Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Современные технологии обработки экономической информации»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта старший преподаватель кафедры экономической информатики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Лыщик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Автоматизированная ИНФОРМАЦИОННАЯ система РАБОТЫ ЦИРКА»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  Жакин Павел Владимирович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

Жакин П.В. Автоматизированная информационная система работы цирка/ П.В. Жакин. – Минск: БГУИР, 2023.

Пояснительная записка 49 с., 33 рис., 11 источников, 3 приложения

*Ключевые слова*: система, автоматизированная, покупка билетов, цирк, разработка, информационная, модель, схема, моделирование, диаграмма, анализ, пользователь, статистика, менеджер, пользователь, администратор.

*Объект исследования*: информационная система, содержащая сведения о пользователях и выступлениях.

*Предмет исследования*: способы и пути повышения качества покупки билетов на выступления.

*Цель курсового проекта*: повышение уровня автоматизации информационной системы покупки билетов на выступления с ориентацией на качество и скорость обслуживания клиента.

*Методология проведения работы*: в процессе разработки автоматизированной системы использованы методы анализа, систематизации, классификации, обобщения данных, качественно-количественные методы обработки данных, функциональный анализ процессов, принципа построения баз данных, моделирование системы с помощью UML-диаграмм.

*Результаты работы:* проведен теоретический анализ, а также моделирование деятельности по осуществлению покупки билетов на цирковые выступления. Выполнено проектирование и конструирование автоматизированной системы работы цирка. Предоставлена инструкция по развертыванию автоматизированной системы, а также руководство пользователя.

*Область применения результатов*: услуги по продаже билетов на цирковые выступления онлайн.

**СОДЕРЖАНИЕ**

|  |  |  |  |
| --- | --- | --- | --- |
| Перечень условных обозначений, символов и терминов……………………. | | | 6 |
| Введение ………………………………………………………..…………... | | | 7 |
| 1 | Анализ и моделирование предметной области системы работы цирка | | 9 |
|  | 1.1 | Описание предметной области …………………………………........ | 9 |
|  | 1.2 | Разработка функциональной модели системы работы цирка…….... | 10 |
|  | 1.3 | Анализ требований к разрабатываемому программному средству.. | 15 |
|  | 1.4 | Разработка информационной модели автоматизированной системы работы цирка……………………….…........... | 15 |
|  | 1.5 | UML-модели представления автоматизированной системы работы цирка ……………………………........ | 18 |
| 2 | Проектирование и конструирование программного средства ………….. | | 28 |
|  | 2.1 | Постановка задачи ……...…………………………………………….. | 28 |
|  | 2.2 | Обоснование выбора компонентов и технологий для реализации автоматизированной системы работы цирка …………….................. | 29 |
|  | 2.3 | Архитектурные решения……………………………........................... | 30 |
|  | 2.4 | Проектирование пользовательского интерфейса………………........ | 31 |
| 3 | Руководство по развёртыванию и использованию автоматизированной системы работы цирка………………………………………….…….......... | | 32 |
|  | 3.1 Вход под ролью пользователя………………………………………... | | 34 |
|  | 3.2 Вход под ролью администратора…………………………………….. | | 37 |
|  | 3.3 Вход под ролью менеджера…………………………………………... | | 39 |
| Заключение ……………………………………………………………………... | | | 39 |
| Список использованных источников ……………………………………......... | | | 40 |
| Приложение А (обязательное) Листинг скрипта генерации базы данных… | | | 41 |
| Приложение Б В Листинг кода алгоритмов, реализующих бизнес-логику… | | | 47 |
| Приложение В Отчет о проверке на заимствования в системе «Антиплаиат»…………………………………………….................................. | | | 49 |

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ

СУБД – Система управления базами данных

AWS – *Amazon Web Services* – облачная платформа для развертывания приложений

DI – *Dependency Injection* – принцип получения объектами других объектов по требованию

HTTP – *Hypertext Transfer Protocol* – протокол передачи данных в сети

IDEF0 – *Integrated DEFinition 0* – методология функционального моделирования

IDEF1X – *Integrated DEFinition 1 Extended* – методология моделирования описания данных

Java EE – *Java Enterprise Edition* – набор спецификаций, описывающий архитектуру для средних и крупных предприятий

JPA – *Java Persistence API* – Java API, предоставляющий механизм взаимодействия с базой данных посредством технологии ORM

JTA – *Java Transaction API* – Java API, предоставляющий механизм управления транзакциями в Java EE

JWT – *JSON Web Token* – стандарт создания токенов доступа в формате JSON

MVC – *Model View Controller* – архитектурный шаблон построения приложений, при котором логика разделена на модель, представление и контроллер

ORM – *Object-Relational Mapping* – технология отображения сущностей базы данных на POJO-классы

POJO – *Plain Old Java Object –* объект Java, не наследующийи не реализующий какой-либо функционал

REST – *Representational State Transfer* – архитектурный стиль взаимодействия компонентов распределенного приложения в сети

SQL – *Structured Query Language –* декларативный язык взаимодействия с реляционной базой данных

UML – *Unified Modeling Language –* графический язык визуализации, специфицирования, конструирования систем

**ВВЕДЕНИЕ**

В настоящее время цирковые представления являются популярным способом развлечения современного человека. Как и любая другая сфера развлечений, со временем данная сфера развивается и становится удобнее для потребителя. Любой человек может изучить афишу цирка, заказать себе билет, или посмотреть интересующую его информацию о выступлении.

На данный момент существуют сайты-афиши, на которых можно поделиться информацией о выступлении и, соответственно, продавать билеты. Однако, данные сайты требуют постоянных вложений, и они не имеют необходимых инструментов для работы персонала и не собирают важные для компании данные.

В связи с увеличенным числом покупок онлайн, информационные технологии стали одной из важнейших частей любого бизнеса.

Использование современных разработок позволяет клиенту работать в системе заказа билетов с любого компьютера, имеющего доступ к интернету и оснащенного стандартным программным обеспечением.

Значимость создания темы автоматизированного учета обусловлена тем, что на данный момент менеджеры могут сталкиваться с огромными объемами информации. Системы обязаны представлять собой мощные средства, способные обрабатывать гигантские потоки данных высокой структурной сложности за минимум затраченного времени, при этом предоставляя все данные в понятном виде пользователю.

Целью данного проекта является сокращение временных затрат и повышение удобства проведения продаж продукции посредством автоматизации процесса продажи билетов.

В ходе решения поставленной цели должны быть решены следующие задачи:

* изучить предметную область;
* разработать и описать руководство по развертыванию автоматизированной системы
* разработать программное обеспечение автоматизированной системы работы цирка
* составить блок-схемы процесса заказа билетов на выступление;
* спроектировать и разработан удобный интерфейс приложения;

Объектом исследования данного проекта является автоматизированная система, содержащая сведения о пользователях и выступлениях.

Предметом исследования являются способы понижения временных затрат на обслуживание клиентов и автоматизация работы цирка.

Языком разработки приложения является Java, среда разработки Intellij Idea. База данных содержит необходимые данные о пользователях, а также данные о покупках. В качестве базы данных в курсовом проекте выступает MySQL

Обращение к базе данных производит сервер. В данном проекте в качестве сервера используется Tomcat.

С целью облегчения проектирования программного комплекса будут разработаны стандартизированные способы моделирования и разработки систем, такие как IDEF0, UML-диаграммы и блок-схемы.

|  |  |  |
| --- | --- | --- |
| **1** | **АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ СИСТЕМЫ РАБОТЫ ЦИРКА** | |
| **1.1** | | **Описание предметной области** |

Автоматизированная система – это система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая автоматизированную технологию выполнения установленных функций.

Автоматизированная система состоит из взаимосвязанной совокупности подразделений организации и комплекса средств автоматизации деятельности, и реализует автоматизированные функции по отдельным видам деятельности. Разновидностью АС являются информационные системы, основной целью которых является хранение, обеспечение эффективного поиска и передачи информации по соответствующим запросам.

Информационная система – взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

При этом автоматизированные информационные системы являются областью информатизации, механизмом и технологией, эффективным средством обработки, хранения, поиска и представления информации потребителю. Автоматизированные информационные системы представляют совокупность функциональных подсистем сбора, ввода, обработки, хранения, поиска и распространения информации. Процессы сбора и ввода данных необязательны, поскольку вся необходимая и достаточная для функционирования автоматизированной информационной системы информация может уже находиться в составе её базы данных.

Предметная область автоматизированной информационной системы цирка – это совокупность программных и аппаратных средств, предназначенных для управления и автоматизации деятельности цирка. Она позволяет цирку эффективно управлять многими процессами, такими как продажа билетов, управление репертуаром, учет финансовых средств и многое другое.

Задачи АИС цирка:

* управление продажей билетов – включая продажу билетов через интернет, по телефону и в кассах цирка;
* учет финансовых средств – включая учет доходов и расходов, контроль за бюджетом цирка и управление финансовыми ресурсами;
* увеличение скорости выполнения процессов обработки и преобразования информации;
* повышение качества и эффективности обслуживания клиентов.

Основные преимущества АИС цирка:

* увеличение эффективности работы цирка;
* сокращение затрат на обработку и хранение информации;
* повышение качества обслуживания посетителей цирка;
* увеличение доходов цирка.

Однако, как и у любой системы, у автоматизированной информационной системы есть и недостатки.

Недостатки АИС цирка:

* высокая стоимость внедрения и обслуживания;
* сложность в использовании для неподготовленных пользователей;
* риск потери информации при сбоях системы.

История автоматизированных информационных систем в цирковом искусстве начинается с конца 20 века. В то время цирки начали осознавать необходимость использования компьютерных технологий для управления своими мероприятиями. Одним из первых успешных цирковых проектов АИС стал проект «CircusPro», разработанный в США в 1990-х годах. Система позволяет управлять продажей билетов, контролировать наличие мест и стоимость билетов, управлять репертуаром и финансами цирка. Данная система позволяет управлять продажей билетов, контролировать наличие мест и стоимость билетов, управлять репертуаром и финансами цирка.

|  |  |
| --- | --- |
| **1.2** | **Разработка функциональной модели системы работы цирка** |

Основной процесс предметной области данного проекта – автоматизирование работы цирка­­­­­­­­­­­­­­­­­­­­ ­. Данное действие включает в себя большое количество особенностей, а также требует соблюдение множества правил и нормативов. Исходя из этого, существует необходимость в изложении основных пунктов и действий каждой из сторон данной операции.­­­­­­­­

При определении бизнес-функции системы, необходимо иметь в виду цель моделирования и точку зрения на данную модель.

Создание функциональной модели производилось при помощи программы Erwin. Данный продукт предназначен для поддержки процесса создания информационных систем. Использовалась методология функционального моделирования IDEF0.

IDEF0 – метод функционального моделирования, а также графическая нотация, которая используется для описания и формализации бизнес-процессов. Особенность IDEF0 заключается в том, что эта методология ориентирована на соподчиненность объектов.

Базовые требования IDEF0:

* главный элемент – в верхнем левом углу;
* каждый элемент должен иметь входящие и исходящие стрелки. Причем входящие стрелки находятся слева, справа – исходящие;
* сверху располагаются управляющие элементы, снизу – механизмы;
* при расположении нескольких блоков на одном листе или экране последующие размещаются справа внизу от предыдущего;
* схемы следует создавать так, чтобы стрелки пересекались минимальное количество раз.

Также стоит рассмотреть и типы стрелок и их обозначения:

* входящими ставятся задачи;
* исходящими выводят результат деятельности;
* управляющие (стрелки сверху вниз) – это механизмы управления;
* механизмы (стрелки снизу вверх) – то, что преобразовывает вход в выход: сотрудники или техника. Считается, что за один цикл процесса не происходит изменения механизма.

Входными данными представленной модели будут являться данные пользователя, которые в дальнейшем будут использованы для авторизации, и информация о выступлениях (наименование, время начала, дата, постер, фотографии).

Контроль над покупкой билетов и управлением пользователями производит администратор или менеджер. Для покупки мест на выступление нужен пользователь, который вводит данные в специальные поля. Конечный результат – приобретение билета на выступление.

Диаграмма верхнего уровня модели IDEF0 по приобретению билета на выступление представлена на рисунке 1.1.

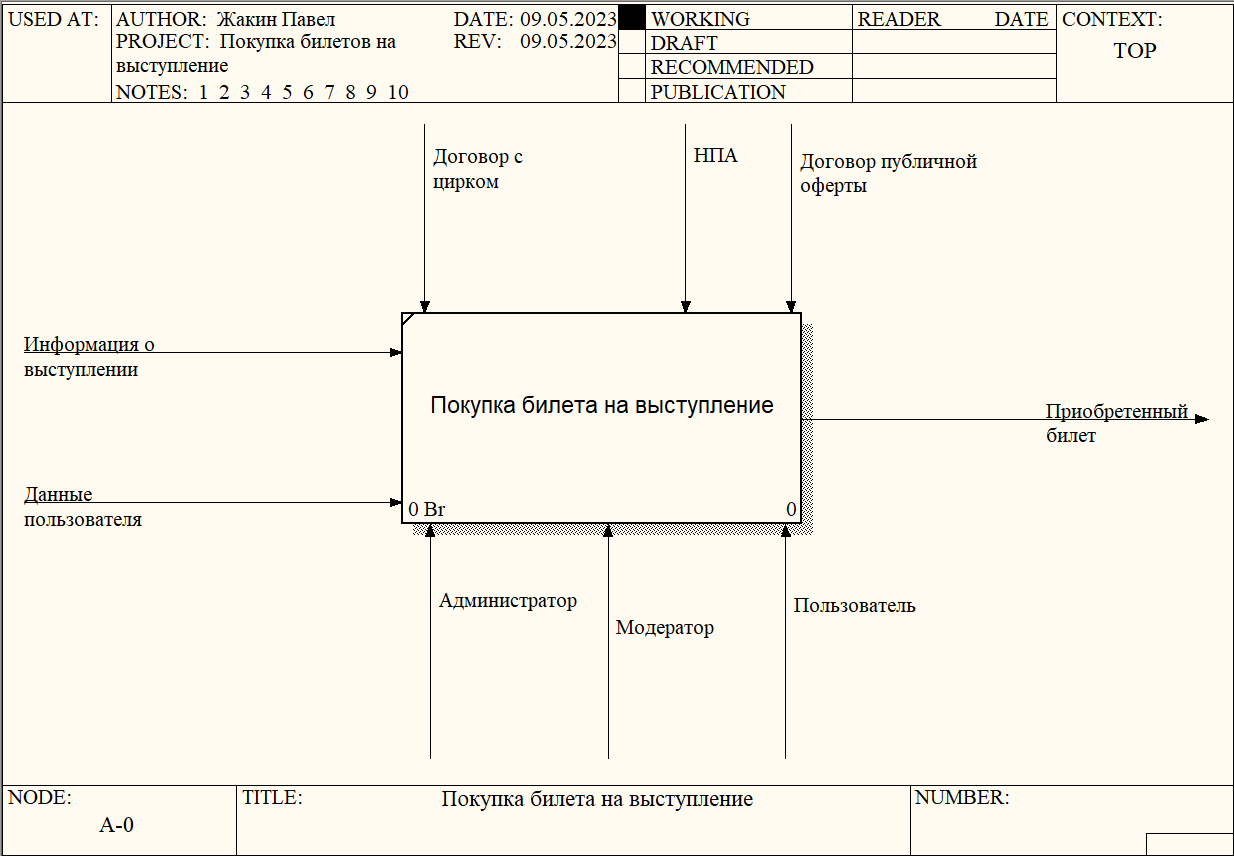


Рисунок 1.1 - Диаграмма IDEF0

Далее представлена декомпозиция контекстной диаграммы, состоящая из четырех блоков (рисунок 1.2):

* Регистрация в системе;
* Авторизация пользователя;
* Выбор выступления для посещения;
* Приобретение билета на выступление.

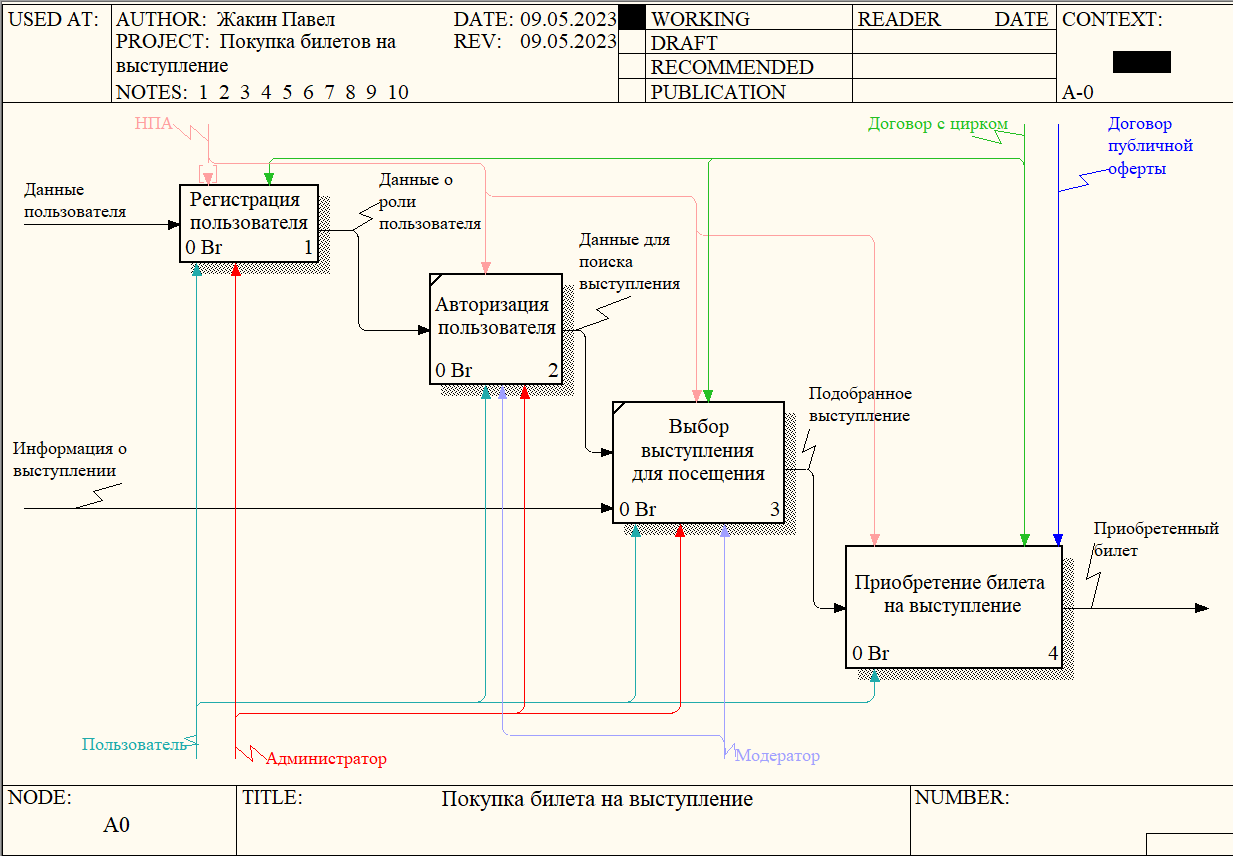


Рисунок 1.2 - Декомпозиция диаграммы IDEF0

Этап «Регистрация в системе» разбит на 3 функциональных блока:

* ввод данных в поля регистрации;
* согласие с договором цирка;
* завершение регистрации.

По окончании данного блока пользователю будут присвоены роли, назначенные администратором.

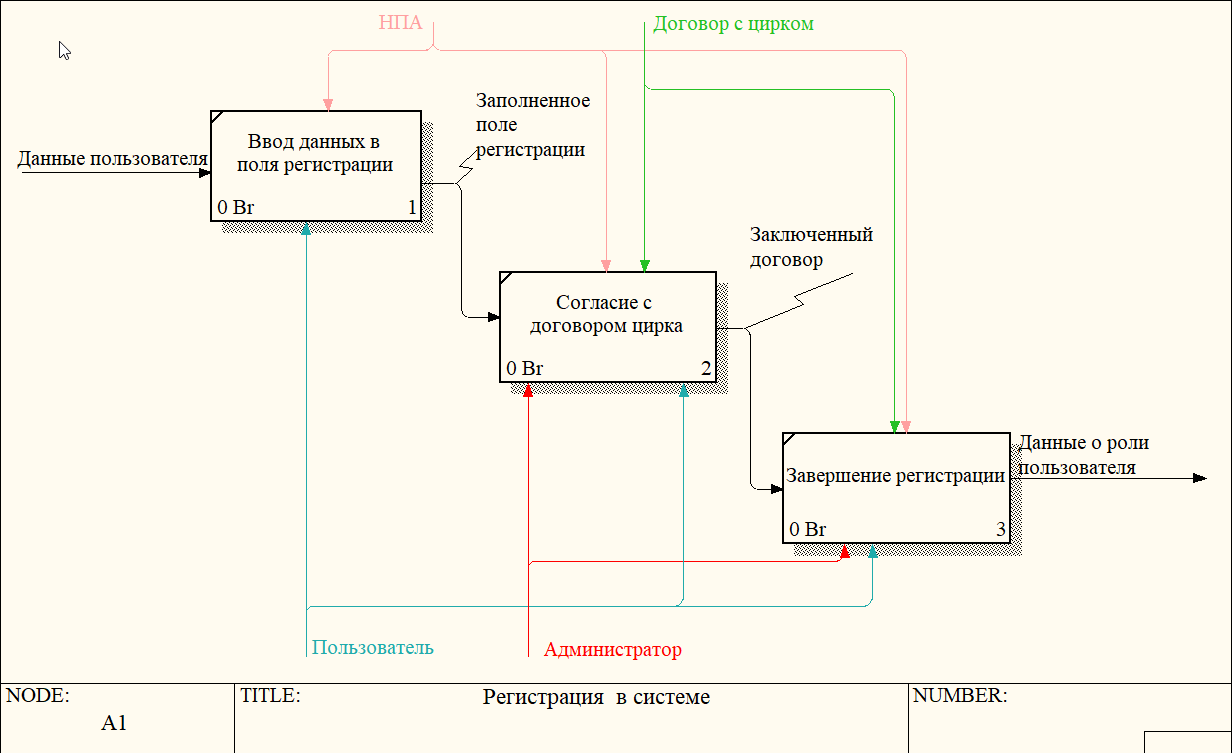


Рисунок 1.3 - Декомпозиция блока «Регистрация в системе»

Этап «Приобретение билета на выступление» представлен тремя блоками (рисунок 1.4):

* переход на страницу выбора выступления;
* утверждение количества билетов и других данных;
* совершение оплаты.

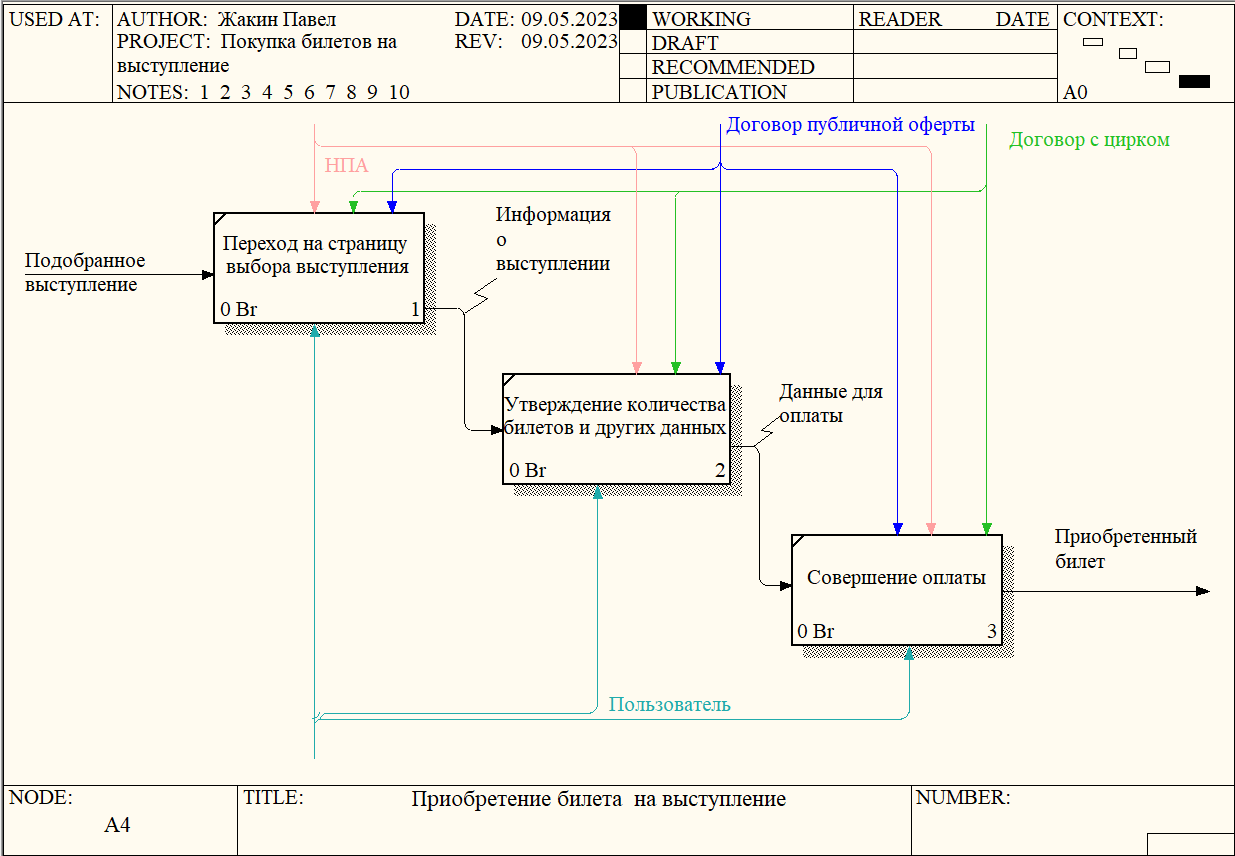


Рисунок 1.4 - Декомпозиция блока «Приобретение билета на выступление»

Исходя из диаграмм IDEF0, можно сделать вывод, что сам процесс приобретения билетов на выступление является сложным, и требует комплексного подхода от администрации.

|  |  |
| --- | --- |
| **1.3** | **Анализ требований к разрабатываемому программному средству** |

Базовые требования к разрабатываемому web-приложению включают в себя понятный интерфейс, который включает в себя инструменты управления другими пользователями со стороны администратора, настройки афиши, и присвоения ролей. Со стороны же пользователя, приложение должно представлять удобный интерфейс в совокупности с высокой скоростью работы. Пользователь имеет возможность просматривать представления и покупать билеты, просматривать основную информацию о цирке и следить за своими заказами в личном кабинете.

Также был разработан мини-форум под каждым выступлением, чтобы пользователи могли делиться своими впечатлениями. Администратор имеет возможность составить отчетность о доходах цирка на основе приобретенных у компании билетов на выступления.

Для работы с приложением используется база данных, реализованная при помощи программного обеспечения MySQL, в которой настроена связь между клиентом и сервером.

|  |  |
| --- | --- |
| **1.4** | **Разработка информационной модели автоматизированной системы работы цирка** |

UML (Unified Modeling Language) - это язык моделирования, который используется для описания различных аспектов программных систем. Одним из наиболее важных аспектов является информационная модель, которая описывает классы объектов, их атрибуты и методы, а также связи между ними.

Информационная модель в UML представляется в виде диаграммы классов. Для ее разработки необходимо провести анализ бизнес-процессов и определить требования к системе. Затем необходимо определить классы объектов, их атрибуты и методы, а также связи между объектами. Для этого используются различные методы моделирования, такие как диаграммы классов и диаграммы объектов

Логическая модель представляет собой модель базы данных, которая не привязана к конкретной СУБД. В ней выделяют основные объекты БД и определяют связи между этими объектами. Иногда определятся типы данных отдельных объектов. Данная модель построена методом сущность-связь.

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания данных конкретной СУБД. Физическая модель базы данных содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей и т. п.

Приложением для составления информационной модели базы данных было выбрано приложение PhpMyAdmin. Выбор обусловлен простотой использования и универсальностью, легкой расширяемостью.

Используемая база данных – MySQL. Выбор обусловлен высокой производительностью, гибкостью и надежностью, широким функционалом. База данных хранит в себе данные о пользователях, заказах, сессиях, комментариях на форумах и доходе компании.

На рисунке 1.5 приведена информационная модель базы данных с таблицами и связи между ними.

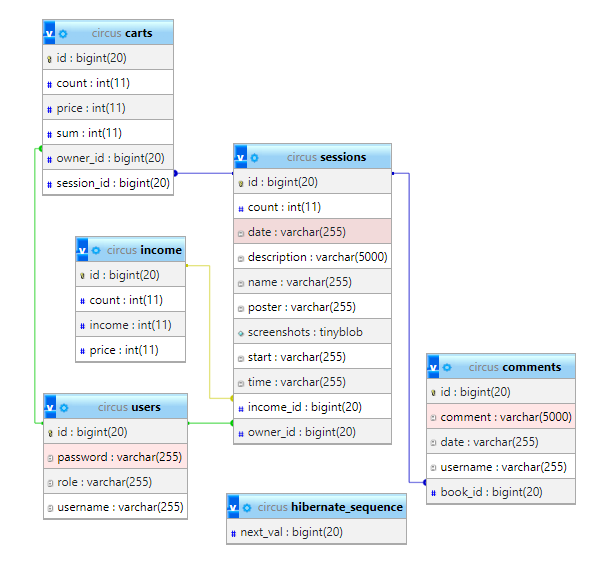


Рисунок 1.5 - Информационная модель базы данных

Ниже описаны основные сущности и их атрибуты.

Сущность «carts» хранит информацию о заказах пользователей.

Атрибуты сущности «carts»:

* id (тип данных bigint) – первичный ключ;
* count (тип данных int) – количество билетов;
* price (тип данных int) – цена одного билета;
* sum (тип данных int) – общая цена заказа;
* owner\_id (тип данных bigint) – идентификатор клиента;
* session\_id (тип данных bigint) – идентификатор выступления.

Сущность «users» хранит информацию о данных пользователей.

Атрибуты сущности «users»:

* id (тип данных bigint) – первичный ключ;
* password (тип данных varchar) – пароль пользователя;
* role (тип данных varchar) – роль пользователя, влияет на права;
* username (тип данных varchar) – имя пользователя.

Сущность «sessions» хранит информацию о цирковых выступлениях.

Атрибуты сущности «sessions»:

* id (тип данных bigint) – первичный ключ;
* count (тип данных int) – общее количество билетов;
* date (тип данных varchar) – дата выступления;
* description (тип данных varchar) – описание выступления;
* name (тип данных varchar) – название выступления;
* poster (тип данных varchar) – постер выступления.
* screenshots (тип данных tinyblob) – фотографии на странице;
* start (тип данных varchar) – год начала программы;
* time (тип данных varchar) – время начала;
* income\_id (тип данных bigint) – идентификатор дохода;
* owner\_id (тип данных bigint) – идентификатор создателя.

Сущность «income» хранит информацию о заказах пользователей.

Атрибуты сущности «income»:

* id (тип данных bigint) – первичный ключ;
* count (тип данных int) – общее количество приобретенных билетов;
* income (тип данных int) – общая прибыль от продаж билетов;
* price (тип данных int) – цена одного билета.

Сущность «comments» хранит информацию о комментариях под выступлением.

Атрибуты сущности «comments»:

* id (тип данных bigint) – первичный ключ;
* comment (тип данных varchar) – текст комментария;
* date (тип данных varchar) – дата комментария;
* username (тип данных varchar) – имя пользователя;
* book\_id (тип данных bigint) – идентификатор выступления;

Доступ к базе данных производится с сервера, что обеспечивает высокую безопасность данных пользователей.

Информационная модель была построена с целью наглядно проследить все связи между сущностями и их атрибутами.

|  |  |
| --- | --- |
| **1.5** | **UML-модели представления автоматизированной системы работы цирка** |

UML (Unified Modeling Language) – это язык моделирования, который используется для описания и проектирования систем. UML был разработан в 1997 году и стал стандартом OMG (Object Management Group). UML представляет собой набор графических нотаций и семантических правил, которые позволяют создавать диаграммы, описывающие различные аспекты системы.

Основными элементами UML являются диаграммы, которые классифицируются по типу. Существует девять типов диаграмм UML, каждая из которых описывает определенный аспект системы. Некоторые из наиболее распространенных диаграмм UML включают в себя диаграммы классов, диаграммы последовательностей, диаграммы состояний и диаграммы деятельности.

Преимущества использования UML заключаются в том, что он позволяет создавать единый язык для коммуникации между различными участниками проекта. Это позволяет устранить возможные несоответствия в понимании требований и спецификаций проекта. Кроме того, UML обеспечивает возможность создания наглядных диаграмм, которые легко читаются и понимаются всеми участниками проекта.

Словарь UML включает в себя различные термины, такие как классы, объекты, атрибуты, методы, связи и т.д. Классы представляют собой шаблоны для создания объектов, которые имеют общие свойства и методы. Атрибуты определяют состояние объекта, а методы определяют его поведение. Связи между объектами могут быть однонаправленными или двунаправленными.

Диаграмма классов в UML описывает структуру классов и их взаимосвязи в системе. Классы являются основными элементами объектно-ориентированного программирования и используются для описания поведения и свойств объектов в системе. Диаграмма классов содержит классы, интерфейсы, абстрактные классы, перечисления и другие узлы, а также связи между классами, такие как наследование, агрегация, ассоциация и зависимость. На рисунках 1.6-1.9 представлены диаграммы классов автоматизированной системы работы цирка.

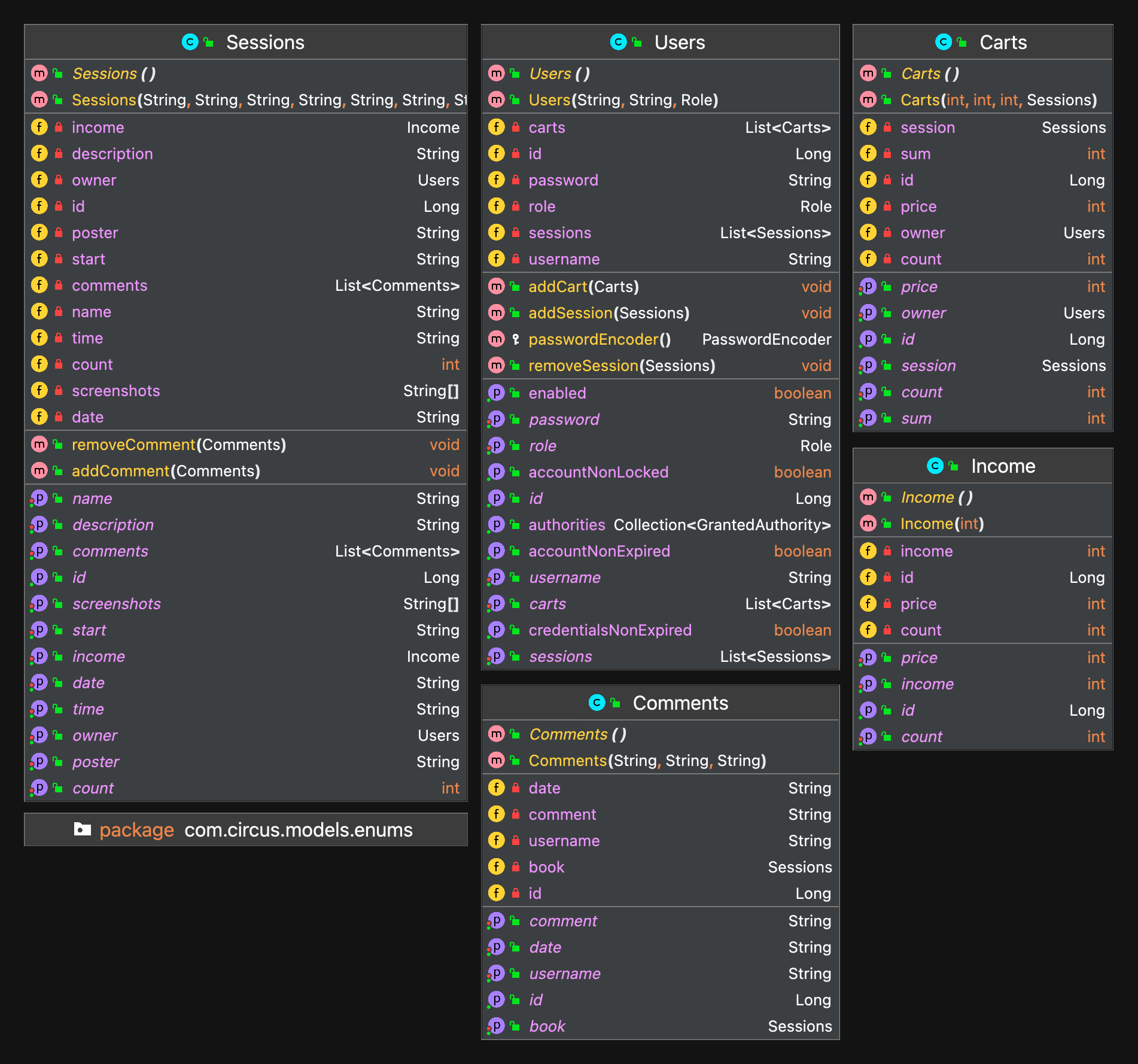


Рисунок 1.6 – Диаграмма классов пакета models

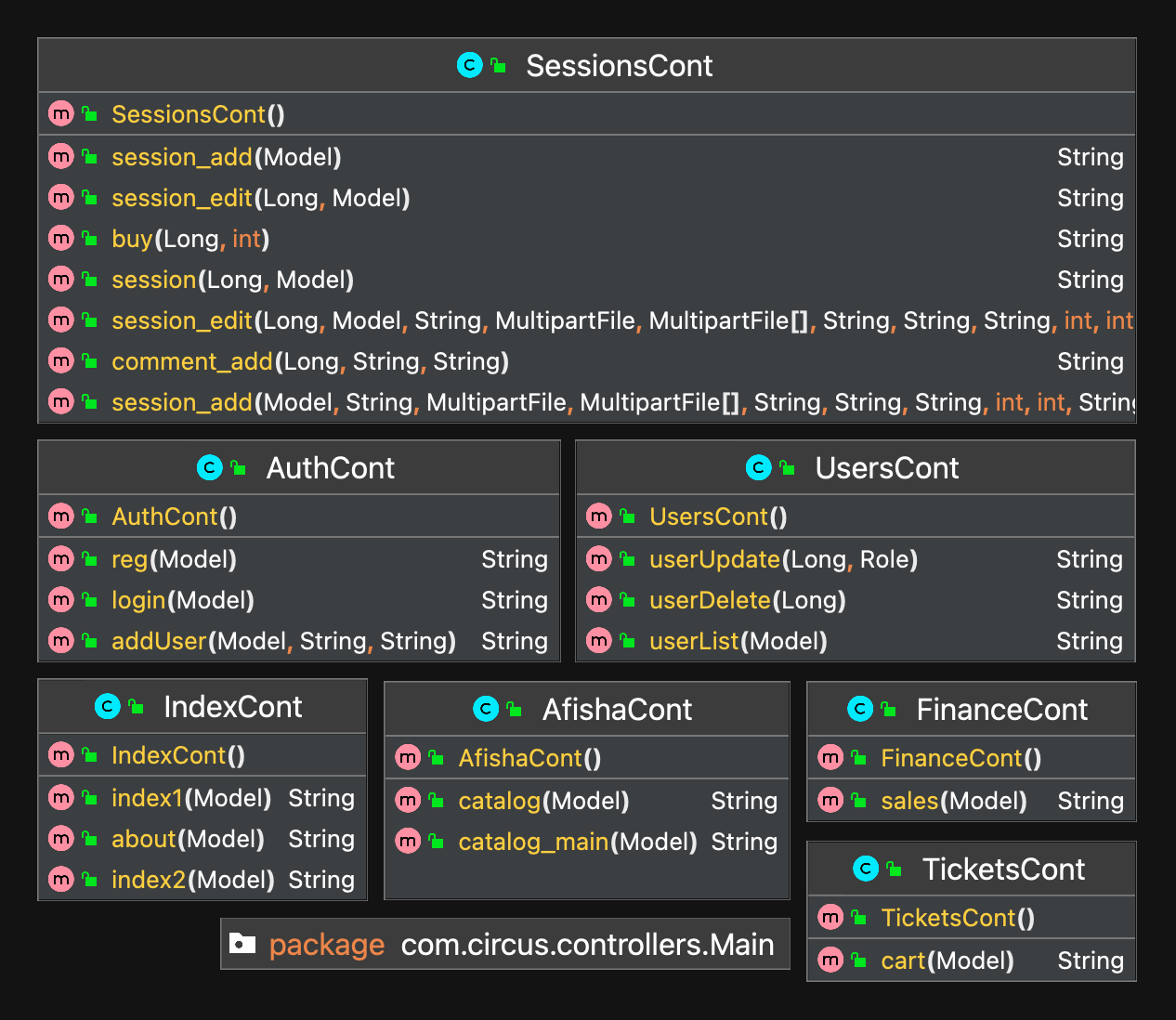


Рисунок 1.7 – Диаграмма классов пакета controllers

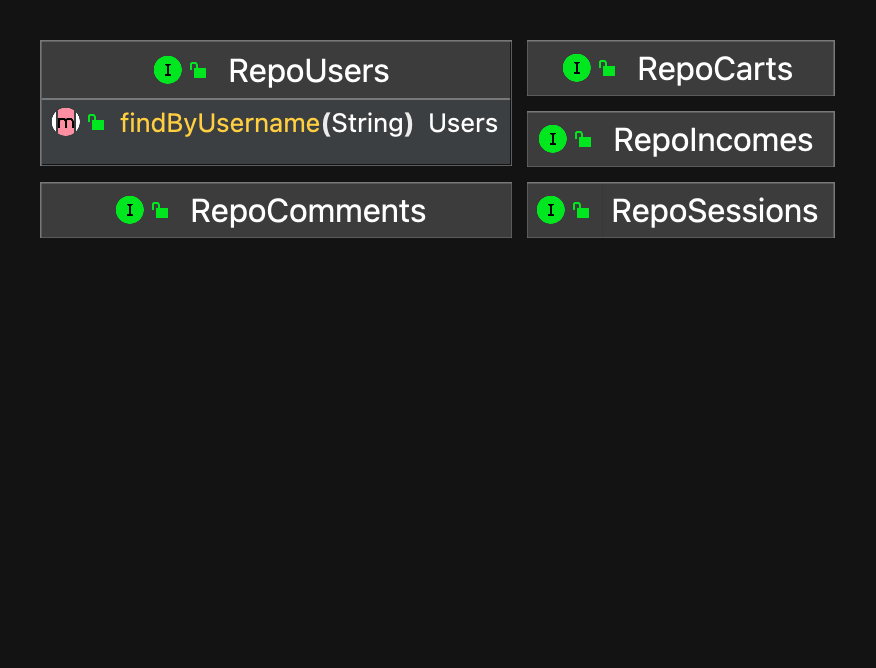


Рисунок 1.8 – Диаграмма классов пакета repo

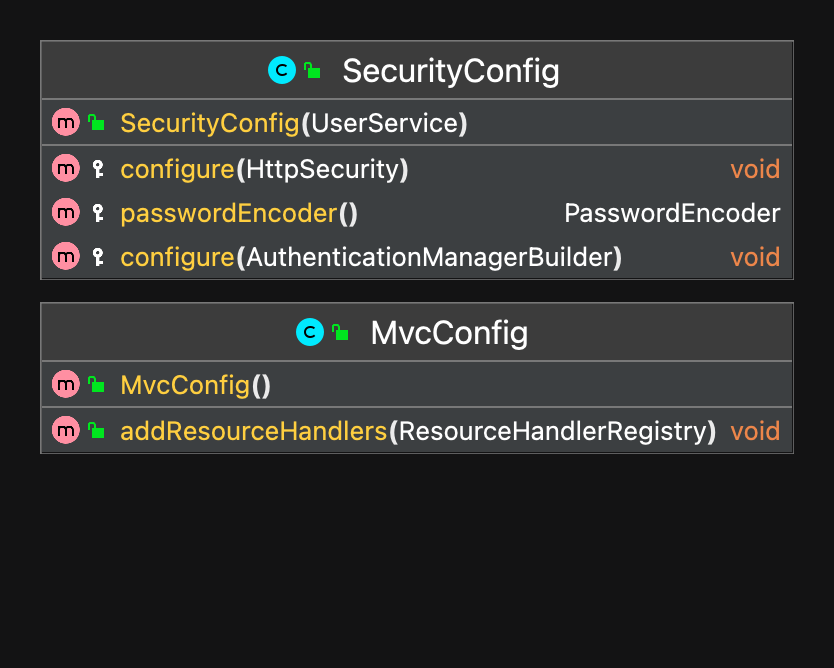


Рисунок 1.9 – Диаграмма классов пакета config

Диаграмма вариантов использования UML используется для описания функциональности системы или приложения с точки зрения пользователя. Она позволяет определить, какие действия может выполнить пользователь, какие варианты использования доступны и как они связаны друг с другом.

Диаграмма вариантов использования помогает разработчикам лучше понять требования пользователей и на основе этого определить функциональность системы. Она также может использоваться для проверки соответствия системы требованиям заказчика и для улучшения процесса разработки.

Кроме того, диаграмма вариантов использования может быть использована для обучения пользователей и для создания документации по проекту. Она помогает представить функциональность системы в понятной форме, что упрощает взаимодействие между разработчиками и пользователями.

Диаграмма вариантов использования автоматизированной системы работы цирка представлена на рисунке 1.10.



Рисунок 1.10 – Диаграмма вариантов использования автоматизированной системы работы цирка

Веб-приложение разделяет пользователей на три роли: администратор, менеджер и пользователь. Администратор обладает всеми возможностями, менеджер обладает всеми возможностями кроме возможности управления пользователями и данными аккаунтов.

Для просмотра афиши и покупки билетов пользователю нужно зарегистрироваться и авторизироваться. Если логин и пароль уникальны и верны, то он сможет зайти в раздел «афиша» и приобрести билет на выступлением. После покупки билета, пользователь сможет просмотреть приобретенные билеты в личном кабинете.

При проектировании сложной системы принято делить ее на части, каждую из которых затем рассматривать отдельно. Таким образом, при объектной декомпозиции система разбивается на объекты или компоненты, которые взаимодействуют друг с другом, обмениваясь сообщениями. Сообщения описывают или представляют собой некоторые события. Получение объектом сообщения активизирует его и побуждает выполнять предписанные его программным кодом действия.

При данном подходе система становится событийно управляемой, поэтому разработчикам зачастую важно знать, как должен реагировать тот или иной объект на определенные события. Инициаторами событий могут быть как объекты самой Системы, так и её внешнее окружение.

Описать поведение отдельно взятого объекта помогает диаграмма состояний.

Также зачастую диаграмма состояний используется аналитиками для описания последовательности переходов объекта из одного состояния в другое.

Основными элементами диаграммы состояний являются «Состояние» и «Переход». Диаграмма состояний имеет схожую семантику с диаграммой деятельности, только деятельность здесь заменена состоянием, переходы символизируют действия. Таким образом, если для диаграммы деятельности отличие между понятиями «Деятельность» и «Действие» заключается в возможности дальнейшей декомпозиции, то на диаграмме состояний деятельность символизирует состояние, в котором объект находится продолжительное количество времени, в то время как действие моментально. На рисунке 1.11 изображены основные элементы диаграммы состояний.

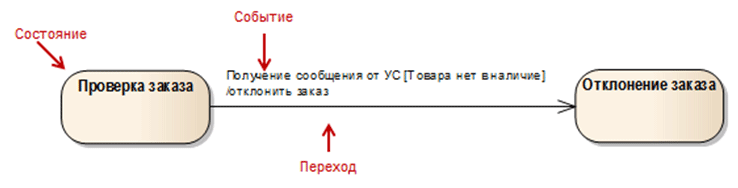


Рисунок 1.11 – Основные элементы диаграммы состояний

Диаграмма состояний показывает все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния.

Чтобы описать основную функциональность системы и продемонстрировать взаимоотношения и зависимости между группами вариантов использования была создана диаграмма вариантов использования, представленная на рисунке 1.12.

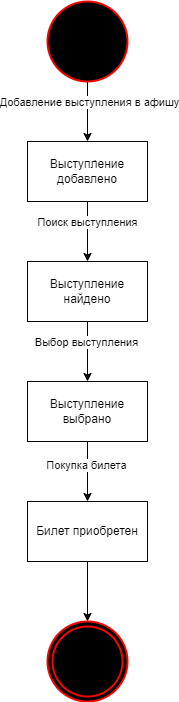


Рисунок 1.12 – Диаграмма состояния процесса покупки билета на выступление

Диаграмма развертывания UML используется для описания физической архитектуры системы или приложения. Она показывает, как компоненты системы распределены на устройства и как они взаимодействуют между собой. Данная диаграмма помогает разработчикам понять, как система будет работать в реальной среде и какие устройства и сетевые ресурсы потребуются для работы системы. Также, ее можно использоваться для обучения пользователей.

Устройство(телефон, компьютер, планшет) используется для взаимодействия с сервером при помощи браузера посредством протокола HTTP.

Сервер данного веб-приложения – Tomcat. Причина выбора – полная совместимость с Java, простота установки и надежность. Сервер взаимодействует с базой данных с помощью JDBC. Выбор JDBC обусловлен легкостью разработки и простотой подключения.

База данных приложения – MySQL, причины выбора которого уже были описаны. Диаграмма развертывания представлена на рисунке 1.13.

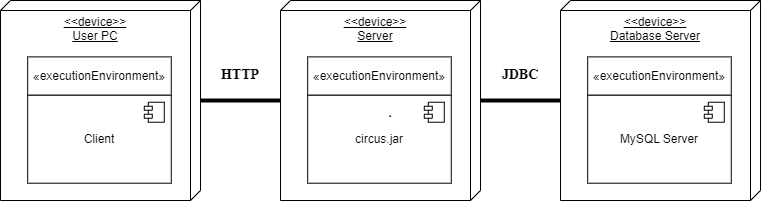


Рисунок 1.13 – Диаграмма развертывания

Диаграмма компонентов – это диаграмма, которая показывает компоненты системы и их зависимости. Компоненты могут быть библиотеками, модулями, приложениями и так далее. Диаграмма компонентов помогает понять структуру системы и ее архитектуру. В диаграмме компонентов показываются компоненты, интерфейсы, зависимости между компонентами, а также связи с другими элементами системы, такими как базы данных и внешние системы. Диаграмма компонентов UML может использоваться для проектирования новых систем или для анализа уже существующих систем. Диаграмма компонентов отображена на рисунке 1.14.

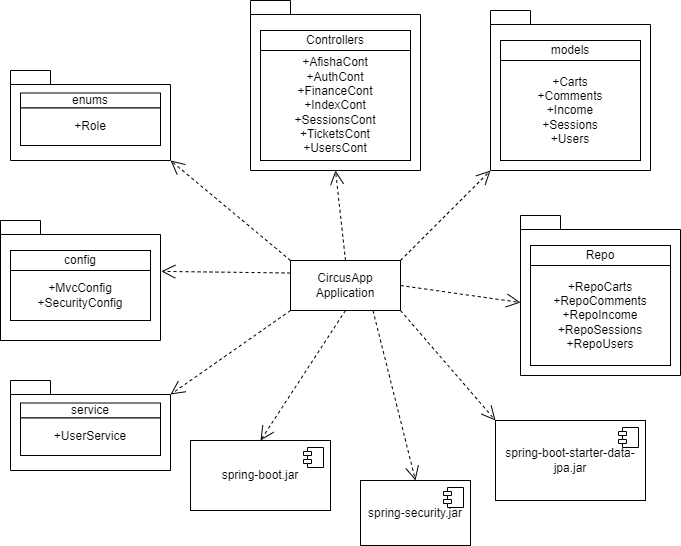


Рисунок 1.14 – Диаграмма компонентов

Диаграмма последовательности – UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие действующих лиц информационной системы в рамках прецедента.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции прямоугольники на пунктирной «линии жизни» –фокусы контроля, и стрелки, показывающие обмен сигналами или сообщениями между объектами. Диаграмма последовательности процесса авторизации пользователя изображена на рисунке 1.15.

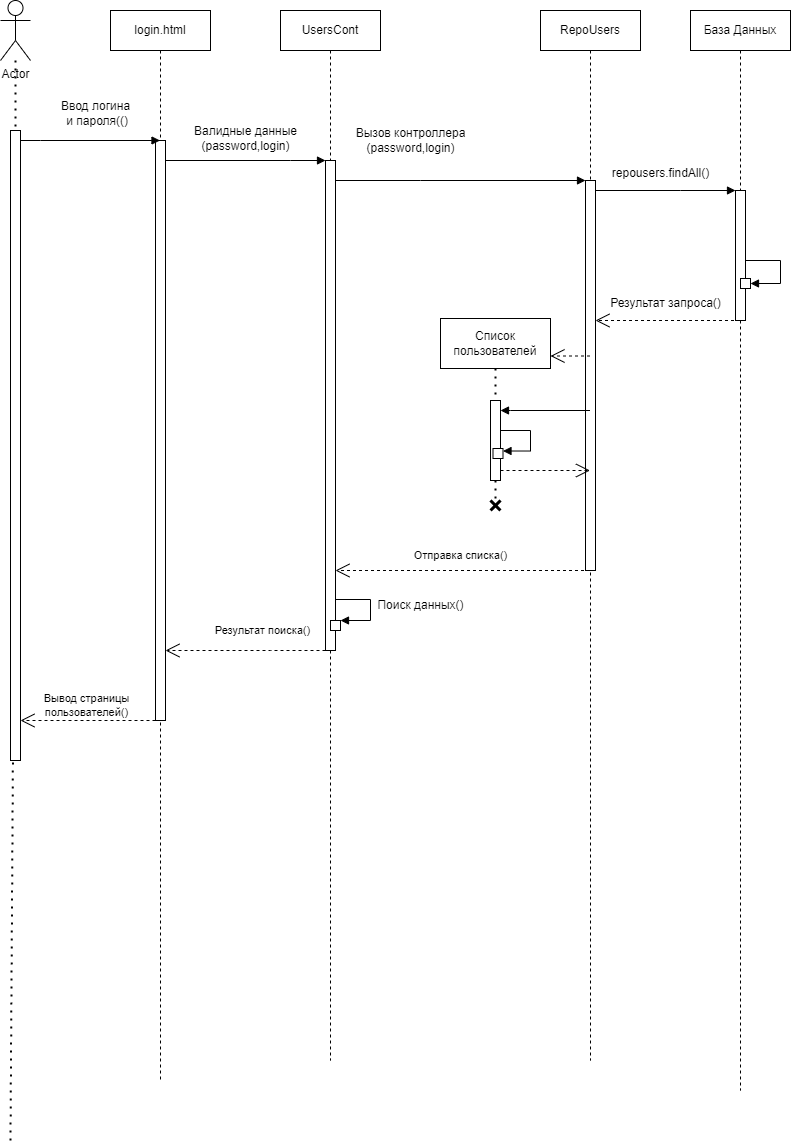


Рисунок 1.15 – Диаграмма последовательности авторизации пользователя

На данной диаграмме отображено, что пользователь выполняет вход в программу, вводит данные (логин и пароль) в форме авторизации. В случае корректно введенных данных сервер посылает логин и пароль контроллеру, который посылает запрос в СУБД на получение списка пользователей, после чего данные проверяются и происходит переход на следующую страницу.

|  |  |  |
| --- | --- | --- |
| **2** | **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА** | |
| **2.1** | | **Постановка задачи** |

В данном курсовом проекте необходимо разработать веб-приложение, взаимодействующее с базой данных с использованием объектно-ориентированного языка программирования Java, современных технологий и фреймворков.

Система должна автоматизировать процессы, которые упростят заказ билета онлайн для пользователя. Пользователь может просматривать основную информацию о цирке, просматривать афишу и покупать билеты на выступления, которые в дальнейшем он сможет просматривать. Также пользователь может оставлять отзывы о выступлениях. Администратор имеет доступ к данным всех пользователей, может изменять их роли. Также администратор может управлять афишей, анализировать статистику продаж и модерировать форум. Роль «менеджер» представляет собой менеджера цирка, который имеет возможность управления афишей (добавление, редактирование, удаление выступлений), имеет возможность модерировать форум и анализировать статистику продаж билетов.

Программа должна иметь интуитивно понятный любому пользователю интерфейс. Интерфейс должен иметь проверку на корректность введенных пользователем данных. Сервер Приложений – используется для размещения “серверной” части приложения, представленной Моделью на основе ORM технологии Hibernate, Бизнес-логика приложения (серверная часть) должна быть реализована на основе фреймворка Spring. При выполнение работы должны быть построены диаграммы функционального моделирования и декомпозиции отдельных процессов, информационной модели базы данных, последовательности, состояния, классов и развертывания.

|  |  |
| --- | --- |
| **2.2** | **Обоснование выбора компонентов и технологий для реализации автоматизированной системы работы цирка** |

Исходя из требования к курсовому проекту, серверная часть автоматизированной системы должна строиться на основе фреймворка Spring или набора спецификаций Java EE.

Из-за того, что разрабатываемое приложение является корпоративным, был выбран фреймворк Spring. Также фреймворк Spring облегчает создание приложений, облегчает подключение модулей, таких как модуль доступа к данным, которая отвечает за взаимодействие с СУБД. Для доступа к данным в Spring используется стандарт JDBC, или Java DataBase Connectivity – соединение с базой данных на Java. База соединяется с приложением по уникальному URL с помощью особых сущностей – драйверов.

Одной из причин использования Spring Framework является модуль MVC. MVC – это популярная схема приложения, при которой оно как бы разделяется на три основных части: модель данных, отображение и контроллер. Модель представляет собой данные, с которыми работает приложение, отображение отвечает за интерфейс и общение с пользователем, а контроллер – это логика изменения модели в ответ на действия пользователя. Чаще всего схему используют для веб-приложений.

Для проектирования пользовательского интерфейса использовались такие технологии как: механизм шаблонов Thymeleaf, язык программирования Javascript, язык описания внешнего вида страницы CSS. Thymeleaf способствует разработке клиентской части системы автоматизированной работы цирка, созданию расширяемого механизма шаблонизации, который помогает создавать шаблоны, соответствующие стандартам Web.

Используемая база данных – MySQL. Выбор обусловлен высокой производительностью, гибкостью и надежностью, высокой скоростью обработки запросов и широким функционалом.

|  |  |
| --- | --- |
| **2.3** | **Архитектурные решения** |

Использование клиент-серверной архитектуры обусловлено тем, что результатом курсового проекта является веб-приложение, способное обслуживать несколько клиентов одновременно. Клиент-серверная архитектура подразумевает собой то, что сетевая нагрузка распределена между клиентом и сервером, а сервер обрабатывает запрос клиента, отправляя запрос в базу данных, после чего возвращает результат клиенту.

В данном курсовом проекте использовались паттерны MVC и Singleton.

MVC разделяет работу приложения на три отдельные функциональные роли: модель данных (model), пользовательский интерфейс (view) и управляющую логику (controller). Таким образом, изменения, вносимые в один из компонентов, оказывают минимально возможное воздействие на другие компоненты. В данном паттерне модель не зависит от представления или управляющей логики, что делает возможным проектирование модели как независимого компонента и, например, создавать несколько представлений для одной модели. На рисунке 2.1 представлена логическая схема MVC-приложения.

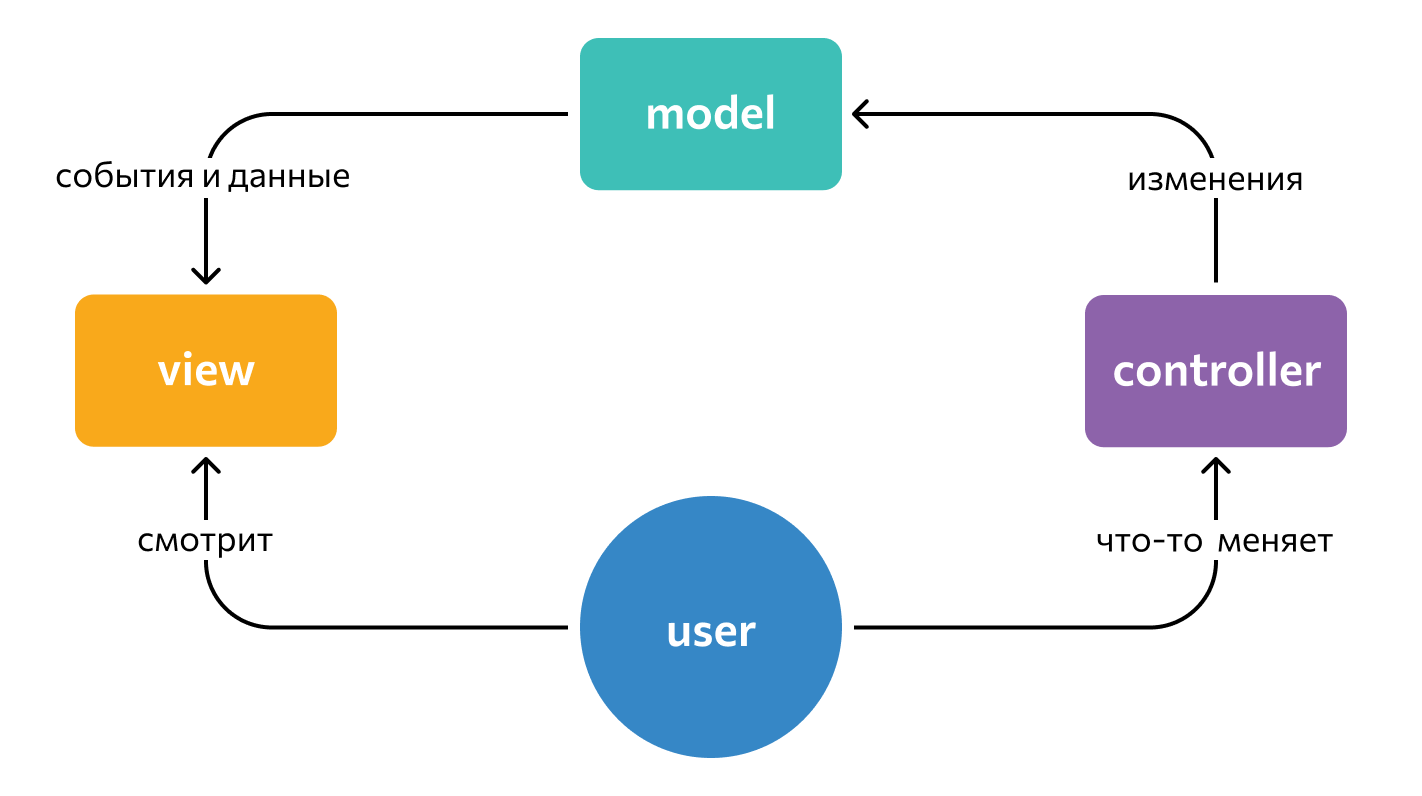


Рисунок 2.1 – Логическая схема MVC-приложения

У схемы MVC могут быть вариации. Например, в некоторых пользователь взаимодействует с контроллером не напрямую, а через интерфейс. Но основной принцип работы шаблона остается тем же.

Для реализации клиент-серверного взаимодействия используется архитектура REST. REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было легко масштабировать.

|  |  |
| --- | --- |
| **2.3** | **Описание алгоритмов, реализующих ключевую бизнес-логику системы заказа билетов на цирковое выступление** |

На рисунке 2.2 изображена блок-схема процесса покупки билета на выступление. Сначала пользователь переходит на страницу афиши, где он получает список выступлений, после чего выбирает нужное ему выступление. На странице самого выступления он может выбрать количество необходимых билетов после чего введенные данные проверяются на наличие мест и соответствие введенных значений в базе данных. Если данные неверны, пользователь получает уведомление с просьбой ввести другие данные. Если данные верны, билеты приобретаются.

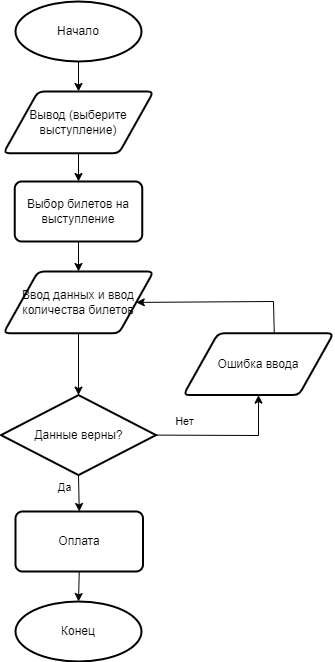


Рисунок 2.2 – Блок-схема процесса приобретения билета

Данная блок-схема отображает поток деятельности через систему и нужна для визуализации ее работы.

|  |  |
| --- | --- |
| **2.4** | **Проектирование пользовательского интерфейса** |

Необходимость разработки качественного пользовательского интерфейса является одной из самых необходимых частей разработки любого проекта. Это обусловлено тем, что даже самый хороший проект с плохим и запутанным интерфейсом не будет таким же эффективным как средний проект с хорошим и простым интерфейсом. Для разработки хорошего интерфейса, важно учитывать базовые требования к нему:

* стремиться к единообразию. Весь интерфейс должен быть выполнен в одном стиле и цветовой гамме, кнопки и блоки меню должны быть однотипными для всех страниц в веб-приложении;
* отталкиваться от проблемы пользователя. Наше веб-приложение создано для покупки билетов на выступления цирка, поэтому афиша должна быть на виду у пользователя;
* не перегружать интерфейс. Хороший интерфейс должен быть минималистичным, понятным и простым. Пользователь не должен искать нужные ему кнопки на разных страницах;
* элементы должны быть выразительными и заметными, при наведении на кнопку должна воспроизводиться анимация, которая покажет пользователю, что на нее можно нажать (hover).

Соблюдая данные правила, был разработан пользовательский интерфейс, представленный на рисунке 2.3.

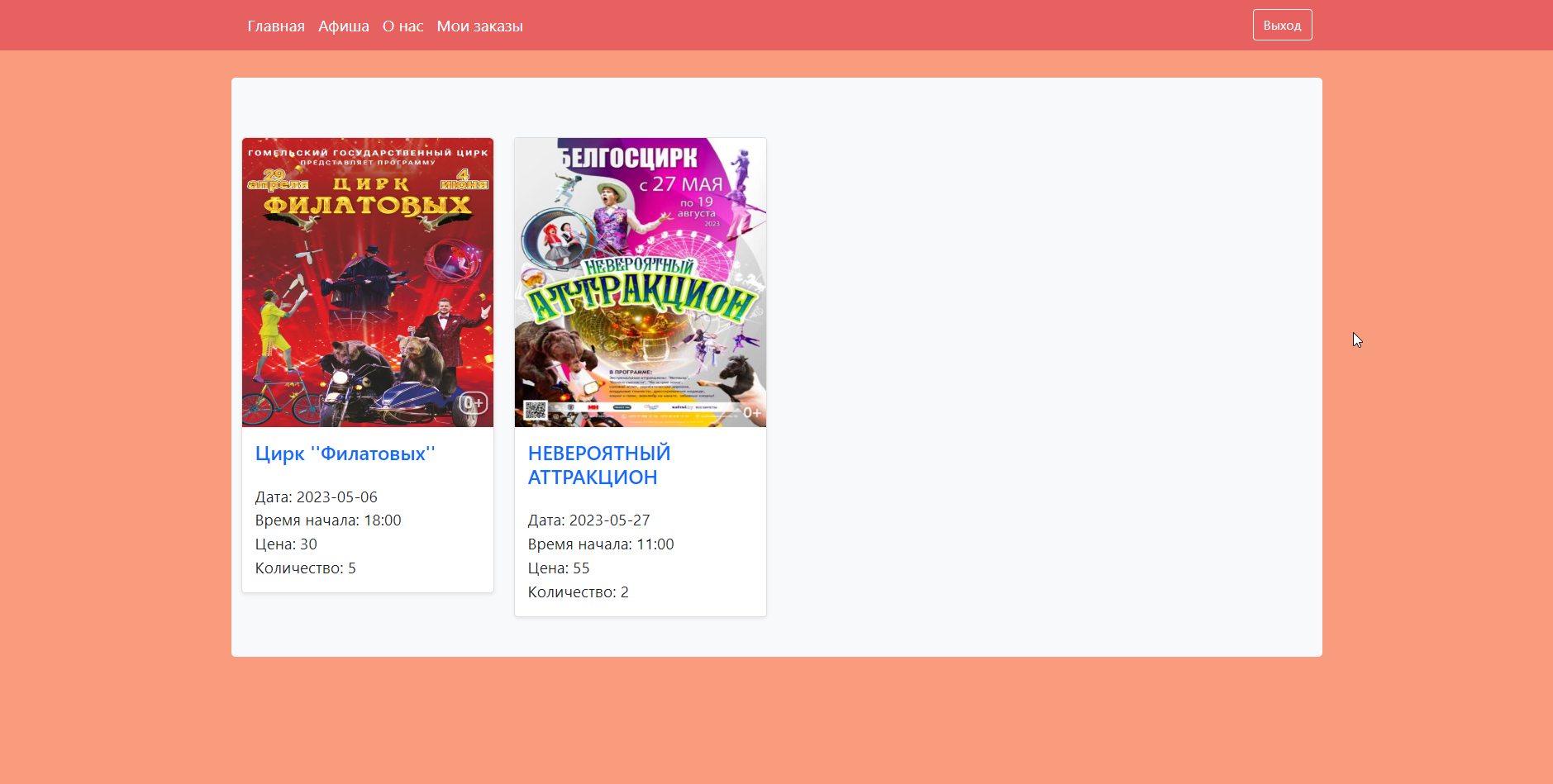


Рисунок 2.3 – Пользовательский интерфейс веб-приложения

Цвета страниц были сделаны в теплых тонах, так как цирк чаще всего посещают семьи, и тематика цирка больше ассоциируется с яркими тонами.

|  |  |
| --- | --- |
| **3** | **РУКОВОДСТВО РАЗВЕРТЫВАНИЯ И ИСПОЛЬЗОВАНИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ РАБОТЫ ЦИРКА** |

Проект упакован в jar-архив. Для запуска необходимо разместить его на сервере и выбрать в качестве запускаемого проекта.

После запуска приложения пользователь окажется на главной странице с «каруселью» фотографий цирка (рисунок 3.1).

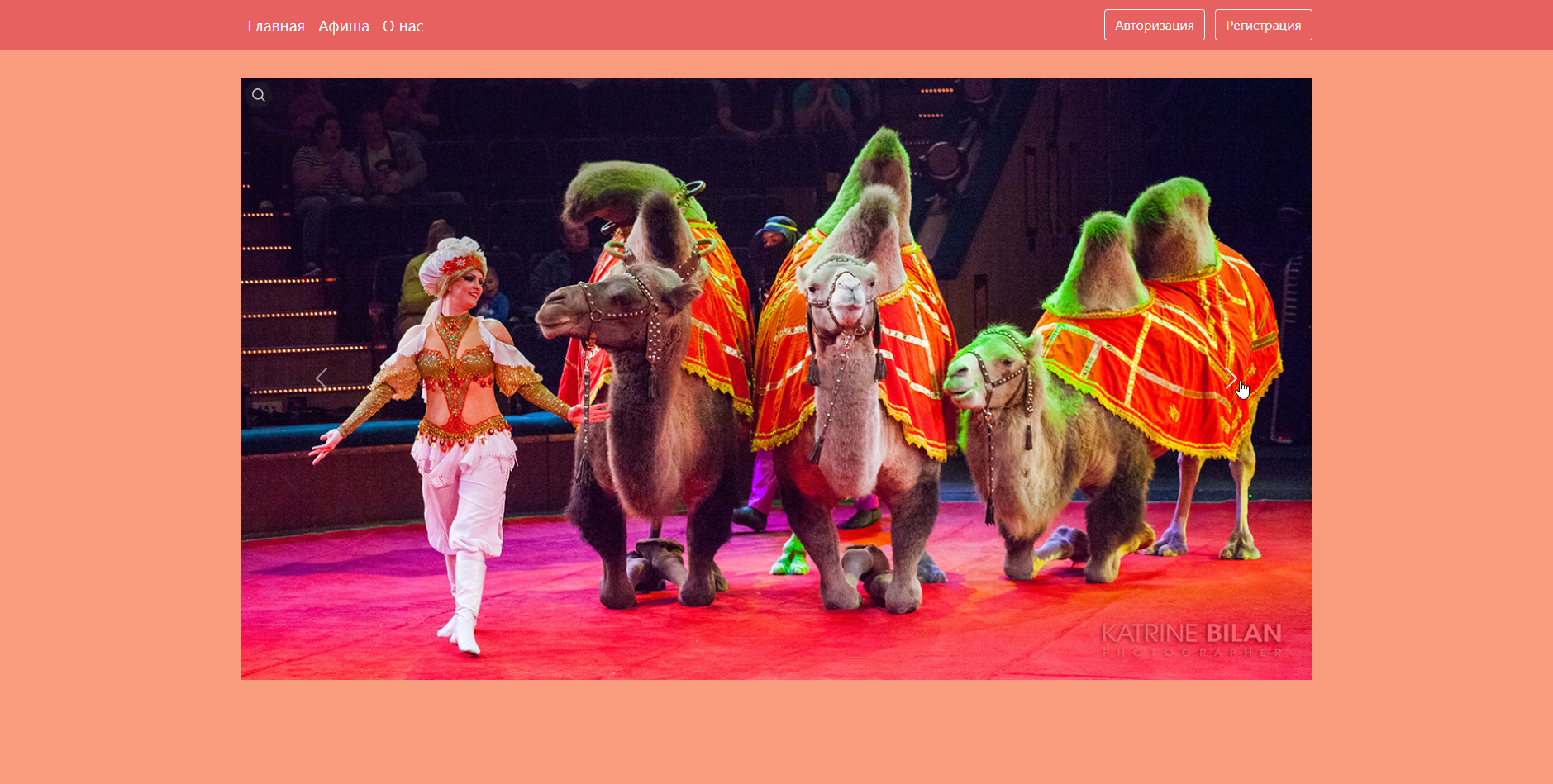


Рисунок 3.1 – Главная страница цирка

При попытке доступа к другим частям сайта или при нажатии кнопки «регистрация». Форма регистрации отображена на рисунке 3.2.

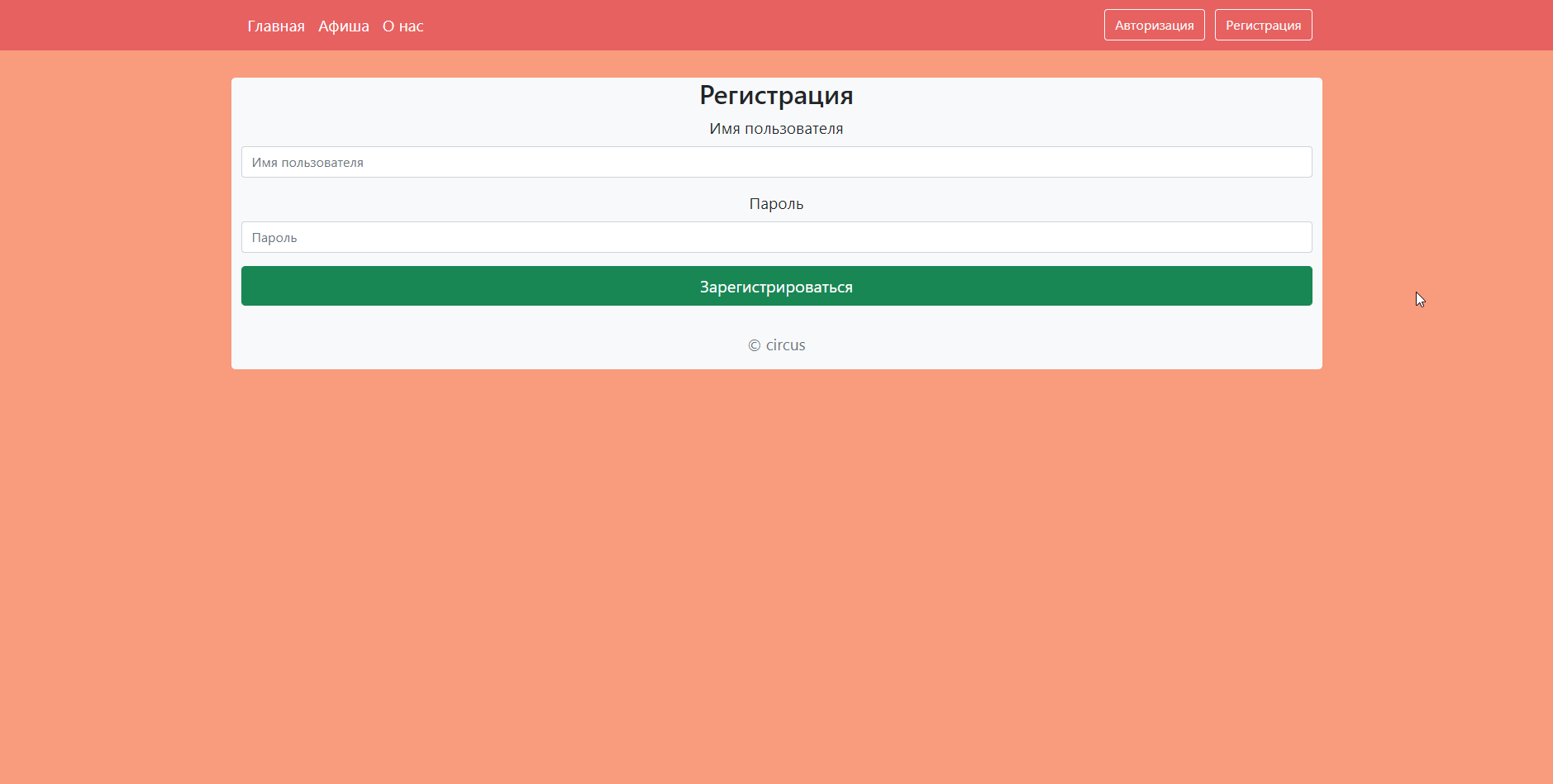


Рисунок 3.2 – Форма регистрации пользователя

Для регистрирующихся пользователей была предусмотрена система обработки типичных ошибок. Если пользователь не ввел пароль или указал имя, которое уже зарегистрировано, он получит ошибку с просьбой изменить введенные данные (рисунок 3.3).

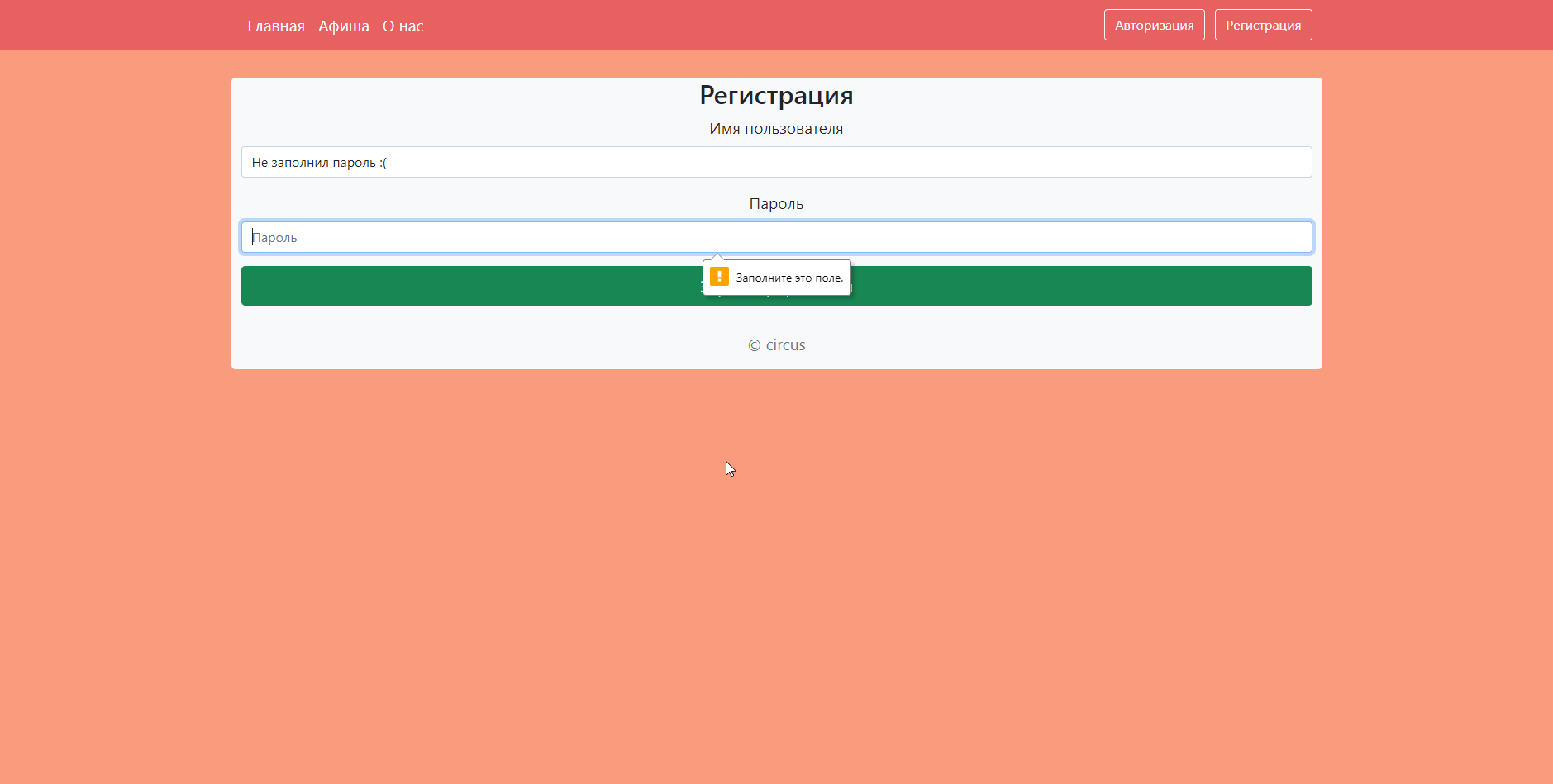


Рисунок 3.3 – Ошибка при регистрации пользователя

После того, как регистрация завершена, пользователь будет перенаправлен на страницу «Авторизация», где должен будет ввести данные, указанные при регистрации (рисунок 3.4). Первый зарегистрировавшийся пользователь автоматически становится администратором, остальным пользователям автоматически устанавливается роль «Клиент», которую сможет изменять Администратор.

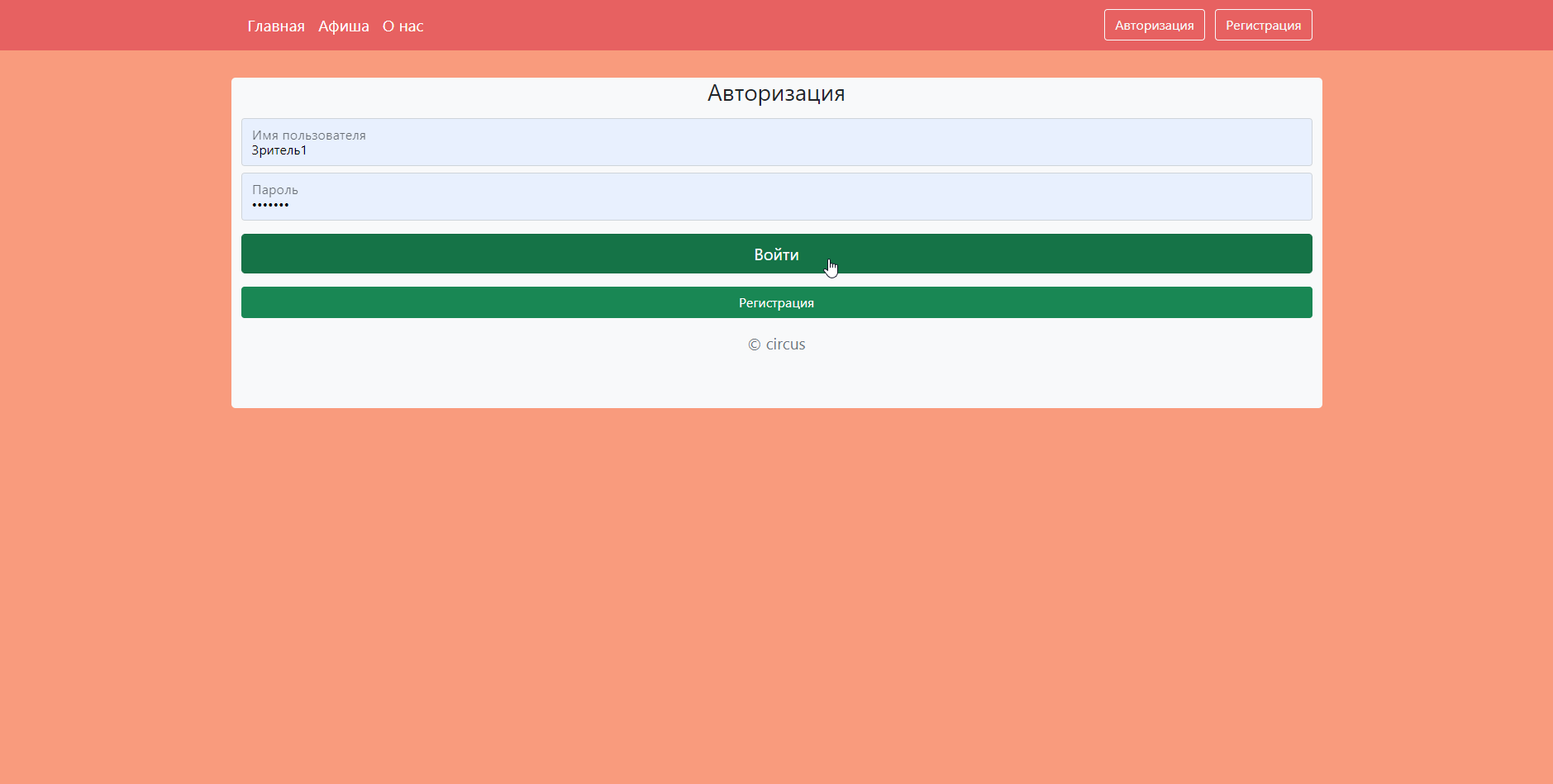


Рисунок 3.4 – Заполненная форма авторизации пользователя

|  |  |
| --- | --- |
| **3.1** | **Вход под ролью пользователя** |

У пользователя есть возможность выйти из своего профиля нажав кнопку «Выход». Нажав кнопку «Афиша», пользователь будет направлен на соответствующую страницу (рисунок 3.5), где может перейти на страницу представления. При покупке большего количества билетов, чем осталось, пользователя попросят изменить введенные значения (рисунок 3.6 и рисунок 3.7), к тому же, пользователь может оставить отзыв о выступлении(рисунок 3.8); нажав кнопку «О нас» пользователь будет перенаправлен на страницу с информацией о цирке (рисунок 3.9); нажав клавишу «Заказы» он сможет посмотреть приобретенные билеты (рисунок 3.10).

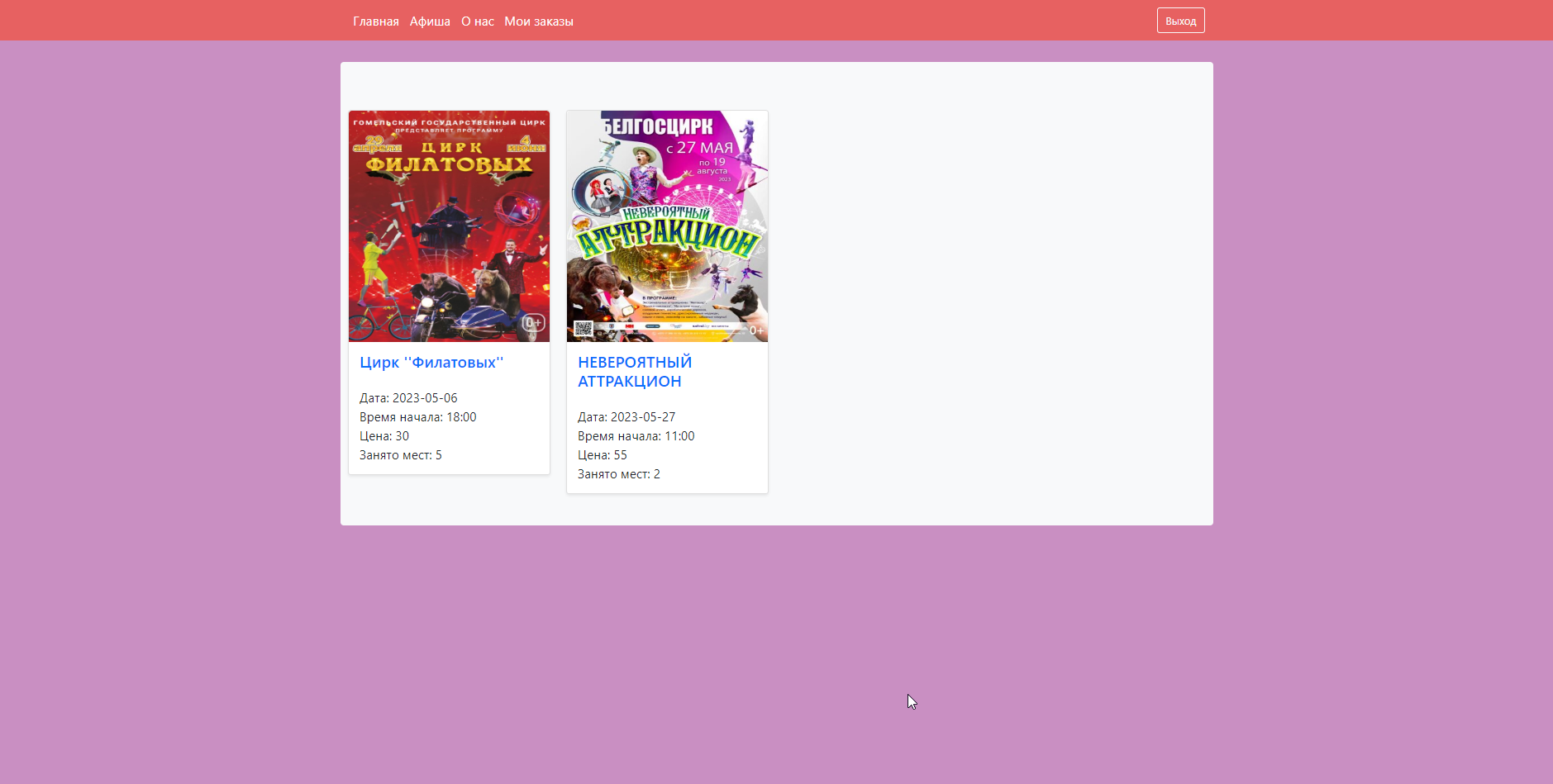


Рисунок 3.5 – Страница «Афиша»

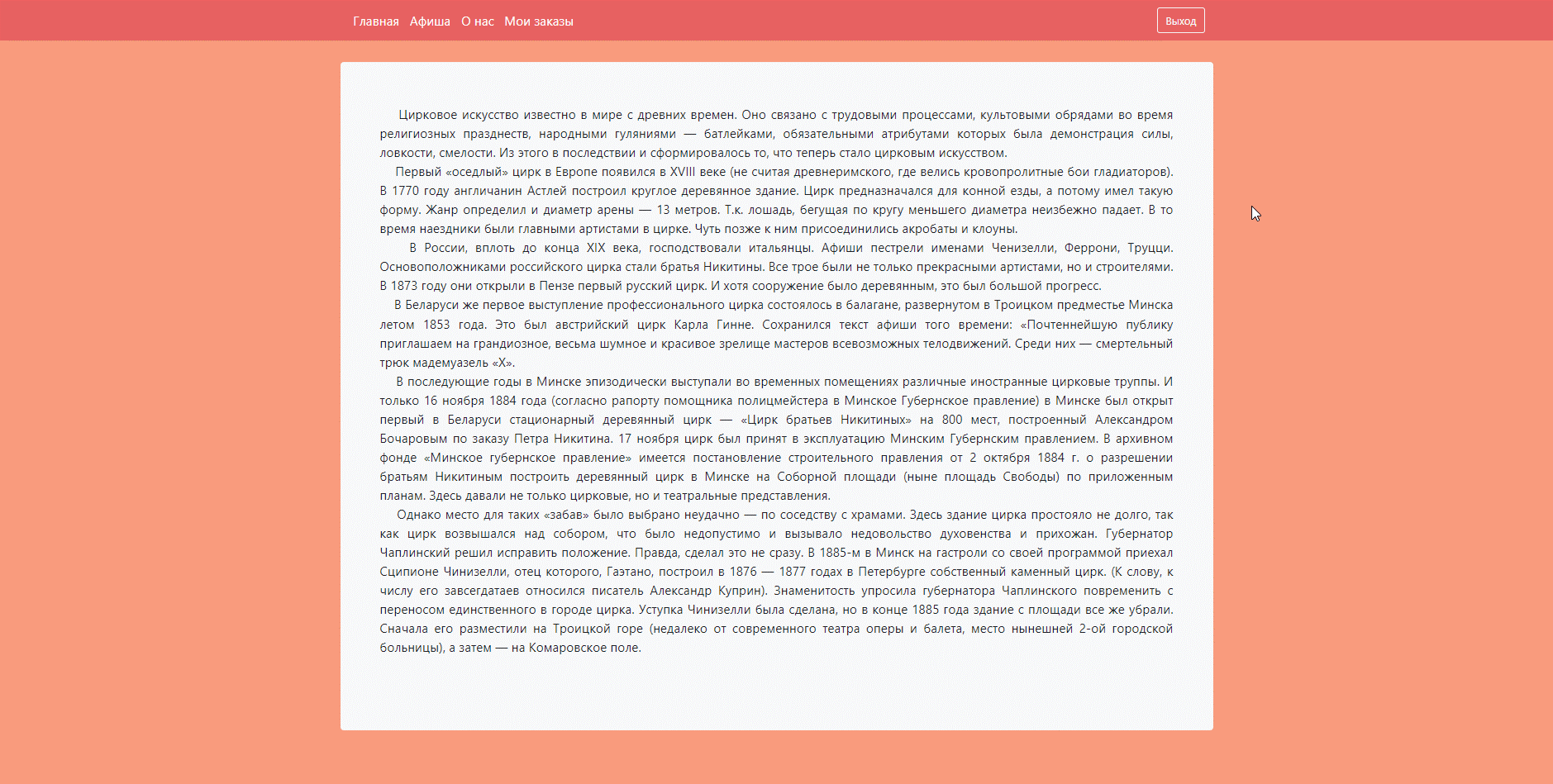


Рисунок 3.6 – Страница «О нас»

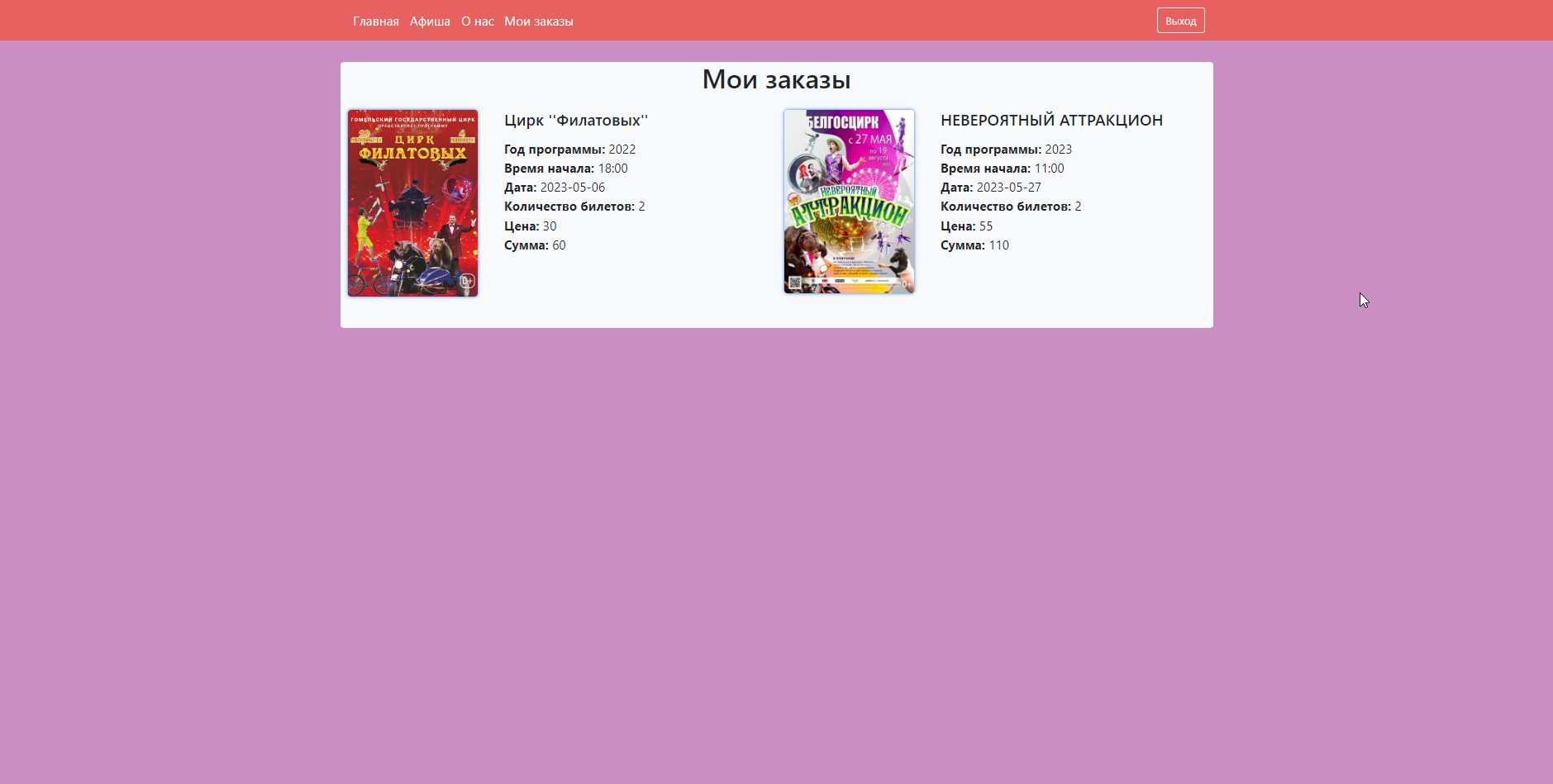


Рисунок 3.7 – Страница «Мои заказы»

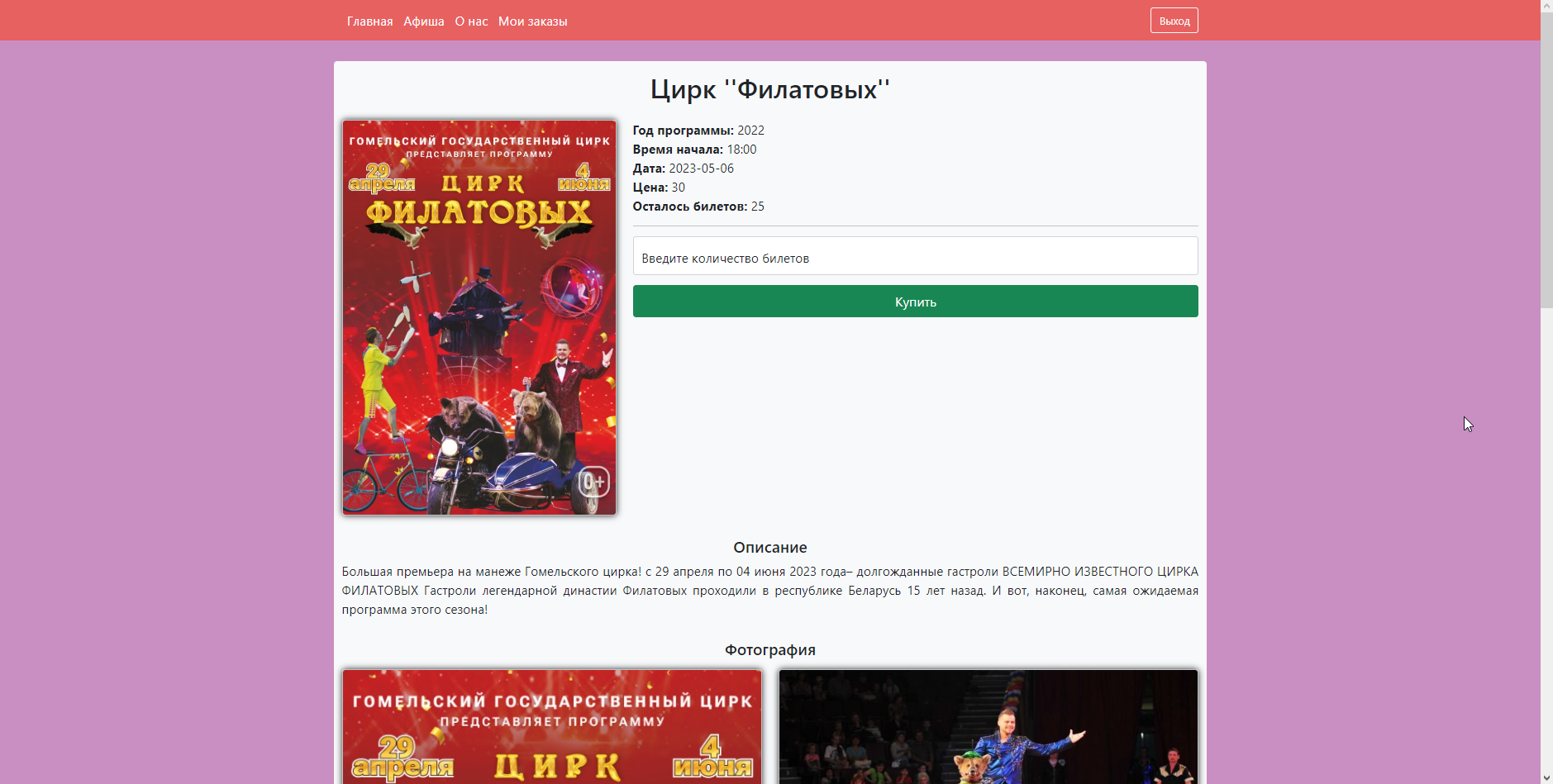


Рисунок 3.8 – Страница покупки билета на выступление

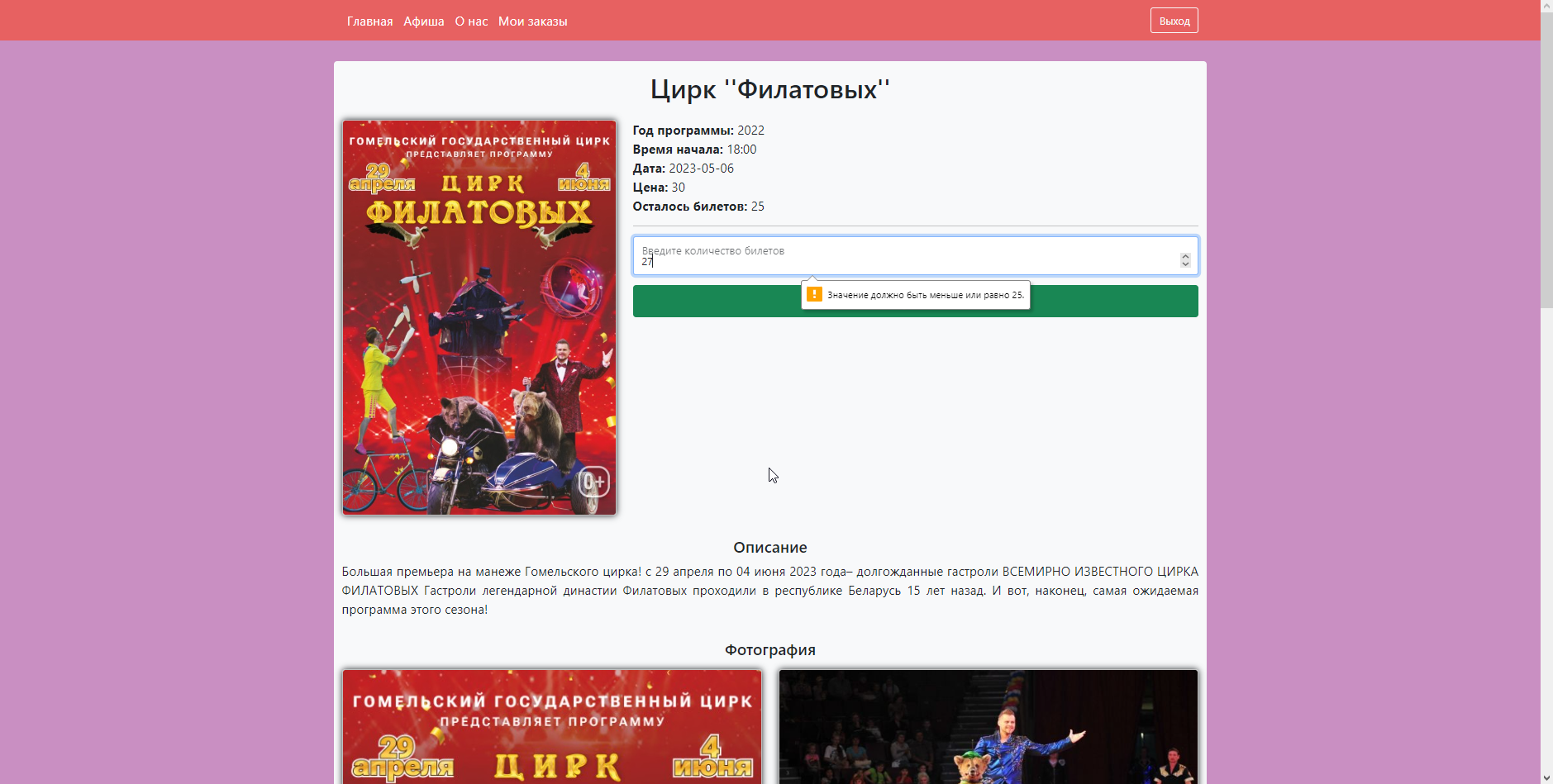


Рисунок 3.9 – Ошибка при вводе данных для покупки билета

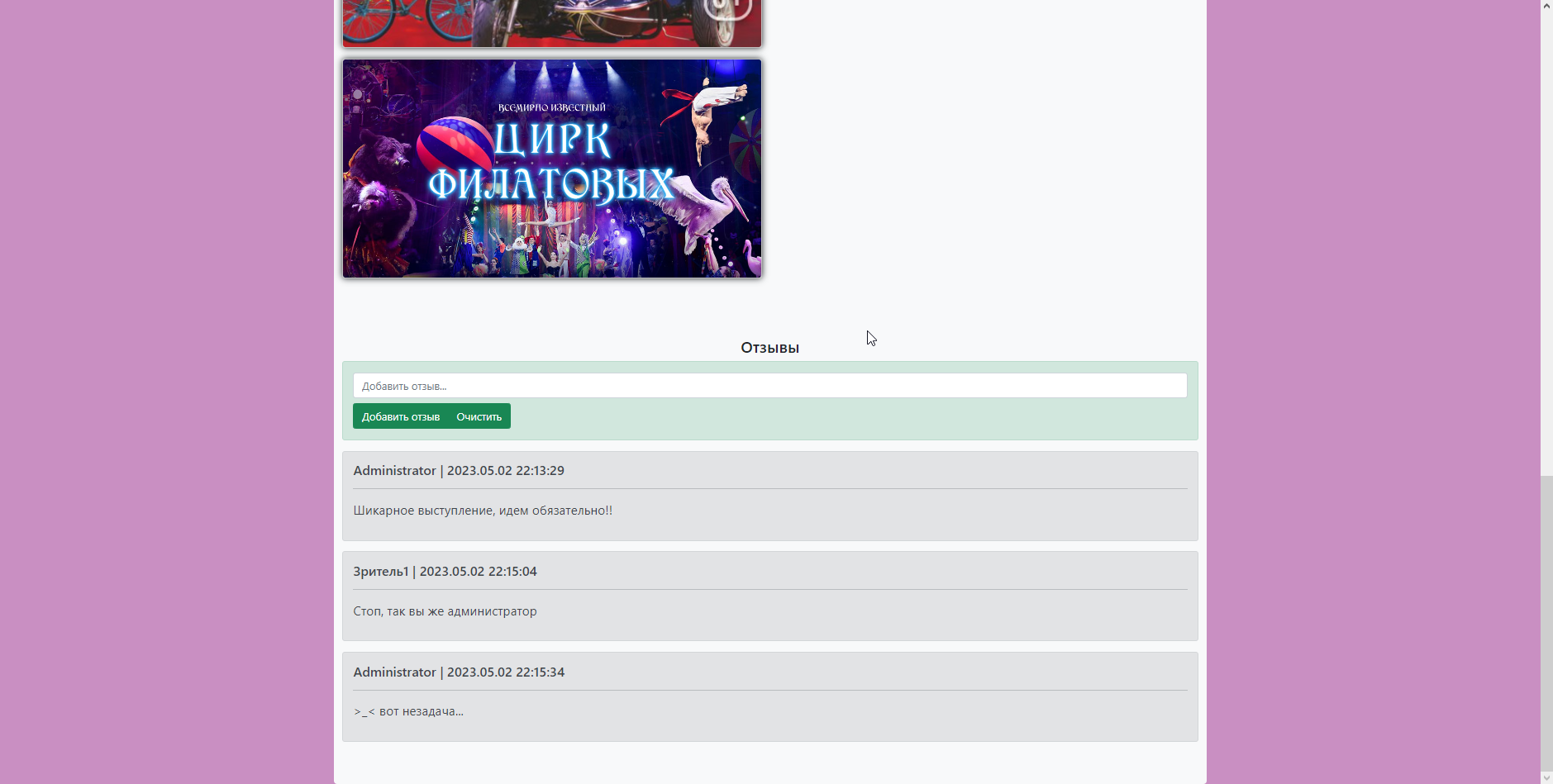


Рисунок 3.10 – Форма создания отзыва о выступлении

|  |  |
| --- | --- |
| **3.2** | Вход под ролью администратора |

Администратор имеет такие же возможности как и пользователь, а также он может перейти на страницу добавления выступления в афишу (рисунок 3.11); может редактировать уже существующее выступление (рисунок 3.12); может перейти в раздел «Финансы» для формирования отчета о каждом выступлении и суммарной выручке (рисунок 3.13); также администратор имеет право регулировать роли других пользователей и удалять их аккаунты (рисунок 3.14).

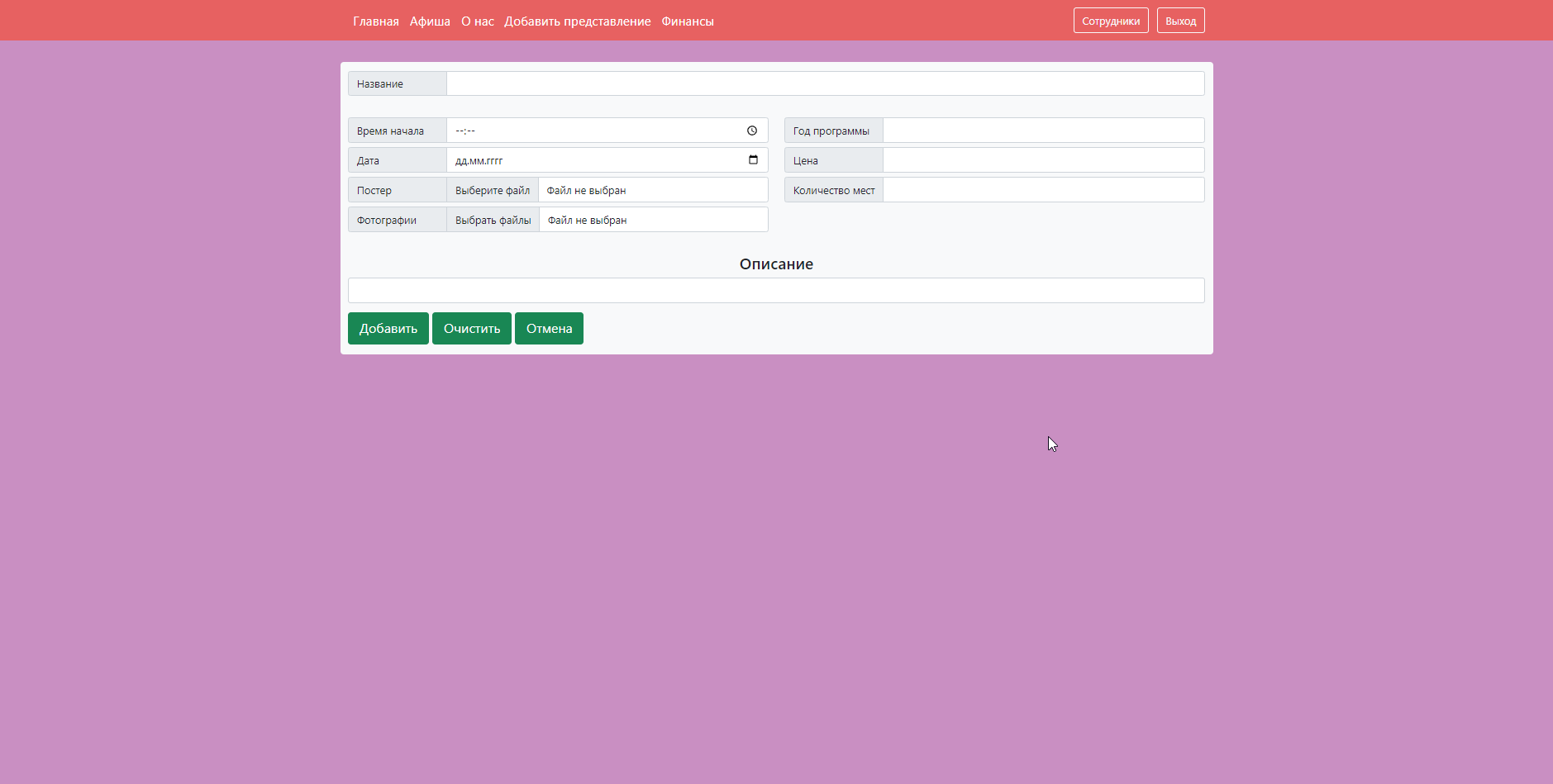


Рисунок 3.11 – Форма добавления выступления

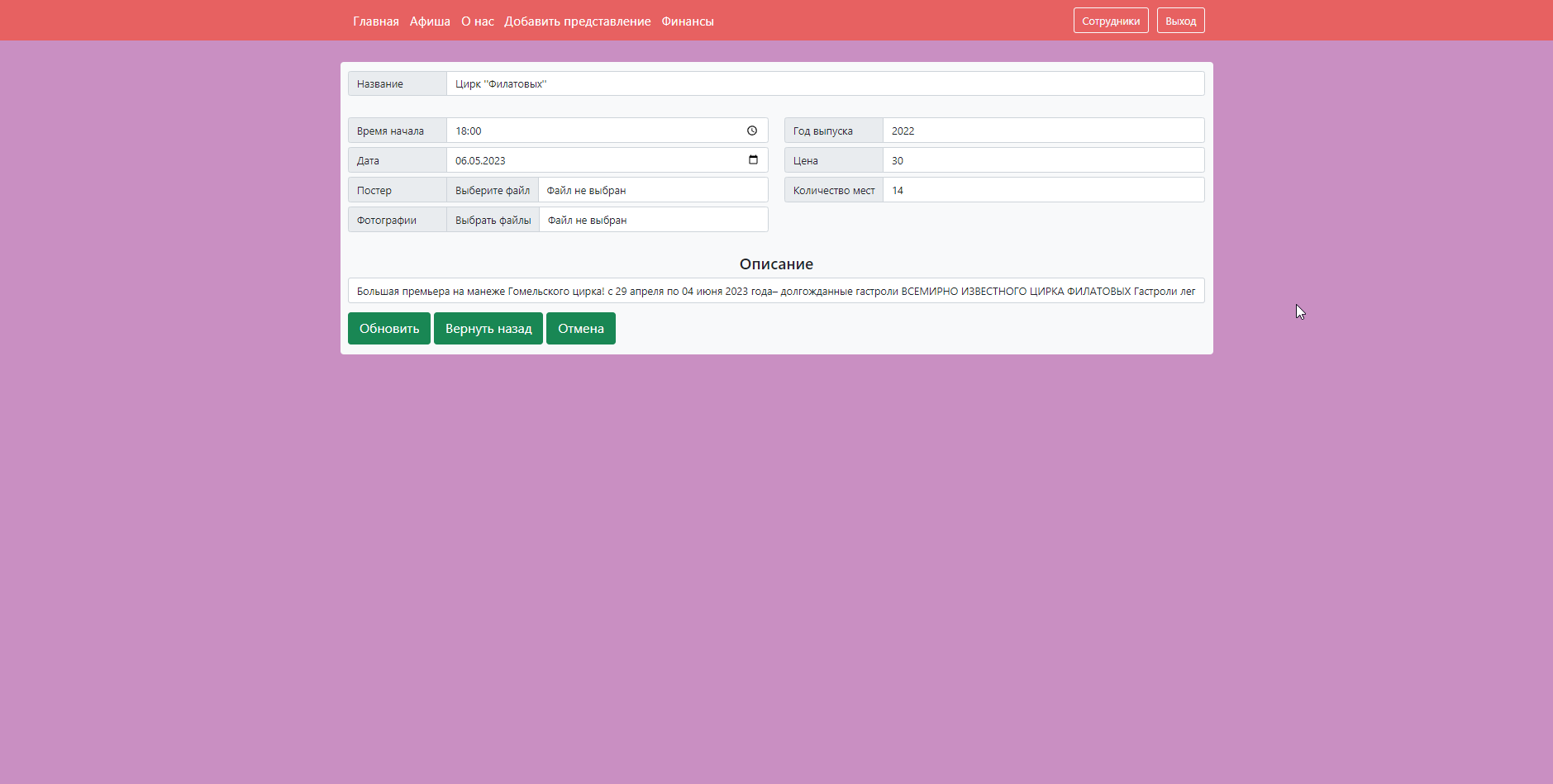


Рисунок 3.12 – Форма изменения выступления

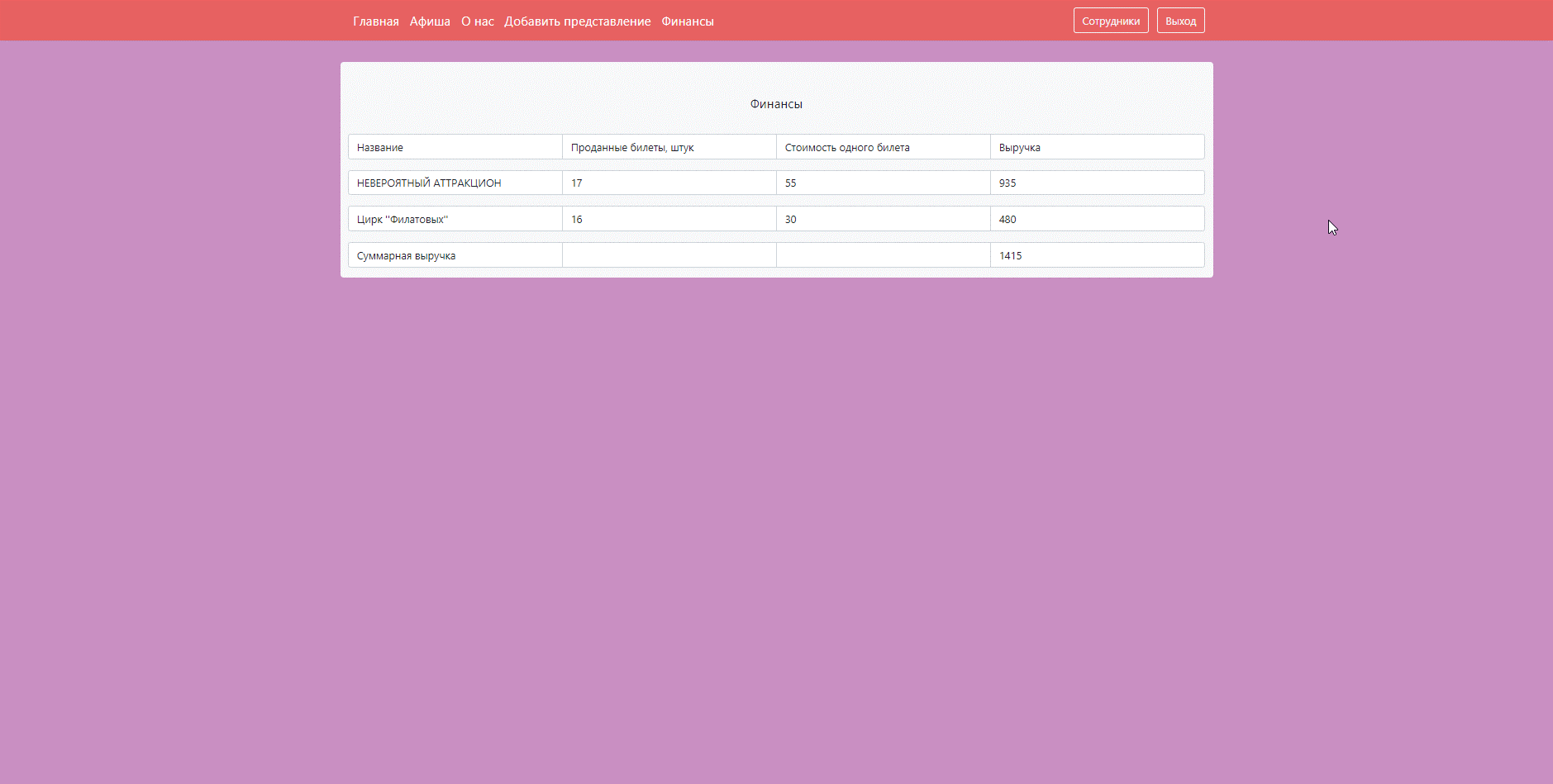


Рисунок 3.13 – Финансовый отчет по выступлениям цирка

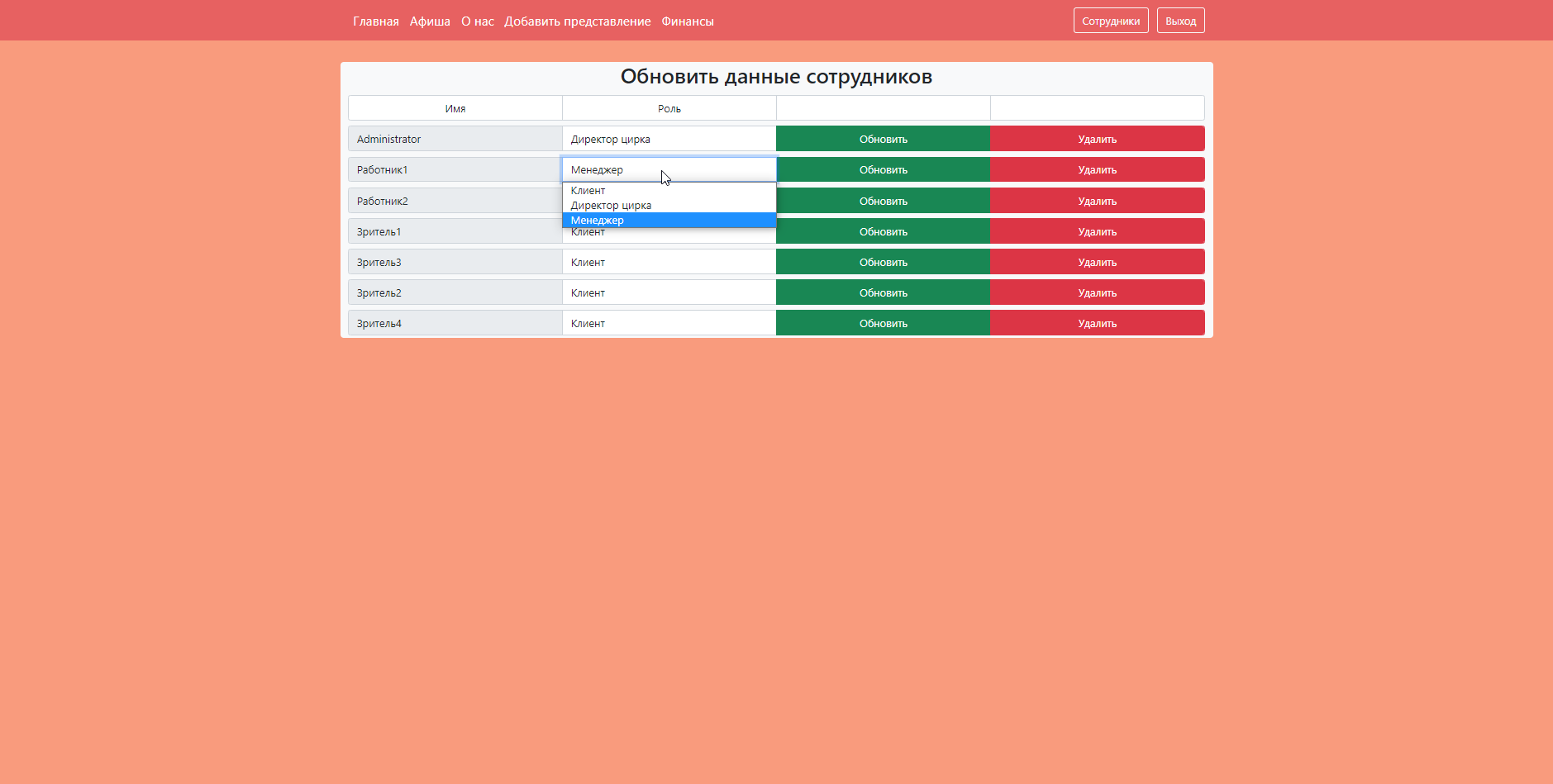


Рисунок 3.14 – Управление данными сотрудников

|  |  |
| --- | --- |
| **3.3** | Вход под ролью менеджера |

При авторизации под ролью менеджера, у пользователя остаются все права администратора, кроме управления данными пользователей и их аккаунтами. Пример того, как выглядят кнопки навигации менеджера представлен на рисунке 3.15.

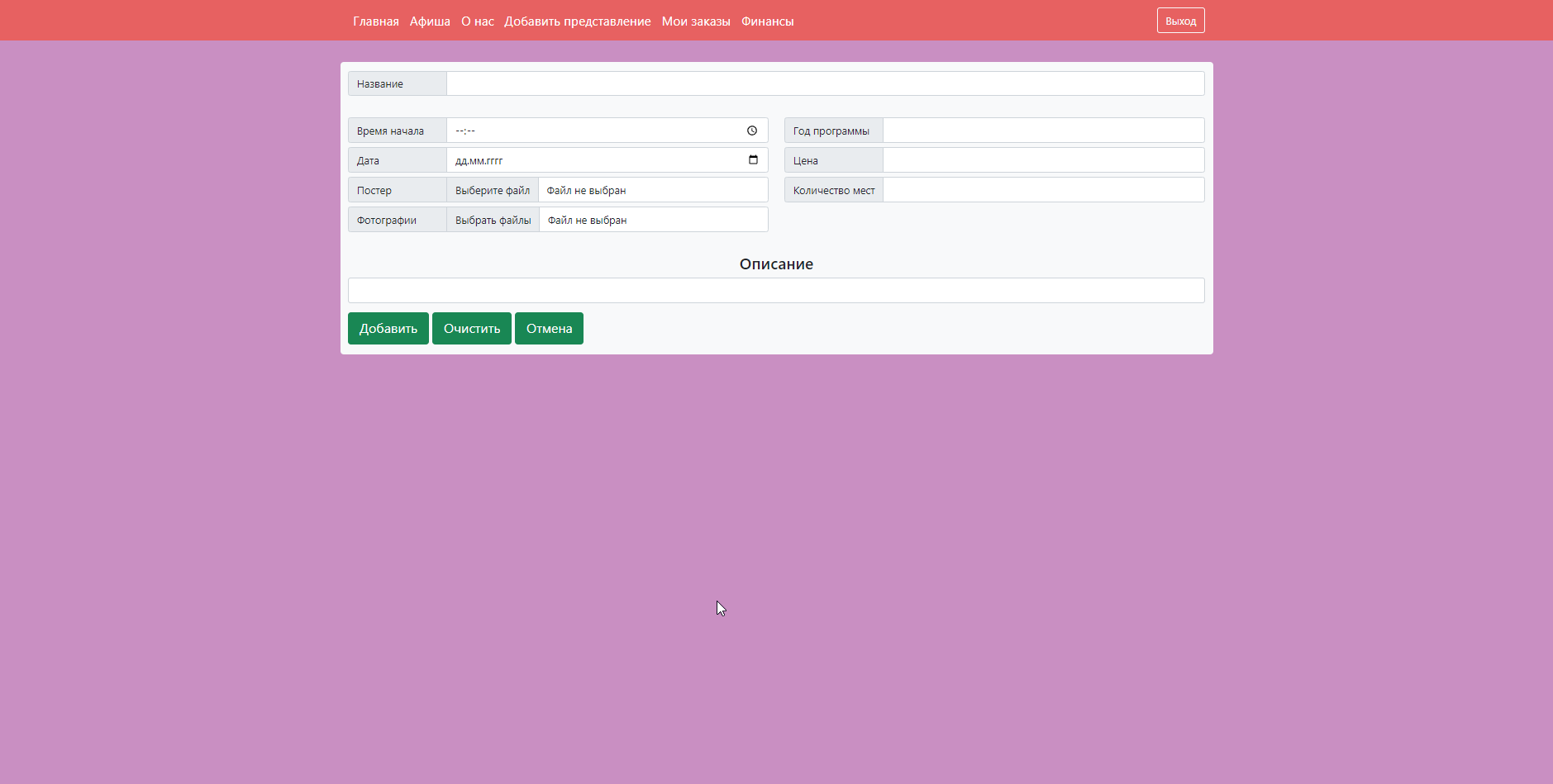


Рисунок 3.15 – Веб-приложение от лица менеджера

Для того, чтобы выйти из аккаунт нужно нажать кнопку «Выход» в меню навигации.

**ЗАКЛЮЧЕНИЕ**

В ходе курсового проекта была изучена предметная область, что позволило выявить наиболее значимые функции для автоматизированной системы работы цирка. Данная автоматизированная система была написана на языке программирования Java, для запуска сервера использовался Tomcat. База данных, примененная в проекте – MySQL.

В результате выполнения курсового проекта было спроектировано и разработано распределенное web-приложение, которое позволяет приобретать билеты на выступления, а также заносить новые в базу данных. Также был реализован функционал для администрирования системы.

При разработке данного приложения были поставлены и достигнуты следующие задачи:

* Изучена предметная область;
* Выявлены методы решения предоставленной задачи;
* Построена информационная модель базы данных;
* Разработаны UML-диаграммы бизнес-процессов;
* Составлены блок-схемы процесса заказа билетов на выступление;
* Разработано приложения на языке Java;
* Спроектирован и разработан удобный интерфейс приложения;
* Написана документация к приложению.

В данном проекте была описана пошаговая инструкция по развёртыванию web-приложения, продемонстрированы функции пользователя, администратора и менеджера.

В разработке web-приложения были использованы различные паттерны проектирования, позволяющие оптимизировать его и упростить процесс создания, различные фреймворки, методы и стили языков гипертекстовой разметки, а для работы с данными использованы возможности реляционной базы данных MySQL.

Разработанное приложение может быть улучшено путем усовершенствования интерфейса, а также введением дополнительных функций. Преимущество данного приложения заключается в том, что доступ к нему можно получить из любой точки мира с доступом к сети Интернет. Также оно является достаточно быстрым по скорости обработки данных.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Кондратьева И.В., Кравченко В.В. Автоматизированные информационные системы в цирковом искусстве: учебное пособие./ И.В. Кондратьева, В.В. Кравченко. – Москва: издательство «Юрайт», 2020.

[2] Цирковое искусство в эпоху цифровых технологий: материалы международной научно-практической конференции / Н.В. Королева, В.А. Смирнова. - Москва: Издательство МГУПС, 2018.

[3] Автоматизированная информационная система [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/334809

[4] Spring Словарь [Электронный ресурс]. – Электронные данные. – Режим доступа: https://blog.skillfactory.ru/glossary/spring/#:~:text=Spring%2C%20или%20Spring%20Framework%20—%20фреймворк,каркас%20для%20разных%20видов%20приложений.

[5] Habr. Справочник Java программиста 1: JPA и Hibernate [Электронный ресурс]. – Электронные данные. – Режим доступа: https://habr.com/ru/articles/265061/

[6] Академик – словари и энциклопедии [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dic.academic.ru/dic.nsf/business/18231

[7] JetBrains – профессиональные инструменты для разработки [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.jetbrains.com/ru-ru/opensource/idea/

[8] MVC Skillfactory [Электронный ресурс] – Электронные данные. – Режим доступа: https://blog.skillfactory.ru/glossary/mvc/

[9] Простое руководство по диаграммам активности UML [Электронный ресус]. – Электронные данные. – Режим доступа: https://creately.com/blog/ru/uncategorized-ru/учебник-по-диаграмме-активности/#:~:text=Диаграмма%20активности%20UML%20позволяет%20более,потока%20событий%20в%20бизнес-процессе.

[10] Businessstudio.com.ua – заметки управленца [Электронный ресурс]. -– Электронные данные. – Режим доступа: http://www.businessstudio.com.ua/bp/bs/overview/notation\_idef0.php

[11] Создание пользовательского интерфейса от Яндекс. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://habr.com/ru/post/74330/

**Приложение А**

**Листинг скрипта генерации базы данных**

**CREATE TABLE `carts` (**

**`id` bigint(20) NOT NULL,**

**`count` int(11) NOT NULL,**

**`price` int(11) NOT NULL,**

**`sum` int(11) NOT NULL,**

**`owner\_id` bigint(20) DEFAULT NULL,**

**`session\_id` bigint(20) DEFAULT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;**

**CREATE TABLE `hibernate\_sequence` (**

**`next\_val` bigint(20) DEFAULT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;**

**CREATE TABLE `income` (**

**`id` bigint(20) NOT NULL,**

**`count` int(11) NOT NULL,**

**`income` int(11) NOT NULL,**

**`price` int(11) NOT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;**

**CREATE TABLE `sessions` (**

**`id` bigint(20) NOT NULL,**

**`count` int(11) NOT NULL,**

**`date` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,**

**`description` varchar(5000) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,**

**`name` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,**

**`poster` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,**

**`screenshots` tinyblob,**

**`start` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,**

**`time` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,**

**`income\_id` bigint(20) DEFAULT NULL,**

**`owner\_id` bigint(20) DEFAULT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;**

**Продолжение приложения А**

**CREATE TABLE `users` ( `id` bigint(20) NOT NULL,**

**`password` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,**

**`role` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,**

**`username` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;**

**ALTER TABLE `carts`**

**ADD PRIMARY KEY (`id`),**

**ADD KEY `FK97nj98gckexxfvmfcqsix8oq2` (`owner\_id`),**

**ADD KEY `FKmey5migt5dl8lu34i1wyrae2a` (`session\_id`);**

**ALTER TABLE `comments`**

**ADD PRIMARY KEY (`id`),**

**ADD KEY `FK5qn47fqhw2x6emyvt555al2gs` (`book\_id`);**

**ALTER TABLE `income`**

**ADD PRIMARY KEY (`id`);**

**ALTER TABLE `sessions`**

**ADD PRIMARY KEY (`id`),**

**ADD KEY `FKhi1kaya99b8m9tvfe7xjynuoi` (`income\_id`),**

**ADD KEY `FKh4dp7hgldljujvtrils5ivxuu` (`owner\_id`);**

**ALTER TABLE `users`**

**ADD PRIMARY KEY (`id`);**

**ALTER TABLE `carts`**

**ADD CONSTRAINT `FK97nj98gckexxfvmfcqsix8oq2` FOREIGN KEY (`owner\_id`) REFERENCES `users` (`id`),**

**ADD CONSTRAINT `FKmey5migt5dl8lu34i1wyrae2a` FOREIGN KEY (`session\_id`) REFERENCES `sessions` (`id`);**

**ALTER TABLE `comments`**

**ADD CONSTRAINT `FK5qn47fqhw2x6emyvt555al2gs` FOREIGN KEY (`book\_id`) REFERENCES `sessions` (`id`);**

**ALTER TABLE `sessions`**

**ADD CONSTRAINT `FKh4dp7hgldljujvtrils5ivxuu` FOREIGN KEY (`owner\_id`) REFERENCES `users` (`id`),**

**ADD CONSTRAINT `FKhi1kaya99b8m9tvfe7xjynuoi` FOREIGN KEY (`income\_id`) REFERENCES `income` (`id`);**

**COMMIT;**

**Приложение Б**

**Листинг кода алгоритмов, реализующих бизнес-логику**

**Afishacont.java**

package com.circus.controllers;  
  
import com.circus.controllers.Main.Main;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
  
@Controller  
public class AfishaCont extends Main {  
 public AfishaCont() {  
 }  
  
 @GetMapping({"/catalog"})  
 public String catalog(Model model) {  
 model.addAttribute("sessions", this.repoSessions.findAll());  
 model.addAttribute("role", this.getRole());  
 return "catalog";  
 }  
  
 @GetMapping({"/catalog/all"})  
 public String catalog\_main(Model model) {  
 model.addAttribute("sessions", this.repoSessions.findAll());  
 model.addAttribute("role", this.getRole());  
 return "catalog";  
 } **}**

**Ticketscont.java**

package com.circus.controllers;  
import com.circus.controllers.Main.Main;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
  
@Controller

public class TicketsCont extends Main {  
 public TicketsCont() { }

**Продолжение приложения Б**

@GetMapping({"/cart"})  
 public String cart(Model model) {  
 model.addAttribute("user", this.getUser());  
 model.addAttribute("role", this.getRole());  
 return "cart";  
 }  
}

**Carts.java**

package com.circus.models;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.ManyToOne;  
import javax.persistence.OneToOne;  
  
@Entity  
public class Carts {  
 @Id  
 @GeneratedValue(  
 strategy = GenerationType.*AUTO* )  
 @Column(  
 nullable = false,  
 updatable = false  
 )  
 private Long id;  
 private int count;  
 private int price;  
 private int sum;  
 @OneToOne(  
 fetch = FetchType.*LAZY* )  
 private Sessions session;  
 @ManyToOne(  
 fetch = FetchType.*LAZY* )

**Продолжение приложения Б**

private Users owner;  
  
 public Carts(int count, int price, int sum, Sessions session) {  
 this.count = count;  
 this.price = price;  
 this.sum = sum;  
 this.session = session;

}  
 public Long getId() {  
 return this.id;  
 }  
  
 public int getCount() {  
 return this.count;  
 }  
  
 public int getPrice() {  
 return this.price;  
 }  
  
 public int getSum() {  
 return this.sum;  
 }  
  
 public Sessions getSession() {  
 return this.session;  
 }  
  
 public Users getOwner() {  
 return this.owner;  
 }  
  
 public void setCount(final int count) {  
 this.count = count;  
 }  
 public void setPrice(final int price) {  
 this.price = price;

}

public void setSum(final int sum) {

**Окончание приложения Б**

this.sum = sum;  
 }  
  
 public void setSession(final Sessions session) {  
 this.session = session;  
 }  
  
 public void setOwner(final Users owner) {  
 this.owner = owner;  
 }  
  
 public Carts() {  
 }  
}

**UserService.java**

package com.circus.service;  
  
import com.circus.repo.RepoUsers;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
  
@Service  
public class UserService implements UserDetailsService {  
 @Autowired  
 private RepoUsers repoUsers;  
  
 public UserService() {  
 }  
  
 public UserDetails loadUserByUsername(String s) throws UsernameNotFoundException {  
 return this.repoUsers.findByUsername(s);  
 }

**Приложение В**

**Отчёт о проверке на заимствования в системе «Антиплагиат»**

