# ВВЕДЕНИЕ

В век современных технологий все большее применение получают компьютерные методы учёта продукции, тем самым всё больше предприятий нуждаются в цифровизации своих данных.

Целью данного курсового проекта является создание прототипа информационной системы «Кинотеатр» с помощью полученных знаний при изучении курса «Визуальное программирование». А также, самостоятельное обучение технологиям создания сложных программных продуктов.

Интерфейс программы должен быть удобным для пользователя. Основная функция программы должна заключается в том, чтобы пользователь программы мог легко работать с базой кинотеатра, добавлять и искать необходимую ему информацию о фильмах.

Задачи, поставленные перед началом процесса создания проекта:

• проанализировать предметную область;

• разработать прототип информационной системы с использованием Windows Form;

• создать таблицы в PostgreSQL, хранимые процедуры для добавления, изменения и поиска данных.

• изучить возможность подключения PostgreSQL к Visual Studio и разработать прототип, которым в дальнейшем можно пользоваться.

# Глава 1. рАЗРАБОТКА пРОЕКТА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ.

# 1.1 Исследование предметной области

Имеем небольшой многозальный кинотеатр, в котором располагаются места разной комфортности и соответственно разной ценовой категории. Для просмотра фильма необходимо приобрести билет, в котором указан номер билета, название фильма, зал, время, место и цена. При этом кинотеатром ведется учёт занято или свободно место для продажи и возврата билета. Кроме того, клиентам предоставляются услуги по бронированию билетов.

В функционирование предметной области входит следующее:

• Продажа билетов

• Контроль наполняемости зала

• Услуги бронирования билетов и снятия брони

• Возврат билетов (допускается возврат билета и снятие его с брони)

• Предоставление информации о фильмах

Сотрудникам кинотеатра предоставляется возможность модифицировать данные, вносить необходимую информацию о фильмах, просматривать текущие фильмы, которые идут в кинотеатре, а также им доступна возможность делать возврат билета.

Работа программы устроена следующим образом: в базе данных хранится информация о фильмах. С помощью визуального приложения на Windows нам доступны такие функции как бронирование и покупка билетов, просмотр свободных мест в зале, редактирование данных и возврат билетов.

# 1.2. Действующие лица и варианты использования системы.

Действующими лицами разрабатываемой системы являются пользователь (кассир) и администратор. Диаграмму вариантов использования можно увидеть на рисунке 1.

Рисунок 1. Диаграмма вариантов использования

Далее подробно рассмотрим варианты использования системы.

**Вариант использования “Формирование заказа”:**

Данный вариант использования начинает исполняться, когда пользователь продаёт билет клиенту.

Основной поток событий:

Данный вариант использования начинается, когда клиент указал что хочет приобрести билет.

1.Система предоставляет пользователю возможность выбора фильма, времени сеанса и зала

2.Пользователь выбирает сеанс из тех, что предоставляет система, выбрав из списка фильмов.

3.Пользователь выбирает время начала фильма из тех, что предоставляет система, выбрав из списка время сеанса.

4.Пользователь выбирает место в зале из доступных в зрительном зале.

5.Пользователь соглашается с введенными требованиями.

6.Система формирует заказ на основании введенных требований.

7.Система предлагает пользователю проверить правильность введённых данных в билете.

8.Клиент подтверждает, что данные корректны

9.Система выводит сообщение о том, что данные записаны, что подразумевает покупку билета

10.Система записывает покупку билета в базу данных.

Альтернативный поток:

Нет свободных мест. Если во время выполнения основного потока обнаружится, что на данный сеанс нет свободных мест, система выдаёт сообщение о том, что нет свободных мест.

**Вариант использования “Возврат билета”:**

Данный вариант использования начинается исполняться, когда клиент хочет вернуть билет.

Основной поток:

Данный вариант использования начинается, когда клиент сообщает пользователю-кассиру, что хочет вернуть билет.

1. Пользователь выбирает возможность «Вернуть билет»

2. Пользователь вводит номер билета или сканирует штрих-код

3. Если данный билет действительно существует, то пользователь может продолжить процедуру возврата билета, иначе запускается альтернативный поток.

4. Система запрашивает подтверждение на возврат билета.

5. Пользователь соглашается с запросом системы.

6. Система отмечает в базе данных те места, что были в билете как свободные.

Альтернативный поток:

Введен неверный номер билета. Система выдает сообщение, что билет не найден.

**Вариант использования “Бронирование билета”:**

Данный вариант использования начинает исполняться, когда клиент хочет забронировать билет.

Основной поток событий:

Данный вариант использования начинается, когда клиент указал что хочет забронировать билет.

1.Система предоставляет пользователю таблицу с выпадающими списками для заполнения.

2.Пользователь выбирает сеанс из тех, что предоставляет система, выбрав из списка расписания сеансов.

3.Пользователь выбирает время начала сеанса из тех, что предоставляет система.

4.Пользователь выбирает место из доступных в зрительном зале.

5.Система формирует заказ на основании введенных требований.

6.Система предлагает пользователю проверить правильность введённых данных в билете.

7.Клиент подтверждает, что данные корректны

9.Пользователь указывает что это «бронирование»

10.Система записывает бронь в базу данных

Альтернативный поток:

Нет свободных мест. Если во время выполнения основного потока обнаружится, что на данный сеанс нет свободных мест, система выдаёт сообщение, что нет свободных мест.

**Вариант использования “Снятия с брони”**

Данный вариант использования начинает исполняться, когда клиент хочет снять бронь.

Основной поток:

Данный вариант использования начинается, когда клиент сообщает пользователю-кассиру, что хочет снять с брони билет.

1. Пользователь выбирает возможность «Снять бронь»

2. Пользователь вводит номер билета или сканирует штрих-код

3. Если данный билет действительно существует, то пользователь может продолжить процедуру снятия с брони билета, иначе запускается альтернативный поток.

4. Система запрашивает подтверждение на отмену бронирования.

5. Пользователь соглашается с запросом системы.

6. Система отмечает в базе данных те места что были в билете как свободные.

Альтернативные потоки:

Введен неверный номер билета. Система выдает сообщение, что билет не найден.

**Вариант использования “Просмотр информации о сеансах”**

Данный вариант использования начинает исполняться, когда пользователь желает просмотреть информацию о сеансах в кинотеатре.

*Основной поток событий:*

Данный вариант использования начинается, когда клиент хочет узнать какие фильмы можно посмотреть в кинотеатре.

1.Система предоставляет пользователю возможность выбора фильма, времени сеанса и зала

2.Пользователь выбирает фильм из тех, что предоставляет система, выбрав из списка фильмов.

3.Пользователь выбирает время начала фильма из тех, что предоставляет система, выбрав из списка время сеанса.

4.Система в графическом виде предоставляет информацию о свободных местах в зале на выбранный пользователем сеанс.

Альтернативные потоки:

Нет.

**Вариант использования “Внесение информации”**

Данный вариант использования начинает исполняться, когда администратор хочет внести информацию о доступных местах в зале, скидках и расписании фильмов.

Основной поток:

1.Данный вариант исполняется, когда администратор выбирает возможность внесение информации.

2.Администратор заполняет или изменяет необходимые поля для внесения информации о фильме.

3. Система предлагает сохранить изменения в базе данных.

4. Администратор соглашается с запросом системы.

5.Система записывает в базу данных изменения, введенные администратором.

Альтернативные потоки:

Нет.

**Вариант использования “Внесение информации о сеансах”**

Данный вариант использования начинает исполняться, когда администратор хочет внести информацию о расписании фильмов, времени и зала.

Основной поток:

1.Данный вариант исполняется, когда администратор открывает вкладку внесение информации.

2.Администратор заполняет или изменяет необходимые поля внесения необходимой информации о фильме.

3. Система предлагает сохранить изменения в базе данных.

4. Администратор соглашается с запросом системы.

5.Система записывает в базу данных изменения, введенные администратором.

Альтернативные потоки:

Нет.

**Вариант использования “Внесение информации о скидках”**

Данный вариант использования начинает исполняться, когда администратор хочет добавить информацию о фильме со скидками.

Основной поток:

1. Данный вариант начинает исполняется, когда администратор открывает вкладку внесение информации.

3.Администратор выбирает и вводит необходимую информацию для внесения данных о скидках.

4. Система предлагает сохранить изменения в базе данных.

5. Администратор соглашается с запросом системы.

6.Система записывает в базу данных изменения, введенные администратором.

Альтернативные потоки:

Нет.

**Вариант использования войти в систему:**

Данный вариант использования описывает вход пользователя в систему кинотеатра.

Основной поток событий

Данный вариант использования начинает выполняться, когда пользователь хочет войти в систему кинотеатра.

1. Система запрашивает имя пользователя и пароль.
2. Пользователь вводит имя и пароль.
3. Система проверяет имя и пароль, после чего открывается доступ в систему и отправляет пользователя на вкладку формирование заказа.
4. Для выхода из системы пользователь может выбрать возможность выход в углу основного окна программы.
5. Для последующего входа в систему у пользователя появляется возможность войти в систему.

Альтернативные потоки

Неправильное имя/пароль.Если во время выполнения Основного потокаобнаружится, что пользователь ввел неправильное имя и/или пароль, система выводит сообщение об ошибке. Пользователь может вернуться к началу Основного потокаили отказаться от входа в систему, при этом выполнение варианта использования завершается.

# 1.3. Концептуальное проектирование базы данных

Концептуальное проектирование — создание БД, включающая в себя определение таблиц, но с высоким уровнем абстракции от модели бд и физической реализации.

Первым делом мы определяем какую информацию необходимо хранить в базе данных. Входные данные данного проекта:

• Информация о фильмах;

• Информация о залах.

• Информация о купленных билетах.

## 1.3.1 Перечень сущностей И АТРИБУТОВ.

Сущность — это некоторая абстракция реального мира, которая может существовать независимо. Сущность может иметь экземпляры, отличающиеся друг от друга значениями атрибутов и допускающие однозначную идентификацию. [2] В нашей системе кинотеатра будет храниться информация о фильмах и расписании сеансов.

Выделим базовые сущности данной предметной области:

1.сущность Movie(фильмы) содержит информацию о фильмах;

2. сущность Showtime (расписание сеансов) содержит информацию о сеансе;

3. сущность Ticket (билет) содержит сведения о приобретенном билете.

4. сущность Hall (зал) содержит сведения о залах в кинотеатре.

Атрибутом является свойство объекта сущности.[4]

Определим список атрибутов от каждой сущности:

1. Для сущности Movie атрибутами являются:

- movieID

-year\_movie

-title

-duration

-director

-genre

-Address

-cast\_movie

2. Для сущности Showtime атрибутами являются:

- showtime\_id

- date

- time

- price

- movieid

3. Для сущности Ticket атрибутами являются:

- ticket\_id

- hall\_id

- price

- seat

- shouwtime\_id

- role

4. Для сущности Roles атрибутами являются:

- role

- login

- password

5. Для сущности Hall атрибутами являются:

- hall\_id

- hall\_name

- seats

6. Для сущности Hall\_Showtime атрибутами являются:

- hSh\_id

- showtime\_id

- hall\_id

- seats\_map

Результат диаграммы (ER-модель) можно увидеть на рисунке 1.

Рисунок 1 – ER-модель

Чтобы пользователю иметь представление как будет выглядеть система, создаём и согласовываем макет интерфейса программы. Данный макет можно увидеть на рисунках 2,3,4,5 и 6. Где, рисунок 2 — это вкладка для смены пользователя, рисунок 3 — это вкладка формирования заказа, рисунок 4 — это вкладка просмотра информации о сеансе и рисунок 5 — это вкладка для внесения информации, рисунок 6 — это вкладка для возврата билета.

Рисунок 2 – Смена пользователя

Рисунок 3 – Формирование заказа

Рисунок 4- Информация о сеансе

Рисунок 5- Внесение информации

Рисунок 6 - возврат

**ГЛОССАРИЙ**

1. **PostgreSQL** — это свободная и открытая система управления реляционными базами данных, подчеркивающая расширяемость и соответствие SQL.
2. **Диаграмма вариантов использования** — представление взаимодействия пользователя с системой, которое показывает связь между пользователем и различными вариантами использования, в которых участвует пользователь.
3. **Интерфейс программы** — это общая граница, через которую два или более отдельных компонента компьютерной системы обмениваются информацией.
4. **ER-модель**— модель отношений между сущностями, описывает взаимосвязанные вещи, представляющие интерес в конкретной области знаний. Базовая модель ER состоит из типов сущностей и определяет отношения, которые могут существовать между сущностями
5. **Цифровизация** — процесс перехода предприятий или экономических отраслей на новые модели ведения бизнеса, основанных на информационных технологиях.

# Глава 2. рАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ.

# 2.1. подключение баз данных к Visual Studio, создание программы и форм

После того как схема базы данных была определена, переходим к созданию визуального приложения в среде Visual Studio. Для этого подключаем postgres к Visual Studio, используя entity framework. Entity Framework — это набор библиотек API для работ с бд, которое находит применение в программировании на языке C#. [3] Он позволяет взаимодействовать с СУБД с помощью сущностей, а не таблиц. Для того, чтобы использовать entity framework в нашем проект добавляем такой компонент как entity framework и NpgSql в менеджере пакетов NuGet. [1] Проверяем соединение и подключаем базу данных к проекту. Подключенную базу данных можно увидеть на рисунке 2.

Рисунок 3- соединение базы данных с Visual Studio

Для удобства использования приложение будет написано с оконным интерфейсом Windows Form. Forms представляет из себя диалоговое окно, являющейся пользовательским интерфейсом приложения. В Windows Forms формой является площадь окна, на которую выводится информация для пользователя. Чтобы построить приложение Windows Form, нужно поместить элементы управления на форму и написать код для обработки действий пользователя. [6] Windows Forms включает в себя множество элементов управления, которые можно использовать при создании приложения: текстовые поля, раскрывающиеся списки, кнопки, переключатели и даже веб-страницы. В данном приложении на основную форму помещены кнопки быстрого доступа, а весь основной функционал, такой как покупка билета, возврат билета, просмотр информации о сеансах реализован на вкладках, что позволяет нам быстро переключаться с одной задачи на другую.

В приложении “заказ билета в кинотеатр” созданы формы, которые представлены на рисунках 2,3,4 и 5.

Первая и самая главная форма — это вкладка для входа пользователя в систему, рисунок 2. Она позволяет нам сберечь систему от посторонних лиц и защитить от несанкционированных действий. Однако, перед тем как войти в систему, необходимо ввести логин и пароль, чтобы система открыла доступ определённых функций для определенного сотрудника. При входе в систему, система выводит картинки, которые показывают нам в какой роли мы вошли, что очень удобно для пользования и дальнейшей работы с программой.

После того, как был произведён вход в систему, пользователь переходит во вкладку с формированием заказа, рисунок 3, чтобы продать билет, но при этом ему доступна и другие вкладки для работы, а именно оформить заказ, посмотреть информацию о сеансе, вернуть билет и внести данные о фильме или же он может перейти через кнопки к необходимой ему вкладке.

На вкладке формирование заказа, которая представлена на рисунке 4, пользователям доступны две кнопки “забронировать билет” и “оформить заказ”, при помощи который можно забронировать и купить билет на определённое место и увидеть есть ли свободные места на определенный фильм, однако перед этим нужно сначала выбрать тот фильм, на который хотят приобрести билет, время сеанса, зал и место, после этого можно оформить заказ, который выводит купленный билет для пользователя. Если же пользователи после того, как закончили свою работу, хотят выйти из системы, они могут выбрать возможность выход в углу основного окна программы и им больше не будет доступна продажа билетов, внесение информации и их возврат.

На второй вкладке, представленной на рисунке 4, сотрудникам кинотеатра доступен просмотр текущих фильмов в определенное время и определённом зале. Рядом с ними находится кнопка “очистить”, куда нужно обратиться, чтобы очистить текущий фильм и посмотреть доступные места на другие фильмы. Тем самым, мы можем посмотреть свободные места на сеанс в удобном графическом виде.

На третьей вкладке, представленной на рисунке 5, доступно внесение информации о фильме и предоставляется скидка на билет для того или иного фильма, но лишь для администратора, так как потом могут возникнуть проблемы при продаже билетов.

На рисунке 6 можно увидеть, как вернуть купленный билет. Для этого пользователю необходимо ввести номер билета, который он может взять на вкладке оформления заказа и нажать на кнопку вернуть билет. В программе уже есть все некоторые данные о фильмах, поэтому нам доступны такие функции как возврат и снятие брони, а как их внести, удалить и произвести еще какие-либо действия над программой показано в подпункте 2.2.

# 2.2. КЛАССЫ ДЛЯ СОЗДАНИЯ, УДАЛЕНИЯ И ПОИСКА ДАННЫХ

Чтобы пользователь системы мог изменять, добавлять, удалять и находить данные в системе, необходимо создать методы и классы в Visual Studio. [5]

К примеру, на рисунке 7 представлена метод для поиска фильмов, а также класс на рисунке 8, который необходим для работы данного метода.

Рисунок 7 – метод для поиска фильмов по его названию

Рисунок 8 – класс для работы с базой данных.

На рисунке 9 показана работа класса:

Рисунок 9- выбор фильма для сортировки по его названию

На данной форме есть элемент tabcontrol, которая занимает большую часть формы. Ее основные свойства:

BackColor: MintcCream – цвет фона

Font: Microsoft Sans Serif; 8 pt – шрифт

Size: 963; 431 – размер панели

BorderStyle: none; - отображает сведения о стиле границы элементы управления

Name: tabControl1; - название панели

Modifiers: Public; - модификаторы

На этой панели есть 5 вкладок, на которых располагаются такие элементы управления как кнопки button, combobox, textbox, richtextbox и label. Рассмотрим кратко каждую из элементов. Основные свойства кнопки AllContact:

Button – Кнопка на которую пользователь нажимает мышкой

Combobox – поле для ввода текста с возможностью раскрытия списка подсказок

Textbox – поле для ввода текста пользователем

Richtextbox – поле для ввода- вывода большого количества текста

Label – тестовая надпись на форме

Рисунок 10- Результат работы программы

Алгоритм работы класса databaseAccess: создаем экземпляр базы данных db, который устанавливает связь с базой данных. Далее вызываем метод класса selectTimeByMovieTitle, который можно увидеть на рисунке 7. После этого следует обращение к базе данных с запросами о фильме, доступных сеансах и залах. Чтобы передать информацию, которая хранится в базе данных, нужно чтобы параметры совпадали. Такой алгоритм будет действовать во всех методах.

Далее идёт заполнение comboBox информацией, полученной из базы данных, принимая во внимание, что количество значений будет больше 1, используя цикл while и foreach.

Второй метод класса databaseAccess - GetTickenNummber. Этот метод нужен для определения наличия билета в базе данных с определённым номером. Создаём переменную result для хранения номера билета и присваиваем ей значение 0, что для нас будет индикатором, что билет не найден. Формируем и отправляем запрос в базу данных с параметрами seat, hall и showtime\_id, которые передаются при вызове метода. Если из базы данных вернулся номер билета, то записываем её в переменную result и возвращаем из метода, иначе возвращается 0. На рисунке 11 показан данный метод.

Рисунок 11 – метод для поиска билета

Третий метод класса databaseAccess - DeleteTicket. Как следует из названия, данный метод предназначен для удаления билета из базы данных. Объявляем переменные для хранения всех id базы данных, отправляем запросы в базы данных и получаем необходимые промежуточные параметры для формирования запроса на удаление билета из базы данных. Подготавливаем для удаления из списка занятых мест определенное место, приобретенное покупателем. Далее отправляем запрос на удаление билета в базу данных. Данный метод можно увидеть на рисунке 12. Результат работы этого метода можно увидеть на рисунке 13.

Рисунок 12 – метод DeleteTicket

Рисунок 13 –результат работы метода DeleteTicket

Для оформления заказа применяется событие btnPlaceOrder\_Click, которое срабатывает по кнопке “оформить заказ”. Всплывает окно уточнения «Проверьте правильность данных» (Рисунок 14).

Рисунок 14- окно уточнения

Нажав на кнопку «Yes», можно увидеть успешное добавление в базу данных, которое показано на рисунке 15.

Рисунок 15 – добавление

Далее открывается форма с билетом, на котором указан фильм, зал, место, время, цена и номер билета. Сформированный билет можно увидеть на рисунке 16.

Рисунок 16 – сформированный билет

Для передачи информации в открывающуюся форму с билетом, мы используем класс tickFRM, который содержит такие поля, которые необходимы для отображения картинки билета. Данный класс можно увидеть на рисунке 17.

Рисунок 17 – класс tickFRM

Для того, чтобы записать в базу данных новые фильмы и сеансы существует метод addMovie, который вызывается событием btnAddDataTab3\_Click. В метод передаются comboBox с необходимой для записи в бд информацией. Первым делом запрашиваем количество записей в таблице фильмов. Далее проверяем нет ли вводимого фильма в бд. Если такого фильма нет, мы добавляем, а иначе обновляем информацию о сеансах. Записываем данные в таблицу сеансов и залов, в которых идут фильмы. Далее добавляем пустой зал к сеансу и сохраняем изменение в базе данных. Данный метод можно увидеть на рисунке 18,19 и 20.

Рисунок 18 – метод addMovie

Рисунок 19 – метод addMovie

Рисунок 20 – метод addMovie

Так как логин и пароль у нас также находятся в базе данных, то для идентификации пользователя, мы используем метод loginFunc класса databaseAccess. Данный метод принимает строку логина и пароля из textbox формы входа, запрашивает в базе данных существует ли такая комбинация и, если такая комбинация находится в бд, он возвращает нам роль, с которой выполнен вход, в случае неудачи возвращается строка failed. Эти данные используются в последствии для указания роли заголовки окна. Данный метод можно увидеть на рисунке 21.

Рисунок 21 – метод идентификации пользователя

ЗАКЛЮЧЕНИЕ

Таким образом, в результате выполнения курсового проектирования, можно сказать о том, что было разработано визуальное приложение «Кинотеатр», а также была подключена и протестирована база данных, созданная в системе PostgreSQL. Поставленные цели и задачи были успешно выполнены.

Система “Кинотеатр” полностью готова к использованию сотрудникам кинотеатра, а также позволяет оперативно вносить и получать необходимую информацию о фильмах, доступ посторонним пользователям закрыт, что является необходимым для оперативной работы сотрудников кинотеатра.

Полученные знания в ходе работы над курсовым проектом явились хорошей практикой для курса визуального программирования на языке Visual Studio и PostgreSQL.

В заключении хочется сказать о том, что данное приложение позволит кинотеатру автоматизировать процесс поиска взаимодействия с клиентами, продавать билеты быстрее, избегая длинных очередей.