МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное  
 учреждение высшего образования  
«Санкт-Петербургский политехнический университет Петра Великого»   
(ФГАОУ ВО «СПБПУ»)

Институт среднего профессионального образования

Отделение информационных технологий

Специальность: 09.02.07 Информационные системы и программирование  
Дисциплина: МДК.11.01 Технология разработки и защиты баз данных  
Группа: 32919/8

КУРСОВОЙ ПРОЕКТ

на тему:

«Информационная система театральные кассы»

Студент Моисеев Евгений Васильевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

(ФИО)

Руководитель Сынкова Алина Денисовна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО)

(подпись)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Сынкова А.Д.

(оценка)

(ФИО)

(подпись)

Санкт-Петербург  
2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное  
учреждение высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»  
 (ФГАОУ ВО «СПБПУ»)

Институт среднего профессионального образования

Отделение информационных технологий

Задание на курсовое проектирование

по МДК.11.01 «Технология разработки и защиты баз данных»

Специальность: 09.02.07 Информационные системы и программирование

студенту(ке) Моисееву Е.В.\_\_ группа 32919/8

(фамилия, инициалы)

Тема: Информационная система театральные кассы

* программа должна обладать функционалом бронирования (покупки) билета, отмены покупки билета (возврата);
* программа должна иметь поисковый функционал с возможностью, фильтрации по: участвующим актёрам, дате показа, названию события;
* программа должна иметь возможность добавления, удаления и изменения самого события, а также отображать все поисковые атрибуты в каждом событии;
* программа должна обладать функционалом добавления новых сотрудников в БД,
* программа должна обладать функционалом назначения сотрудников на роли в спектакле, снятие с роли и возможность замещения какого-либо из актёров;
* программа также предполагает разделение возможностей пользователей по ролям. Ожидаемые роли: Администратор, менеджер персонала, организатор событий, кассир. Администратору будут доступны все возможности. Менеджеру персонала доступны будут таблицы и функции, связанные с наймом и увольнением работников (занесение и исключение из таблиц БД), а также редактирование таких таблиц. Организатору событий доступен функционал назначения и снятие сотрудников с ролей, создание и изменение событий и программ, добавление, удаление изменение данных о сценариях в БД, добавление. Кассиру доступны создание, изменение данных о билетах, возможность регистрации и изменения данных о пользователях, доступно добавление, удаление, изменение промокодов;
* для приложения предполагается работа с потенциальным внешним сервисом касс, для этого в информации о билете будет указан код транзакции оплаты. То есть информация о платёжных операциях должна храниться в независимом сервисе;

**Руководитель**  Сынкова А.Д.

(подпись, дата) (инициалы, фамилия)

**Задание принял к выполнению:**

**Студент**  Моисеев Е.В.

(подпись, дата) (инициалы, фамилия)

Содержание

[Введение 5](#_Toc200557870)

[1 Общая часть 6](#_Toc200557871)

[1.1 Анализ предметной области 6](#_Toc200557872)

[1.2 Анализ рынка существующих решений 7](#_Toc200557873)

[1.3 Сравнение технологий 10](#_Toc200557874)

[2 Специальная часть 15](#_Toc200557875)

[2.1 Назначение разработки 15](#_Toc200557876)

[2.2 Требования к программе или программному изделию 16](#_Toc200557877)

[2.3 Проектирование 18](#_Toc200557878)

[2.3.1 Диаграмма прецедентов использования 18](#_Toc200557879)

[2.3.2 Концептуальная модель 19](#_Toc200557880)

[2.3.3 Даталогическая модель 20](#_Toc200557881)

[2.3.4 Физическая модель 21](#_Toc200557882)

[2.4 Описание алгоритма 22](#_Toc200557883)

[2.4.1 Формы редактирования данных таблицы 22](#_Toc200557884)

[2.4.2 Защита данных 24](#_Toc200557885)

[2.4.3 Разграничение прав 25](#_Toc200557886)

[2.5 Описание программы 26](#_Toc200557887)

[2.6 Инструкция пользователя 32](#_Toc200557888)

[2.7 Тестирование 34](#_Toc200557889)

[Заключение 38](#_Toc200557890)

[Список использованных источников 39](#_Toc200557891)

[ПРИЛОЖЕНИЕ А (обязательное) Листинг SQL-скрипта 40](#_Toc200557892)

[ПРИЛОЖЕНИЕ Б (обязательное) Листинг приложения 42](#_Toc200557893)

Введение

Театральное искусство — это вид исполнительского творчества, в котором актёры, используя драматургический текст, режиссёрскую концепцию, сценографию и другие выразительные средства, создают художественный образ перед зрителями в реальном времени. Театральное искусство появилось тысячелетия назад, будь то первобытные ритуалы, мистерии Древнего Египта или представления Античной Греции и Древнего Рима, ставшие колыбелью европейского театра.

Это искусство, которое, проходя сквозь века, не просто сохраняет свою актуальность, но и постоянно трансформируется, отвечая на вызовы времени. От древних ритуалов до постановок модернизма он остаётся живым диалогом между эпохами, культурами и людьми. Его сила — в способности переосмыслять традиции, не теряя при этом сути: живого взаимодействия между сценой и зрителем.

Если раньше люди вручную считали кассу, писали программы и заполняли зрительские журналы, то сегодня автоматизированные системы берут на себя рутину, освобождая время и сохраняя деньги.

Целью данного курсового проекта станет обеспечение удобства организаторам театральных событий в отслеживании билетов, настройке программ постановок, отслеживанию текущих, запланированных и прошедших событий, ведение учёта персонала на актёрских или других позициях постановкой.

В задачи данного курсового входят разработка и тестирование системы, обеспечивающая такое удобство, обеспечение и настройка связи базы данных и приложения.

# Общая часть

## Анализ предметной области

Сперва важно обозначить, что данная система предназначена для внутреннего использования среди работников компании, а не для широкого круга пользователей. Поэтому яркая, почти рекламная составляющая пользовательского интерфейса будет опущена и в первую очередь соблюдена функциональная пригодность.

Главными сущностями театрального учреждения являются:

1. Билеты;

* Имеют владельцев;

1. Программы;

* Основаны на каком-либо сюжете;
* Показываются на определённых сценах;
* Имеют определённый актёрский состав.

Как и описано выше программы это и есть по сути события, они основаны на каких-либо сценариях, которые в свою определённый жанр. У каждого события должно быть описания для того, чтоб получить более подробное представление о программе. Программа показывается в определённое время определённой даты. На каждую программу назначается определённый актёрский состав, где у каждого актёра может быть одна или несколько ролей. Каждая программа показывается на определённой сцене, чаще всего сцены имеют разные адреса.

## Анализ рынка существующих решений

Несмотря на широкий спектр учреждений, сталкивающихся с данными трудностями организации событий, каких-либо узконаправленных решений не было найдено. Зачастую для данных задач организовывают собственные информационные системы или и вовсе хранят эти данные в электронных таблицах. Поэтому будут рассмотрены альтернативы между:

* Разрабатываемая информационная система «Театральные кассы» с собственным приложением и базой данных на сервере;
* Информационная система, созданная на базе Microsoft Access;
* Хранение данных в файлах электронных таблиц для Microsoft Excel;

Будем рассматривать по очереди по каждому из критериев.

Простота создания. Самым простым в реализации станет создание и организация электронной таблицы в Microsoft Excel, ввиду универсальности приложения по работе с таблицами. Следующим в порядке возрастания сложности станет создание информационной системы на базе Microsoft Access, поскольку создателю такой информационной системы потребуются специальные знания и навыки работы с средствами Microsoft Access, понимание логики построения запросов, умения настройки форм, отчётов. И самым сложным в создании является разработка информационной системы «Театральные кассы» с собственным приложением и базой данных на сервере, поскольку требуют специализированного программиста, владеющего набором технологий по работе с базами данных, а также разработке приложений.

Удобство пользования. Самым удобным решением должна быть информационная система «Театральные кассы» с приложением и базой данных на сервере, ввиду широких возможностей пользовательского интерфейса и возможности применять бизнес-логику почти любой сложности в приложении. Вторым по убыванию удобства пользования становится информационная система, созданная на базе Microsoft Access, поскольку встроенные средства проектирования позволяют настроить весьма широкий спектр решаемых задач, однако всё же ограничены эффективности организации сложных информационных и бизнес-логических процессов. Последним в удобстве пользования оказываются электронные таблицы Microsoft Excel, поскольку почти не имеют средств настройки качественного пользовательского интерфейса и полностью опираются на заранее спроектированный, то есть базовый.

Удобство сопровождения. В долгосрочной перспективе грамотно спроектированную информационную систему «Театральны кассы» значительно проще расширять и сопровождать, поскольку исходный код, а также сервер с базой данных находятся в распоряжении владельцев. Базу данных в Microsoft Access значительно труднее сопровождать и масштабировать, ввиду ограничений средства самого MS Access, а также отсутствия полного исходного кода в расположении у владельцев. Самой трудно масштабируемым и трудно сопровождаемым решением станет ведение электронных таблиц Microsoft Excel, поскольку изменения логики или некоторых данных зачастую будут требовать почти полного пересоздания имеющейся системы с большим объёмом ручного труда по копированию и переносу данных.

Таблица 1 содержит сводку по проведённом анализу возможных решений.

Таблица 1 – Сводная таблица оценок решений по выбранным критериям

| Критерий | Разрабатываемая информационная система «Театральные кассы» с собственным приложением и базой данных на сервере, оценка в баллах | Информационная система, созданная на базе Microsoft Access, оценка в баллах | Хранение данных в файлах электронных таблиц для Microsoft Excel, оценка в баллах |
| --- | --- | --- | --- |
| Простота создания | 1 | 2 | 3 |
| Удобство пользования | 3 | 2 | 1 |
| Удобство сопровождения | 3 | 2 | 1 |
| Итоги | 7 | 6 | 5 |

По итогам сводки можно сделать вывод о преимуществе именно разработки собственной информационной системы из приложения и сервера на БД перед общим универсальными решениями.

## Сравнение технологий

В первую очередь следовало выбрать язык программирования и платформу, на которой будет происходить разработка, а также СУБД.

Среди возможных вариантов языка программирования были C++, C# или же python.

Язык C++ обладает рядом ключевых преимуществ, делающих его одним из наиболее востребованных языков для разработки производительных и сложных приложений. Благодаря поддержке как процедурного, так и объектно-ориентированного программирования, C++ обеспечивает гибкость в проектировании архитектуры, позволяя выбирать оптимальный подход для решения конкретных задач. Высокая производительность программ, написанных на C++, достигается за счёт компиляции в машинный код, что особенно критично для ресурсоёмких приложений, таких как игры, системы реального времени или высоконагруженные сервисы.

C# является современным и универсальным языком программирования, который сочетает в себе высокую производительность, простоту разработки и широкие возможности, что делает его отличным выбором для создания разнообразных приложений. Одним из ключевых преимуществ C# является его тесная интеграция с платформой .NET, которая предоставляет богатую стандартную библиотеку, включающую средства для работы с сетью, базами данных, многопоточностью, шифрованием и многим другим, что значительно ускоряет процесс разработки. Благодаря строгой типизации и автоматическому управлению памятью (сборка мусора) C# снижает риск ошибок, связанных с утечками памяти и неверным использованием указателей, что особенно важно в крупных проектах. Язык поддерживает множество парадигм программирования, включая объектно-ориентированный, функциональный и асинхронный подходы, что позволяет разработчикам выбирать наиболее подходящий стиль для решения конкретных задач.[1]

Python является одним из самых популярных и востребованных языков программирования благодаря своей универсальности и большим банком пользовательских библиотек, что делает его отличным выбором для разработки самых разных приложений. Главным преимуществом Python является его лаконичный и интуитивно понятный синтаксис, который позволяет писать код быстро. Python поддерживает множество парадигм программирования, включая объектно-ориентированный, функциональный и процедурный подходы, что делает его гибким инструментом для решения разнообразных задач. Кроссплатформенность Python позволяет разрабатывать приложения для Windows, macOS и Linux без необходимости существенных изменений в коде.

Приложение для информационной системы «Театральные кассы» может быть написано на любом из языков, однако выбор был сделан в пользу C#. C++ высокопроизводительный и очень гибкий инструмент для решения широкого спектра задач, однако высокий необходимы уровень теоретического и практического опыта, а также усилия по отладке продукта будут не целесообразны в сравнении с Python или C#. На Python в свою очередь производство программного кода происходило бы значительно эффективней за количество вложенного труда, однако показатели производительности значительно могут убавиться в процессе разработки. C# в свою очередь является промежуточным вариантом, обладая достаточно высокой эффективностью производства кода за вложенный труд и производительность. К тому же C# тесно интегрирован с платформой .Net.

Однако C# имеет две основные платформы, более новую .Net и более старую .Net Framework. .NET (ранее известный как .NET Core) и .NET Framework — это две платформы от Microsoft для разработки приложений, но они существенно различаются по архитектуре, возможностям и сценариям использования. .NET является более современной, кроссплатформенной и высокопроизводительной эволюцией .NET Framework, которая сохраняет совместимость с его ключевыми технологиями, но устраняет многие ограничения.

Главное отличие заключается в том, что .NET Framework работает только на Windows и тесно интегрирован с операционной системой, что делает его идеальным для приложений, которые должны исправно запускаться на устаревших (по современным меркам) операционным системах Windows (7, 8), WPF, Windows Forms и старых версий ASP.NET. Однако его развитие практически остановлено, а новые функции добавляются только в .NET. В свою очередь, .NET — это модульная, открытая и кроссплатформенная платформа (поддерживает Windows, Linux и macOS), что делает её предпочтительной для современных облачных, контейнеризированных и микросервисных решений. Производительность .NET значительно выше благодаря оптимизированной runtime-среде (CoreCLR), более эффективной работе с памятью. Кроме того, .NET активно развивается, получая ежегодные крупные обновления (например, .NET 5, 6, 7, 8), в то время как .NET Framework прекратил своё развитие с версии 4.8.x без существенных нововведений. С точки зрения экосистемы, .NET поддерживает современные стандарты разработки, включая C# 10/11, F#, Blazor, MAUI и ASP.NET Core (более быстрый и гибкий аналог ASP.NET MVC). В то же время .NET Framework остаётся привязан к старым технологиям, таким как Web Forms, WCF (Windows Communication Foundation; в .NET заменён на gRPC и REST API) и Remoting. Ещё одно важное преимущество .NET — это его модульность и гибкость развертывания. В отличие от .NET Framework, который требует установки в систему, .NET позволяет публиковать приложения как самостоятельные (self-contained).

Выбор СУБД производился из следующих 3-ёх вариантов: MongoDB, MySQL и Microsoft SQL server.

Хотя MySQL и Microsoft SQL Server являются реляционными СУБД и во многом схожи по базовому функционалу (поддержка SQL, транзакций, индексов), ключевое отличие заключается в интеграции систем и инструментах управления, где SQL Server с SQL Server Management Studio предлагает более целостное и удобное решение для поставленных задач. Обе СУБД поддерживают стандартные операции — от проектирования схемы до оптимизации запросов, но SQL Server интегрирован в единую платформу Microsoft. SSMS здесь выступает как мощный централизованный интерфейс, предоставляющий уникальные возможности: визуализацию планов выполнения запросов с детализацией стоимости операций, встроенные средства профилирования (SQL Profiler), автоматическую настройку индексов (Database Engine Tuning Advisor) и удобное управление безопасностью (интеграция с Active Directory). Для MySQL аналогичный функционал требует использования разрозненных инструментов — например, MySQL Workbench для проектирования, pt-query-digest для анализа запросов и сторонних решений для мониторинга, что усложняет поддержку.

MongoDB — это документоориентированная NoSQL-СУБД, обладающая рядом уникальных преимуществ, но имеющая и определённые ограничения по сравнению с реляционными системами. Главное достоинство MongoDB — гибкость модели данных, позволяющая хранить информацию в виде JSON-подобных документов (BSON) с динамической схемой, что идеально подходит для проектов с быстро меняющимися требованиями или неструктурированными данными (например, контент-платформы, каталоги товаров). Высокое потребление ресурсов (особенно памяти из-за работы с документами целиком) и отсутствие оптимизированных JOIN-операций усложняют выполнение сложных аналитических запросов, где реляционные СУБД (например, SQL server) демонстрируют лучшую производительность. Безопасность в MongoDB, несмотря на поддержку аутентификации, шифрования и ролевой модели, исторически считалась менее проработанной, чем в SQL-решениях.

MS SQL Server превосходит MongoDB в производительности сложных JOIN-операций, агрегации данных и выполнении аналитических запросов благодаря оптимизированному движку хранения, а главное преимущество в виде способности быстро адаптироваться под часто трансформирующуюся структуру хранимых данных для нас несущественно.

Таким образом выбранная СУБД стала однозначно SQL server.

Для разработки приложения выбрана технология Windows Forms, а для обеспечения связи базы данных и приложения выбран официальный Nuget пакет от Microsoft – Entity Framework для работы с SQL server. [2] Обширные встроенные средства реализации интерфейса с помощью конструктора форм обеспечит целесообразное использование трудовых и временных ресурсов при разработке пользовательского интерфейса, а использование Entity Framework позволит работать с объектами базы данных аналогично сущностям Объектно-ориентированного программирования, что так же обеспечит повышенную простоту сопровождения и тестирования проекта.

# Специальная часть

Назначение разработки

Основное назначение программного продукта: управление театральными событиями.

Эксплуатационное назначение программного продукта: программа не предназначена для широкого круга пользователей, без ограничения по возрасту (на пользование самой программы, но на спектакли ограничения могут отличаться). Система подразумевает последующее подключение платёжных систем.

## Требования к программе или программному изделию

* 1. Требования к функциональным характеристикам:
     + программа должна обладать функционалом бронирования (покупки) билета, отмены покупки билета (возврата);
     + программа должна иметь поисковый функционал с возможностью, фильтрации по: участвующим актёрам, дате показа, названию события;
     + программа должна иметь возможность добавления, удаления и изменения самого события, а также отображать все поисковые атрибуты в каждом событии;
     + программа должна обладать функционалом добавления новых сотрудников в БД,
     + программа должна обладать функционалом назначения сотрудников на роли в спектакле, снятие с роли и возможность замещения какого-либо из актёров;
     + программа также предполагает разделение возможностей пользователей по ролям. Ожидаемые роли: Администратор, менеджер персонала, организатор событий, кассир. Администратору будут доступны все возможности. Менеджеру персонала доступны будут таблицы и функции, связанные с наймом и увольнением работников (занесение и исключение из таблиц БД), а также редактирование таких таблиц. Организатору событий доступен функционал назначения и снятие сотрудников с ролей, создание и изменение событий, добавление, удаление изменение данных о сценариях в БД. Кассиру доступны создание, изменение данных о билетах, возможность регистрации и изменения данных о пользователях, доступно изменение промокодов;
     + для приложения предполагается работа с потенциальным внешним сервисом касс, для этого в информации о билете будет указан код транзакции оплаты. То есть информация о платёжных операциях должна храниться в независимом сервисе;
  2. Требования к надежности:
     + использование лицензированного программного обеспечения;
     + проверка программы на наличие вирусов;
     + организация бесперебойного питания.

Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

* + - процессор Intel или другой совместимый;
    - объем свободной оперативной памяти ~10 Mб;
    - объем необходимой памяти на жестком диске ~300Мб;
    - стандартный VGA-монитор или совместимый;
    - стандартная клавиатура;
    - манипулятор «мышь».

Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows 10 или совместимой. Язык интерфейса – русский.

* 1. Требования к маркировке и упаковке

Программа должна поставляться в виде проекта, исполняемого (еxе) файла, установщика и документации.

* 1. Требования к транспортировке и хранению

Программа распространяется в электронном виде. Требования к транспортировке и хранению не предъявляются.

* 1. Специальные требования

Теоретическая часть включает подробное описание работы с приложением, разработанные на этапе проектирования программной системы. Практическая часть включает разработку и реализацию программных модулей программного продукта с использованием среды программирования.

## Проектирование

Проектирование структуры Базы данных происходило, исходя из требований. А для проектирования базы данных был выбран подход создания схем: UML диаграмма, концептуальная модель, даталогическая модель, физическая модели.[3] На рисунке 1 представлена диаграмма прецендентов использования.

* + 1. Диаграмма прецедентов использования

Рисунок 1 представляет из себя UML диаграмму, отображающая основной функционал доступных ролей и системы в целом.

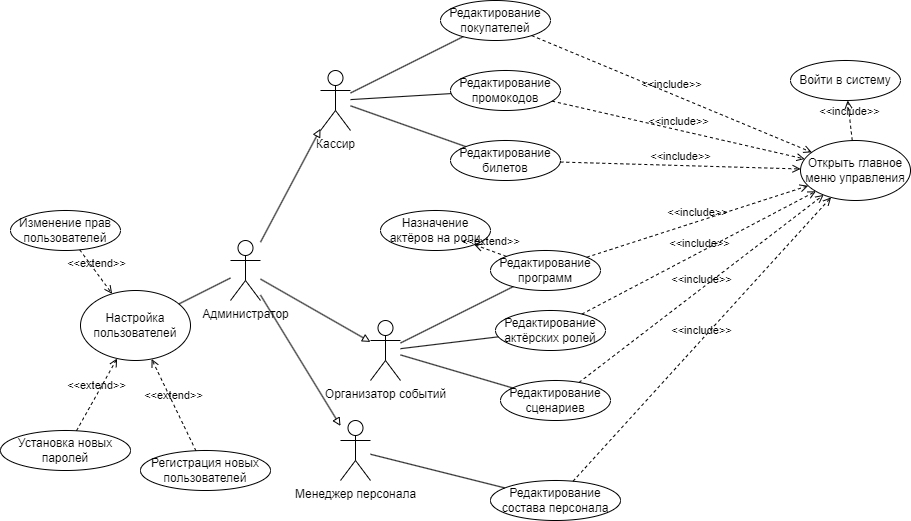


Рисунок 1 – UML диаграмма

* + 1. Концептуальная модель

Рисунок 2 представляет из себя концептуальную модель, на которой изображены отношения и их атрибуты (поля), без конкретных типов данных.

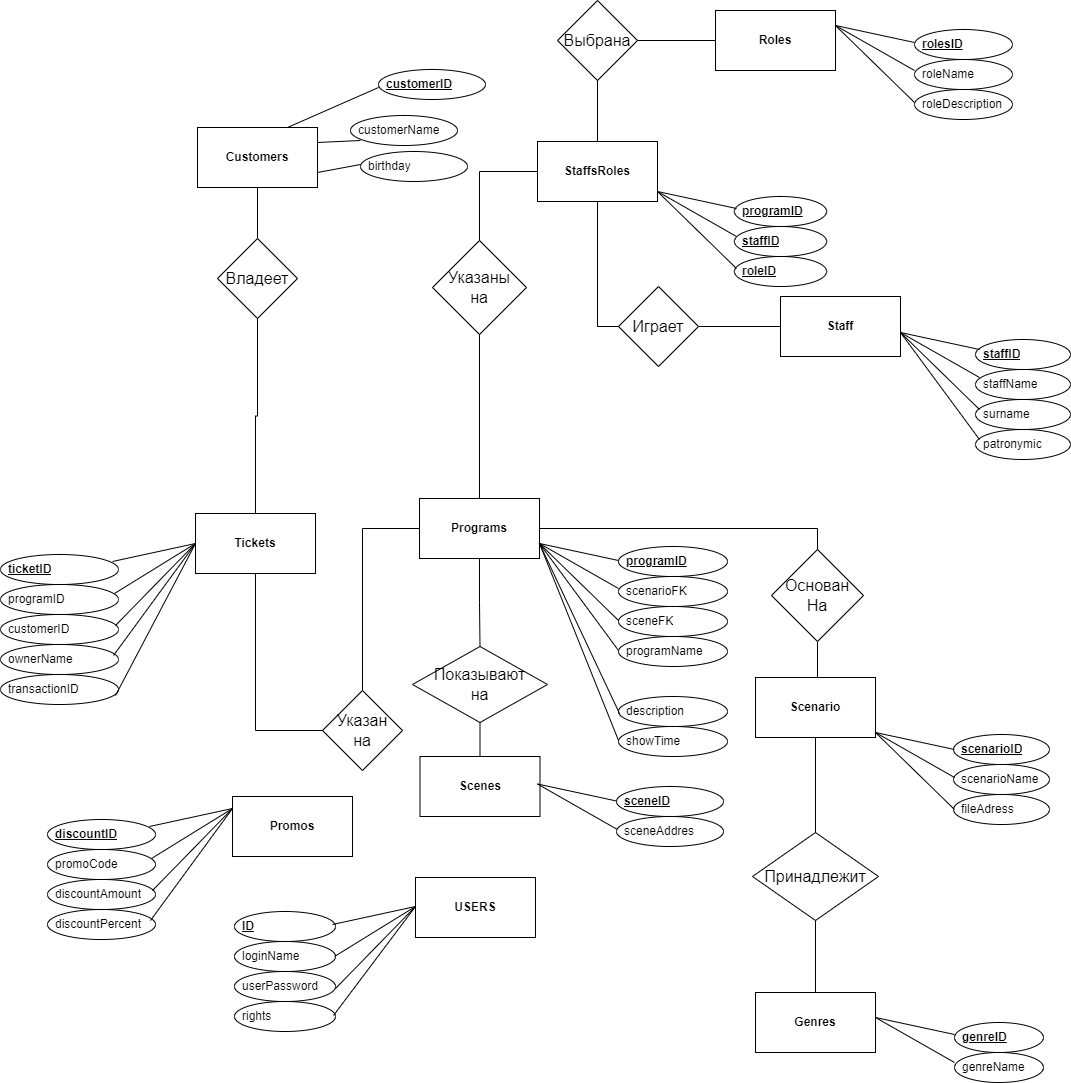


Рисунок 2 – Концептуальная модель

* + 1. Даталогическая модель

Рисунок 3 представляет из себя даталогическую модель, на которой представлены связи между отношениями.

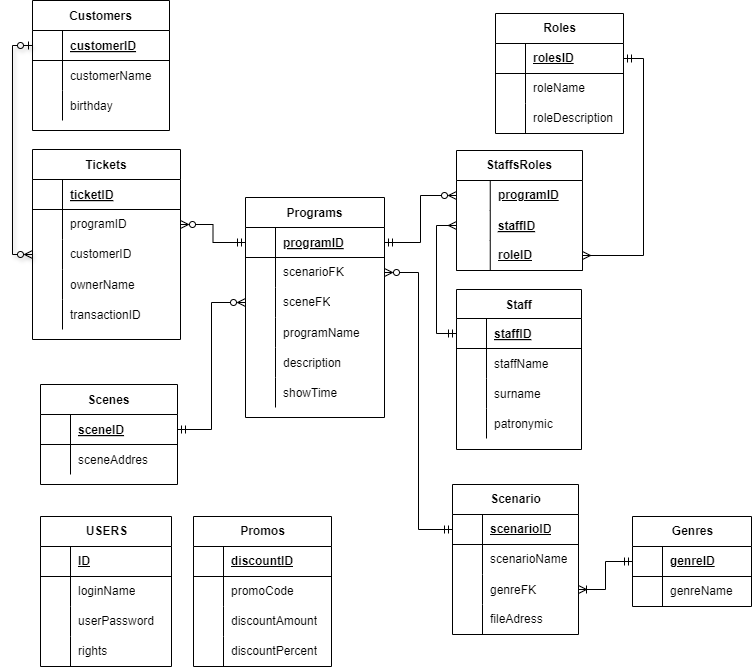


Рисунок 3 – Даталогическая модель

* + 1. Физическая модель

Рисунок 4 представляет из себя физическую модель базы данных, на которой указаны связи отношений, первичные, внешние ключи, а также типы данных.



Рисунок 4 – Физическая модель

## Описание алгоритма

* + 1. Формы редактирования данных таблицы

Большинством форм информационной системы «Театральные кассы» являются формы редактирования таблиц. Поэтому логично подробно рассмотреть их общие алгоритмы.

На всех подобных формах присутствует компонент DataGridView (рисунок 5), который позволяет просматривать данные в виде таблицы. C DataGridView удобно работать, поскольку он имеет свойство DataSource, что позволяет указать источник данных, например список (List) объектов с Базы данных, а DataGridView.[4] автоматически преобразует свойства объектов в подходящие колонки.

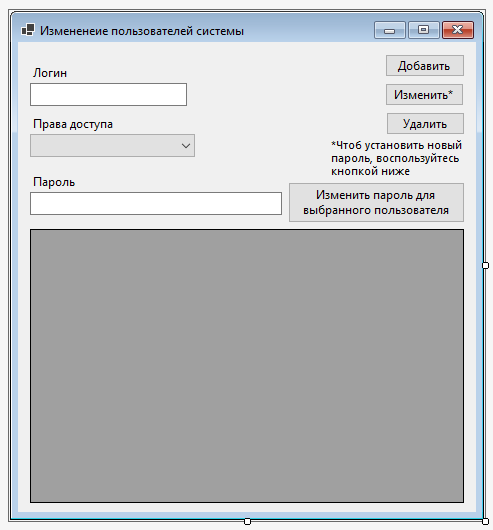


Рисунок 5 – Типичная форма изменения данных таблицы, крупный серый прямоугольник — это компонент DataGridView.

Кроме того, DataGridView позволяет менять заголовки столбцов и скрывать отдельны столбцы, тем самым при разработке обеспечивается изоляция пользователей от технических данных, например коды записей, хэши паролей, а также замена программно записанных названий полей на пользовательские, в частности так как код пишется преимущественно на английском языке, за счёт изменения заголовков происходит замена английских программных названий полей на русские, адаптированные для пользователей.

Почти на всех формах изменения данных таблицы присутствуют три основные кнопки: «Добавить», «Изменить», «Удалить». Функционал каждой из кнопок реализован схоже.

При добавлении поочерёдно проверяются вводимые поля, в случае ошибки выводится сообщение пользователю, а в случае успеха происходит создание объекта, занесение его в таблицу средствами Entity Framework, а затем сохранение изменений базы данных.

При изменении проверяются вводимые поля, в случае ошибки выводится сообщение пользователю, а в случае успеха создаётся объект. Объект создаётся в соответствии с выбранной строкой в компоненте DataGridView за счёт свойства строк DataBoundItem. средствами Entity Framework этот объект заменяется ссылкой на эту же запись с базы данных, затем у объекта меняются свойства в соответствии с введёнными данных, и в конце происходит сохранение изменений базы данных.

При изменении проверяются вводимые поля, в случае ошибки выводится сообщение пользователю, а в случае успеха создаётся объект. Объект создаётся в соответствии с выбранной строкой в компоненте DataGridView за счёт свойства строк DataBoundItem. средствами Entity Framework этот объект заменяется ссылкой на эту же запись с базы данных, затем у объекта меняются свойства в соответствии с введёнными данных, и в конце происходит сохранение изменений базы данных.

При удалении создаётся объект в соответствии с выбранной строкой в компоненте DataGridView за счёт свойства строк DataBoundItem. средствами Entity Framework этот объект заменяется ссылкой на эту же запись с базы данных, затем объект удаляется и происходит сохранение изменений базы данных.

Кроме того, при изменении выбора выделенной строки в компоненте DataGridView соответствующие поля ввода заполняются данными выбранного объекта выделенной строки, так же за счёт свойства DataBoundItem. При это при каждой такой смене выделения обращения к Базе данных не происходит, и средства Entity Framework не задействуются, все данные берутся локально из DataGridView.

* + 1. Защита данных

Пользователи перед началом работы в системе должны войти в неё, используя логин и пароль. Для обеспечения сохранности паролей они хранятся в зашифрованном хэш функциями виде. Для этой задачи был написан сервис (программный модуль) Password Service. На данный момент актуальным алгоритмов хэширования является Pbkdf2, с количеством итераций около 600тыс.[5] Для его работы также стоит использовать «соль». Солью является какой-то случайный набор данных с хорошей энтропией, при «подмешивании» которой к паролю, в алгоритме получается другой хэш. В C# этот алгоритм можно вызвать через Rfc2898DeriveBytes.Pbkdf2(). В Базе данных будут храниться «соль» и хэш пароля вместе.

Одна из техник защит, которая применяется на этапе валидации паролей – это константное время проверки равности полученного и ожидаемого хэша. Поскольку обычный алгоритм сравнения массива байтов останавливается на первом несовпадении байтов, злоумышленник может начать подбирать по времени ответа номера совпавших байтов. Для устранения такой уязвимости следует использовать метод сравнения байт массивов с константным временем сравнения. В C# это осуществляется через CryptographicOperations.FixedTimeEquals().

Помимо всего, связанного непосредственно защитами, при работе над этим сервисом была использована технология спанов (Span). Главное преимущество спана перед обычным массивом, это то, что новых выделений памяти и копирования содержимого не происходит, что экономит операции и оперативную память.

* + 1. Разграничение прав

Для разграничений прав был также написан отдельный сервис RightsService, отвечающий за подтверждение доступности пользователю той или иной формы (функционала). В составе класса находиться Enum (перечисление) всех доступных ролей системы, это:

* Без прав (например, заблокирован),
* Администратор,
* Кассир,
* Организатор событий,
* Менеджер персонала.

Главная функция этого сервиса принимает два аргумента: объект пользователя и объект запускаемой формы. В теле этой функции за счёт технических средств языка C#, а именно – конструкции switch case написаны заранее определённые доступные классы запускаемых форм (функционала).

Применяется этот сервис для того, чтоб отключать кнопки, отвечающие за недопустимы для текущего пользователя функционал.

## Описание программы

При самом первом запуске программы автоматически создаётся база данных, а также таблицы средствами Entity Framework. В таблицу пользователей автоматически добавляется первый пользователь admin с паролем admin123 и ролью администратора. Это необходимо для создания всех остальных пользователей, а также крайне настоятельно рекомендуется при первом запуске изменить пароль первого администратора на более защищённый. Рисунок 6 представляет собой форму, в которой заполняются два поля «Логин» и «Пароль», а нажатие кнопки «Вход» запускает процесс авторизации и входа в система. Кнопка «Выход» в свою очередь закрывает окно.

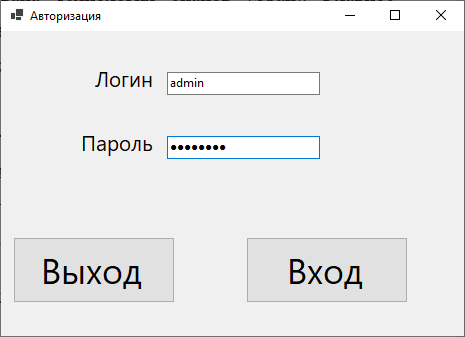


Рисунок 6 – форма авторизации

После входа в систему открывается основное окно выбора редактируемых таблиц. Рисунок 7 представляет собой именно такое окно выбора.

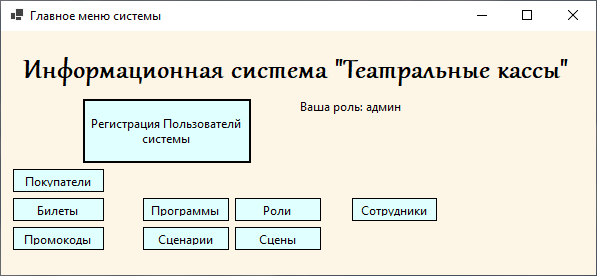


Рисунок 7 – форма выбора изменяемой таблицы от лица администратора

Данное окно специфично для каждой роли. Так обеспечивается разделение по ролям (правам), соответственно доступные кнопки по ролям:

* 1. Администратор,
* Регистрация Пользователей системы,
* Покупатели,
* Билеты,
* Промокоды,
* Программы,
* Сценарии,
* Роли,
* Сцены,
* Сотрудники;
  1. Кассир,
* Покупатели,
* Билеты,
* Промокоды;
  1. Организатор событий,
* Программы,
* Роли,
* Сценарии,
* Сцены;
  1. Менеджер персонала,
* Сотрудники;

Пример того, как выглядит форма от лица кассира представлена на рисунке 8.

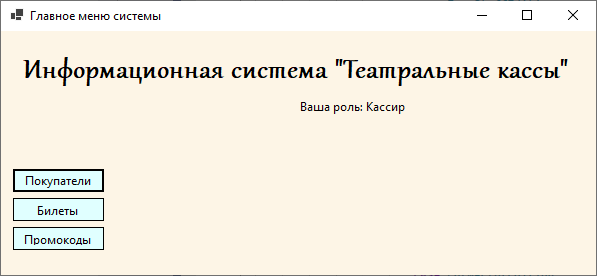


Рисунок 8 – форма выбора изменяемой таблицы от лица кассира

А основные задачи кассира выполняются на форме, изображенной на рисунке 9.

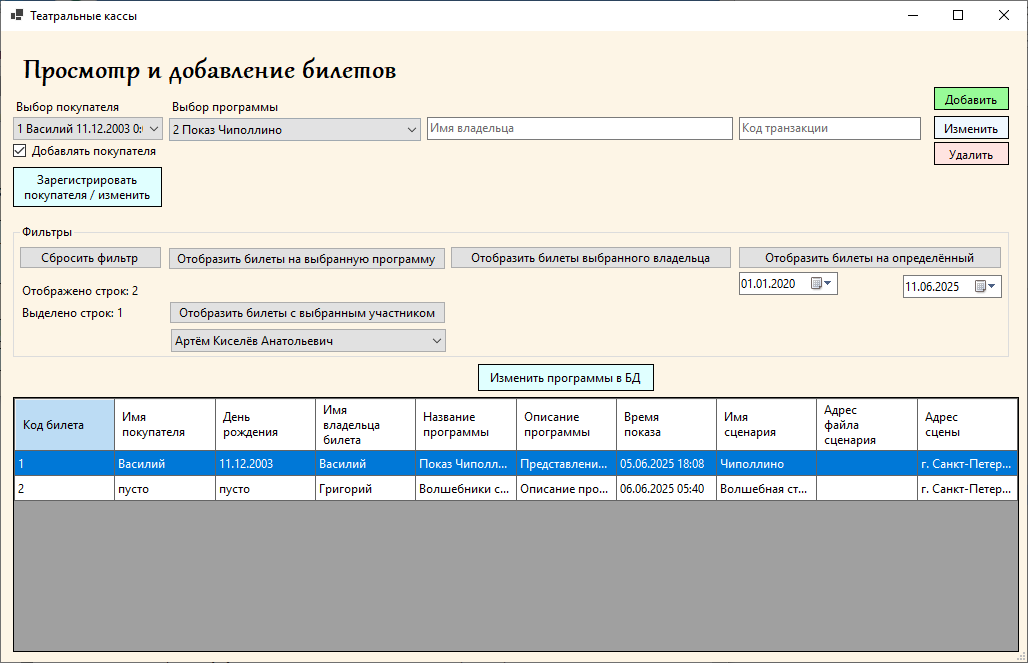


Рисунок 9 – основная форма кассира, форма билетов

Для организатора событий основные задачи вынесены на форму, показанную на рисунке 10.

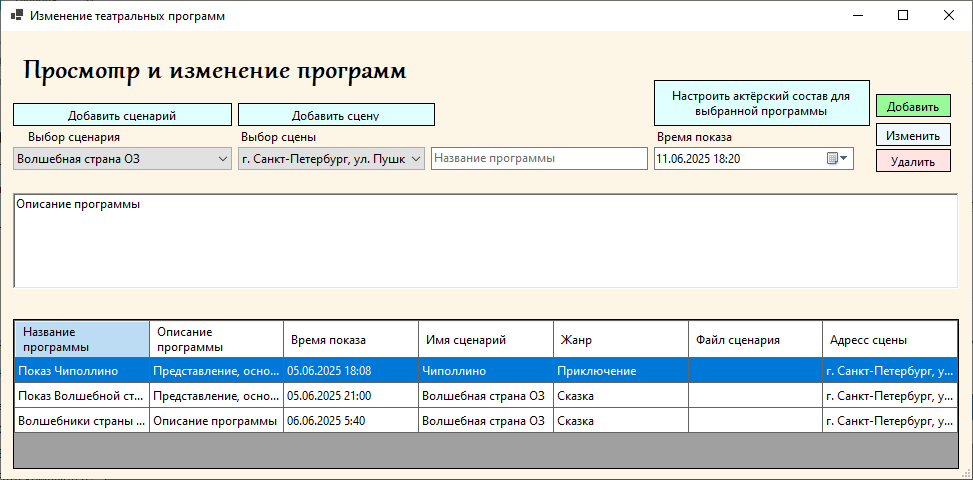


Рисунок 10 – основная форма организатора событий, форма программ

Менеджер персонала в свою очередь в системе только руководит персоналом через форму, показанную на рисунке 11.

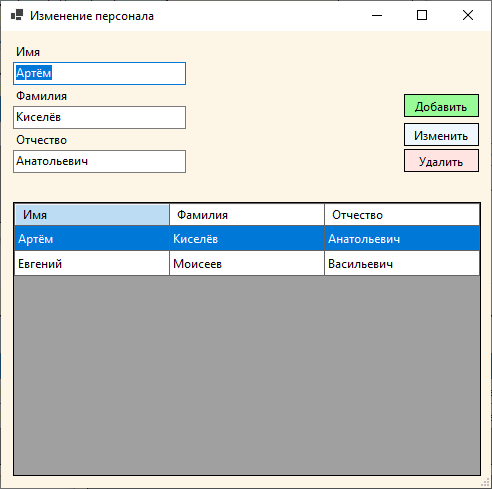


Рисунок 11 – основная форма менеджера персонала, форма персонала

Эксклюзивной для администратора формой можно считать форму пользователей системы, поскольку администратор имеет исключительное право доступа и работы с этой формой, которая отвечает за перечень пользователей, их роли, логины и пароли. Форма изменения пользователей системы показана на рисунке 12.

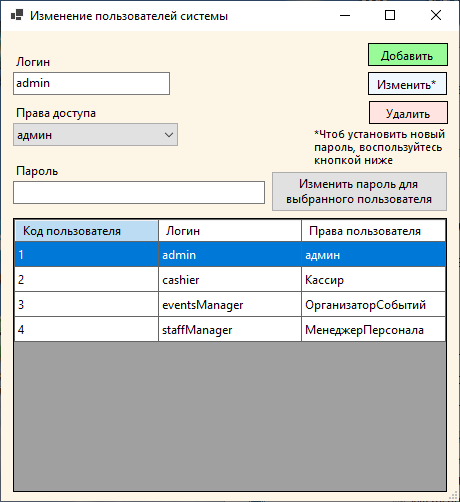


Рисунок 12 – форма, предназначенная исключительно для администратора, форма пользователей системы.

## Инструкция пользователя

При первом запуске системы по умолчанию доступен только один пользователь – логин: admin, пароль: admin123. Введите на странице авторизации admin в поле логин и admin123 в поле пароль. Нажмите кнопку «Вход». Откроется форма выбора изменяемой таблицы. Откройте таблицу пользователей, нажатием кнопки «Регистрация пользователей системы». Откроется форма «Изменение пользователей системы» (рисунок 9), выберите пользователя admin, нажатием на строку с его логином в таблице снизу и введите новый надёжный пароль в поле пароль, а затем нажмите кнопку «Изменить пароль для выбранного пользователя». Таким образом у пользователя admin будет новый пароль, который был установлен в поле пароль. Далее следует добавить других пользователей системы, для этого необходимо вводить логины в поле «логин», пароли в поле Пароль и выбирать права доступа из выпадающего списка «Права доступа» для каждого из требуемых под нужды предприятия пользователя. Все введённые логины должны быть уникальны.

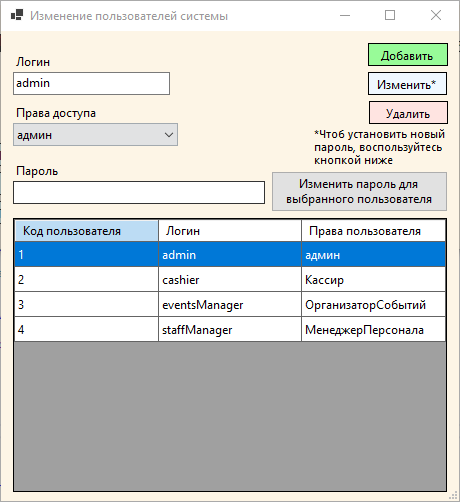


Рисунок 13 – форма регистрации новых пользователей.

После того, как все требуемые пользователи были добавлены в систему следует начать заполнение самой системы данными. Поскольку одни данные зависят от других следует заполнять таблицы в следующем порядке:

* 1. Сотрудники, Сцены, Роли, Покупатели, Промокоды, Жанры (доступно изменение из формы сценариев);
  2. Сценарии;
  3. Программы;
  4. Билеты, Актёрский состав программ (доступно изменение, если на форме изменения программ открыть «изменение актёрского состава для выбранной программы»);

## Тестирование

Тестирование информационной системы осуществляется через протокол испытаний, представленный ниже. (Таблица 2)

Таблица 2 – Протокол испытаний информационной системы системы

|  |  |  |  |
| --- | --- | --- | --- |
| Номер и название испытания | Осуществляемые в процессе испытания действия | Ожидаемые результаты | Результат испытания |
| 1. Способность программы обеспечить контроль ввода пустых полей. | Открыть форму изменения сценария.  Если поле название не пусто, то убрать текст из поля.  Нажать кнопку «добавить» | Новый сценарий не будет добавлен, а пользователю будет выведено сообщение об ошибки:  «Название сценария пусто. Введите другое название» | Пройден.  См. рисунок 14 |
| 2. Способность программы не допустить блокировку системы из-за удаления всех пользователей. | Войти как администратор (например, под логином admin) и открыть форму изменения пользователей.  Выбрать пользователя, под которым осуществлён вход в систему (например, admin). | Текущий пользователь не будет удалён, пользователю будет выведено сообщение об ошибки:  «Вы не можете удалить текущий аккаунт", "Ошибка удаления текущего пользователя» | Пройден.  См. рисунок 15 |
| 3. Способность программы обеспечить целостность данных каскадным удалением. | Создать сценарий «Сказки братьев Гримм»  Создать программу «Показ сказок братьев Гримм».  Удалить сценарий «Сказки братьев Гримм». | Программа «Показ сказок братьев Гримм» будет каскадно удалена. | Пройден См. рисунок 16, рисунок 17 |

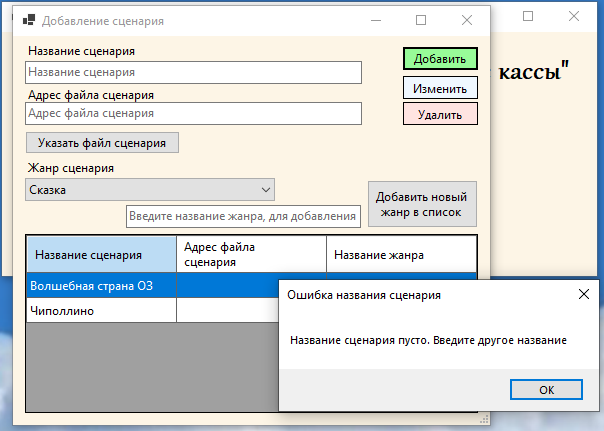


Рисунок 14 – Обеспечение ввода непустых данных

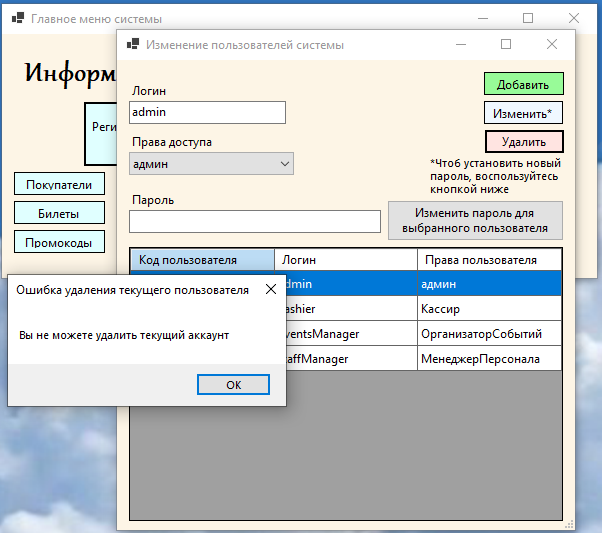


Рисунок 15 – сообщение об ошибке удаления текущего аккаунта

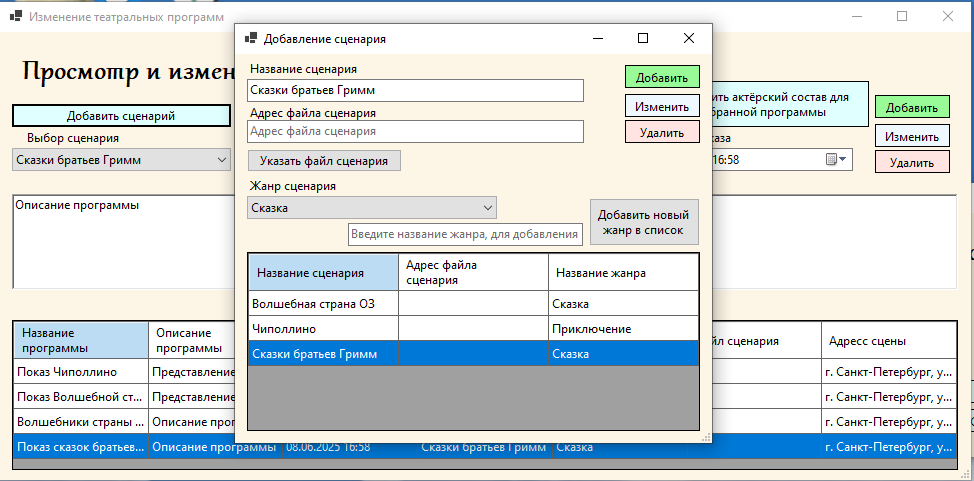


Рисунок 16 – Созданные программа и сценарий связанные со сказками братьев Гримм

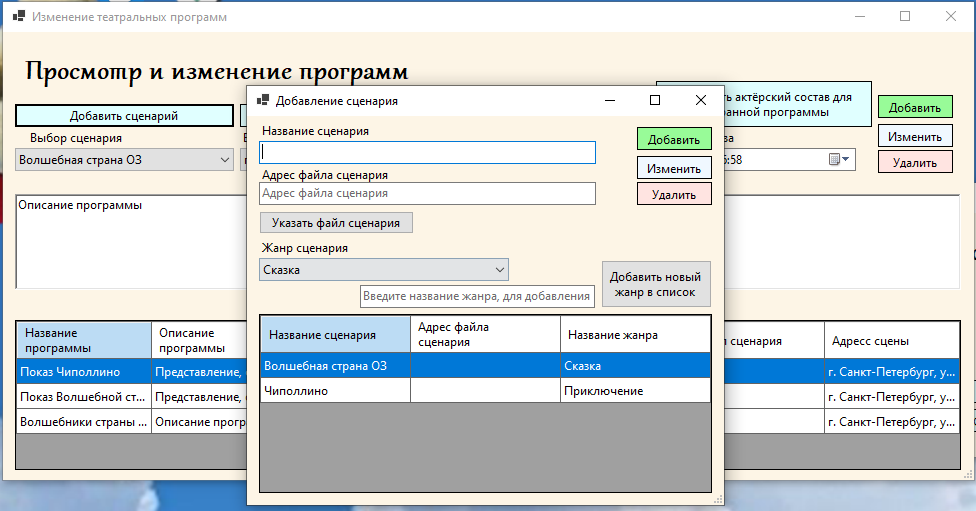


Рисунок 17 – Каскадное удаление программы при удалении сценария

Ручное тестирование системы пройдено успешно и в полном объёмы, что означает, что должное качество системы обеспечено.

Заключение

В ходе данного курсового проекта были разработаны база данных и приложение для работы с этой базой данных, вместе образующие информационную систему «Театральные кассы». Главной технологией, которая была освоена в процессе работы над информационной системой, стала управление базой данных с помощью средств устанавливаемого пакета Entity Framework.

Использование интегрированной среды разработки Visual Studio 2022, а также средств проектирование Draw.Io способствовали повышению качества и скорости разработки архитектуры и алгоритмов приложения.

Через Visual Studio 2022 так же происходило обращение к SQL серверу во время отладки и для написания SQL скриптов, было использовано это же ПО.

Поставленные цели и задачи по обеспечению пользователей контролем над информационной системой «Театральные Кассы» были достигнуты и должное требуемое качество приложения было обеспечено.

Список использованных источников

1. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .Net Framework 4.5 на языке C#. 4-е изд. – СПБ.: Питер, 2022 – 896 с.: ил. (Серия «Мастер-класс»)
2. Центр документации Entity Framework [электронный ресурс] / Режим доступа: https://learn.microsoft.com/ru-ru/ef/, свободный. (дата обращения 01.06.2025)
3. С. А. Нестеров Базы данных : учебник и практикум для вузов / С. А. Нестеров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023 — 258 с. — (Высшее образование). — Текст : непосредственный.
4. Элемент управления DataGridView (Windows Forms) [электронный ресурс] / Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/datagridview-control-windows-forms, свободный. (дата обращения 16.04.2025)
5. Р 1323565.1.040–2022. Информационная технология КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ Парольная защита ключевой информации. – ВЗАМЕН P 50.1.111—2016; введ. 20-10-2022 г. – Приказом Федерального агентства по техническому регулированию и метрологии.; Москва: Российский институт стандартизации 2022. – с.5

ПРИЛОЖЕНИЕ А

(обязательное)

**Листинг SQL-скрипта**

CREATE TABLE USERS(

id int IDENTITY(1,1),

loginName nvarchar(MAX) NOT NULL,

userPassword varBinary(MAX) NOT NULL,

rights nvarchar(MAX) NOT NULL

);

CREATE TABLE Genres(

genreID INT IDENTITY(1,1) PRIMARY KEY,

genreName nvarchar(Max),

);

CREATE TABLE Scenario(

scenarioID INT IDENTITY(1,1) PRIMARY KEY,

genreFK int FOREIGN KEY(genreFK) REFERENCES Genres(genreID),

scenarioName nvarchar(MAX) NOT NULL,

fileAddress nvarchar(MAX)

);

CREATE TABLE Scenes(

sceneID INT IDENTITY(1,1) PRIMARY KEY,

sceneAddress nvarchar(MAX) NOT NULL

);

CREATE TABLE Roles(

roleID INT IDENTITY(1,1) PRIMARY KEY,

roleName nvarchar(MAX) NOT NULL,

roleDescription nvarchar(MAX) NOT NULL

)

CREATE TABLE Customers(

customerID INT IDENTITY(1,1) PRIMARY KEY,

customerName nvarchar(MAX) NOT NULL,

birthday date

);

CREATE TABLE Staff(

staffID INT IDENTITY(1,1) PRIMARY KEY,

staffName nvarchar(MAX) NOT NULL,

surname nvarchar(MAX) NOT NULL,

patronymic nvarchar(MAX)

);

CREATE TABLE Promos(

promoID INT IDENTITY(1,1) PRIMARY KEY,

promoCode nvarchar(256) NOT NULL,

discountAmount Money,

discountPercent Money

);

CREATE TABLE Programs(

programID INT IDENTITY(1,1) PRIMARY KEY,

scenarioID INT NOT NULL FOREIGN KEY(scenarioID) REFERENCES Scenario(scenarioID)

ON UPDATE CASCADE ON DELETE CASCADE,

sceneID INT NOT NULL FOREIGN KEY(sceneID) REFERENCES Scenes(sceneID)

ON UPDATE CASCADE ON DELETE CASCADE,

programName nvarchar(256) NOT NULL,

description nvarchar(256),

showTime dateTime NOT NULL

)

CREATE TABLE Tickets(

ticketID INT IDENTITY(1,1) PRIMARY KEY,

programID INT FOREIGN KEY(programID) REFERENCES Programs(programID) ON UPDATE CASCADE ON DELETE CASCADE,

customerID INT FOREIGN KEY(customerID) REFERENCES Customers(customerID) ON UPDATE CASCADE ON DELETE CASCADE,

ownerName nvarchar(MAX) NOT NULL,

transactionID varchar(64)

)

CREATE TABLE StaffsRoles(

programID INT FOREIGN KEY (programId) REFERENCES Programs(programId) ON UPDATE CASCADE ON DELETE CASCADE,

staffID INT FOREIGN KEY (staffID) REFERENCES Staff(staffID) ON UPDATE CASCADE ON DELETE CASCADE,

roleID INT FOREIGN KEY (roleID) REFERENCES Roles(roleID) ON UPDATE CASCADE ON DELETE CASCADE,

Constraint StaffsRolets\_PK PRIMARY KEY(programID,staffID,roleID)

)

ПРИЛОЖЕНИЕ Б

(обязательное)

**Листинг приложения**

Код модуля Паролей

using System.Security.Cryptography;

namespace BoxOfficeMainForm.Services

{

public class PasswordService : IPasswordService

{

const int DEFAULT\_SALT\_LEN = 20;

byte[] IPasswordService.HashPassword(string password) => HashPassword(password);

bool IPasswordService.ValidatePassword(string password, byte[] expectedHash) => ValidatePassword(password, expectedHash);

void IPasswordService.GetNewSalt(Span<byte> salt) => GetNewSalt(salt);

public static byte[] HashPassword(string password)

{

byte[] result = new byte[64];

Span<byte> resultSpan = result;

GetNewSalt( resultSpan.Slice(start:2) );

Rfc2898DeriveBytes.Pbkdf2(

password,

resultSpan.Slice(start: 2, DEFAULT\_SALT\_LEN),

resultSpan.Slice(start: 2+DEFAULT\_SALT\_LEN),

iterations: 600\_015,

HashAlgorithmName.SHA256

);

//0 версия хэша

resultSpan[0] = 0b\_0000\_0001;

//1 hints

resultSpan[1] = 0b\_0000\_0000;

return result;

}

public static bool ValidatePassword(string password, byte[] expectedHash)

{

bool result = false;

Span<byte> expectedHashSpan = expectedHash;

switch (expectedHashSpan[0])

{

case 0b\_0000\_0001:

{

byte[] passwordHash = new byte[64];

Span<byte> passwordHashSpan = passwordHash;

Span<byte> salt = expectedHashSpan.Slice(start: 2, length: DEFAULT\_SALT\_LEN);

Rfc2898DeriveBytes.Pbkdf2(

password,

salt,

passwordHashSpan.Slice(start: 2 + DEFAULT\_SALT\_LEN),

iterations: 600\_015,

HashAlgorithmName.SHA256

);

result = CryptographicOperations.FixedTimeEquals(

passwordHashSpan.Slice(start: 2 + DEFAULT\_SALT\_LEN),

expectedHashSpan.Slice(start: 2 + DEFAULT\_SALT\_LEN)

);

break;

}

default:

{

throw new NotImplementedException();

}

}

return result;

}

public static void GetNewSalt(Span<byte> salt)

{

RandomNumberGenerator.Fill(salt);

}

}

}

Код модуля проверки прав

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using BoxOfficeMainForm.EntityFrameWorkEssentials;

using BoxOfficeMainForm.Forms.Tables;

using Microsoft.VisualBasic.ApplicationServices;

namespace BoxOfficeMainForm.Services

{

public class RightsService

{

public enum Rights

{

НетПрав,

админ,

Кассир,

ОрганизаторСобытий,

МенеджерПерсонала

}

public static bool isThisFormAllowedForUser(Form form, UserBoxOffice userBoxOffice)

{

switch (userBoxOffice.rights)

{

case "админ":

switch(form)

{

case FormUSERSEdition:

case FormActorEdition:

case FormCustomersEdition:

case FormProgramsEdition:

case FormPromosEdition:

case FormRolesEdition:

case FormScenarioEdition:

case FormSceneEdition:

case FormStaffEdition:

case FormTicketsEdition:

return true;

}

break;

case "Кассир":

switch (form)

{

case FormCustomersEdition:

case FormPromosEdition:

case FormTicketsEdition:

return true;

}

break;

case "ОрганизаторСобытий":

switch (form)

{

case FormActorEdition:

case FormProgramsEdition:

case FormRolesEdition:

case FormScenarioEdition:

case FormSceneEdition:

return true;

}

break;

case "МенеджерПерсонала":

switch (form)

{

case FormStaffEdition:

return true;

}

break;

case "None":

return false;

}

return false;

}

}

}