# ВВЕДЕНИЕ

Современные информационные системы являются не просто средством автоматизации и повышения эффективности, но неотъемлемым элементом архитектуры компании. Организации все чаще вкладывают значительные средства в системы, способные помочь компании выжить в стремительно изменяющейся внешней среде и условиях конкуренции.

Использование базы данных значительно повышает производительность труда пользователя и упрощает использование различных средств по преобразованию данных. Появление БД явилось едва ли не важнейшим событием для людей, которые каждый день имеют дело с огромным количеством данных.

В наше время огромное количество фирм используют персональные компьютеры для сохранения и обработки любого вида информации. Эта информация содержится в базах данных. Базы данных играют важную роль в развивающемся мире технологий. Всё, с чем мы каждый день взаимодействуем в жизни, по всей видимости, зафиксировано в какой-нибудь базе. Работа с базами данных является важнейшим навыком в работе с компьютером, а специалисты данной области становятся всё более востребованными. Главные идеи нынешней информационной методики базируются на представлении, в соответствии чему информация должна быть образована в базы данных с задачей отображения динамически изменяющегося мира и удовлетворения всех потребностей в информации у пользователей. Базы данных формируются и работают под управлением специальных программных средств, называемых системами управления базами данных.

База данных, которая представлена в объективной форме, это совокупность таких материалов: статей, счетов, нормативных актов, судебных решений или иных подобных материалов, собранные вместе таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины.

Целью курсового проекта является создание информационной системы «Театр». Следовательно нужно проанализировать предметную область театра и принципы его работы. Прежде всего театры это заведения из сферы развлечений, которые напрямую связаны с социальной сферой общества Существует огромное количество разновидностей театров, особенности работы каждого из них стоит рассматривать отдельно. Исходя из каждого определения и описания, задачи и цели заведения могут сильно отличаться.

1. **Общий раздел**
   1. **Постановка задачи**

Целью курсового проекта является разработка информационной системы «Театр».

В рамках курсового проектирования необходимо разработать приложение, которое состоит из девяти окон:

Первое окно − авторизация пользователя.

Второе окно − регистрация пользователя (если у пользователя отсутствует аккаунт в приложении).

Третье окно − форма администратора.

Четвертое окно − форма контактной информации о театре.

Пятое окно – форма покупки билетов.

Шестое окно – основная форма приложения, на которую пользователь попадает после авторизации.

Седьмое окно – форма описания спектакля «Мэрри Поппинс».

Восьмое окно – форма описания спектакля «Обломов».

Девятое окно – форма расписания постановок.

В данном курсовом проекте исследуется предметная область театра. В ходе анализа данной предметной области были сформированы и выделены основные элементы значимые элементы системы.

Определены основные предметно-значимые сущности: билеты, учет.

Основными предметно-значимыми атрибуты сущностей будут:

* + билеты – код, вид билета;
  + учет – код, вид билета, имя, фамилия, отчество, дата рождения; Итак, требуется разработать информационную систему для автоматизации работы театра. Объекты в системе могут иметь уникальный внутренний номер и полное наименование. Каждый билет, представляемый в информационном ресурсе, характеризуется следующими параметрами:
* уникальный номер;
* вид билета;

Пользователи могут иметь схожие данные, но они различаются по своему уникальному шифру.

На каждого к в базу данных заносятся следующие сведения:

* + код;
  + вид билета;
  + имя;
  + фамилия;
  + отчество;
  + дата рождения;

С данной информационной системой должны работать следующие группы пользователей:

* администраторы;
* пользователи-гости;
* зарегистрированные пользователи

При работе с системой администратор должен иметь возможность решать следующие задачи:

* контроль учетных записей других пользователей;
* просмотр записей других пользователей;

Пользователь должен иметь возможность решать следующие задачи:

* просматривать список актуальных постановок;
* просматривать информацию о конкретном спектакле;
* покупать билеты

Основной услугой организации, рассматриваемой в рамках данного проекта является постановка и организация продажи билетов на театральные спектакли. Продаваемые билеты различаются по видам, то есть распределены по спектаклям на которые они направлены. Определены основные предметно-значимые сущности: билеты, учет.

В том или ином порядке администрация имеет право получать все внесенные пользователями данные. ER – диаграмма отображает поля представленные в таблицах Билеты и Учет (представлено на рисунке 1.1).

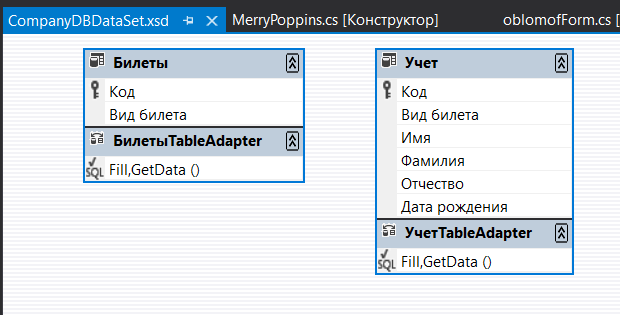


Рисунок 1.1 – ER – диаграмма

* 1. **Выбор и основания средств методов разработки**

В данном курсовом проекте использовались следующие средства разработки:

C# это объектно-ориентированный язык с большим количеством синтаксического сахара, что позволяет экономить время и силы. Он используется во многих крупны организациях, а также начинающими. Имеется удобная и бесплатная среда разработки. Достаточно прост в освоении на базовом уровне, но содержит вагон и маленькую тележку скрытых особенностей, чтобы разобраться в которых потребуется много времени.

Позволяет разрабатывать универсальные мобильные приложения и игры, но используется для этого достаточно редко

При этом следует понимать, что язык очень мощный и на изучение всех тонкостей потребуется много времени, но изучить основы синтаксиса, понять принципы ООП и разработки в целом вполне возможно любому заинтересованному человеку.

C# разрабатывался как язык программирования прикладного уровня для [CLR](https://ru.wikipedia.org/wiki/Common_Language_Runtime) и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает [BCL](https://ru.wikipedia.org/wiki/Base_Class_Library). Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим [.NET](https://ru.wikipedia.org/wiki/.NET_Framework)-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, [сборка мусора](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0) не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на [VB.NET](https://ru.wikipedia.org/wiki/Visual_Basic_.NET), [J#](https://ru.wikipedia.org/wiki/Visual_J%E2%99%AF) и др. В C# существуют две разновидности типов: ссылочные типы и типы значений. Переменные типа значений содержат непосредственно данные, а в переменных ссылочных типов хранятся ссылки на нужные данные, которые именуются объектами. Две переменные ссылочного типа могут ссылаться на один и тот же объект, поэтому может случиться так, что операции над одной переменной затронут объект, на который ссылается другая переменная. Каждая переменная типа значения имеет собственную копию данных, и операции над одной переменной не могут затрагивать другую

Microsoft Word - мощный текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов, с локальным применением простейших формтаблично-матричных алгоритмов. Выпускается корпорацией Microsoft в составе пакета Microsoft Office.Одним из многих полезных свойств Word является автоматическая коррекция текста по границам, автоматический перенос слов и правка правописания слов, сохранение текста в определенный устанавливаемый

промежуток времени, наличие мастеров текстов и шаблонов, позволяющих в считанные минуты создать деловое письмо, факс, автобиографию, расписание, календарь и многое другое.

Word позволяет применять различные шрифты, вставлять в документ электронные таблицы, диаграммы, сложные математические формулы, графические иллюстрации и звуковые комментарии. Он обеспечивает фоновую проверку орфографии. По своим возможностям Word приближается к настольным издательским системам.

При создании больших документов (таких как бизнес-план или курсовая) приходится часто менять шрифты, чтобы выделить заголовки, примечания, сноски и тому подобное.

Для того, чтобы упростить работу со шрифтами, их начертанием, размерами, выравниванием в Word предусмотрена работа со стилями. Данная функция позволяет сохранять художественное оформление и единство документа. Для каждого объекта документа задается определенный стиль (Формат \ Стиль) - совокупность параметров шрифта и абзаца.

Встроенный редактор формул позволяет создавать самые сложные математические и химические формулы.

Основные элементы текстового документа

Слово – произвольная последовательность ограничена с обоих концов служебными символами (пробелы и знаки препинания).

Строка – это последовательность символов между границами страницы.

Абзац – это последовательность символов расположенная между двумя <Enter>.

Перечислим дополнительные функции, которые ускоряют и облегчают работу с текстами:

* расстановки переносов;
* изменение масштаба документа при предварительном просмотре;
* автоматическая нумерация страниц;
* создание границ вокруг текста и страниц;
* разработка своего стиля и использование его в дальнейшем для быстрого форматирования текста;
* работа с макросами;
* поддержка нескольких окон;
* удобная справочная система;
* проверка орфографии, грамматики и синтаксиса;
* настройка панелей инструментов (добавление и удаление кнопок и команд);
* возможность предварительного просмотра перед печатью;
* настройка параметров страницы и поддержка нестандартных форматов бумаги;
* вставка в текст специальных символов и т.д;

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами. Первая версия Entity Framework - 1.0 вышла еще в 2008 году и представляла очень ограниченную функциональность, базовую поддержку ORM (object-relational mapping - отображения данных на реальные объекты) и один единственный подход к взаимодействию с БД - Database First. С выходом версии 4.0 в 2010 году многое изменилось - с этого времени Entity Framework стал рекомендуемой технологией для доступа к данным, а в сам фреймворк были введены новые возможности взаимодействия с БД - подходы Model First и Code First.

Дополнительные улучшения функционала последовали с выходом версии 5.0 в 2012 году.

И наконец, в 2013 году был выпущен Entity Framework 6.0, обладающий возможностью асинхронного доступа к данным.Центральной концепцией Entity Framework является понятие сущности или entity.

Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами. Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес.

Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность.

Подобные свойства называют ключами. При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из БД, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга). На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных. Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных
* model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.
* code first: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.
* code first: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

Платформа ASP.NET Core представляет технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов.

С одной стороны, ASP.NET Core является продолжением развития платформы ASP.NET. Но с другой стороны, это не просто очередной релиз. Выход ASP.NET Core фактически означает революцию всей платформы, ее качественное изменение.

Разработка над платформой началась еще в 2014 году. Тогда платформа условно называлась ASP.NET vNext. В июне 2016 года вышел первый релиз платформы. А в ноябре 2020 года вышла версия ASP.NET Core 5.0, которая собственно и будет охвачена в текущем руководстве.

ASP.NET Core теперь полностью является opensource-фреймворком. Все исходные файлы фреймворка доступны на [GitHub](https://github.com/aspnet/).

ASP.NET Core может работать поверх кросс-платформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS, Linux. И таким образом, с помощью ASP.NET Core мы можем создавать кросс-платформенные приложения. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь уже мы не ограничены только этой операционной системой. То есть мы можем запускать веб-приложения не только на ОС Windows, но и на Linux и Mac OS. А для развертывания веб-приложения можно использовать традиционный IIS, либо кросс-платформенный веб-сервер Kestrel.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку System.Web.dll.

ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages. В предыдущих версии платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности.

Кроме объединения вышеупомянутых технологий в одну модель в MVC был добавлен ряд дополнительных функций.

Одной из таких функций являются тэг-хелперы (tag helper), которые позволяют более органично соединять синтаксис html с кодом С#.

ASP.NET Core характеризуется расширяемостью. Фреймворк построен из набора относительно независимых компонентов. И мы можем либо использовать встроенную реализацию этих компонентов, либо расширить их с помощью механизма наследования, либо вовсе создать и применять свои компоненты со своим функционалом.

Также было упрощено управление зависимостями и конфигурирование проекта. Фреймворк теперь имеет свой легковесный контейнер для внедрения зависимостей, и больше нет необходимости применять сторонние контейнеры, хотя при желании их также можно продолжать использовать.

В качестве инструментария разработки мы можем использовать последние выпуски Visual Studio, начиная с версии Visual Studio 2015. Кроме того, мы можем создавать приложения в среде Visual Studio Code, которая является кросс-платформенной и может работать как на Windows, так и на Mac OS X и Linux.

Windows Forms — это технология интеллектуальных клиентов для NET Framework. Она представляет собой набор управляемых библиотек, упрощающих выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью такой среды разработки, как Visual Studio, можно создавать интеллектуальные клиентские приложения Windows Forms, которые отображают информацию, запрашивают ввод от пользователей и обмениваются данными с удаленными компьютерами по сети.

В Windows Forms форма— это визуальная поверхность,на которой отображаются сведения для пользователя.

Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие.

Приложение реагирует на эти события с помощью кода и обрабатывает события при их возникновении. Подробнее см. в разделе [Создание обработчиков событий в Windows Forms](https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/creating-event-handlers-in-windows-forms?view=netframeworkdesktop-4.8).

Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы.

 Список всех элементов управления, которые можно использовать в форме, представлены в разделе [Элементы управления для использования в формах Windows Forms](https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/controls-to-use-on-windows-forms?view=netframeworkdesktop-4.8). Если существующий элемент управления не удовлетворяет потребностям, в Windows Forms можно создать пользовательские элементы управления с помощью класса [UserControl](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.usercontrol).

В состав Windows Forms входят многофункциональные элементы пользовательского интерфейса, позволяющие воссоздавать возможности таких сложных приложений, как Microsoft Office. Используя элементы управления [ToolStrip](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.toolstrip) и [MenuStrip](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.menustrip), можно создавать панели инструментов и меню, содержащие текст и рисунки, подменю и другие элементы управления, такие как текстовые поля и поля со списками.

С помощью **конструктор Windows Forms** перетаскивания в Visual Studio можно легко создавать Windows Forms приложения.

 Достаточно выделить элемент управления курсором и поместить его в нужное место на форме. Для преодоления трудностей, связанных с выравниванием элементов управления, конструктор предоставляет такие средства, как линии сетки и линии привязки.

А также при использовании Visual Studio или компиляции из командной строки можно использовать [FlowLayoutPanel](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.flowlayoutpanel) [TableLayoutPanel](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.tablelayoutpanel) [SplitContainer](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer) элементы управления, и для создания расширенных макетов форм за меньшее время.

Наконец, если нужно создать свои собственные элементы пользовательского интерфейса, пространство имен [System.Drawing](https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing) содержит широкий набор классов, необходимых для отрисовки линий, кругов и других фигур непосредственно на форме.

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений.  Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

Ниже перечислены некоторые популярные возможности Visual Studio, которые помогут вам повысить продуктивность разработки программного обеспечения.

Волнистые линии обозначают ошибки или потенциальные проблемы кода прямо во время ввода.

Эти визуальные подсказки позволяют устранять проблемы немедленно и не ждать, пока ошибка будет обнаружена во время сборки или запуска программы. Если навести указатель мыши на волнистую линию, на экран будут выведены дополнительные сведения об ошибке

 Кроме того, в поле слева может появляться значок лампочки с быстрыми действиями по устранению ошибки.

Вы можете одним нажатием кнопки отформатировать код и применить к нему исправления, предложенные [параметрами стиля кода](https://docs.microsoft.com/ru-ru/visualstudio/ide/reference/options-text-editor-csharp-formatting?view=vs-2019), [соглашениями в файле EditorConfig](https://docs.microsoft.com/ru-ru/visualstudio/ide/create-portable-custom-editor-options?view=vs-2019) и (или) [анализаторами Roslyn](https://docs.microsoft.com/ru-ru/visualstudio/code-quality/roslyn-analyzers-overview?view=vs-2019). **Очистка кода** помогает устранить многие проблемы в коде еще до проверки кода.

Рефакторинг включает в себя такие операции, как интеллектуальное переименование переменных, извлечение одной или нескольких строк кода в новый метод, изменение порядка параметров методов и многое другое.

IntelliSense — это набор функций, отображающих сведения о коде непосредственно в редакторе и в некоторых случаях автоматически создающих небольшие отрывки кода. По сути, это базовая документация, встроенная в редактор, с которой вам не приходится искать информацию где-то еще. Функции IntelliSense зависят от языка. Дополнительные сведения см. в руководствах по [IntelliSense для C#](https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-csharp-intellisense?view=vs-2019), [IntelliSense для Visual C++](https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-cpp-intellisense?view=vs-2019), [IntelliSense для JavaScript](https://docs.microsoft.com/ru-ru/visualstudio/ide/javascript-intellisense?view=vs-2019) и [IntelliSense для Visual Basic](https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-basic-specific-intellisense?view=vs-2019).

Среда Visual Studio может показаться сложной, ведь там столько разных меню, параметров и свойств.  Чтобы быстро находить функции интегрированной среды разработки и элементы кода, в Visual Studio представлен единый компонент поиска (**CTRL**+**Q**).

Предоставляет возможности совместного редактирования и отладки в реальном времени независимо от типа приложения или языка программирования. Вы можете мгновенно и безопасно поделиться своим проектом и, при необходимости, сеансами отладки, экземплярами терминалов, веб-приложениями localhost, голосовыми звонками и многим другим.

Функция отладки предусматривает прогон программы и построчную отладку на выбор. Таким образом, пользователь получает возможность протестировать каждый отдельный этап программы или же рассмотреть ее работу в целом. Это особенно полезно и актуально во время модульного тестирования и тестирования «белого ящика»

В окне **Иерархия вызовов** показаны методы, вызывающие выбранный метод. Это может быть полезно, если вы собираетесь изменить или удалить метод или хотите отследить ошибку.

CodeLens помогает находить ссылки на код, изменения кода, связанные ошибки, рабочие элементы, проверки кода и модульные тесты — все это не выходя из редактора.

В окне **Просмотр определений** показано определение метода или типа, при этом не нужно открывать отдельный файл.

[Microsoft Azure](https://azure.microsoft.com/ru-ru/) — это сервис облачных вычислений, который работает аналогично [Amazon Web Services (AWS)](https://aws.amazon.com/ru/) и [облачной платформе Google](https://cloud.google.com/).

Под «облачными вычислениями» мы не подразумеваем туманный термин, который часто применяется к потребительским сервисам, которые хранят Ваши данные на удаленном сервере. Мы имеем в виду фактические вычисления как услугу для компаний, организаций и даже частных лиц, которые хотят этим воспользоваться.

Традиционно предприятия и другие организации размещают свою собственную инфраструктуру. У бизнеса будет собственный веб-сервер (или почтовый сервер, или что-то еще) на своем собственном оборудовании. Если бы потребовалось больше мощности, бизнесу пришлось бы покупать больше серверного оборудования. Компании также придется платить кому-то за администрирование этого оборудования и платить за надежное подключение к Интернету для обслуживания своих клиентов. Кроме того, есть хостинговые компании, которые размещают Ваши сервисы на некоторых из своих аппаратных средств в своих центрах обработки данных, за деньги.

[PhpMyAdmin](https://www.phpmyadmin.net/) - это удобная рабочая среда для новичков и профессионалов. Интуитивно понятное расположение основных элементов помогает быстро получить доступ к базам данных. Интерфейс переведен на несколько языков, включая русский.

Свободная лицензия поспособствовала широкому распространению программы по всему миру. Продукт интегрирован в Openserver, Денвер и другие разработки.

Первая версия появилась в 1998 году, разработчик Тобиас Ротшильд. Поддерживается на всех операционных системах.

PhpMyAdmin может управлять как целым MySQL сервером (для этого необходимы права супер-пользователя), так и отдельной базой данных. Возможно мультипользовательское использование (см. мануал MySQL "Добавление новых пользователей в MySQL").

В последнем случае пользователи могут пользоваться только назначенным им базами.

В данный момент phpMyAdmin позволяет:

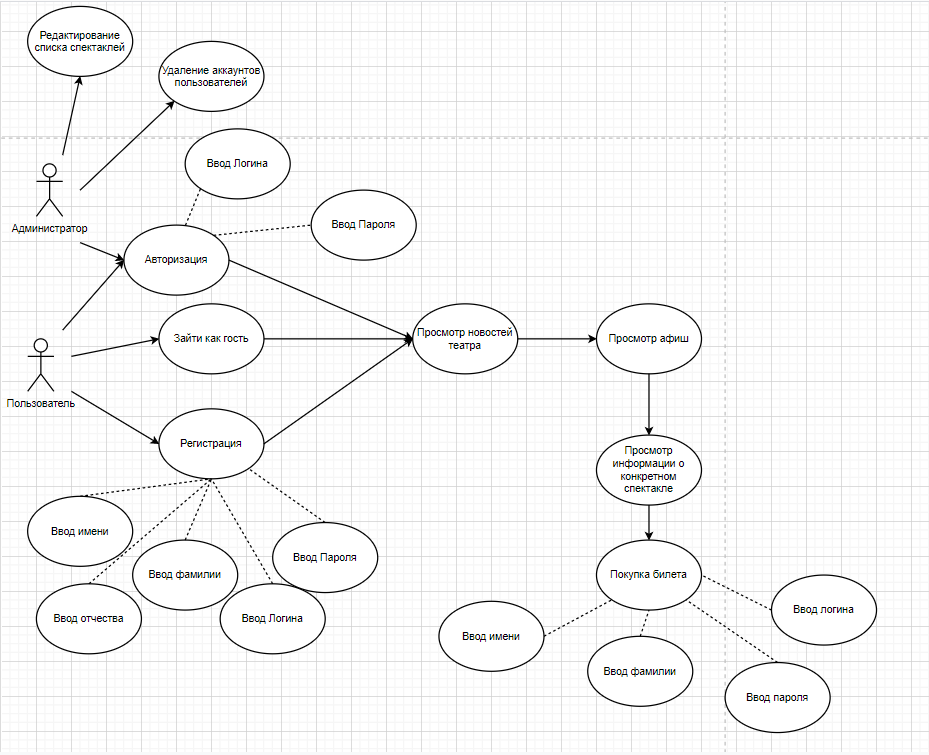
* создавать и удалять базы данных;
* создавать, копировать, удалять, переименовывать и изменять таблицы;
* осуществлять сопровождение таблиц;
* удалять, править и добавлять поля;
* выполнять SQL-запросы, в том числе пакетные SQL-запросы;
* управлять ключами;
* загружать текстовые файлы в таблицы;
* создавать (\*) и просматривать дампы таблиц;
* экспортировать (\*) данные в форматах CSV, XML, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, Excel и LATEX;
* администрирование нескольких серверов;
* управлять пользователями MySQL и привилегиями;
* проверять целостность ссылочных данных в таблицах MyISAM;
* использовать запрос по образцу (Query-by-example - QBE), создавать комплексные запросы, автоматически соединяясь с указанными таблицами;
* создавать графическую схему базы данных в формате PDF;
* осуществлять поиск в базе данных или в её разделах модифицировать хранимые данные в различные форматы, использующиеся в предустановленных функциях, например, отображение BLOB-данных как изображений или как загружаемые ссылки и т.д.;
* поддерживает InnoDB таблицы и внешние ключи;
* поддерживает mysqli, улучшенное расширение MySQL;
* переведен более чем на 50 языков;

В свою очередь благодаря дополнительным модулям становится возможным редактирование и обращение к БД с помощью таких средств как PHPMyAdmin

* 1. **Анализ требований и определений спецификации ПП**

Для решения поставленных задач, были сформированы и проанализированы основные проблемы, благодаря которым стало возможным вывести определенные требования к ИС

* совместимость ИС «Театр» с операционными системами Windows 10 и Windows 7.
* требования по функциональности. Разрабатываемая ИС «Театр» должна обеспечивать отслеживание и контроль актуальных театральных постановок. Для этого создаваемая система должна обеспечивать ввод и удаление данных о пользователях, услугах, пользователи использующие АИС «Театр», не должны обладать какими либо особыми навыками, так как интерфейс предоставляет простой дизайн и удобство пользования. Для отображен основных сценариев действия пользователей была разработана диаграмма вариантов использования − представлена на рисунке 1.3.1.

Рисунок 1.3.1 – Диаграмма вариантов использования

* 1. **Специальный раздел**
  2. **Проектирование логической структуры программы продукта**

На всех этапах разработки информационной системы идет использование и взаимодействие и информацией предоставленной пользователями и системой. Сформирована логическая структура информационной системы «Театр», которая отображает взаимодействие библиотек и исполняемых файлов (рисунок 2).

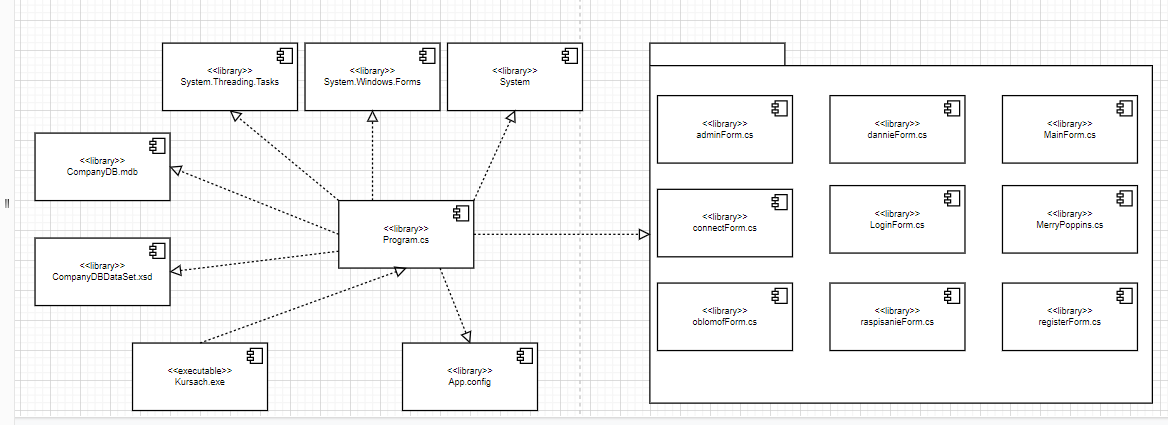


Рисунок 2 – Диаграмма компонентов

Вся структура пользовательских классов изображена на UML диаграмме классов, которая отображает связи и зависимости классов, а также используемые в них методы и переменные (рисунок 2.1).

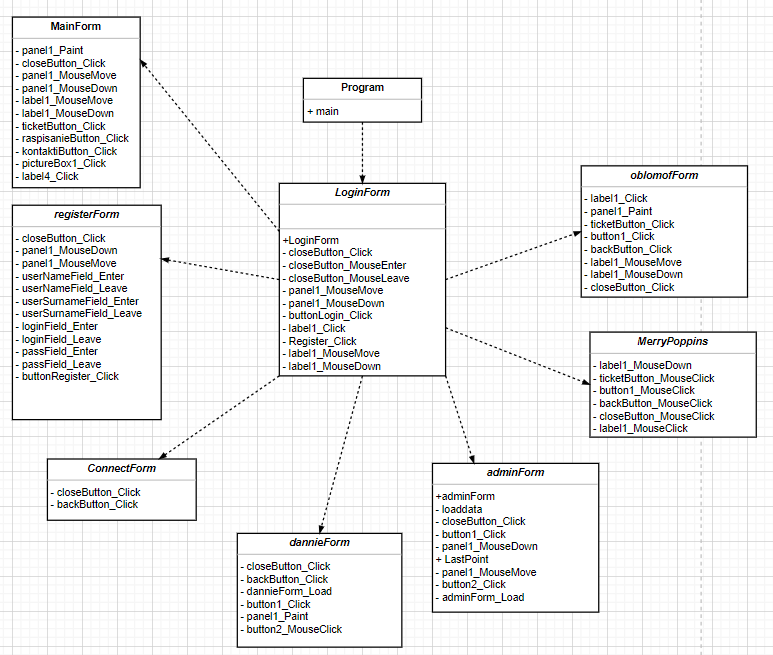


Рисунок 2.1 – Диаграмма классов

* 1. **Разработка физической структуры программы продукта**

Физическая структура информационной системы «Театр» разделена на две таблицы:

– информация о билетах.

– информация о пользователях (рисунок 2.2).

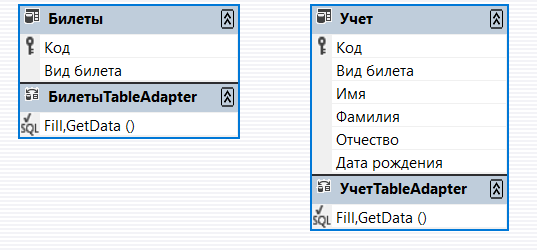


Рисунок 2.2 – Таблица билеты и учет

В таблице билет содержатся данные об актуальных билетах. Полями этой таблицы являются:

* Код билета;
* Вид билета.

В таблице учет содержатся данные о билетах, купленные пользователями. Полями этой таблицы являются:

* Код операции;
* Вид билета;
* Имя пользователя;
* Фамилия пользователя;
* Отчество пользователя;
* Дата рождения пользователя;
  1. **Разработка интерфейсных компонентов**

После составления требований по созданию программному обеспечению

Были созданы макеты следующих интерфейсов. Интерфейс окна авторизации (представлен на рисунке 2.3.1)



Рисунок 2.3.1 – Макет экрана авторизации

Окно содержит :

* поле ввода логина;
* поле ввода пароля;
* кнопку «Войти» которая будет проверять наличие пользователя в БД и открывать другую форму если введенные данные верны;
* кнопку «Войти как гость»

Интерфейс главного окна представлен на рисунке 2.3.1

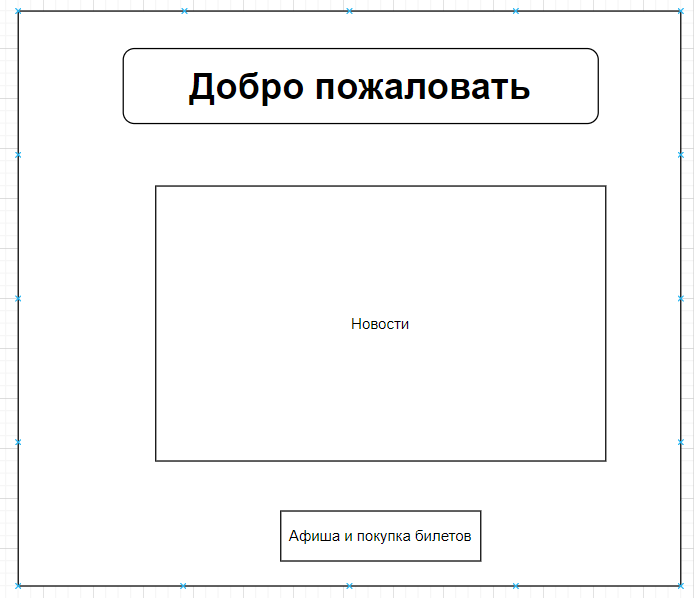


Рисунок 2.3.2 – Макет главного окна

Окно содержит

* вкладку «новости» на которой будут отображены актуальные новости театра
* кнопку «Афиша и покупка билетов»

Интерфейс окна просмотра постановок представлен на рисунке 2.3.3



Рисунок 2.3.3 – Макет окна просмотра постановок

Окно содержит

* окно просмотра актуального спектакля, при нажатие на которое произойдет переход на форму с подробной информацией о данной постановке;
* кнопку назад для возвращения на предыдущую форму.
  1. **Описание разработанного приложения**

Разработанная информационная система представляет собой площадку для размещения информации о спектаклях и новостей разного рода связанные с театром. Каждый пользователь имеет возможность просматривать афиши актуальных постановок и покупать билеты на них. Информация о всех купленных билетах будет отображена в учетной таблице и сохранена даже при выходе из приложения.

Зайдя в аккаунт администратора можно просматривать все заявки на покупку билета полученные в данный момент, а также редактировать список актуальных спектаклей . Таким образом система поддерживает взаимодействие нескольких пользователей одновременно Структура диаграммы последовательности действий отображает стандартный сценарий действий пользователя (представлена на рисунке 2.4).

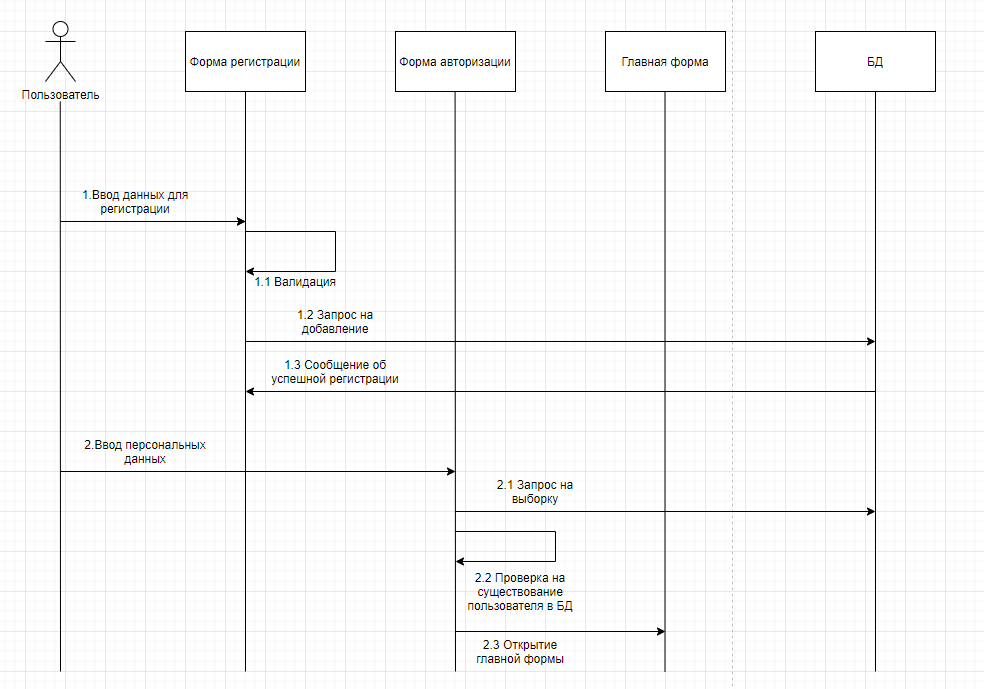


Рисунок 2.4 – Диаграмма последовательности действий

* 1. **Выбор стратегии тестирования программного продукта**

В процессе разработки информационных систем и программных обеспечений, очень часто используются различные методы тестирования. Информационное сообщество рассматривает следующие виды тестирования:

* **Системное тестирование программного обеспечения** — это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям. Системное тестирование относится к методам тестирования чёрного ящика, и, тем самым, не требует знаний о внутреннем устройстве системы. Основной задачей системного тестирования является проверка как функциональных, так и не функциональных требований к системе в целом. При этом выявляются дефекты, такие как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д. Для минимизации рисков, связанных с особенностями поведения системы в той или иной среде, во время тестирования рекомендуется использовать окружение максимально приближенное к тому, на которое будет установлен продукт после выдачи. модульное тестирование является одной из ключевых практик методологии экстремального программирования. Если в результате исправления ошибок интеграции меняется исходный код, в нем с большой вероятностью появляются ошибки. Если в результате добавления новой функциональности меняется исходный код, в нем с большой вероятностью появляются ошибки. И искать их лучше с помощью ранее созданных модульных тестов;

Каждое из видов тестирования используется под свою задачу и возможную проблематику приложений. При работе над курсовым проектом были использованы лишь несколько из этих видов тестирования – минимально достаточный вид тестирования:

Тестирование «белого ящика» зачастую помогло при начале отладки программы обнаружить огромное количество ошибок, которые вскоре были исправлены. Данный вид тестирования показал свою эффективность за счет сужения области поисков нахождения ошибки.

Тестирование «черного ящика» так же помогало решить большинство проблем на начальных этапах проведения тестирований. Благодаря своей низкой затратности и упрощенной процедуры ввода и вывода данных, которые позволяли не задумываться разработчику о том какие именно данные необходимо ввести, такой вид тестирования показал себя крайне положительно и позволил устранить большинство ошибок до того как это стало критично

При введении новых функций в программу, таких как например удаление или добавление объявления, осуществление поиска по категориям и по текстовому запросу, процедурах авторизации и регистрации, а также валидации проводилось модульное тестирование, которое позволяло тестировать отдельные участки кода, не затрагивая остальную программу. Такой вид тестирования сократил время необходимое для того чтобы добраться до конкретной ошибки. Например временно упразднив процедуру авторизации, которая необходима для добавления объявлений

Так же использовался метод ручного тестирования. Проверка работоспособности полей ввода, срабатывание кнопок, открытие различных окон, а так же проверка валидации введенных данных..

* + 1. **Отладка программного продукта**

Одним из самых главных средств отладки программы была непосредственно интегрированная среда разработки Visual Studio. В случае нахождения критических ошибок IDLE подсвечивала их красной волнистой линией и выдавала предупреждение перед попыткой запустить процедуру отладки. В случае если ошибки были не критическими, выводилось предупреждение, с которым можно было запустить программу, но проблемы связанные с этими замечаниями могли проявиться в будущем и привести к ошибочному завершению программы. На момент итогов тестирования все ошибки были устранены, о чем свидетельствует сама среда разработки. Однако присутствуют некритичные предупреждения (рисунок 2.5.1).

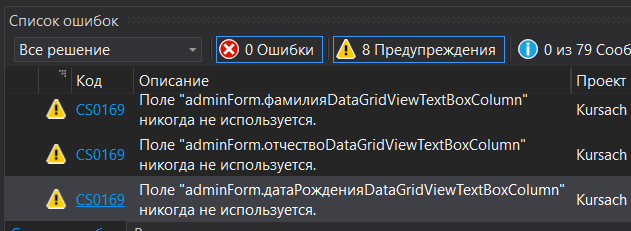


Рисунок 2.5.1 – Visual Studio не обнаружила критических ошибок, однако есть несущественные предупреждения

Помимо стандартной процедуры «прогона» неоднократно, в особенности при модульном тестировании и тестировании белого ящика использовалась пошаговая процедура отладки.

* 1. **Руководство пользователя**

Для безошибочного пользования информационной системой, пользователю рекомендуется просмотреть руководство пользователя:

При первом входе пользователю будет предоставлена возможность авторизации с помощью ввода логина и пароля либо же можно войти не имея аккаунта – как гость (рисунок 2.6.1).

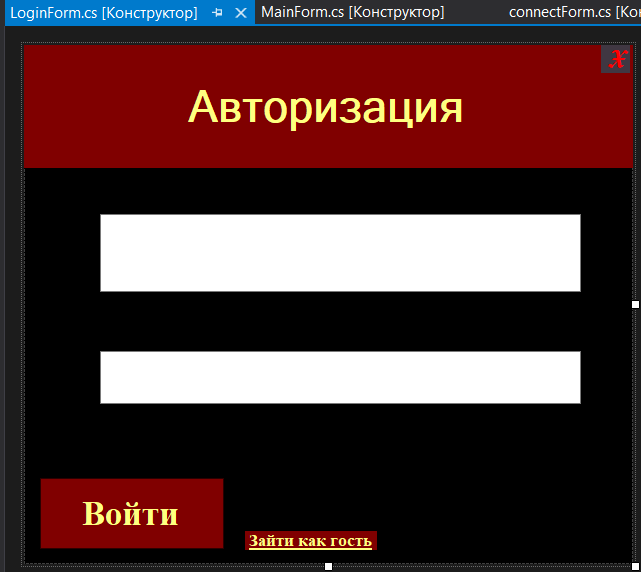


Рисунок 2.6.1 – Окно авторизации

При желании пользователь может зарегистрироваться открыв форму регистрации. Для регистрации необходимо ввести имя, фамилию, логин и пароль для последующего использования при авторизации. Без данных введенных в поля регистрация не представляется возможной.(рисунок 2.6.2)



Рисунок 2.6.2 – Окно регистрации

После успешной авторизации пользователь видит основное окно приложения, на котором отображены последние новости театра и кнопка для просмотра афиш (рисунок 2.6.3).

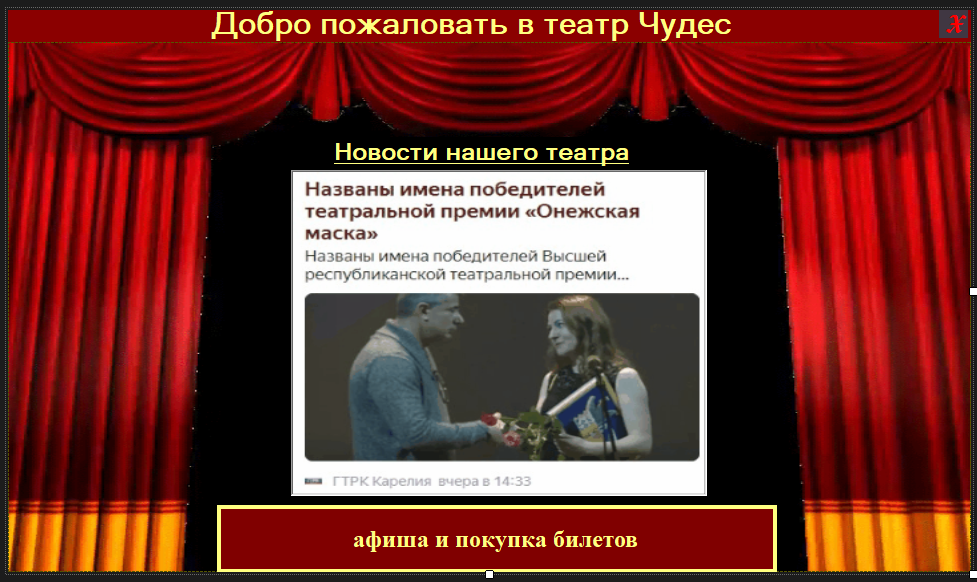


Рисунок 2.6.3 – Основное окно приложения

В центре располагается новостная, сменяющаяся сводка из мира искусства. Внизу находится кнопка «Афиша и покупка билетов» При нажатии на эьту кнопку будет осуществлен переход на форму просмотра афиш, это представлено на рисунке 2.6.4.

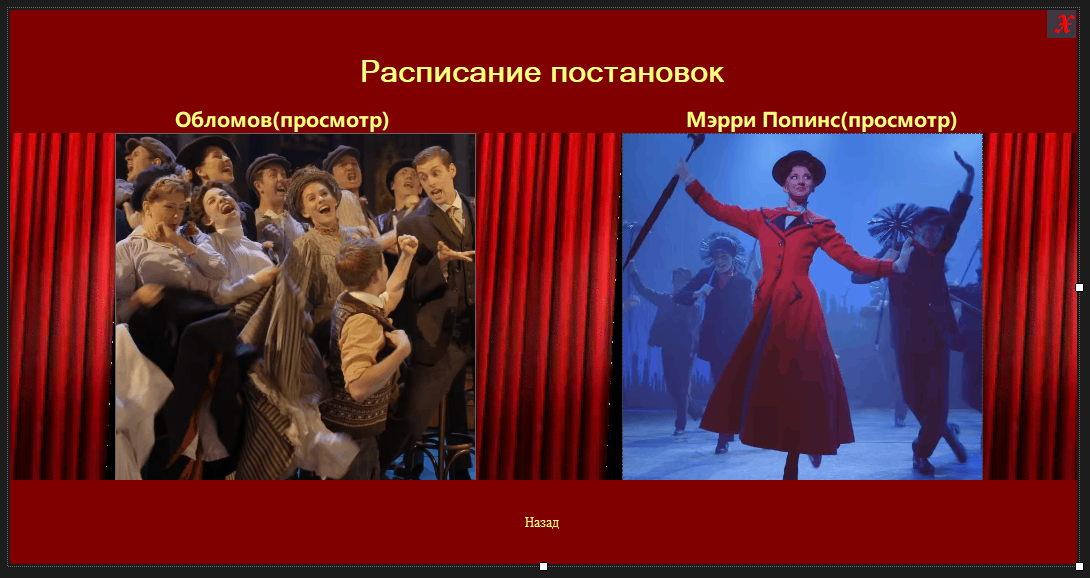


Рисунок 2.6.4 – Расписание постановок

Пользователь может нажать кнопку «Назад» для возвращения на основную форму либо нажать на картинку, чтобы перейти на форму просмотра подробной информации о представлении (рисунок 2.6.5 и 2.6.6).



Рисунок 2.6.5 – Информация о представлении



Рисунок 2.6.6 – Информация о представлении

При нажатии одной из кнопок с наценкой произойдет переход на форму оплаты. Цена и параметры оплаты зависят соответственно от даты (рисунок 2.6.7).



Рисунок 2.6.7 – Покупка билета

В представленных полях необходимо выбрать тип билета, ввести свои имя, фамилию, отчество и дату рождения, после чего станет доступным непосредственно оплата билета.

В случае если пользователя интересует контактная информация об организации или сфере услуг в целом он может нажать на кнопку «Контакты» в главном меню и перейти на окно описания услуг.(рисунок 2.6.8)

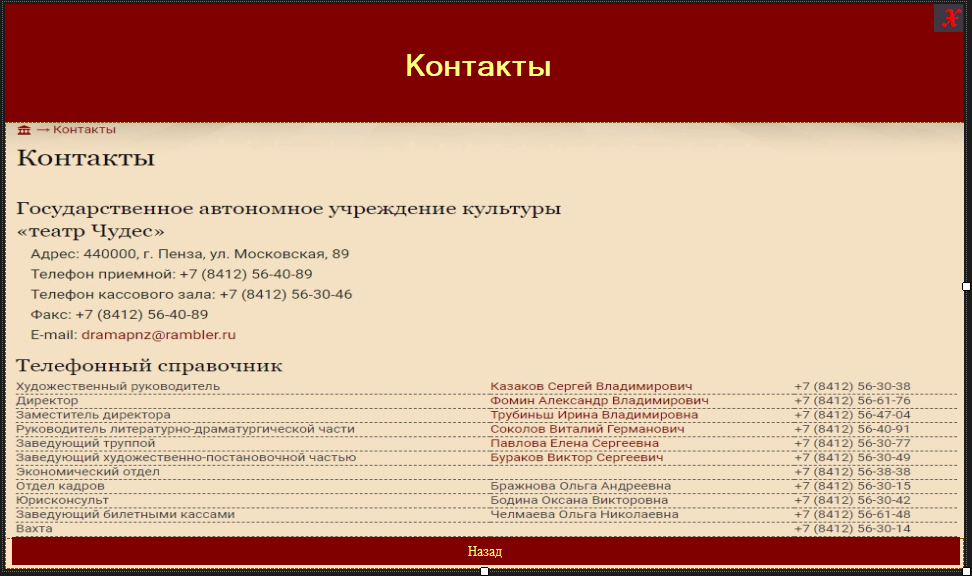


Рисунок 2.6.8 – Контактная информация

Если же зайти под аккаунтом с полномочиями администратора то доступной станет другая форма, специальная для просмотра оформленных билетов, а также списка актуальных постановок в театре.(рисунок 2.6.9)

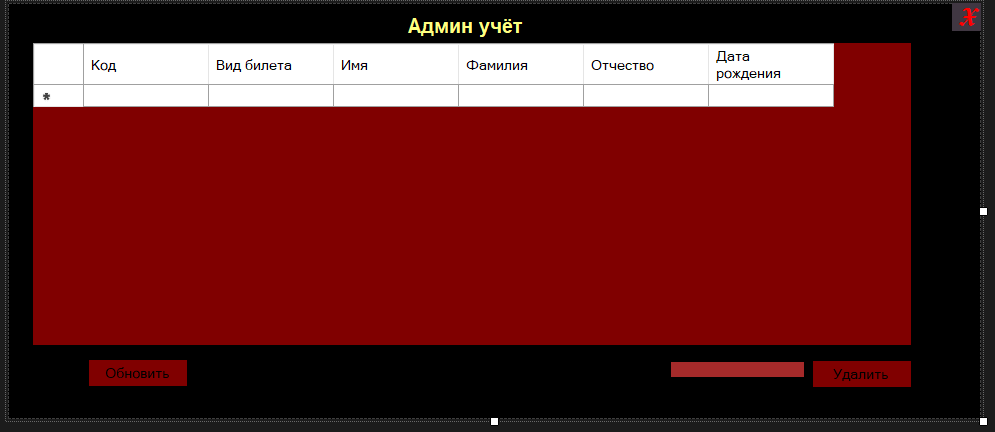


Рисунок 2.6.9 – Окно администратора

**ЗАКЛЮЧЕНИЕ**

В ходе разработки курсового проекта информационной системы «Театр» были изучены и проанализированы пособия по разработке на языке программирования C#.

Неоднократно для разработки использовалась официальная техническая документация от Microsoft для разработки Windows Forms приложений в Visual Studio, а также сторонние информационные ресурсы.

Были изучены способы работы с подключениями через облачную платформу Microsoft Azure, которая обеспечивала работу с базами данных.

Так же во время анализа предметной области было проведено ознакомление с сайтами и web-приложениями посвященными тематике аренды автомобилей и других транспортных средств. Были сделаны выводы, необходимые для формирования требований и принципов работы приложения, а также для распределения функций на приоритетные и второстепенные.

Была разработана, протестирована информационная система «Театр

Были сформированы техническая документация, руководство пользователя, построены диаграмма классов, вариантов использований, компонентов, последовательности действий и ER-диаграмма

Готовый проект обладает простотой использования и быстротой действия в виду малых затрат к аппаратным и программным средствам.

Таким образом, все задачи поставленные курсовым проектом были выполнены, а цель достигнута

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. ГОСТ 3.1102-2011 Единая система технологической документации (ЕСТД). Стадии разработки и виды документов. Общие положения (Издание с Поправками).
2. ГОСТ Р 56939-2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования.
3. Введение в .NET 4.0 и Visual Sudio 2010 для профессионалов. Алекс Макки[книга] (дата обращения: 10.04.21)
4. Изучаем C# (Head First). Эндрю Стиллмен, Дженнифер Грин [книга] (дата обращения: 08.04.21)
5. C# 7.0 и .NET Core. Кросс-платформенная разработка для профессионалов. Прайс Марк. [книга] (дата обращения: 08.04.21)
6. C# 7.0. Справочник. Полное описание языка. Албахари Бен , Албахари Джозеф [книга] (дата обращения: 10.04.21)
7. Microsoft Visual C# в задачах и примерах [книга] (дата обращения: 08.04.21)
8. Microsoft Visual C#. Джон Шарп [книга] (дата обращения: 10.04.21)
9. Базы данных [Базы данных. Учебное пособие [cайт] (siblec.ru)](https://siblec.ru/informatika-i-vychislitelnaya-tekhnika/bazy-dannykh). (дата обращения: 10.04.21)
10. Базы данных [Базы данных. Учебное пособие [cайт] (siblec.ru)](https://siblec.ru/informatika-i-vychislitelnaya-tekhnika/bazy-dannykh). (дата обращения: 10.04.21)
11. Введение в Windows Forms – пишем первую программу [cайт] https://code-live.ru/post/first-windows-form/ (дата обращения: 08.04.21)
12. Диаграмма вариантов использования (use case diagram) [cайт] [Леоненков. Самоучитель UML. (kharkiv.edu)](http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html). (дата обращения: 11.04.21)
13. Диаграммы классов. Википедия [cайт] https://ru.wikipedia.org/wiki/Диаграмма\_классов#/media/Файл:Class\_diagram\_relations.png (дата обращения: 24.03.21)
14. Назначение и состав диаграммы компонентов [cайт] https://www.sites.google.com/site/anisimovkhv/learning/pris/lecture/tema15/tema15\_2. (дата обращения: 11.04.21)
15. НОУ ИНТУИТ | Лекция | Элементы графической нотации диаграммы компонентов [cайт] https://intuit.ru/studies/courses/32/32/lecture/1022 (дата обращения: 31.03.21)
16. Основы программирования. Знакомство с курсом. [cайт] https://www.youtube.com/watch?v=wSa5f1F-J4I. (дата обращения: 04.04.21)
17. Отладка ошибок StackOverflow [cайт] https://docs.microsoft.com/ru-ru/dotnet/core/diagnostics/debug-stackoverflow (дата обращения: 02.04.21)
18. [Проектирование ER-диаграммы — Национальная сборная [cайт] Worldskills Россия](https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/). (дата обращения: 11.04.21)
19. Регистрация пользователей. <https://www.cyberforum.ru/windows-forms/thread1720420.html>. [cайт] (дата обращения: 26.03.21)
20. Руководство по программированию в Windows Forms [cайт] https://metanit.com/sharp/windowsforms/ (дата обращения: 09.04.21)
21. Уроки C# .NET Windows Forms. [cайт] <https://www.youtube.com/watch?v=gp2rA0rgq_0&list=PL0lO_mIqDDFWOMqSKFaLypANf1W7-o87q>. (дата обращения: 26.03.21)
22. Inserting Draw.io Flowcharts into Word [cайт] <https://www.youtube.com/watch?v=74Moceg2YAo>. (дата обращения: 04.04.21)
23. MySQL :: MySQL Connectors [cайт] https://www.mysql.com/products/connector/ (дата обращения: 04.04.21)
24. Usability Testing от А до Я: подробный гид [cайт] https://habr.com/ru/company/luxoft/blog/508146/ (дата обращения: 21.03.21)
25. Visual Studio 2019. Быстрое написание кода. Автоматизация работы. Интегрированная среда разработки будущего. [cайт] <https://visualstudio.microsoft.com/ru/vs/>. (дата обращения: 08.04.21)
26. Visual Studio. Лучшие в своем классе средства для разработчиков [cайт] https://visualstudio.microsoft.com/ru/. (дата обращения: 09.04.21)
27. Windows Forms | Microsoft вики | Fandom [cайт] https://microsoft.fandom.com/ru/wiki/Windows\_Forms (дата обращения: 11.04.21)
28. Windows Forms C#. [cайт] https://itproger.com/course/csharp-app/4. (дата обращения: 01.04.21)