист	№ докум.	Подпись	Дата

– Оптимальна для OLTP-систем.

База данных веб-приложения состоит из 11 таблиц.

Таблица 5 – Таблица «User»

Поле	Тип данных	Описание
id	INT	Идентификатор пользователя
password	VARCHAR(128)	Хэш пароля
last_login	DATETIME	Время последнего входа в систему
is_superuser	TINYINT(1)	Права суперпользователя (1/0)
is_staff	TINYINT(1)	Права администратора (1/0)
date_joined	DATETIME	Дата и время регистрации
email	VARCHAR(100)	Электронная почта
name	VARCHAR(30)	Имя пользователя
surname	VARCHAR(30)	Фамилия пользователя
number	VARCHAR(20)	Номер телефона
telegram_chat_id	VARCHAR(15)	ID чата в Telegram
telegram_username	VARCHAR(32)	Имя пользователя в Telegram
is_telegram_verified	TINYINT(1)	Статус привязки Telegram (1 - подтвержден)
telegram_verification_code	VARCHAR(10)	Код для привязки Telegram
is_email_verified	TINYINT(1)	Статус верификации email (1 - подтвержден)
email_verification_code	VARCHAR(6)	Код подтверждения email
email_verification_code_sent_at	DATETIME	Дата и время отправки кода подтверждения email
is_active	TINYINT(1)	Активен ли пользователь (1/0)

Таблица 6 – Таблица «Hall»

Поле	Тип данных	Описание
id	INT	Идентификатор
name	VARCHAR(30)	Название
rows	INT	Количество рядов
seats_per_row	INT	Количество мест в ряду
description	TEXT	Описание

Таблица 7 – Таблица «Genre»

Поле	Тип данных	Описание
id	INT	Идентификатор
name	VARCHAR(30)	Название
created_at	DATETIME	Дата создания

Таблица 8 – Таблица «AgeRating»

Поле	Тип данных	Описание
id	INT	Идентификатор возрастного рейтинга
name	VARCHAR(10)	Название рейтинга
description	VARCHAR(100)	Описание
created_at	DATETIME	Дата и время создания
updated_at	DATETIME	Дата и время последнего обновления

Таблица 9 – Таблица «Movie»

Поле	Тип данных	Описание
id	INT	Идентификатор фильма
title	VARCHAR(100)	Название фильма
short_description	VARCHAR(200)	Короткое описание фильма
description	TEXT	Полное описание фильма
duration	INT	Продолжительность в минутах
poster	VARCHAR(500)	Путь к файлу постера
genre_id	INT	Идентификатор жанра
age_rating_id	INT	Идентификатор возрастного рейтинга

Таблица 10 – Таблица «Screening»

Поле	Тип данных	Описание
id	INT	Идентификатор
start_time	DATETIME	Дата и время начала
end_time	DATETIME	Дата и время окончания
ticket_price	DECIMAL(6,2)	Стоимость
movie_id	INT	Идентификатор фильма
hall_id	INT	Идентификатор зала

Таблица 11 – Таблица «Seat»

Поле	Тип данных	Описание
id	INT	Идентификатор
row	INT	Ряд
number	INT	Место
hall_id	INT	Идентификатор зала

Таблица 12 – Таблица «TicketStatus»

Поле	Тип данных	Описание
id	INT	Идентификатор статуса
code	VARCHAR(20)	Код статуса
name	VARCHAR(30)	Название статуса
description	TEXT	Описание статуса
is_active	TINYINT(1)	Активен ли статус (1/0)

Продолжение таблицы 12

can_be_refunded	TINYINT(1)	Можно ли вернуть билет из этого статуса (1/0)
created_at	DATETIME	Дата и время создания
updated_at	DATETIME	Дата и время обновления

Таблица 13 – Таблица «Ticket»

Поле	Тип данных	Описание
id	INT	Идентификатор билета
purchase_date	DATETIME	Дата и время покупки
qr_code	VARCHAR(255)	Путь к файлу QR-кода
group_id	VARCHAR(40)	Идентификатор группы билетов
refund_requested_at	DATETIME	Дата и время запроса возврата
refund_processed_at	DATETIME	Дата и время обработки возврата
user_id	INT	Идентификатор пользователя
screening_id	INT	Идентификатор сеанса
seat_id	INT	Идентификатор места
status_id	INT	Идентификатор статуса

Таблица 14 – Таблица «Operationlog»

Поле	Тип данных	Описание
id	INT	Идентификатор
user_id	INT	Идентификатор пользователя
action_type	VARCHAR(10)	Тип действия
module_type	VARCHAR(15)	Модуль системы
description	TEXT	Описание операции
ip_address	VARCHAR(45)	IP-адрес
user_agent	TEXT	User Agent браузера
object_id	INT	ID объекта операции
object_repr	VARCHAR(100)	Представление объекта
additional_data	JSON	Дополнительные данные
timestamp	DATETIME	Время операции

Таблица 15 – Таблица «BackupManager»

Поле	Тип данных	Описание
id	INT	Идентификатор бэкапа
name	VARCHAR(100)	Название бэкапа
backup_file	VARCHAR(255)	Имя файла бэкапа
created_at	DATETIME	Дата и время создания записи
backup_type	VARCHAR(15)	Тип бэкапа
backup_date	DATE	Дата выполнения бэкапа



Продолжение таблицы 15

restored_at	DATETIME	Дата и время восстановления из бэкапа
restoration_status	VARCHAR(20)	Статус восстановления
restoration_log	TEXT	Лог процесса восстановления

Спроектированная база данных полностью соответствует требованиям предметной области кинотеатра. ER-модель в третьей нормальной форме обеспечивает оптимальную структуру для хранения данных, минимизируя избыточность и обеспечивая целостность. Четко определенные связи между сущностями позволяют эффективно реализовать все бизнес-процессы системы, от бронирования билетов до управления расписанием. Выбранные типы данных и ограничения гарантируют корректное хранение информации и предотвращают возникновение противоречивых состояний в базе данных.

#### 4.4 Проектирование интерфейса

Для проектирования пользовательского интерфейса веб-приложения «Кинотеатр» использовался профессиональный онлайн-инструмент Draw.io. Данный инструмент был выбран за его кроссплатформенность, простоту использования, широкий набор элементов для построения схем и удобство совместной работы, что позволило быстро и наглядно создать прототипы всех основных экранов.

Были разработаны прототипы всех ключевых окон веб-приложения, соответствующих функциональным требованиям, предъявляемым к клиентской и административной частям приложения.

The image shows a login page for 'КИНОТЕАТР ПРЕМЬЕРА'. The page has a clean, minimalist design with a light gray background and rounded corners. At the top, the title 'КИНОТЕАТР ПРЕМЬЕРА' is displayed in a bold, sans-serif font. Below the title is the word 'Вход' (Login). A dashed box labeled 'Сообщения системы' (System messages) is positioned above the input fields. There are two input fields: 'Email' and 'Пароль' (Password). The password field has a 'Показать' (Show) button next to it. Below the input fields is a large 'Войти' (Login) button. Underneath the login button, there is a link that says 'Нет аккаунта? Зарегистрироваться\nЗабыли пароль?' (No account? Register\nForgot password?). Below this link is a button labeled 'Панель администратора' (Administrator panel). At the bottom, there is a link that says 'Или продолжите как гость' (Or continue as guest) and a button labeled 'Продолжить как гость' (Continue as guest).

Рисунок 13 – Страница Login

Страница авторизации содержит стандартную форму для ввода учетных данных, кнопку для входа и кнопку для входа без авторизации, а также ссылки для восстановления пароля и регистрации нового пользователя. Дизайн выполнен в минималистичном стиле, фокусирующем внимание на процессе аутентификации.

КИНОТЕАТР  
ПРЕМЬЕРА

Регистрация

Сообщения системы

Общие ошибки формы

Email

Имя

Фамилия

Телефон

Пароль

Показать

Подсказка требований к паролю

Подтверждение пароля

Показать

Зарегистрироваться

Уже есть аккаунт? Войти

Панель администратора

Или продолжите как гость

Продолжить как гость

Рисунок 14 – Страница Register

Пользователю предлагается ввести основные данные для регистрации: имя, фамилию, адрес электронной почты, номер телефона, пароль и подтвердить пароль. После успешного заполнения формы пользователь создает нажимает кнопку регистрации, далее его отправляет на страницу подтверждения почты и после успешного подтверждения создаётся учётная запись.

Изм.	Лист	№ докум.	Подпись	Дата

КИНОТЕАТР ПРЕМЬЕРА

Подтверждение email

Здравствуйте, [Имя пользователя]!

Для завершения регистрации введите следующий код подтверждения:

123456

Код действителен в течение 10 минут

Если вы не регистрировались в нашем кинотеатре, просто проигнорируйте это письмо

С уважением, Команда Кинотеатра Премьера

Рисунок 15 – Страница email\_verification

После заполнения формы регистрации и нажатия на кнопку регистрации, пользователь переходит на страницу подтверждения почты. На почту ему приходит шестизначный код, который он вводит в поле для ввода и тем самым подтверждает, что является владельцем почты.

КИНОТЕАТР ПРЕМЬЕРА

Войти

Регистрация

Панель пользователя

О кинотеатре

Фильмы

Сообщения системы

Сегодня

Завтра

Послезавтра

Дата 1

Дата 2

Фильтры

Поиск фильма

Зал

Жанр

Возрастной рейтинг

Применить

Сбросить

Постер

Название фильма 1

Жанр: Боевик | Возраст: 6+ | Длительность: 120 мин

Краткое описание фильма...

Все сеансы

10:00

13:30

17:00

20:30

Постер

Название фильма 2

Жанр: Комедия | Возраст: 8+ | Длительность: 95 мин

Краткое описание фильма...

Все сеансы

11:15

14:45

18:30

21:45

Рисунок 16 – Страница Home

Изм.	Лист	№ докум.	Подпись	Дата

КП.09.02.07.23.221.19.ПЗ

Лист

36

Главная страница является центральным узлом. На ней представлена афиша – сетка фильмов, идущих в прокате. Для каждого фильма отображается постер, название, жанр, длительность и описание. Страница предоставляет навигацию ко всем основным функциям для клиента.

**КИНОТЕАТР ПРЕМЬЕРА**

На главную
Админ-панель
Выйти

## Личный кабинет

Сообщения системы

Аватар

Имя Фамилия

**Смена email**  
Текущий email: user@example.com  


Запросить смену

Имя  
  
Фамилия  
  
Телефон  
  

Сохранить изменения

**Telegram**  
**Уведомления**  
Telegram привязан  

Управление Telegram

### История покупок

Постер

**Название фильма 1**

Скачать билеты

Группа из 2 билетов

Дата покупки: 01.01.2024
Места: Ряд 3, Места 4,5  
Сеанс: 01.01.2024 18:00
Стоимость: 1000 Р  
Зап: Красный

Постер

**Название фильма 2**

Скачать билеты

Группа из 1 билет

Дата покупки: 15.12.2023
Место: Ряд 5, Место 8  
Сеанс: 15.12.2023 20:30
Стоимость: 500 Р  
Зап: Синий

Нет купленных билетов

Рисунок 17 – Страница Profile

Личный кабинет пользователя отображает персональную информацию пользователя, а также история покупок и забронированные билеты. Пользователь может просматривать детали своих прошлых заказов, может привязать свой телеграмм аккаунт для упрощения взаимодействия с сайтом, а также поменять свои персональные данные и даже привязать свой аккаунт к другой электронной почте.

					КП.09.02.07.23.221.19.ПЗ	Лист
						37
Изм.	Лист	№ докум.	Подпись	Дата		

КИНОТЕАТР ПРЕМЬЕРА

На главную

Все сеансы

Войти

← Вернуться

Название фильма

Сообщения системы

Telegram бот

Билеты успешно куплены!

Личный кабинет

Войти

Требуется авторизация

Регистрация

Дата: 01.01.2024 18:00 | Зал: Красный | Цена: 500 Р

Описание фильма...

Схема зала

ЭКРАН

Ряд 1

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Ряд 2

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Ряд 3

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Ряд 4

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Ряд 5

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Ряд 6

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Выбрано мест: 0

Общая стоимость: 0 Р

Забронировать билеты

Рисунок 18 – Страница Screening–details

Страница информации о фильме и его сеансах открывается после выбора его на главной странице. Данный экран детализирует информацию о ближайшем сеансе на выбранный фильм: информация о фильме, дата и время, кинозал, стоимость билетов. Отсюда пользователь переходит к интерактивной схеме зала для выбора мест, что является следующим шагом в процессе бронирования.

Информация о сеансе

Начало: 01.01.2024 18:00

Окончание: 01.01.2024 20:00

Зал: Красный

Цена за место: 500 Р

ЭКРАН

Выбрано мест: 0

Ряд 1

Ряд 2

Ряд 3

Ряд 4

1-1

1-2

1-3

1-4

1-5

1-6

1-7

1-8

2-1

2-2

2-3

2-4

2-5

2-6

2-7

2-8

3-1

3-2

3-3

3-4

3-5

3-6

3-7

3-8

4-1

4-2

4-3

4-4

4-5

4-6

4-7

4-8

Купить выбранные билеты

Требуется авторизация

Для покупки билетов необходимо войти в систему

Войти

Регистрация

Рисунок 19 – screening–partial

Страница информации о конкретном сеансе открывается, когда пользователь на главной странице у фильма выбрал конкретный сеанс или на странице фильма выбрал сеанс. В отличие от прошлой страницы, это упрощённая её версия, в которой отсутствует описание фильма, а имеется информация только о выбранном сеансе.

Интерфейс спроектирован интуитивно понятным и удобным для конечного пользователя, с четкой последовательностью шагов от выбора фильма до получения билета.

Использование прототипа на раннем этапе позволило утвердить логику взаимодействия, визуальную структуру и навигацию до начала этапа разработки клиентской части, что способствовало снижению количества доработок на поздних стадиях проекта.

## 5 Разработка веб-приложения

### 5.1 Разработка интерфейса веб-приложения

Веб-приложение «Кинотеатр» было разработано с использованием фреймворка Django. Реализация интерфейса выполнена с акцентом на пользовательский опыт, безопасность и производительность. Структура шаблонов соответствует архитектуре MVT.

Все функции, указанные в техническом задании, были реализованы полностью, создав целостную экосистему для онлайн-бронирования:

1. Система аутентификации и авторизации:
  - кастомная модель пользователя;
  - Email-верификация при регистрации;
  - восстановление пароля;
  - разграничение ролей.
2. Основной функционал для пользователей:
  - просмотр афиши с фильтрацией по дате, залам и жанрам;
  - детальный просмотр информации о фильме;
  - выбор мест на интерактивной схеме зала;
  - покупка билетов с группировкой;
  - личный кабинет с историей покупок;
  - скачивание билетов в PDF формате с QR-кодами;
  - интеграция с Telegram для получения уведомлений.
3. Административный функционал:
  - CRUD операции для фильмов, залов, сеансов;
  - управление пользователями через стандартную Django админку;
  - система отчетности и логирования операций.
4. Дополнительные функции:
  - смена email с подтверждением;



– привязка/отвязка Telegram аккаунта.

На рисунке 20 изображён код интерактивной схемы зала с выбором мест в реальном времени.

```
<div class="cinema-hall">
  <div class="screen">ЭКРАН</div>

  {% for row_num, seats in rows.items %}
  <div class="row">
    <div class="row-number">{{ row_num }}</div>

    {% for seat in seats %}
    <div class="seat {% if seat.id in booked_seat_ids %}booked{% endif %}"
      data-seat-id="{{ seat.id }}"
      data-row="{{ seat.row }}"
      data-number="{{ seat.number }}">
      {{ seat.number }}
    </div>
    {% endfor %}
  </div>
  {% endfor %}
</div>

<script>
// JavaScript для выбора мест
document.querySelectorAll('.seat:not(.booked)').forEach(seat => {
  seat.addEventListener('click', function() {
    this.classList.toggle('selected');
    updateSelectedSeats();
  });
});

function updateSelectedSeats() {
  const selectedSeats = Array.from(document.querySelectorAll('.seat.selected'))
    .map(seat => seat.dataset.seatId);
  document.getElementById('selected_seats').value = JSON.stringify(selectedSeats);
}
</script>
```

Рисунок 20 – Интерактивная схема зала

На главной странице реализована динамическая фильтрация афиши по жанру, возрастному рейтингу и по дате. Код фильтрации изображён на рисунке 21.

```

def home(request):
    # Применение фильтров из GET-параметров
    search_query = request.GET.get('search', '')
    hall_filter = request.GET.get('hall', '')
    genre_filter = request.GET.get('genre', '')
    selected_date = request.GET.get('date', today.isoformat())

    # Динамический фильтр по нескольким критериям
    movies = Movie.objects.prefetch_related('screening_set__hall')

    if search_query:
        movies = movies.filter(Q(title__icontains=search_query) | Q(description__icontains=search_query))

    if genre_filter:
        movies = movies.filter(genre__name=genre_filter)

    # Отображение только сеансов на выбранную дату
    movies_data = []
    for movie in movies:
        screenings_on_date = movie.screening_set.filter(
            start_time__date=selected_date,
            start_time__gt=local_now
        ).order_by('start_time')

        movies_data.append({
            'movie': movie,
            'upcoming_screenings': screenings_on_date[:3],
            'screening_count': screenings_on_date.count()
        })

    return render(request, 'ticket/home.html', {'movies': movies_data})

```

Рисунок 21 – Фильтрация афиши

Интерфейс веб-приложения «Кинотеатр» представляет собой законченное, профессиональное решение, которое не только полностью удовлетворяет изначальным требованиям технического задания, но и закладывает фундамент для будущего развития. Сбалансированное сочетание удобства для конечного пользователя, мощности инструментов для администратора и эффективности работы на уровне кода делает систему готовой к промышленной эксплуатации. Она способна обслуживать сотни одновременных пользователей при соблюдении минимальных аппаратных требований, демонстрируя отличное соотношение функциональности и ресурсоемкости.

## 5.2 Разработка базы данных веб-приложения

Информационная модель веб-приложения «Кинотеатр» построена на основе реляционной базы данных PostgreSQL и реализована с использованием Django ORM.

Создание базы данных происходило с помощью Django ORM, которое автоматически генерировала миграции на основе прописанных моделей.

					КП.09.02.07.23.221.19.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		42

Подключение к базе данных PostgreSQL сконфигурировано в файле cinematic/settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'cinema',
        'USER': 'postgres',
        'PASSWORD': '123',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Рисунок 22 – Подключение к базе данных

С 23 по 33 рисунок изображён код создания таблиц.

```
migrations.CreateModel(
    name='BackupManager',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('name', models.CharField(max_length=255)),
        ('backup_file', models.CharField(max_length=500)),
        ('created_at', models.DateTimeField(auto_now_add=True)),
        ('backup_type', models.CharField(max_length=50)),
        ('backup_date', models.DateField(blank=True, null=True)),
    ],
    options={
        'verbose_name': 'Backup',
        'verbose_name_plural': 'Backups',
    },
),
```

Рисунок 23 – создание BackupManager

```
migrations.CreateModel(
    name='Genre',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('name', models.CharField(max_length=50, unique=True, verbose_name='Название жанра')),
        ('created_at', models.DateTimeField(auto_now_add=True)),
    ],
    options={
        'verbose_name': 'Жанр',
        'verbose_name_plural': 'Жанры',
    },
),
```

Рисунок 24 – создание Genre

```
migrations.CreateModel(
    name='AgeRating',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('name', models.CharField(help_text='Например: 0+, 6+, 12+, 16+, 18+', max_length=10, unique=True, verbose_name='Возрастной рейтинг')),
        ('description', models.TextField(blank=True, max_length=500, null=True, verbose_name='Описание ограничения')),
        ('created_at', models.DateTimeField(auto_now_add=True)),
        ('updated_at', models.DateTimeField(auto_now=True)),
    ],
    options={
        'verbose_name': 'Возрастной рейтинг',
        'verbose_name_plural': 'Возрастные рейтинги',
        'ordering': ['name'],
    },
),
```

Рисунок 25 – создание AgeRating

```
migrations.CreateModel(
    name='Hall',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('name', models.CharField(max_length=50)),
        ('rows', models.IntegerField()),
        ('seats_per_row', models.IntegerField()),
        ('description', models.TextField(blank=True, null=True)),
    ],
    options={
        'verbose_name': 'Зал',
        'verbose_name_plural': 'Залы',
    },
),
```

Рисунок 26 – создание Hall

```
migrations.CreateModel(
    name='User',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('password', models.CharField(max_length=128, verbose_name='password')),
        ('last_login', models.DateTimeField(blank=True, null=True, verbose_name='last login')),
        ('is_superuser', models.BooleanField(default=False, verbose_name='superuser status')),
        ('first_name', models.CharField(blank=True, max_length=150, verbose_name='first name')),
        ('last_name', models.CharField(blank=True, max_length=150, verbose_name='last name')),
        ('is_staff', models.BooleanField(default=False, verbose_name='staff status')),
        ('is_active', models.BooleanField(default=True, verbose_name='active')),
        ('date_joined', models.DateTimeField(default=django.utils.timezone.now, verbose_name='date joined')),
        ('email', models.EmailField(max_length=100, unique=True)),
        ('name', models.CharField(max_length=50)),
        ('surname', models.CharField(max_length=50)),
        ('number', models.CharField(max_length=50)),
        ('telegram_chat_id', models.CharField(blank=True, max_length=20, null=True)),
        ('telegram_username', models.CharField(blank=True, max_length=100, null=True)),
        ('is_telegram_verified', models.BooleanField(default=False)),
        ('telegram_verification_code', models.CharField(blank=True, max_length=10, null=True)),
        ('is_email_verified', models.BooleanField(default=False)),
        ('email_verification_code', models.CharField(blank=True, max_length=6, null=True)),
        ('email_verification_code_sent_at', models.DateTimeField(blank=True, null=True)),
    ],
    options={
        'verbose_name': 'user',
        'verbose_name_plural': 'users',
        'abstract': False,
    },
),
```

Рисунок 27 – создание User

Изм.	Лист	№ докум.	Подпись	Дата

```
migrations.CreateModel(
    name='Movie',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('title', models.CharField(max_length=100)),
        ('short_description', models.CharField(blank=True, max_length=200, null=True, verbose_name='Короткое описание')),
        ('description', models.TextField(max_length=1000, verbose_name='Полное описание')),
        ('duration', models.DurationField()),
        ('poster', models.ImageField(blank=True, null=True, upload_to='movie_posters/', verbose_name='Постер фильма')),
        ('genre', models.ForeignKey(on_delete=django.db.models.deletion.PROTECT, to='ticket.genre', verbose_name='Жанр')),
    ],
    options={
        'verbose_name': 'Фильм',
        'verbose_name_plural': 'Фильмы',
    },
),
```

Рисунок 28 – создание Movie

```
migrations.CreateModel(
    name='Screening',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('start_time', models.DateTimeField(verbose_name='Время начала')),
        ('end_time', models.DateTimeField(blank=True, null=True, verbose_name='Время окончания')),
        ('price', models.DecimalField(decimal_places=2, max_digits=6, verbose_name='Цена')),
        ('hall', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='ticket.hall', verbose_name='Зал')),
        ('movie', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='ticket.movie', verbose_name='Фильм')),
    ],
    options={
        'verbose_name': 'Сеанс',
        'verbose_name_plural': 'Сеансы',
    },
),
```

Рисунок 29 – создание Screening

```
migrations.CreateModel(
    name='Seat',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('row', models.IntegerField()),
        ('number', models.IntegerField()),
        ('hall', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='ticket.hall')),
    ],
    options={
        'verbose_name': 'Место',
        'verbose_name_plural': 'Места',
    },
),
```

Рисунок 30 – создание Seat

```
migrations.CreateModel(
    name='TicketStatus',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('code', models.CharField(max_length=20, unique=True, verbose_name='Код статуса')),
        ('name', models.CharField(max_length=30, verbose_name='Название статуса')),
        ('description', models.TextField(blank=True, null=True, verbose_name='Описание статуса')),
        ('is_active', models.BooleanField(default=True, verbose_name='Активный статус')),
        ('can_be_refunded', models.BooleanField(default=False, verbose_name='Можно вернуть из этого статуса')),
        ('created_at', models.DateTimeField(auto_now_add=True)),
        ('updated_at', models.DateTimeField(auto_now=True)),
    ],
    options={
        'verbose_name': 'Статус билета',
        'verbose_name_plural': 'Статусы билетов',
        'ordering': ['id'],
    },
),
```

Рисунок 31 – создание TicketStatus

```
migrations.CreateModel(
    name='Ticket',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('purchase_date', models.DateTimeField(auto_now_add=True)),
        ('qr_code', models.ImageField(blank=True, null=True, upload_to='qr_codes/')),
        ('group_id', models.CharField(blank=True, db_index=True, max_length=100, null=True)),
        ('screening', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='ticket.screening')),
        ('seat', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='ticket.seat')),
        ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
    ],
    options={
        'verbose_name': 'Билет',
        'verbose_name_plural': 'Билеты',
    },
),
```

Рисунок 32 – создание Ticket

```
migrations.CreateModel(
    name='OperationLog',
    fields=[
        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
        ('action_type', models.CharField(max_length=10, verbose_name='Тип действия')),
        ('module_type', models.CharField(max_length=15, verbose_name='Модуль')),
        ('description', models.TextField(verbose_name='Описание')),
        ('ip_address', models.GenericIPAddressField(blank=True, null=True, verbose_name='IP адрес')),
        ('user_agent', models.TextField(blank=True, null=True, verbose_name='User Agent')),
        ('object_id', models.IntegerField(blank=True, null=True, verbose_name='ID объекта')),
        ('object_repr', models.CharField(blank=True, max_length=255, null=True, verbose_name='Объект')),
        ('additional_data', models.JSONField(blank=True, null=True, verbose_name='Дополнительные данные')),
        ('timestamp', models.DateTimeField(default=django.utils.timezone.now, verbose_name='Время операции')),
        ('user', models.ForeignKey(blank=True, null=True, on_delete=django.db.models.deletion.SET_NULL, to=settings.AUTH_USER_MODEL)),
    ],
    options={
        'verbose_name': 'лог операции',
        'verbose_name_plural': 'Логи операций',
        'ordering': ['-timestamp'],
    },
),
```

Рисунок 33 – создание OperationLog

В ходе работы была разработана структура базы данных, обеспечивающая хранение необходимых данных и поддерживающая целостность информации. Схема отражает основные объекты и их связи, что позволяет эффективно управлять данными и обеспечивать быстрый доступ для дальнейшей обработки и использования в системе.

### 5.3 Разработка веб-приложения

Разработка веб-приложения «Кинотеатр» осуществлялась на основе спроектированной архитектуры с использованием фреймворка Django и реляционной СУБД PostgreSQL. В данном разделе описывается реализация подключения к базе данных, методы работы с ORM и ход выполнения ключевых бизнес-процессов.

Система построена по классической архитектуре Django MVT:

Изм.	Лист	№ докум.	Подпись	Дата

1. Модели – описание структуры данных в ticket/models.py.
2. Представления – бизнес-логика в ticket/views.py.
3. Шаблоны – отображение данных в ticket/templates/.

Ядро системы составляют 11 взаимосвязанных моделей данных, полностью соответствующих диаграмме классов из раздела 4.2 (рисунок 8). Модели реализованы с использованием Django ORM с учетом всех бизнес-ограничений. Например, на рисунке 34 изображена модель Ticket реализует систему бронирования с поддержкой групповых покупок и автоматического возврата.

```
class Ticket(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    screening = models.ForeignKey(Screening, on_delete=models.CASCADE)
    seat = models.ForeignKey(Seat, on_delete=models.CASCADE)
    group_id = models.CharField(max_length=40, blank=True, null=True, db_index=True)
    status = models.ForeignKey(TicketStatus, on_delete=models.PROTECT)

    def can_be_refunded(self):
        """Проверка возможности возврата билета"""
        from django.utils import timezone
        time_until_screening = self.screening.start_time - timezone.now()
        minutes_until = time_until_screening.total_seconds() / 60
        return minutes_until >= 30 # Возврат возможен за 30+ минут до сеанса
```

Рисунок 34 – Модель Ticket

Контроллеры приложения реализуют полный цикл взаимодействия с пользователем. Например, на рисунке 35 изображена реализации возврата билета с автоматической проверкой условий возвратов.

```
@login_required
@require_POST
def request_ticket_refund(request, ticket_id):
    ticket = get_object_or_404(Ticket, id=ticket_id, user=request.user)

    # Автоматическая проверка всех условий возврата
    success, message = ticket.request_refund()

    if success:
        # Автоматическое обновление статуса и логирование
        refunded_status = TicketStatus.objects.get(code='refunded')
        ticket.status = refunded_status
        ticket.refund_processed_at = timezone.now()
        ticket.save()

        OperationLogger.log_operation(
            request=request,
            action_type='UPDATE',
            module_type='TICKETS',
            description=f'Автоматический возврат билета #{ticket_id}'
        )

        messages.success(request, '👍 Билет успешно возвращен!')

    return redirect('profile')
```

Рисунок 35 – Возврат билета

В результате разработки серверной части была реализована структура, обеспечивающая обработку запросов. Использование контроллеров и декораторов позволило централизованно обрабатывать данные, вести логирование действий и безопасно работать с базой данных.

Таким образом, веб-приложение «Кинотеатр» представляет собой полнофункциональную систему, реализующую все запланированные бизнес-процессы – от онлайн-бронирования билетов до комплексного администрирования. Использование Django ORM обеспечивает надежную работу с PostgreSQL, а модульная архитектура позволяет легко расширять функциональность в будущем.

					КП.09.02.07.23.221.19.ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		



## 6 Тестирование веб-приложения

### 6.1 Тестовые сценарии

Тестирование веб-приложения «Кинотеатр» проводилось с использованием встроенного фреймворка Django Test Case. Были разработаны и выполнены тестовые сценарии, охватывающие ключевые функциональные возможности системы. Сценарии разделены на позитивные и негативные.

Позитивные сценарии демонстрируют корректную работу функций при использовании системы по назначению, тогда как негативные позволяют проверить устойчивость системы к ошибочным действиям и некорректным данным.

Первый функциональный сценарий (таблица 16) направлен на тестирование регистрации и верификации пользователя.

Таблица 16 – Сценарий тестирования успешной регистрации и верификации пользователя

Параметр	Описание
ID теста	FPOS-01-тест-аутентификации-1
Название теста	Успешная регистрация пользователя с email-верификацией
Предусловия	1. Пользователь не зарегистрирован в системе. 2. Email не используется другими пользователями.
Шаги	1. Заполнить форму регистрации и отправить запрос на регистрацию. 2. Ввести код верификации. 3. Проверить создание пользователя в БД. 4. Выполнить авторизацию с созданными учетными данными.
Тестовые данные	Email: testuser@example.com, Имя: Иван, Фамилия: Иванов, Телефон: +79123456789, Пароль: TestPassword123
Ожидаемый результат	Пользователь успешно зарегистрирован, email верифицирован, доступна авторизация в системе.

Следующий функциональный сценарий (таблица 17) направлен на тестирование генерации PDF отчётов.

Таблица 17 – Сценарий тестирования успешной генерации PDF отчетов

Параметр	Описание
ID теста	FPOS-02-тест-pdf-генерации-2
Название теста	Успешная генерация PDF отчетов различных типов
Предусловия	<ol style="list-style-type: none"> <li>1. В системе присутствуют тестовые данные для отчетов.</li> <li>2. Установлены шрифты для генерации PDF.</li> </ol>
Шаги	<ol style="list-style-type: none"> <li>1. Создать тестовые данные для финансового отчета, отчета по фильмам и отчетов по залам.</li> <li>2. Вызвать функцию generate_pdf_report для каждого типа отчета.</li> <li>3. Проверить корректность созданных PDF файлов.</li> <li>4. Сохранить PDF во временный файл для визуальной проверки.</li> </ol>
Тестовые данные	Финансовые данные за период, данные о популярности фильмов, статистика загруженности залов.
Ожидаемый результат	Корректная генерация PDF файлов всех типов с правильным форматированием и данными.

Следующий функциональный сценарий (таблица 18) направлен на тестирование генерации аналитических отчётов.

Таблица 18 – Сценарий тестирования успешной генерации аналитических отчетов

Параметр	Описание
ID теста	FPOS-03-тест-отчетов-3
Название теста	Успешная генерация аналитических отчетов.
Предусловия	<ol style="list-style-type: none"> <li>1. В БД созданы тестовые пользователи, фильмы, залы, сеансы и билеты.</li> <li>2. Имеются данные за различные периоды времени.</li> </ol>
Шаги	<ol style="list-style-type: none"> <li>1. Запросить отчет по фильмам.</li> <li>2. Запросить отчет по залам.</li> <li>3. Запросить общую статистику продаж.</li> <li>4. Запросить финансовой статистику.</li> <li>5. Проверить структуру и корректность данных в отчетах.</li> </ol>

## Продолжение таблицы 18

Тестовые данные	Тестовые данные за 30 дней: 5 пользователей, 3 фильма, 2 зала, 10 сеансов, 25 билетов.
Ожидаемый результат	Корректное формирование всех типов отчетов с актуальными и точными данными.

Негативные тестовые сценарии проверяют поведение системы при некорректных действиях пользователя или попытках нарушения правил работы системы. Первый негативный сценарий (таблица 19) направлен на проверку бронирования билетов с нарушениями условий.

Таблица 19 – Сценарий тестирования некорректного бронирования билетов

Параметр	Описание
ID теста	FUNC-NEG-1-тест-бронирования
Название теста	Попытка бронирования билетов с нарушением условий
Предусловия	<ol style="list-style-type: none"> <li>1. Пользователь авторизован в системе.</li> <li>2. В зале существуют как свободные, так и занятые места.</li> <li>3. Создан сеанс на будущее время.</li> </ol>
Шаги	<ol style="list-style-type: none"> <li>1. Попытка бронирования без выбора мест.</li> <li>2. Попытка бронирования уже занятого места.</li> <li>3. Попытка бронирования несуществующего места.</li> <li>4. Попытка бронирования с невалидным JSON в запросе.</li> <li>5. Попытка бронирования без авторизации.</li> </ol>
Тестовые данные	Занятые места: ряд 1 места 1-2; свободные места: ряд 1 места 3-5; несуществующее место ID: 99999.
Ожидаемый результат	Система корректно отклоняет все некорректные попытки бронирования с соответствующими сообщениями об ошибках.

Следующий негативный сценарий (таблица 20) направлен на тестирование возврата билетов с нарушением условий.

Таблица 20 – Сценарий тестирования некорректного возврата билетов

Параметр	Описание
ID теста	FUNC-NEG-2-тест-возврата
Название теста	Попытка возврата билетов с нарушением условий
Предусловия	1. Созданы билеты с различными статусами и временем до сеанса. 2. Пользователь авторизован и является владельцем билетов.
Шаги	1. Попытка возврата билета на прошедший сеанс. 2. Попытка возврата билета за 15 минут до начала. 3. Попытка возврата уже возвращенного билета. 4. Попытка возврата билета со статусом "Запрошен возврат". 5. Попытка возврата билета с неактивным статусом.
Тестовые данные	Билеты: прошедший сеанс, скоро начинающийся, будущий, возвращенный, запрошенный возврат, неактивный статус.
Ожидаемый результат	Система корректно отклоняет все некорректные запросы на возврат, соблюдая бизнес-правила.

Следующий негативный сценарий (таблица 21) направлен на тестирование валидации моделей.

Таблица 21 – Сценарий тестирования намеренных ошибок валидации моделей

Параметр	Описание
ID теста	FUNC-NEG-3-тест-валидации
Название теста	Проверка граничных случаев и намеренных ошибок валидации
Предусловия	1. Существуют базовые тестовые объекты. 2. Система настроена на строгую валидацию данных.
Шаги	1. Создание пользователя с некорректным форматом email. 2. Создание пользователя с дублирующим email. 3. Создание пересекающихся сеансов в одном зале. 4. Создание сеанса в нерабочее время.

## Продолжение таблицы 21

Тестовые данные	Некорректный email, дубликат email, пересекающиеся времена сеансов, ночное время.
Ожидаемый результат	Система валидации корректно обнаруживает и предотвращает все некорректные операции, выбрасывая соответствующие исключения ValidationError.

Разработанные функциональные тестовые сценарии подтвердили корректность работы основных функций веб-приложения и соответствие заданным требованиям. Позитивные сценарии показали, что система правильно обрабатывает запросы. Негативные сценарии продемонстрировали устойчивость приложения к ошибочным действиям. Проведённое тестирование подтвердило корректность ключевых механизмов системы.

### 6.2 Методы тестирования

Для тестирования веб-приложения использовался фреймворк Django Test Case, который предоставляет встроенные инструменты для тестирования моделей, представлений и шаблонов. Тестирование проводилось в изолированной тестовой базе данных, что гарантировало чистоту тестов и отсутствие побочных эффектов.

На рисунках 36 – 41 изображены ключевые фрагменты кода функционального теста.

```
def test_fpos_01_registration_success(self):
    """FPOS-01: Успешная регистрация с верификацией email"""
    # Отправка формы регистрации
    response = self.client.post(reverse('register'), self.test_data)

    # Проверка редиректа на верификацию
    self.assertEqual(response.status_code, 302)
    self.assertRedirects(response, reverse('verify_email'))

    # Проверка создания временной записи
    pending_reg = PendingRegistration.objects.filter(
        email=self.test_email).first()
    self.assertIsNotNone(pending_reg)

    # Верификация email
    verification_data = {
        'verification_code': pending_reg.verification_code
    }
    response = self.client.post(reverse('verify_email'), verification_data)

    # Проверка создания пользователя
    user = User.objects.filter(email=self.test_email).first()
    self.assertIsNotNone(user)
    self.assertTrue(user.is_email_verified)
```

Рисунок 36 – Фрагмент кода теста test\_fpos\_01\_registration\_succes

```

def test_fpos_02_pdf_report_generation(self):
    """FPOS-02: Успешная генерация PDF отчетов"""
    # Генерация финансового отчета
    pdf_buffer = generate_pdf_report(
        data=self.test_data,
        report_type='revenue',
        title='Тестовый финансовый отчет',
        filters={'period': 'daily'}
    )

    # Проверка корректности PDF
    pdf_content = pdf_buffer.getvalue()
    self.assertGreater(len(pdf_content), 0)
    self.assertIn(b'%PDF', pdf_content[:100]) # Сигнатура PDF

    # Сохранение для проверки
    with tempfile.NamedTemporaryFile(suffix='.pdf') as temp_file:
        temp_file.write(pdf_content)
        self.assertTrue(os.path.exists(temp_file.name))

```

Рисунок 37 – Фрагмент кода теста test\_frop\_02\_pdf\_report\_generation

```

def test_fpos_03_report_generation_success(self):
    """FPOS-03: Генерация аналитических отчетов"""
    # Отчет по популярным фильмам
    movies_report = ReportGenerator.get_popular_movies(limit=10)
    self.assertIsInstance(movies_report, list)

    # Отчет по загруженности залов
    halls_report = ReportGenerator.get_hall_occupancy()
    self.assertIsInstance(halls_report, list)

    # Статистика продаж
    sales_stats = ReportGenerator.get_sales_statistics()
    expected_keys = ['total_tickets', 'total_revenue', 'avg_ticket_price']
    for key in expected_keys:
        self.assertIn(key, sales_stats)

```

Рисунок 38 – Фрагмент кода теста test\_03\_report\_generation\_succes

```

def test_func_neg_1_ticket_booking_negative_scenarios(self):
    """FUNC-NEG-1: Тест негативных сценариев бронирования"""
    # Попытка бронирования без выбора мест
    response = self.client.post(reverse('book_tickets'), {
        'screening_id': self.screening.id,
        'selected_seats': '' # Пустой список
    })
    self.assertEqual(response.status_code, 302)

    # Попытка бронирования занятого места
    booked_seat_id = self.booked_seats[0].id
    response = self.client.post(reverse('book_tickets'), {
        'screening_id': self.screening.id,
        'selected_seats': json.dumps([booked_seat_id])
    }, follow=True)

    # Проверка сообщения об ошибке
    messages = list(response.context['messages'])
    error_found = any('уже занято' in str(message) for message in messages)
    self.assertTrue(error_found)

```

Рисунок 39 – Фрагмент кода теста  
test\_func\_neg\_1\_ticket\_booking\_negative\_scenarios

```

def test_func_neg_2_ticket_refund_negative_scenarios(self):
    """FUNC-NEG-2: Негативные сценарии возврата билетов"""
    # Попытка возврата билета на прошедший сеанс
    response = self.client.post(
        reverse('request_ticket_refund', args=[self.past_ticket.id]),
        follow=True
    )
    self.past_ticket.refresh_from_db()
    self.assertEqual(self.past_ticket.status.code, 'active')

    # Попытка возврата за 15 минут до начала
    can_refund, message = self.soon_ticket.can_be_refunded()
    self.assertFalse(can_refund)
    self.assertIn('30 минут', message)

```

Рисунок 40 – Фрагмент кода теста

test\_func\_neg\_2\_ticket\_refund\_negative\_scenarios

```

def test_func_neg_3_intentional_validation_failures(self):
    """FUNC-NEG-3: Тест с намеренными ошибками валидации"""
    # Попытка создания пользователя с некорректным email
    with self.assertRaises(ValidationError):
        user = User(email='некорректный-email')
        user.full_clean()

    # Попытка создания дублирующего email
    User.objects.create_user(email='duplicate@example.com')
    with self.assertRaises(Exception) as context:
        user2 = User(email='duplicate@example.com')
        user2.save()
    self.assertIn('уже существует', str(context.exception))

    # Попытка создания пересекающихся сеансов
    with self.assertRaises(ValidationError) as context:
        screening2 = Screening(
            hall=self.hall,
            start_time=self.screening1.start_time + timedelta(minutes=30)
        )
        screening2.clean()
    self.assertIn('пересекается', str(context.exception).lower())

```

Рисунок 41 – Фрагмент кода теста test\_func\_neg\_3\_intentional\_validation\_failures

Все разработанные тесты были успешно выполнены. Система продемонстрировала высокую устойчивость к некорректным данным и действиям пользователей. Тестирование подтвердило, что веб-приложение «Кинотеатр» соответствует всем функциональным требованиям, указанным в техническом задании, и готово к промышленной эксплуатации.

## 7 Документирование веб-приложения

### 7.1 Руководство по установке веб-приложения

Перед началом установки веб-приложения «Кинотеатр Премьера» необходимо убедиться, что на компьютере установлены все необходимые компоненты:

- Python 3.12 или выше
- PostgreSQL 16 или совместимая версия
- Django 5.2.3
- Pip

Шаги установки:

1. Клонирование и настройка проекта на рисунке 42:

```
# Клонирование проекта (если используется Git)
git clone <репозиторий>
cd cinema-management-system

# Создание виртуального окружения
python -m venv .venv

# Активация виртуального окружения
# Для Windows:
.venv\Scripts\activate
# Для Linux/Mac:
source .venv/bin/activate
```

Рисунок 42 – Клонирование и настройка проекта

2. Установка зависимостей на рисунке 43:

```
pip install -r requirements.txt
```

Рисунок 43 – Установка зависимостей

3. Настройка базы данных PostgreSQL



- Создайте базу данных с именем cinema
- Настройте пользователя postgres с паролем 123 (или измените настройки в cinematic/settings.py)

4. Создание и применение миграций на рисунке 44:

```
python manage.py makemigrations
python manage.py migrate
```

Рисунок 44 – Создание и применение миграций

5. Создание суперпользователя на рисунке 45:

```
python manage.py createsuperuser
```

Рисунок 45 – Создание суперпользователя

6. Заполнение базы данных тестовыми данными на рисунке 46:

```
python manage.py populate_db
или
python manage.py big_populate_db
```

Рисунок 46 – Заполнение базы данных тестовыми данными

7. Запуск сервера разработки на рисунке 47:

```
python manage.py runserver
```

Рисунок 47 – Запуск сервера

## 7.2 Руководство гостя

При переходе по адресу «<http://127.0.0.1:8080>», пользователь попадает на главную страницу кинотеатра, представленную на рисунке 48.

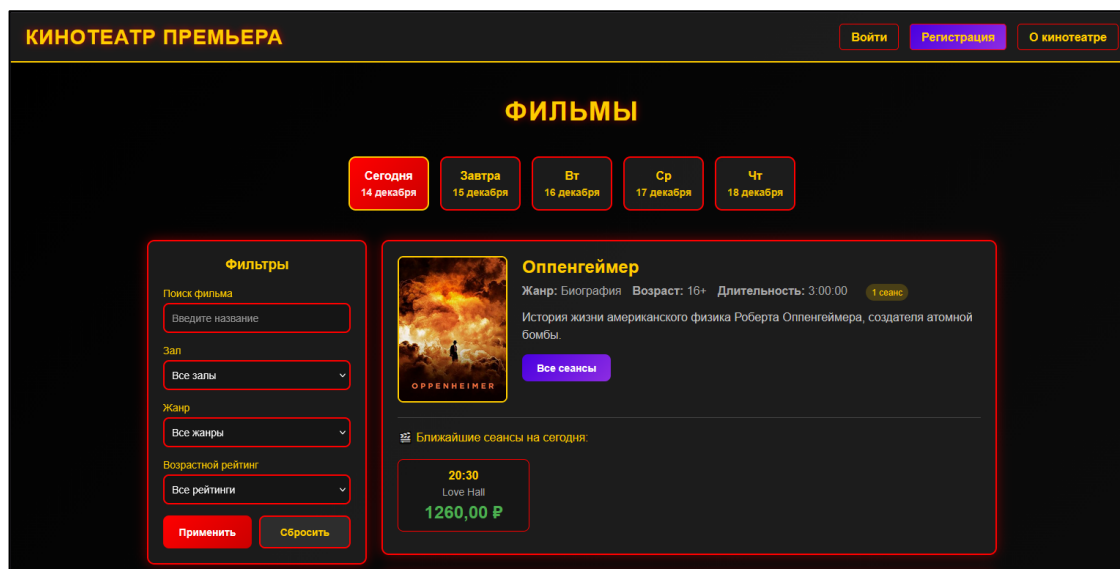


Рисунок 48 – Главная страница

На главной странице в левой секции «Фильтры» доступен поиск фильма по названию, а также фильтрация по жанру, залу и возрастному рейтингу. Над секциями фильтров и фильмов находится фильтр по дате. Здесь же он может посмотреть подробную информацию о фильме и о любом сеансе на этот фильм. Страница информации о фильме изображена на рисунке 49.

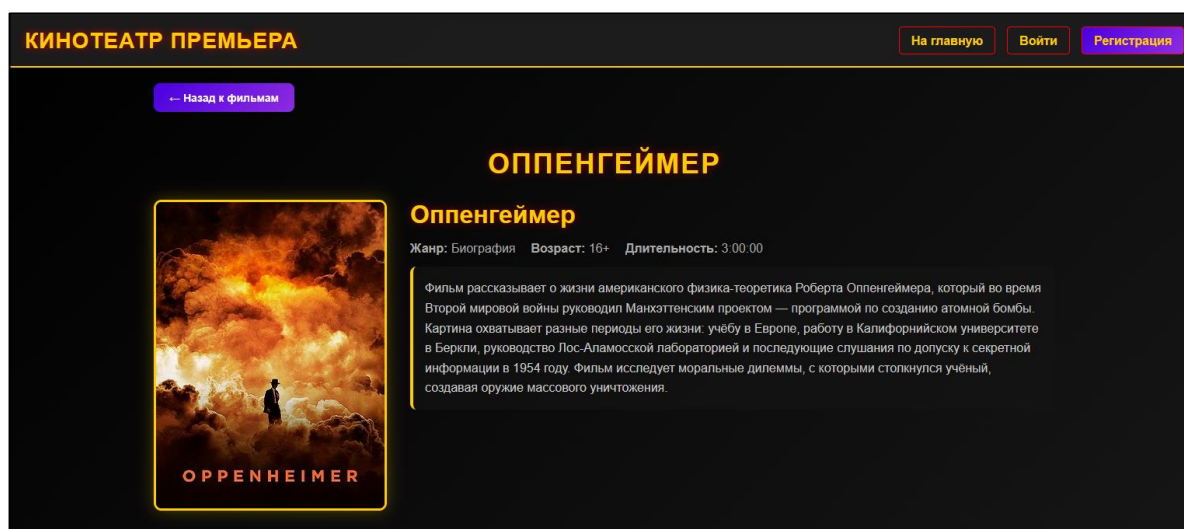


Рисунок 49 – Подробная информация о фильме

Страница информации о сеансе изображена на рисунке 50.

20:30  
14 Декабрь 2025  
Love Hall  
1260,00 Р

08:00  
19 Декабрь 2025  
Стандарт  
244,00 Р

10:30  
10 Декабрь 2025  
Комфорт  
385,00 Р

10:30  
10 Декабрь 2025  
VIP Зал  
770,00 Р

18:00  
10 Декабрь  
Комфорт  
660,00 Р

Информация о сеансе

Начало:  
10 Декабрь 2025 10:30

Окончание:  
10 Декабрь 2025 13:40

Зал:  
VIP Зал

Цена за место:  
770,00 Р

ЭКРАН

Выбрано мест: 0

Ряд 1

1-1, 1-2, 1-3, 1-4, 1-5, 1-6, 1-7, 1-8

Ряд 2

2-1, 2-2, 2-3, 2-4, 2-5, 2-6, 2-7, 2-8

Ряд 3

3-1, 3-2, 3-3, 3-4, 3-5, 3-6, 3-7, 3-8

Ряд 4

4-1, 4-2, 4-3, 4-4, 4-5, 4-6, 4-7, 4-8

Ряд 5

5-1, 5-2, 5-3, 5-4, 5-5, 5-6, 5-7, 5-8

Ряд 6

6-1, 6-2, 6-3, 6-4, 6-5, 6-6, 6-7, 6-8

Требуется авторизация

Для покупки билетов необходимо войти в систему

ВОЙТИ

РЕГИСТРАЦИЯ

Рисунок 50 – Подробная информация о сеансе

Для приобретения билетов пользователю нужно зарегистрироваться. Страница регистрации изображена на рисунке 51.

КИНОТЕАТР  
ПРЕМЬЕРА

РЕГИСТРАЦИЯ

Email

Ваш email

Email обязателен для заполнения

Имя

Ваше имя

Имя обязательно для заполнения

Фамилия

Ваша фамилия

Фамилия обязательна для заполнения

Телефон

+7 (800) XXX-XX-XX

Телефон обязателен для заполнения

Пароль

Придумайте пароль

Пароль обязателен для заполнения

Подтверждение пароля

Повторите пароль

Подтвердите пароль

ЗАРЕГИСТРИРОВАТЬСЯ

Уже есть аккаунт? Войти

Панель администратора

Или продолжите как гость

Продолжить как гость

Рисунок 51 – Страница регистрации

Изм.	Лист	№ докум.	Подпись	Дата

После заполнения регистрации, пользователю отправляется шестизначный код на почту, который он должен ввести в систему для подтверждения подлинности почты. Страница подтверждения почты изображена на рисунке 52.

**КИНОТЕАТР  
ПРЕМЬЕРА**

**Подтверждение Email**

Код подтверждения отправлен на email  
vladislavmix2020@gmail.com

**Код подтверждения**

000100

Введите 6-значный код из письма

**ПОДТВЕРДИТЬ EMAIL**

Не получили код? [Отправить повторно](#)

[← Вернуться к регистрации](#)

Рисунок 52 – Страница подтверждения почты

Далее пользователь считается авторизованным в системе и после регистрации и подтверждения почты попадает на главную страницу.

### 7.3 Руководство авторизованного пользователя

Для того, чтобы у пользователя было больше возможностей, он должен быть зарегистрированным и авторизованным в системе. Если пользователь зарегистрирован, но не авторизован, он переходит на страницу авторизации и вводит свои данные для входа. Страница авторизации изображена на рисунке 53.

Рисунок 53 – Страница авторизации

После авторизации пользователя попадает на главную страницу и уже выбирает фильм и сеанс. После выбора сеанса пользователь выбирает места и покупает билеты. Выбор мест изображён на рисунке 54 и покупка билетов изображена на рисунке 55.

Рисунок 54 – Выбор мест

Рисунок 55 – Покупка билетов

Купленные билеты пользователь может посмотреть в личном кабинете. Там же он может сменить почту, поменять свои персональные данные и привязать телеграмм аккаунт. Личный кабинет изображён на рисунке 56.

Рисунок 56 – Личный кабинет

В личном кабинете пользователь может скачать билеты в PDF формате и оформить возврат билетов.

## 7.4 Руководство администратора

Что бы быть администратором, пользователь должен обладать правами администратора (is\_staff). Управление происходит из встроенной в Django админ-панели. Администратор может авторизоваться на сайт и потом перейти в админ-панель, а может на странице авторизации выбрать «Панель администратора» и перейти на страницу авторизации для администрации. Страница авторизации для администрации изображена на рисунке 57.

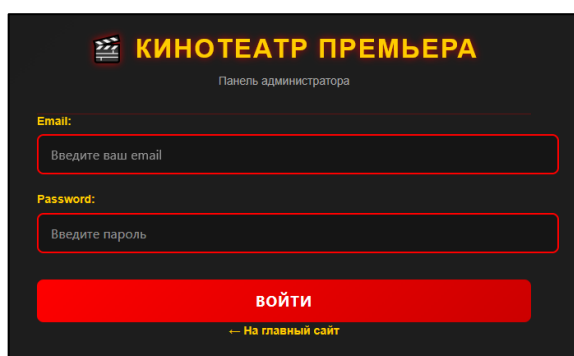


Рисунок 57 – Страница авторизации в админ-панель

После авторизации и перехода в админ-панель, администратора встречают все использующиеся на сайте модели. Админ-панель представлена на рисунке 58.

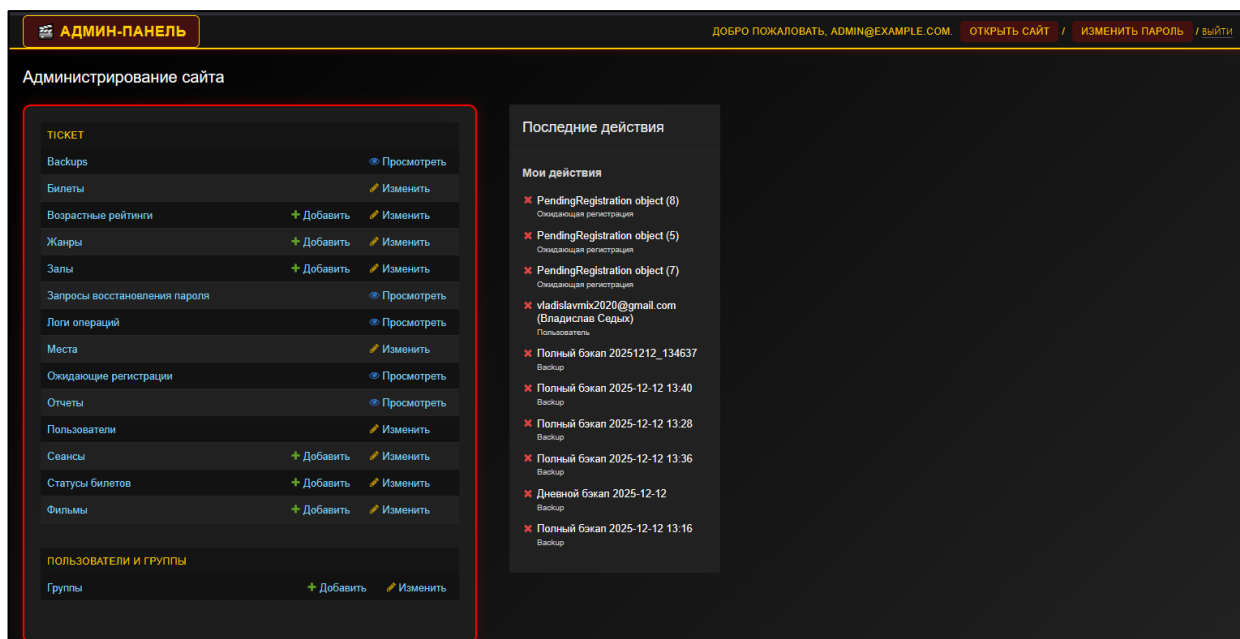


Рисунок 58 – Админ-панель

Администратор может управлять бэкапами. Он может их просматривать, создавать и восстанавливать из них базу данных. Страница модели Backups и страница управления бэкапами изображены на рисунках 59 и 60.

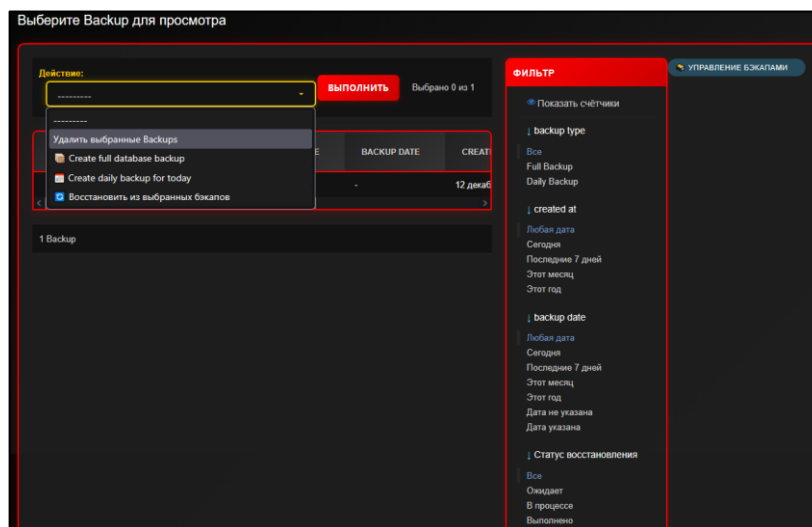


Рисунок 59 – Страница модели Backups

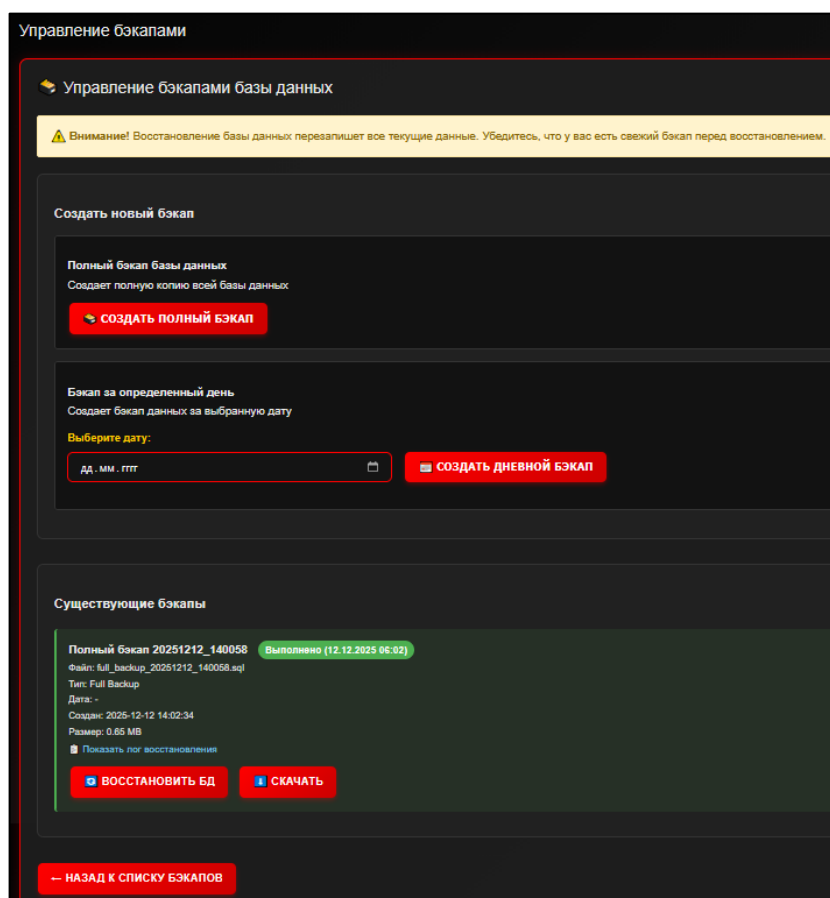


Рисунок 60 – Страница управления бэкапами



Администратор может просматривать, изменять, обрабатывать возвраты, отменять запросы возвратов и удалять билеты. Страница модели «Билеты» изображена на рисунке 61.

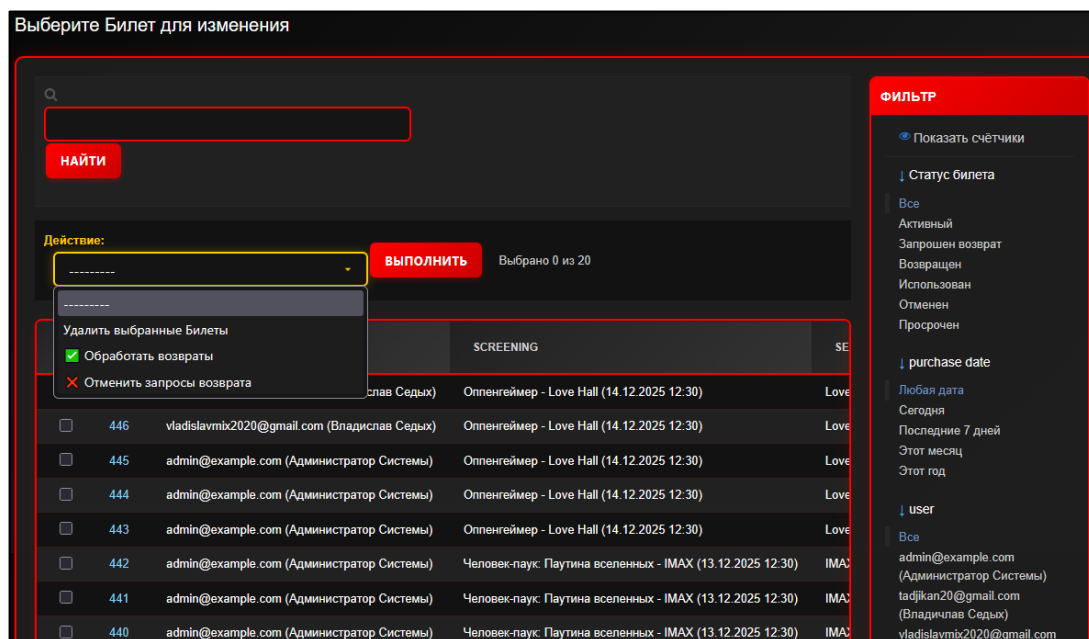


Рисунок 61 – Страница модели «Билеты»

Администратор имеет полный доступ к взаимодействиям с возрастным рейтингом. Страница модели «Возрастные рейтинги» изображена на рисунке 62.

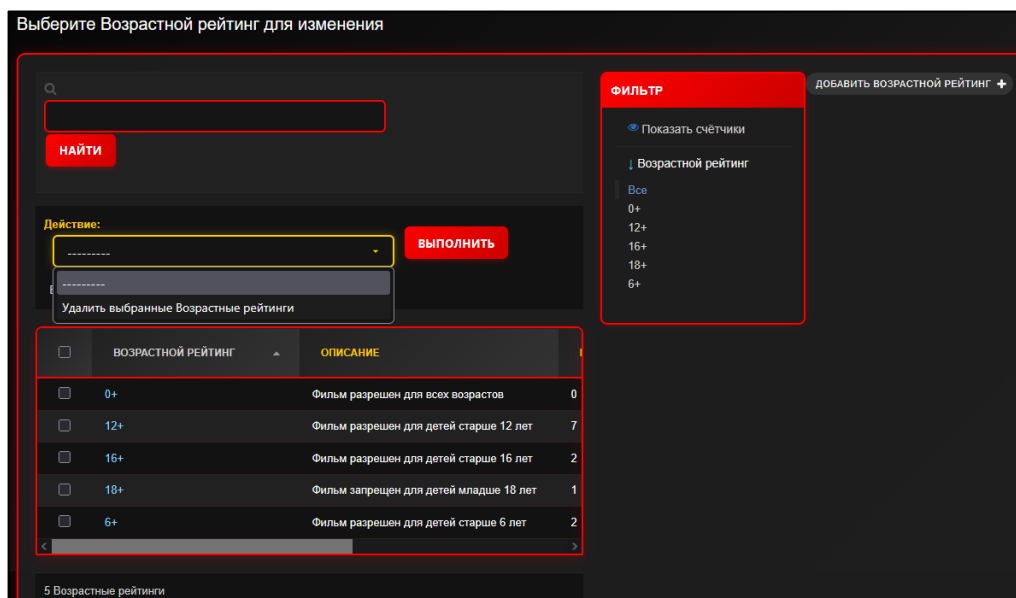


Рисунок 62 – Страница модели «Возрастные рейтинги»

Администратор имеет полный доступ к взаимодействиям с жанрами.  
Страница модели «Жанры» изображена на рисунке 63.

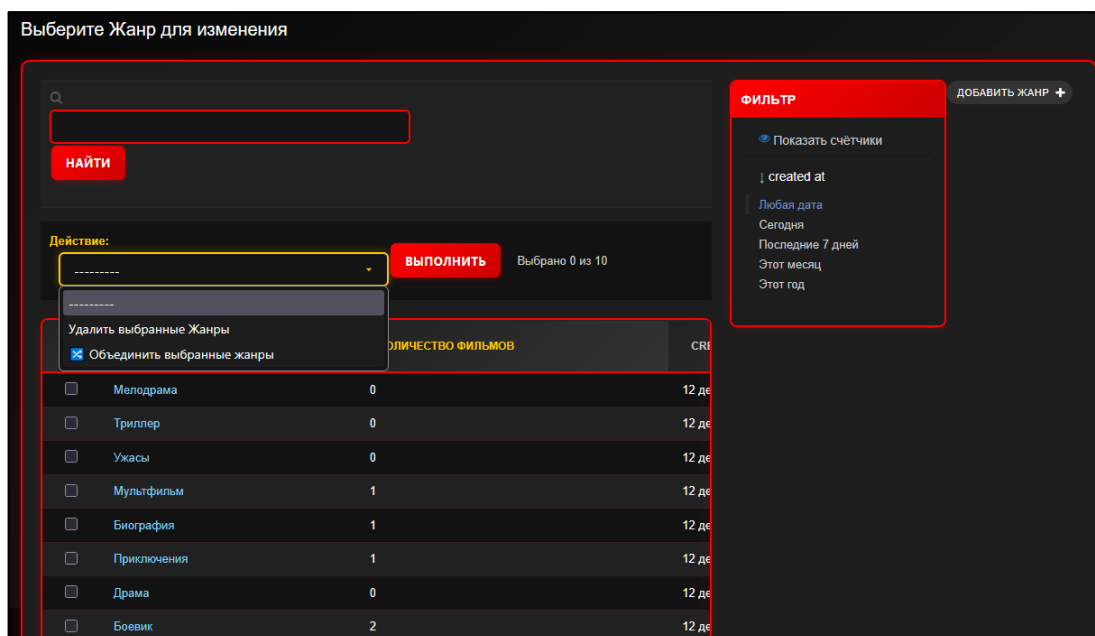


Рисунок 63 – Страница модели «Жанры»

Администратор имеет полный доступ к взаимодействиям с залами.  
Страница модели «Залы» изображена на рисунке 64.

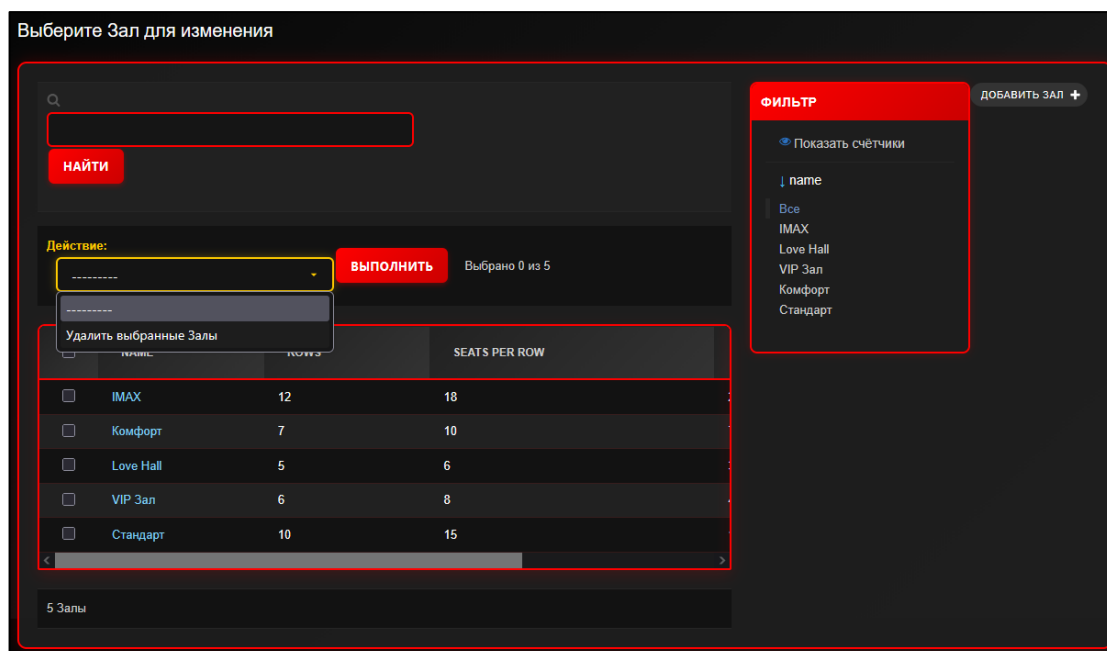


Рисунок 64 – Страница модели «Залы»

Администратор может просматривать пользователей, которые восстанавливают пароль с помощью почты. Страница «Запросы восстановления пароля» изображена на рисунке 65.

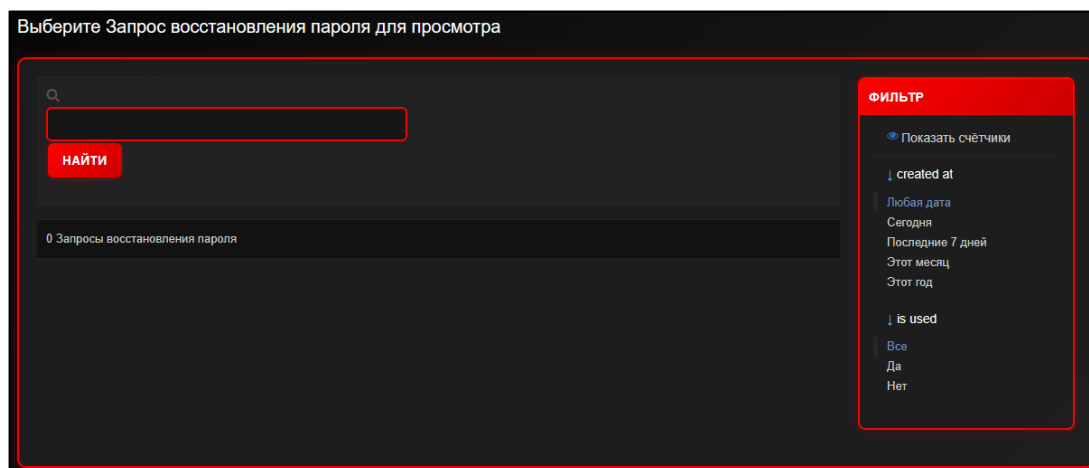


Рисунок 65 – Страница «Запросы восстановления пароля»

Администратор может просматривать логи операций, проходящих в системе, а так же экспортировать их в формате JSON и PDF с фильтрами. Страница модели «Логи операций» и страница экспорта логов изображены на рисунках 66 и 67.

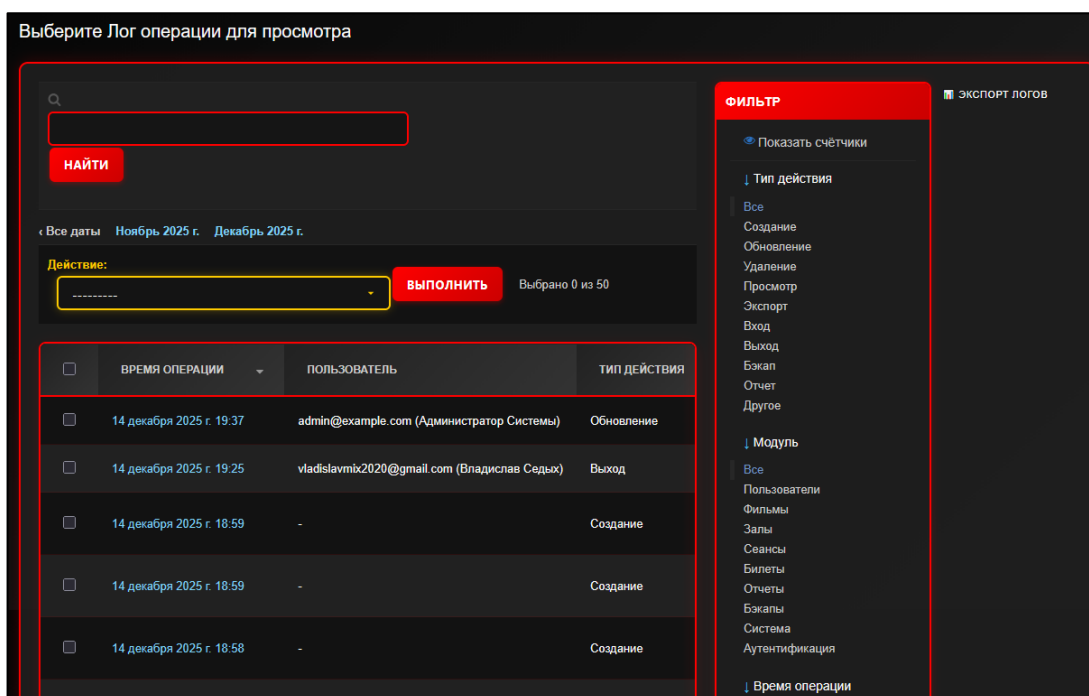


Рисунок 66 – Страница модели «Логи операций»



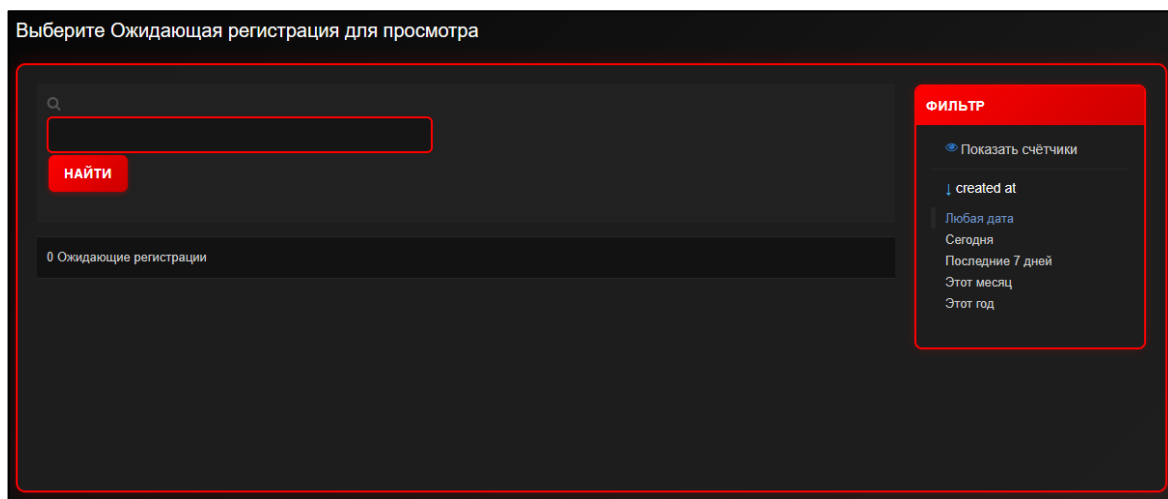


Рисунок 69 – Страница «Ожидающие регистрации»

Администратор может просматривать, генерировать и скачивать в формате PDF 4 вида отчётов: «Финансовая статистика», «Популярность фильмов», «Загруженность залов», «Статистика продаж». Страница отчётов изображена на рисунке 70.

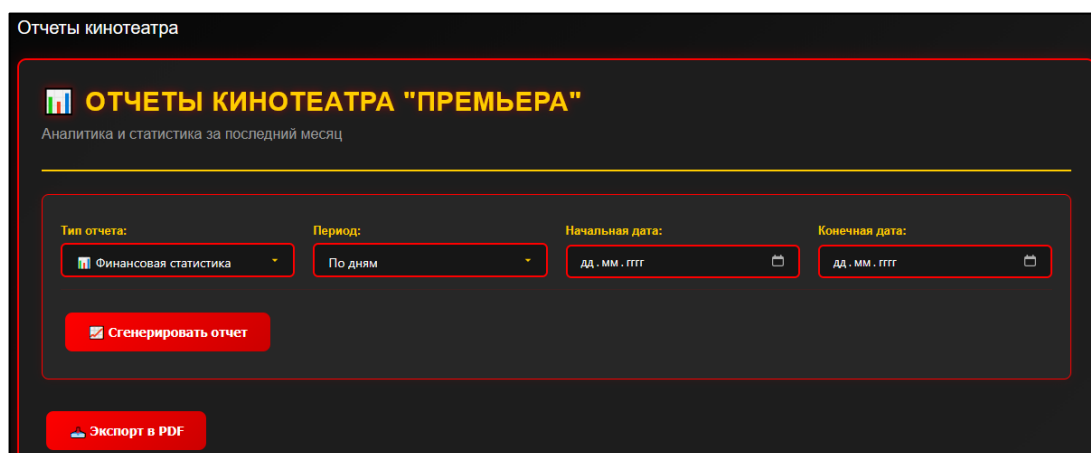


Рисунок 70 – Страница отчётов

Администратор может просматривать, изменять и удалять пользователей. Страница модели «Пользователи» изображена на рисунке 71.

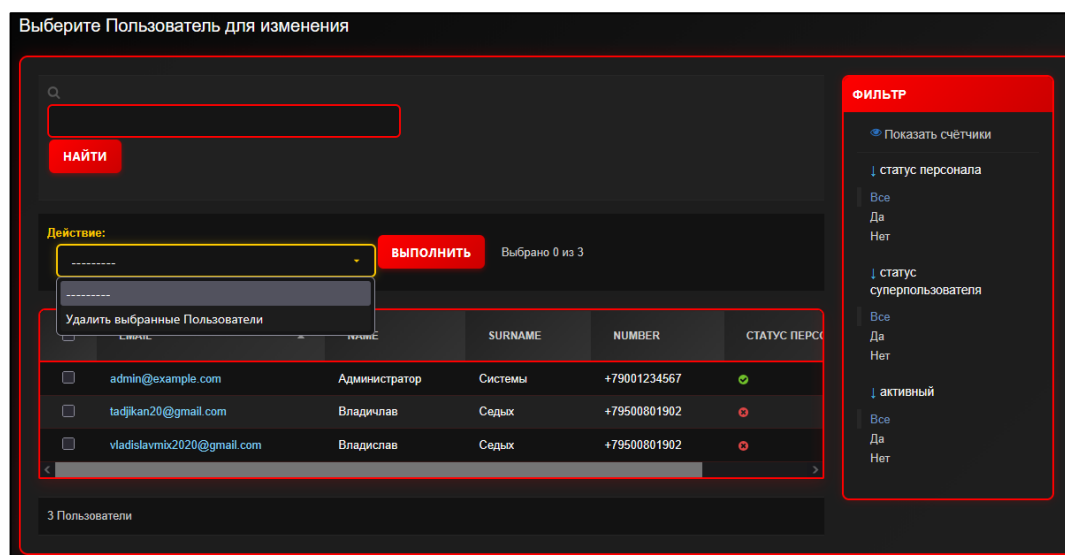


Рисунок 71 – Страница модели «Пользователи»

Администратор имеет полный доступ к взаимодействиям с сеансами. Страница модели «Сеансы» изображена на рисунке 72.

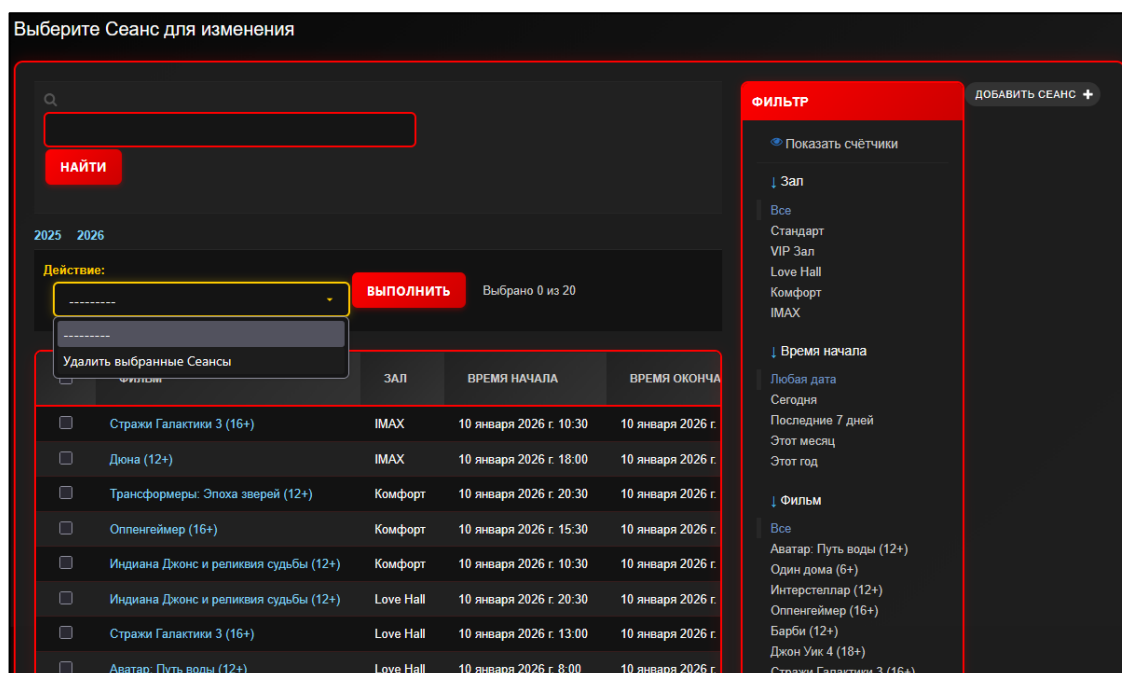


Рисунок 72 – Страница модели «Сеансы»

Администратор имеет полный доступ к взаимодействиям со статусами билетов. Страница модели «Статусы билетов» изображена на рисунке 73.

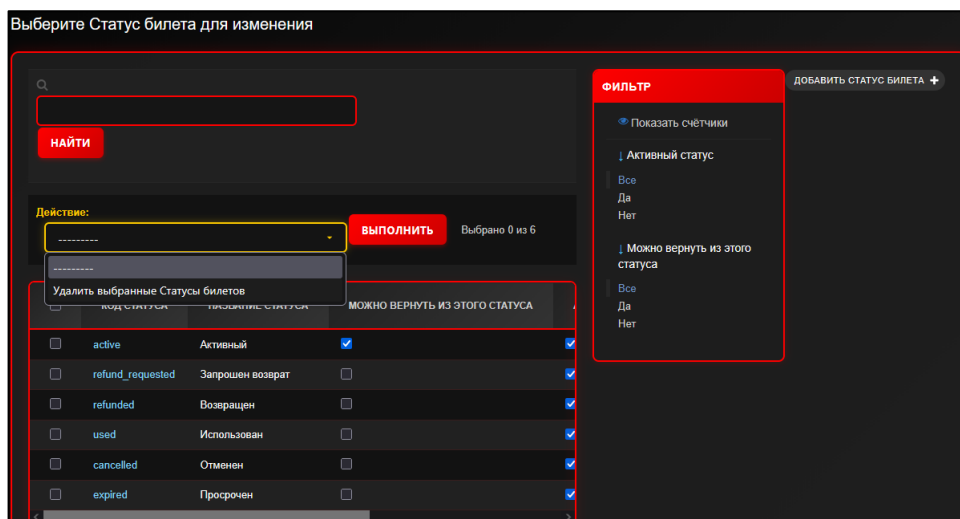


Рисунок 73 – Страница модели «Статусы билетов»

Администратор имеет полный доступ к взаимодействиям с фильмами. Страница модели «Фильмы» изображена на рисунке 74.

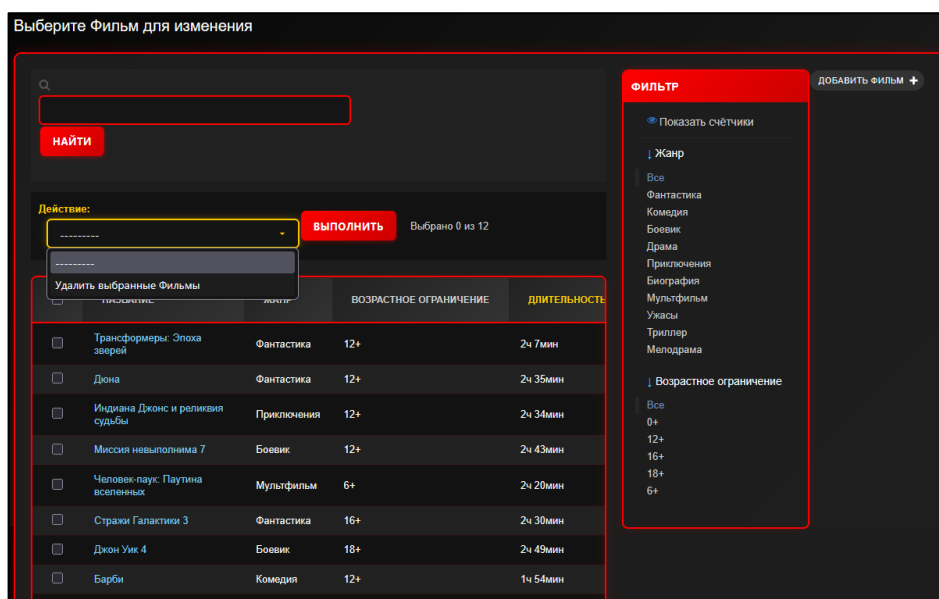


Рисунок 74 – Страница модели «Фильмы»

Документация охватывает полный цикл работы с системой управления кинотеатром: от установки и настройки до ежедневного использования всеми типами пользователей. Представленные инструкции обеспечивают корректное развертывание системы и позволяют эффективно использовать её функционал в повседневной работе.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана полнофункциональное веб–приложение «Кинотеатр» на базе фреймворка Django. Данный проект представляет собой законченное программное решение, направленное на автоматизацию ключевых бизнес-процессов современного кинотеатра, с фокусом на взаимодействие с конечным потребителем и внутреннее администрирование.

Итоги проекта в соответствии с поставленными целью и задачами:

1. Проведено предпроектное исследование и обоснован выбор технологического стека. В результате анализа современных инструментов был сформирован оптимальный стек технологий, полностью соответствующий требованиям проекта по производительности, безопасности, скорости разработки и удобству поддержки.
2. Выполнено комплексное проектирование архитектуры системы. На основе стандартов UML и IDEF0 была детально проработана структурная и функциональная схемы веб-приложения. Спроектирована и нормализована до третьей нормальной формы реляционная база данных, включающая 11 взаимосвязанных сущностей. Созданы детальные прототипы пользовательского интерфейса, утвердившие логику взаимодействия для всех ролей.
3. Разработано и формализовано техническое задание. ТЗ было составлено в соответствии с ГОСТ 34.602–2020.
4. Полностью реализована серверная и клиентская части системы.
5. Создан интуитивно понятный и отзывчивый пользовательский интерфейс. Интерфейс, реализованный с использованием HTML5 и CSS3, обеспечивает удобную навигацию для гостей, авторизованных пользователей и администраторов, полностью соответствуя разработанным прототипам.
6. Реализована и оптимизирована база данных. На основе спроектированной ER-модели с использованием Django ORM развернута



надежная база данных PostgreSQL. Модели данных реализуют все необходимые бизнес-правила.

7. Проведено тестирование и составлена полная документация. Функциональное тестирование с использованием Django Test Case подтвердило корректность работы всех ключевых сценариев. Разработано подробное руководство пользователя, руководство по установке и настоящая пояснительная записка.

Вывод о проделанной работе:

Курсовой проект «Веб-приложение "Кинотеатр"» успешно завершен. Разработанная система является работоспособным, надежным и безопасным продуктом, который полностью удовлетворяет изначально сформулированным требованиям. В процессе работы были получены и закреплены практические навыки полного цикла разработки программного обеспечения: от анализа предметной области и проектирования архитектуры до реализации, тестирования и документирования. Проект демонстрирует глубокое понимание принципов веб-разработки на Django, работы с базами данных, построения API, обеспечения информационной безопасности и создания удобного пользовательского опыта.

Перспективные направления развития проекта:

Несмотря на завершенность, система обладает значительным потенциалом для масштабирования и улучшения:

1. Внедрение полноценной платежной системы.
2. Развитие аналитического модуля. Добавление дашбордов с графиками, прогнозирования спроса и систем рекомендаций фильмов для пользователей на основе истории просмотров.
3. Расширение функционала для кинотеатральной сети. Добавление модуля управления несколькими филиалами, централизованной кассой, системой лояльности и печатью билетов на терминалах самообслуживания.
4. Повышение производительности и отказоустойчивости. Внедрение кеширования, использование асинхронных задач для отправки email и тяжелых

отчетов, балансировка нагрузки и контейнеризация для упрощения развертывания.

5. Расширение интеграций. Подключение сервисов email-рассылок, смс-уведомлений, а также агрегаторов киноафиш.

Таким образом, данный курсовой проект не только представляет собой законченное учебное задание, но и служит качественной основой для создания коммерческого продукта, востребованного на рынке услуг кинопоказа.

					КП.09.02.07.23.221.19.ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Python – Документация – URL: <https://habr.com/ru/hubs/python/articles/> (дата обращения: 05.09.2025) – Текст: электронный.
2. PythonWorld – Документация – URL: <https://pythonworld.ru/samouchitel-python> (дата обращения: 05.09.2025) – Текст: Электронный.
3. PythonTutorial – Документация – URL: <https://metanit.com/python/tutorial/> (дата обращения: 10.09.2025) – Текст: электронный.
4. Django – Документация – URL: <https://docs.djangoproject.com/en/5.2/> (дата обращения: 12.09.2025) – Текст: электронный.
5. PostgreSQL – Документация – URL: <https://www.postgresql.org/docs/> (дата обращения: 15.09.2025) – Текст: электронный.
6. W3Schools – Руководства – URL: <https://www.w3schools.com/> (дата обращения: 18.09.2025) – Текст: электронный.
7. MDN Web Docs – Документация – URL: <https://developer.mozilla.org/ru/> (дата обращения: 20.09.2025) – Текст: электронный.
8. Stack Overflow – Сообщество – URL: <https://stackoverflow.com/> (дата обращения: 22.09.2025) – Текст: электронный.
9. Draw.io – Инструмент – URL: <https://app.diagrams.net/> (дата обращения: 25.09.2025) – Текст: электронный.
10. ГОСТ 34.602-2020 – Стандарт – URL: <https://docs.cntd.ru/document/1200177846> (дата обращения: 28.09.2025) – Текст: электронный.
11. Django Girls – Учебник – URL: <https://tutorial.djangogirls.org/ru/> (дата обращения: 05.10.2025) – Текст: электронный.
12. PyCharm – Документация – URL: <https://www.jetbrains.com/help/pycharm/> (дата обращения: 10.10.2025) – Текст: электронный.

13. Bootstrap – Документация – URL: <https://getbootstrap.com/docs/5.3/> (дата обращения: 15.10.2025) – Текст: электронный.

14. Django REST framework – Документация – URL: <https://www.django-rest-framework.org/> (дата обращения: 20.10.2025) – Текст: электронный.

15. ReportLab – Документация – URL: <https://docs.reportlab.com/> (дата обращения: 25.10.2025) – Текст: электронный.

16. Pytest – Документация – URL: <https://docs.pytest.org/en/stable/> (дата обращения: 30.10.2025) – Текст: электронный.

17. Telegram Bot API – Документация – URL: <https://core.telegram.org/bots/api> (дата обращения: 05.11.2025) – Текст: электронный.

18. UML – Руководство – URL: <https://www.uml-diagrams.org/> (дата обращения: 10.11.2025) – Текст: электронный.

19. Реляционные базы данных – Статья – URL: <https://habr.com/ru/articles/725514/> (дата обращения: 15.11.2025) – Текст: электронный.

20. Git – Книга – URL: <https://git-scm.com/book/ru/v2> (дата обращения: 20.11.2025) – Текст: электронный.

21. PEP 8 – Руководство – URL: <https://peps.python.org/pep-0008/> (дата обращения: 25.11.2025) – Текст: электронный.

22. Two Scoops of Django – Книга – URL: <https://www.feldroy.com/books/two-scoops-of-django-3-x> (дата обращения: 01.12.2025) – Текст: электронный.

23. Django for Professionals – Книга – URL: <https://djangoforprofessionals.com/> (дата обращения: 05.12.2025) – Текст: электронный.

24. JetBrains Academy – Курс – URL: <https://www.jetbrains.com/academy/> (дата обращения: 10.12.2025) – Текст: электронный.

**Приложение А – Техническое задание**  
**Министерство образования Иркутской области**  
Государственное бюджетное профессиональное  
образовательное учреждение Иркутской области  
«Иркутский авиационный техникум»  
(ГБПОУИО «ИАТ»)

**Техническое задание**  
**ВЕБ-ПРИЛОЖЕНИЕ «КИНОТЕАТР»**

Руководитель: \_\_\_\_\_ (Е.С. Кубата)  
(подпись, дата)

Студент: \_\_\_\_\_ (В.М. Седых)  
(подпись, дата)

Иркутск, 2025

					КП.09.02.07.23.221.19.ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

## **1 Введение**

### **1.1 Общие сведения**

Документ представляет собой техническое задание на создание веб-приложения «Кинотеатр», предназначенное для автоматизации процессов бронирования и продажи билетов, управления расписанием сеансов и кинозалами, аналитики деятельности кинотеатра.

Вид программного продукта: веб-приложение с серверной архитектурой.

### **1.2 Цели и задачи**

Целью создания веб-приложения «Кинотеатр» является автоматизация процессов современного кинотеатра.

Задачи веб приложения включают:

- Предоставление клиентам возможности просмотра афиши, выбора сеансов и мест в зале, бронирования и оплаты билетов онлайн.
- Предоставление администраторам инструментов для управления фильмами, кинозалами, расписанием сеансов и мониторинга продаж.
- Устранение очередей, оптимизация работы администрации и повышение удовлетворенности клиентов.

## **2 Основания для разработки**

### **2.1 Нормативные документы**

Документ основывается на следующих нормативных документах:

- ГОСТ 34.602-2020 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы».

					КП.09.02.07.23.221.19.ПЗ	Лист 78
Изм.	Лист	№ докум.	Подпись	Дата		

- ГОСТ Р 56477-2015 "Проектирование и внедрение информационных систем. Общие требования".
- Методические указания к курсовому проекту по МДК 02.01.
- Приказ образовательного учреждения об утверждении тем курсовых проектов.

## **2.2 Проектные документы**

Проектные документы включают:

- Задание на курсовой проект.
- Методические указания по выполнению дипломного проекта.
- Пояснительная записка.
- Руководство пользователя.

## **3 Назначение системы**

### **3.1 Общее описание**

Веб-приложение «Кинотеатр» предназначено для использования клиентами кинотеатра и его административным персоналом. Приложение поддерживает процессы просмотра афиши, выбора и бронирования билетов, электронной оплаты, а также управления кинопоказами и анализа деятельности.

## **4 Требования к системе**

### **4.1 Функциональные требования**

1 Функционал роли «Гость»:

- Просмотр афиши с фильтрацией.
- Просмотр детальной информации о фильме: постер, описание, длительность, возрастной рейтинг.
- Просмотр расписания сеансов для каждого фильма.

					КП.09.02.07.23.221.19.ПЗ	Лист 79
Изм.	Лист	№ докум.	Подпись	Дата		

- Просмотр схемы кинозалов в режиме «только просмотр».
- Регистрация в системе с обязательной email-верификацией.
- Авторизация по email и паролю.
- Просмотр руководства пользователя.

## 2 Функционал роли «Авторизованный пользователь»:

Все функции роли «Гость», а также:

- Выбор одного или нескольких мест на интерактивной схеме зала.
- Бронирование и оформление заказа на билеты с группировкой.
- Получение электронного билета в PDF формате с QR-кодом для

скачивания.

- Просмотр истории своих покупок в личном кабинете.
- Скачивание билетов.
- Запрос возврата билетов с автоматической проверкой условий.
- Привязка и отвязка Telegram аккаунта с верификацией через код.
- Получение уведомлений о покупках через Telegram.
- Смена email с подтверждением через код верификации.
- Сброс пароля через email с временным кодом.

## 3 Функционал роли «Администратор»:

Все функции роли «Авторизованный пользователь», а также:

- Расширенная панель администратора Django.
- Управление каталогом фильмов: CRUD операции, загрузка постеров.
- Управление кинозалами: создание, редактирование, автоматическое

создание мест.

- Управление жанрами фильмов с проверкой уникальности и объединением дубликатов.
- Управление возрастными рейтингами фильмов.
- Формирование расписания сеансов.
- Управление билетами: просмотр, обработка возвратов, массовые операции.

					КП.09.02.07.23.221.19.ПЗ	Лист 80
Изм.	Лист	№ докум.	Подпись	Дата		



- Управление статусами билетов с настройкой возможности возврата.
- Просмотр отчётности.
- Экспорт отчетов в PDF формате.
- Управление бэкапами базы данных.
- Просмотр логов всех действий в системе
- Экспорт логов в JSON, PDF форматах.
- Просмотр и управление ожидающими регистрациями.
- Просмотр и управление запросами сброса пароля.
- Просмотр и управление запросами смены email.

## 4.2 Технические требования

### 4.2.1 Производительность:

- Количество одновременных пользователей: до 200 активных сессий;
- Количество запросов в секунду (RPS/QPS): 100 RPS;
- Среднее время отклика: не более 15 мс.

### 4.2.2 Надежность:

- Валидация пользовательского ввода на клиенте – все формы, поля, файлы и параметры должны проверяться перед отправкой запроса на сервер.
- Валидация пользовательского ввода на сервере – все входные данные, независимо от наличия клиентской проверки, проходят повторную строгую валидацию.
- Защита от SQL–инъекций за счёт применения параметризованных запросов или ORM с встроенной защитой.
- Логирование всех ключевых действий на клиентской части.

## 4.3 Эксплуатационные требования

					КП.09.02.07.23.221.19.ПЗ	Лист 81
Изм.	Лист	№ докум.	Подпись	Дата		

- 1 Среда функционирования:
  - Поддержка современных браузеров: Яндекс браузер, Chrome, Firefox, Edge, Safari.
- 2 Установка и развёртывание:
  - Разработка инструкции по установке и настройке.
- 3 Резервирование и восстановление:
  - Ручное резервное копирование базы данных.
  - Восстановления базы данных из резервной копии.
- 4 Интеграция:
  - Telegram API: интеграция с Telegram Bot API для уведомлений и управления.
  - Email SMTP: интеграция с почтовыми сервисами для отправки писем.
  - PDF Generation: интеграция с ReportLab для генерации PDF документов.
  - QR Code Generation: интеграция с qrcode библиотекой для создания QR-кодов.

#### 4.4 Требования к информационной безопасности

Веб-приложение должно обеспечивать защиту данных и предотвращение несанкционированного доступа в соответствии со следующими требованиями:

1. Аутентификация и авторизация:
  - Аутентификация по электронной почте и паролю.
  - Хранение паролей пользователей в зашифрованном виде с использованием встроенного механизма хэширования Django: PBKDF2 с SHA256.
  - Обязательная email-верификация при регистрации с временным кодом.
2. Защита от распространённых уязвимостей:
  - Защита от несанкционированного доступа к данным - проверка прав доступа к каждому ресурсу на серверной стороне.

## **5 Требования к техническому обеспечению**

### **5.1 Сервер**

- Количество процессоров: 1 шт.
- Количество ядер одного процессора: 2 ядра.
- Тактовая частота одного процессора: 2.5 ГГц.
- Оперативная память: 4 Гб.
- Пропускная способность сети: 100 Мбит/с.

### **5.2 Хранилище данных**

- Тип накопителей: SSD.
- Общий объем хранилища: 100 Гб.
- IOPS на чтение: 1000.
- IOPS на запись: 500.

### **5.3 Клиентские устройства**

#### **5.3.1 Минимальные требования для персонального компьютера (ноутбука)**

- Количество ядер процессора: 2 ядра.
- Тактовая частота процессора: 1.6 ГГц.
- Оперативная память: 4 Гб.
- Тип диска: HDD или SSD.
- Свободного места на диске: не менее 1 Гб.
- Разрешение дисплея: Full HD и выше.
- Пропускная способность сети: 10 Мбит/с.

### **5.4 Сетевые требования**

					КП.09.02.07.23.221.19.ПЗ	Лист 83
Изм.	Лист	№ докум.	Подпись	Дата		

- Доступ к сети Интернет со скоростью не менее 10 Мбит/с.
- Поддержка следующих сетевых протоколов: HTTP/1.1, HTTPS, WebSocket.
- Поддержка IPv4 (обязательно), IPv6 (при наличии).
- Возможность работы через защищённые порты 443 (HTTPS) и 80 (HTTP).

## **6 Требования к программному обеспечению**

### **6.1 Сервер**

- Операционная система Ubuntu Server 24.04 LTS.
- Сервер базы данных: PostgreSQL 16.
- Веб-сервер: Nginx 1.24.
- Интерпретатор: Python 3.12.
- Менеджер пакетов: pip 24.0.
- Система контроля версий: Git 2.45.

### **6.2 Персональный компьютер (ноутбук)**

1. Операционная система Windows 10 / 11 (64-bit), Linux Ubuntu 22.04 / 24.04 LTS, macOS 12 Monterey и выше.
2. Веб-браузеры (актуальные версии, не старше двух релизов): Яндекс.Браузер 24.9+, Google Chrome 128+, Mozilla Firefox 128+, Microsoft Edge 128+, Apple Safari 17+.
3. Необходимые компоненты браузера:
  - Поддержка HTML5, CSS3, ECMAScript 2015+ (ES6+).
  - Поддержка Web Storage API.
  - Поддержка WebSocket.
  - Включена поддержка JavaScript и cookies.

- Поддержка PDF просмотра в браузере.

## **7 Требования к тестированию**

### **7.1 Общие требования**

Тестирование веб-приложения должно охватывать все ключевые аспекты функциональности, производительности и безопасности. Этап тестирования должен включать:

- Документирование.
- Функциональное тестирование.

#### **7.1.1 Документирование**

Тестовые сценарии – охватывающие ключевые функциональные и нефункциональные требования к системе для проверки работы:

- в типовых и исключительных ситуациях;
- при вводе корректных и некорректных данных;
- при различных ролях пользователей (администратор, пользователь, модератор и т. д.).

Тестовая документация должна включать:

- план тестирования;
- перечень тестовых сценариев;
- отчёт о результатах тестирования и выявленных дефектах.

#### **7.1.2 Функциональное тестирование**

- Проверка корректности работы веб-приложения в соответствии с функциональными требованиями.
- Проверка работы пользовательских сценариев.
- Проверка корректности взаимодействия с внешними системами.

					КП.09.02.07.23.221.19.ПЗ	Лист 85
Изм.	Лист	№ докум.	Подпись	Дата		

– Проверка обработки ошибок и сообщений пользователю при некорректных действиях.

### **7.1.3 Нагрузочное тестирование**

– Проверка производительности для подтверждения выполнения требований к производительности.

## **8 Организационно-технические требования**

### **8.1 Этапы разработки**

В таблице 1 представлены сроки и этапы разработки веб-приложения «Кинотеатр».

Таблица 1 – Сроки и этапы разработки веб-приложения

№	Этап	Срок выполнения
1	Предпроектное исследование предметной области и выбор технологического стека	20.09.2025
2	Разработка технического задания и проектной документации	30.09.2025
3	Разработка веб-приложения	02.11.2025
4	Разработка тестирования	05.12.2025
5	Документирование веб-приложения	20.12.2025

## Приложение Б – Листинг Urls

```
import os

from django.urls import path, include
from . import views
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin

urlpatterns = [
    path("", views.home, name='home'),
    path('register/', views.register, name='register'),
    path('login/', views.user_login, name='login'),
    path('logout/', views.user_logout, name='logout'),
    path('movie/<int:movie_id>', views.movie_detail, name='movie_detail'),
    path('screening/<int:screening_id>', views.screening_detail, name='screening_detail'),
    path('screening/<int:screening_id>/partial/', views.screening_partial, name='screening_partial'),
    path('book/', views.book_tickets, name='book_tickets'),
    path('download-ticket/', views.download_ticket, name='download_ticket'),
    # Админка
    path('admin/', admin.site.urls),
    path('admin-dashboard/', views.admin_dashboard, name='admin_dashboard'),
    path('admin/movies/', views.movie_manage, name='movie_manage'),
    path('admin/movies/add/', views.movie_add, name='movie_add'),
    path('admin/movies/edit/<int:movie_id>', views.movie_edit, name='movie_edit'),
    path('admin/movies/delete/<int:movie_id>', views.movie_delete, name='movie_delete'),
    path('admin/hall/', views.hall_manage, name='hall_manage'),
    path('admin/hall/add/', views.hall_add, name='hall_add'),
    path('admin/hall/edit/<int:hall_id>', views.hall_edit, name='hall_edit'),
    path('admin/hall/delete/<int:hall_id>', views.hall_delete, name='hall_delete'),
    path('admin/screening/', views.screening_manage, name='screening_manage'),
    path('admin/screening/add/', views.screening_add, name='screening_add'),
    path('admin/screening/edit/<int:screening_id>', views.screening_edit, name='screening_edit'),
    path('admin/screening/delete/<int:screening_id>', views.screening_delete,
name='screening_delete'),
    path('admin/screening/calculate-price/', views.calculate_screening_price,
name='calculate_screening_price'),
    # Профиль
    path('profile/', views.profile, name='profile'),
    path('download-ticket/<int:ticket_id>', views.download_ticket_single,
name='download_ticket_single'),
    path('download-ticket-group/<str:group_id>', views.download_ticket_group,
name='download_ticket_group'),
    path('ticket/<int:ticket_id>/refund/', views.request_ticket_refund, name='request_ticket_refund'),
    path('ticket/<int:ticket_id>/cancel-refund/', views.cancel_refund_request,
name='cancel_refund_request'),
    # Почта
    path('verify-email/', views.verify_email, name='verify_email'),
    path('resend-verification-code/', views.resend_verification_code, name='resend_verification_code'),
```

					КП.09.02.07.23.221.19.ПЗ	Лист 87
Изм.	Лист	№ докум.	Подпись	Дата		

```

path('password-reset/', views.password_reset_request, name='password_reset_request'),
path('password-reset/code/', views.password_reset_code, name='password_reset_code'),
path('password-reset/confirm/', views.password_reset_confirm, name='password_reset_confirm'),
# Руководство пользователя
path('about/', views.about, name='about'),
]

```

```

if settings.DEBUG:

```

```

    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

```

    urlpatterns += static('/backups/', document_root=os.path.join(settings.BASE_DIR, 'backups'))

```

					КП.09.02.07.23.221.19.ПЗ	Лист
						88
Изм.	Лист	№ докум.	Подпись	Дата		