## Аннотация

## Дипломная работа посвящена разработке автоматизированной системы продажи билетов в кинотеатр на основе API контроллеров. В пояснительной записке описана предметная область, процесс бронирования билетов, концептуальное и физическое проектирование базы данных, целостность данных, безопасность, проектирование логики диалога с пользователем, алгоритмы обработки данных и разработка приложения. Цель работы — создание удобной и эффективной системы продажи билетов, обеспечивающей быстрый и комфортный процесс бронирования для посетителей кинотеатра.

## 

**Содержание**

[Введение 4](#_Toc167928795)

[1 Назначение и область применения 5](#_Toc167928796)

[2 Постановка задачи 6](#_Toc167928797)

[3 Описание программы](#_Toc167928801) 7

[3.1 Общие сведения](#_Toc167928802) 7

[3.2 Функциональное назначение](#_Toc167928803) 9

[3.3 Описание логической структуры](#_Toc167928804) 9

[3.4 Входные и выходные данные 1](#_Toc167928805)2

[4 Тестирование приложения](#_Toc167928806) 13

[4.1 Методы испытаний](#_Toc167928807) 13

[4.2 Тест-кейсы](#_Toc167928808) 13

[5 Руководство пользователя](#_Toc167928809) 17

[5.1 Работа с приложением](#_Toc167928810) 17

[5.2 Сообщение пользователю](#_Toc167928811) 34

[6 Мероприятия по информационной безопасности 3](#_Toc167928812)8

[6.1 Защита данных:](#_Toc167928813) 38

[6.2 Защита от несанкционированного доступа:](#_Toc167928814) 38

[Заключение](#_Toc167928816) 39

[Приложение А](#_Toc167928817) 40

[Источники, использованные при разработке](#_Toc167928822) 44

[Internet – ресурсы](#_Toc167928823) 44

### **Введение**

В современном мире киноиндустрия является одной из самых быстрорастущих и развивающихся отраслей экономики. Кинотеатры играют важную роль в организации досуга людей, предоставляя им возможность просмотра новых фильмов и получения ярких эмоций от посещения киносеансов. Однако, успешное функционирование кинотеатра невозможно без эффективного управления, которое обеспечивает оптимизацию всех процессов, связанных с обслуживанием посетителей и контролем над ресурсами.

Одним из ключевых факторов успешного управления кинотеатром является автоматизированная система учета данных, которая позволяет оперативно собирать, анализировать и хранить информацию о работе кинотеатра. В рамках данной дипломной работы будет рассмотрена проблема создания такой системы, а также ее внедрение и эксплуатация в условиях реального кинотеатра.

Целью данной работы является разработка автоматизированной системы учета данных для кинотеатра, которая позволит оптимизировать процесс управления и улучшить качество предоставляемых услуг. Для достижения этой цели необходимо решить ряд задач, среди которых: анализ существующих систем учета данных, выбор оптимального технического решения, разработка структуры базы данных, создание пользовательского интерфейса и тестирование системы на реальных данных.

## Актуальность данной темы обусловлена тем, что современные кинотеатры сталкиваются с необходимостью повышения эффективности управления и снижения издержек, связанных с обработкой большого объема данных. Автоматизированная система учета позволит решить эти проблемы, а также обеспечит оперативный доступ к информации, необходимой для принятия управленческих решений.

**1 Назначение и область применения**

## Область применения дипломного проекта включает кинотеатры различных форматов и размеров, от небольших местных залов до крупных сетевых кинотеатров. Автоматизированная система продажи билетов на основе API-контроллеров поможет оптимизировать процессы управления билетами, снизить издержки и улучшить взаимодействие с клиентами.

## Вот некоторые аналоги программ для продажи билетов:

## AppEvent — облачная CRM-система для продажи билетов;

## TicketTool.net — система онлайн бронирования и продажи билетов;

* TicketForEvent — сервис для продажи билетов на мероприятия;

### **2 Постановка задачи**

К разработанной в рамках данной работы информационной системе предъявляются следующие требования:

* Дизайн пользовательского интерфейса должен быть интуитивно понятен и удобен для пользователя.
* Должны быть реализованы уровни доступа (Администратор, Менеджер, Пользователь, Гость).
* Система должна иметь распределённую структуру с общей базой данных для обеспечения безопасности хранения и скорости доступа к информации.
* Наличие выполнения всех пунктов раздела «Требования к составу выполняемых функций», которые описаны в техническом задании.

## Среда разработки: «Visual Studio 2022», язык программирования: C#.

## 

### **3 Описание программы**

### **3.1 Общие сведения**

Наименование программы – «Синематограф».

Эксплуатационное назначение:

Разработанное в рамках данной работы ПО предназначено для:

* покупки билетов в кинотеатр на терминале ТЦ;
* продажи билетов на кассе кинотеатра;
* учета данных в системе;
* просмотра информации о киносеансах;
* продажи билетов на кассе кинотеатра;
* контролю билетов на входе в зал
* сбора и выдачи справочной информации;

Система «Синематограф» позволяет отслеживать и редактировать информацию о клиентах, билетах, киносеансах, фильмах и работниках, а также дает возможность обычным пользователям заказывать билеты на киносеансы.

Для неавторизованных пользователей приложение позволяет просматривать текущую информацию о сеансах.

Для авторизованных пользователей программа приложение позволяет просматривать текущую информацию о киносеансах и заказывать билеты на выбранные киносеансы.

Для персонала кинотеатров предусмотрена ролевое разделение прав.

Для Менеджера приложение позволяет просматривать, редактировать, удалять, добавлять данные в таблицу пользователи и просматривать такие таблицы как киносеансы, билеты, сотрудники, а также позволяет получить доступ к сканированию QR-кода на билетах.

Для Администратора приложение позволяет делать все вышеизложенное и осуществлять полную работу с всеми приобретенными билетами клиентов и со всеми киносеансами.

Данные сгруппированы в разрабатываемой системе следующим образом:

* список сотрудников (Id, Имя, Фамилия, Отчество, Возраст, Должность);
* список пользователей (Id, Имя, Фамилия, Отчество, Возраст, Почта, Роль в системе, Логин, Пароль);
* список фильмов (Id, Наименование, Описание, Возрастное ограничение, Превью, Жанр).
* список залов (Id, Номер зала, Кол-во мест в ряду, Кол-во рядов).
* список киносеансов (Id, IdФильма, IdЗала, Дата начала, Дата окончания)
* список билетов (Id, IdСеанса, IdПользователя, IdСотрудника, Ряд, Место, Цена, Дата продажи билета, Способ оплаты)

# Для исправной работы приложения требуются следующие аппаратное и программное обеспечение:

Поддерживаемые ОС:

* Windows 7 (x86 и х64)
* Windows 8 (x86 и х64)
* Windows Server 2008 R2 (x64)
* Windows Server 2012 (x64)

Поддерживаемые архитектуры:

* 32-разрядная
* 64-разрядная

Требования к оборудованию:

* Процессор мощностью 1,6 ГГц или выше
* 1 Гб ОЗУ (1,5 ГБ для виртуальной машины)
* 10 ГБ (NTFS) свободного дискового пространства
* Подключение сервера к Базе данных

Видеоадаптер, совместимый с DirectX 9 и поддерживающий разрешение экрана 1024 x 768 пикселей и выше.

### **3.2 Функциональное назначение**

В проектируемом продукте представлен следующий функционал:

* Просмотр информации о заказанных билетах.
* Просмотр, добавление, удаление, изменение данных о билетах, фильмах, киносеансах, сотрудниках, пользователях.
* Фильтровать и искать данные в вышеперечисленных таблицах.
* Возможность осуществлять заказ билетов.
* Возможность получать информацию о билетах с помощью QR-кода.
* Выдавать пользователю PDF-файл билета с QR-кодом для сканирования.

### **3.3 Описание логической структуры**

Логическая структура программы «Синематограф» состоит из нескольких основных компонентов, обеспечивающих получение данных, управление данными и выдачу вещественных данных пользователю. Ниже приводится подробное описание алгоритмов и функциональных блоков системы.

**Получение данных из БД**

**1. Отправка URL запроса на сервер**

* Пользователь выбирает какие данные должны отображаться в интерфейсе программы.
* На основе выбора создается нужный URL запрос и отправляется на сервер.

**2. Обработка URL запроса на сервере**

* После получения GET запроса сервер выполняет нужную операцию, написанную разработчиком в контроллере.
* Сервер упаковывает данные в JSON файл и отправляет обратно клиенту.

**3. Вывод данные пользователю**

* Полученный JSON файл дисереализуется в список сущностей.
* Список преобразуется в красивый интерфейс у пользователя.

**4. Обработка сообщения на сервере**

* Сервер принимает сообщение, проверяет его на наличие ошибок и корректность формата.

**Регистрация нового пользователя**

**1. Сбор данных пользователя**

* Пользователь заполняет форму регистрации, вводя ФИО, пароль, логин, возраст и адрес электронной почты.

**2. Проверка уникальности имени**

* Система проверяет уникальность введенного логина пользователя, сравнивая его с уже зарегистрированными логинами в базе данных.

**3. Хеширование пароля**

* Введенный пользователем пароль хешируется для обеспечения безопасности хранения.

**4. Сохранение данных в базе данных**

* Данные пользователя сохраняются в базе данных.

**5. Завершение регистрации**

* После добавления в базу данных учетная запись пользователя активируется, и он может войти в систему.

**Управление данными**

**1. Добавление зала**

* Пользователь нажимает на кнопку “Добавить зал”.
* Система открывает форму добавления зала где пользователь заполняет обязательные поля (Номер зала, кол-во рядов, кол-во мест в ряду), после заполнения пользователь нажимает на кнопку “Сохранить”.
* Система делает URL запрос серверу и проверяет есть ли зал с таким же номером. При положительном ответе пользователю выдается информация о невозможности добавления зала с таким номером.
* После успешной проверки система отправляет URL запрос серверу содержащий JSON файл с новым залом.
* Сервер проводит валидацию данных после чего добавляет зал в базу данных.
* Сервер отправляет новый зал обратно клиентскому приложению.
* После получения ответа, пользовательский интерфейс обновляется с учетом нового зала.

**2. Удаление зала**

* Пользователь может удалить зал, отправив соответствующий запрос на сервер.
* Система открывает форму подтверждения удаления для пользователя
* В случае подтверждения система делает URL запрос на удаление зала из базы данных.
* Сервер обрабатывает запрос на удаление и отправляет ответ.
* Данные в пользовательском интерфейсе обновятся.

**Архитектура и взаимодействие**

Программа реализует клиент-серверную архитектуру. Клиентская часть разработана с использованием технологии WF и выполняется на компьютерах под управлением ОС Windows. Серверная часть взаимодействует с базой данных MsSQL, обеспечивая хранение и обработку данных.

Для управления базой данных используется SQL, а взаимодействие с ней осуществляется через CRUD контроллеры, которые в свою очередь используют ORM систему. Для администрирования базы данных может быть использован любой удобный инструмент. Клиентская программа написана на языке C# и использует фреймворк .NET для реализации функционала.

### **3.4 Входные и выходные данные**

К входным данным относятся следующее объекты: Имена, Фамилии, Возраст, Наименования, Стоимость, Почты, Уникальные идентификаторы и.т.п. Информация организована в виде вводимого в форму текста с использованием масок и других ограничений, соответствующих требуемому вводу. Данные хранятся в базе данных для дальнейшего просмотра и корректировки.

К выходным данным относятся: Информация о киносеансах, билетах, пользователях, фильмах и сотрудниках, а также билеты в виде PDF файла. Данные приходят в программу в виде классов, а в дальнейшем организуются в виде табличной формы либо дизайнерской панели.

### **4 Тестирование приложения**

### **4.1 Методы испытаний**

Испытания проводятся для проверки функциональности, надежности и безопасности приложения. В соответствии с задачами, поставленными в разделе 2, методы испытаний включают:

**1. Проверка всех функций**:

* Отправка и получение данных
* Управление данными
* Авторизация и регистрация
* Выдача билетов пользователям

**2. Нагрузочное тестирование**:

* Проверка производительности при высокой нагрузке на сервер

**3. Тестирование на отказоустойчивость**:

* Поведение приложения при сбоях сервера
* Обработка ошибок сети

**4. Проверка безопасности**:

* Хеширование и проверка паролей

### **4.2 Тест-кейсы**

Таблица 1 - Тест работы всех функций приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тест-кейса | 01 | | |
| Название: | Тест работы всех функций приложения | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден |
| Предусловие: | | | |
| Запустить сервер с БД, открыть приложение, выполнить авторизацию | | Приложение запущено без ошибок | Пройден |
| Шаги теста: | | | |
| Выбрать из меню слева любой раздел и открыть его | | Раздел открыт | Пройден |
| Повторить предыдущий шаг для всех существующих разделов | | Все разделы работают и открываются | Пройден |
| Постусловие: | | | |
| Закрыть приложение, выйти из OS | | Приложение закрыто | Пройден |

Таблица 2 - Тест работоспособности интерфейса

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тест-кейса | 02 | | |
| Название: | Проверка отображения интерфейса | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден |
| Предусловие: | | | |
| Запустить приложение | | Приложение запущено без ошибок, интерфейс отображается корректно | Пройден |
| Шаги теста: | | | |
| Проверить отображение основных элементов интерфейса (меню, кнопки, текстовые поля) | | Все элементы отображаются корректно | Пройден |
| Проверить корректность шрифтов и цветов | | Шрифты и цвета соответствуют дизайну | Пройден |
| Постусловие: | | | |
| Закрыть приложение | | Приложение закрыто | Пройден |

Таблица 3 – Тестирование входа в систему

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер тест-кейса | 03 | | | | |
| Название: | Отправка текстового сообщения | | | | |
| Действие | | Ожидаемый результат | | | Результат теста:  пройден |
| Предусловие: | | | | | |
| Запустить приложение | | Приложение запущено | | | Пройден |
| Шаги теста: | | | | | |
| Ввести корректные учетные данные и нажать "Войти" | | Пользователь успешно вошел в систему | | | Пройден |
| Постусловие: | | | | | |
| Выйти из системы | | | Приложение закрыто | Пройден | |

Таблица 4 – Получение и отображение корректной информации

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тест-кейса | 04 | | |
| Название: | Получение корректных данных о залах | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден |
| Предусловие: | | | |
| Авторизоваться в приложении | | Успешная авторизация | Пройден |
| Наличие данных о залах в базе данных | | Более 2 записей | Пройден |
| Шаги теста: | | | |
| Выбрать страницу залы | | Открылась страница с информацией о залах | Пройден |
| На странице должна корректно отображаться информация о залах из базы данных в виде таблице | | Данные отображаются корректно и в полном объёме | Пройден |
| Постусловие: | | | |
| Выйти из системы | | Приложение закрыто | Пройден |

Таблица 5 – Тест удаления зала

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тест-кейса | 05 | | |
| Название: | Удаление зала | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден |
| Предусловие: | | | |
| Авторизоваться в приложении | | Успешная авторизация | Пройден |
| Наличие данных о залах в базе данных | | Более 2 записей | Пройден |
| Шаги теста: | | | |
| Открыть раздел "Залы" и нажать "Удалить" | | Открыто дилоговое окно удаления | Пройден |
| Нажать “Да” | | Успешное удаление | Пройден |
| Корректное отображение информации с учетом удаленной записи | | Информация должна быть отображена с учетом удаленной записи | Пройден |
| Постусловие: | | | |
| Выйти из приложения | | Приложение закрыто | Пройден |

### **5 Руководство пользователя**

### **5.1 Работа с приложением**

Для запуска приложения:

1. Запустить приложение TicketSelling-API для поднятия сервера.
2. Запустить программу WinFormUserInterface.UI (либо существующий .exe файл, либо скачать программу и запустить созданный exe файл).

После запуска приложения открывается форма авторизации(Рис.1)

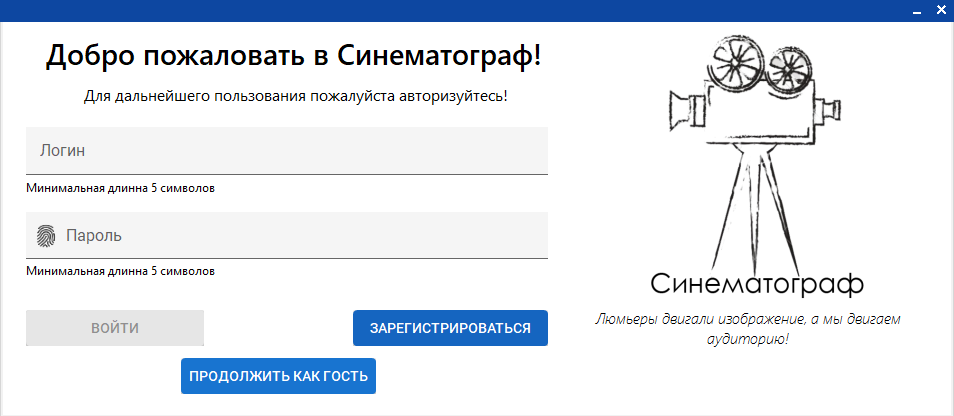


Рисунок 1 – Форма авторизации

Кнопка “Зарегистрировать” переносит нас на форму регистрации(Рис.2), которая позволяет новому пользователю зарегистрироваться и войти, а приложение.

Кнопка “Войти как гость” позволяет пользователю войти в приложение без регистрации что существенно сказывается на функционале приложения.

Кнопка “Войти” позволяет перейти к главному окну (Рис.3) программы, в случае ввода правильного логина и пароля.

Главная форма представлена 8 вкладками с различной информацией и функционалом. Панель вкладок выдвижная.

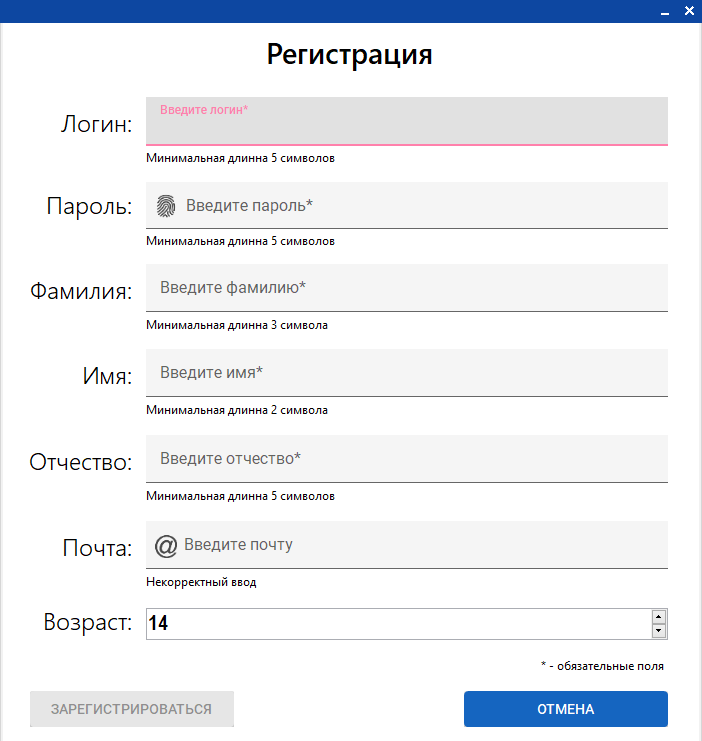


Рисунок 2 – Форма регистрации

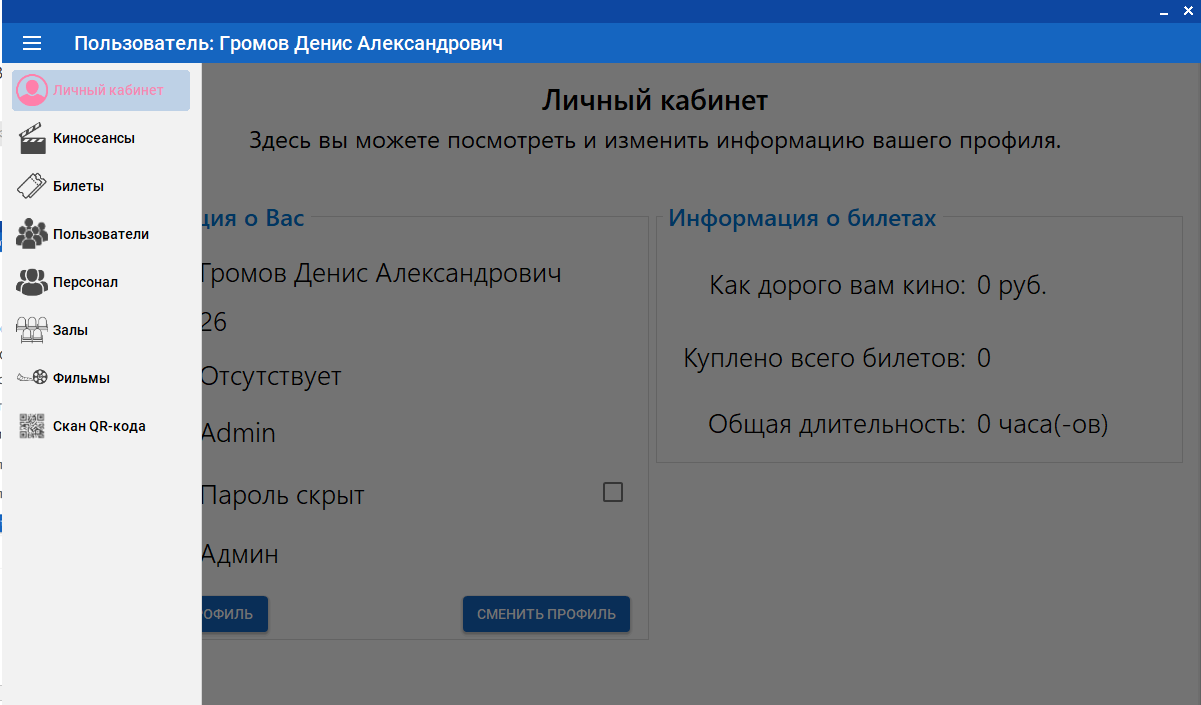


Рисунок 3 – Главная форма приложения

В зависимости от роли пользователя кол—во и порядок вкладок может меняться. Пример на рисунке 3 взят при использовании системы Администратором.

Первая вкладка на которую попадает пользователь — это “Личный кабинет” (Рис. 4), здесь собрана краткая информация о пользователе и предоставлена возможность изменить свои данные либо зайти в другой аккаунт.

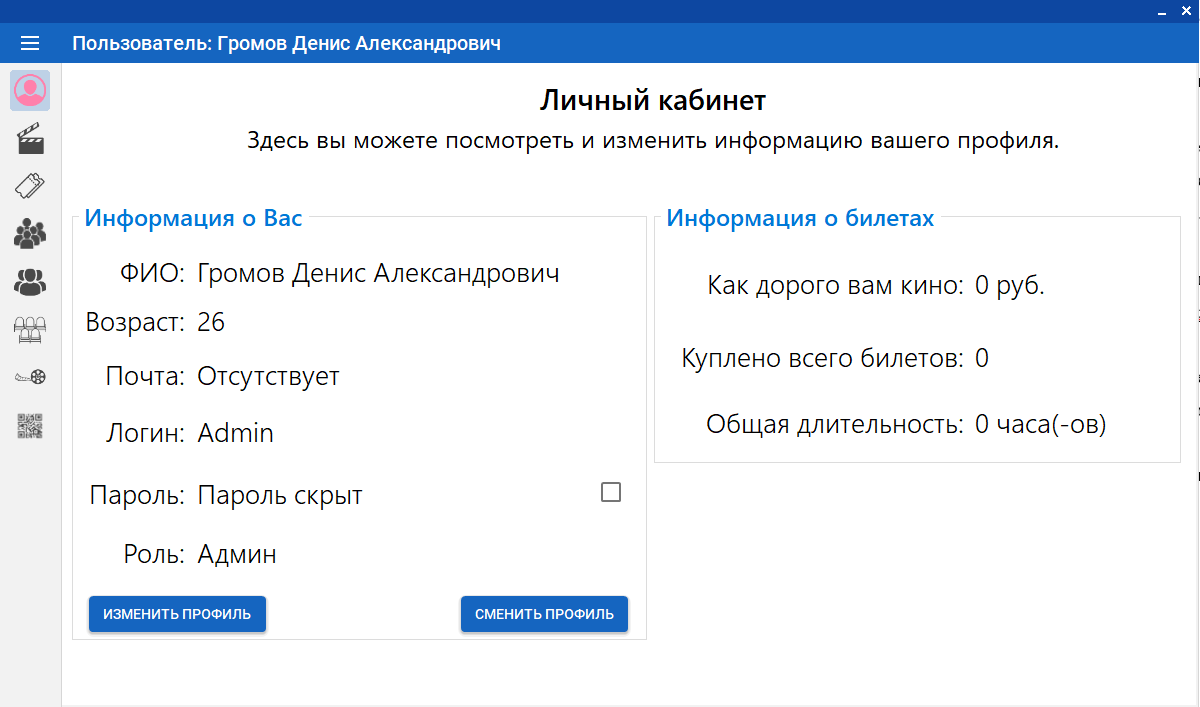


Рисунок 4 – Вкладка “Личный кабинет”

При нажатии на кнопку “Сменить профиль” программа перезапустится и вернет пользователя на окно авторизации (Рис.1).

При нажатии на кнопку “Изменить профиль” открывается окно изменения профиля (Рис.5), которое сразу же заполняется текущими данными.

При нажатии на кнопку “Сменить профиль” программа перезапустится и вернется на окно авторизации (Рис.1).

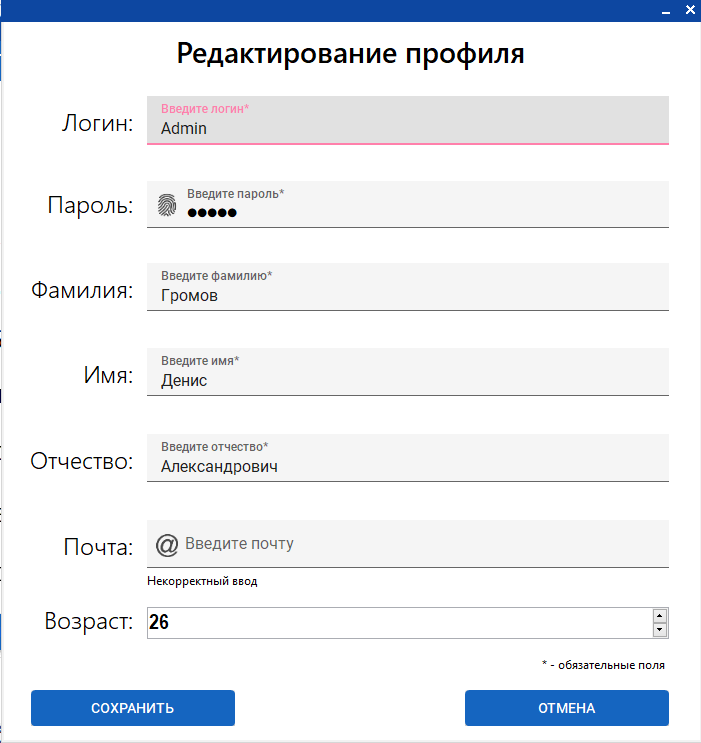


Рисунок 5 – Окно “Редактирование профиля”

Вторая вкладка, представленная пользователю это “Сеансы” (Рис.6), здесь обычный пользователь или менеджер может посмотреть, что идет в кинотеатре и заказать билет, а администратор редактировать эти сеансы. Так же на данной вкладке присутствуют фильтрация и поиск для более удобного поиска нужного пользователю фильма.

Исключительно для Администратора присутствует кнопка “Удалить прошедшие сеансы” при нажатии на которую из БД удаляются все прошедшие сеансы и билеты к ним.

Кнопка Добавить переносит админа в окно “Добавление сеанса” (Рис.7). В данном окне он может выбрать фильм для показа, номер зала (при выборе зала админа видит его загруженность, чтобы не поставить сеанс в одно и тоже время вместе с другим), а также дату и время начала и конца сеанса (сеанс длится от 1 до 3 часов).

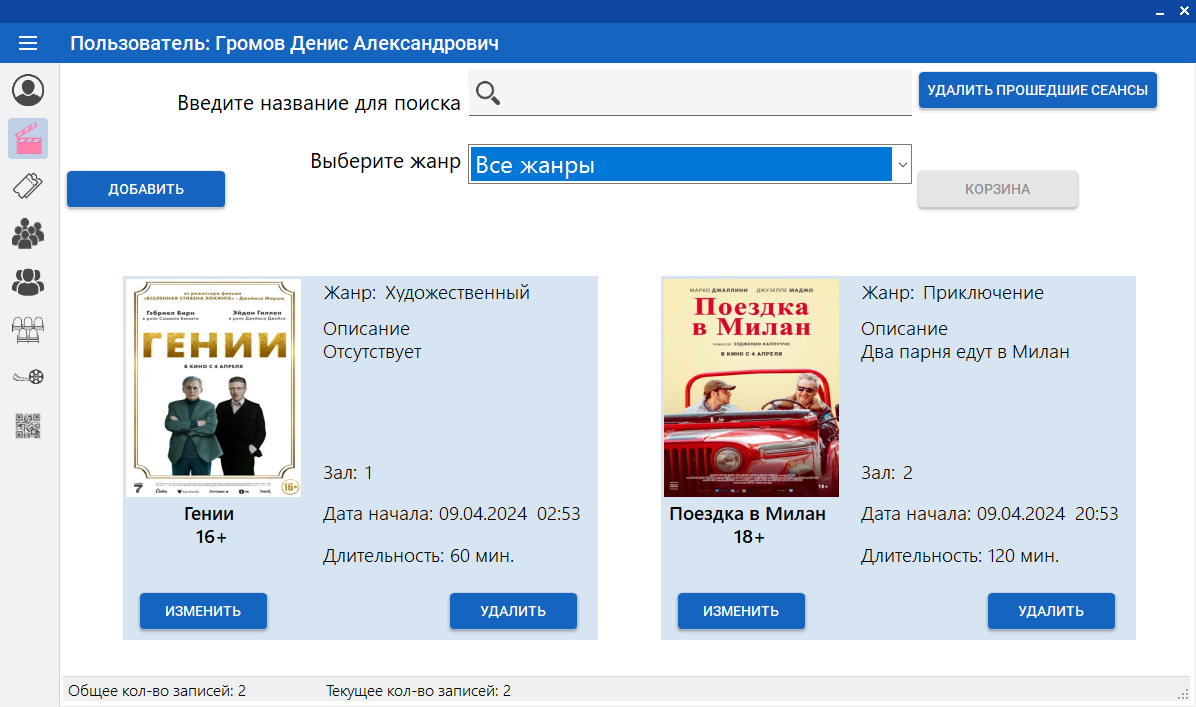


Рисунок 6 – Вкладка “Сеансы”

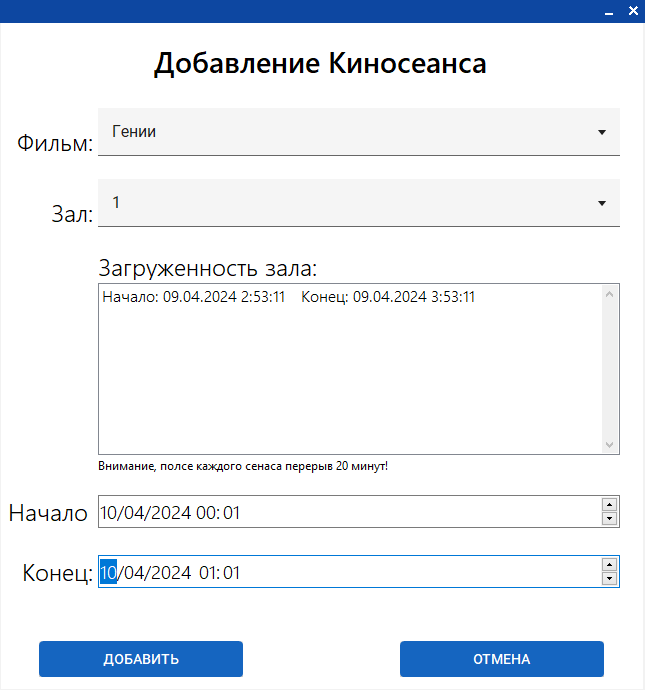


Рисунок 7 – Окно “Добавление киносеанса”

Так же на вкладке “Киносеансы” представлена возможность удалять сеансы. При нажатии на кнопку “Удалить” система оповестит админа если на данный сеанс куплены билеты и попросит подтвердить его, что он берете ответственность за его удаление.

Кнопка “Изменить” открывает окно “Изменения сеанса” (Рис.8) где админ может редактировать выбранный сеанс.

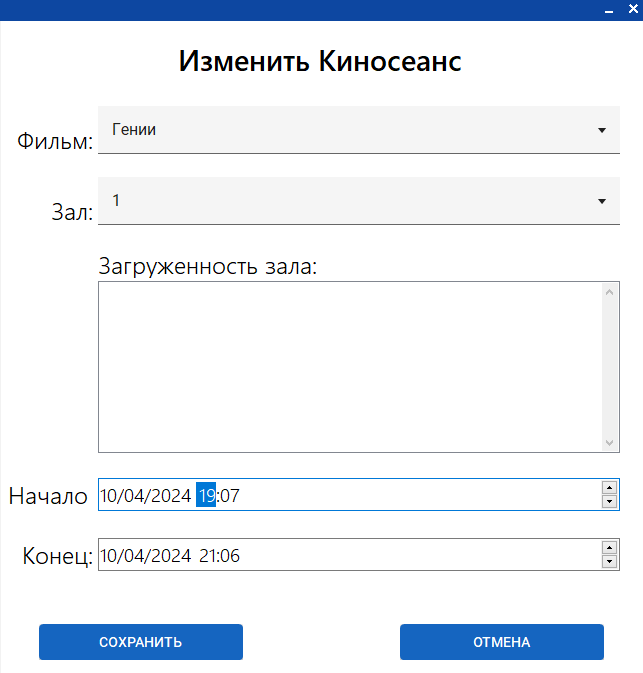


Рисунок 8 – Окно “Изменить киносеанс”

Для пользователей и менеджеров присутствует возможность заказывать билеты. После нажатия на кнопку, откроется окно (Рис.9), которое поможет выбрать место в зале (знаком “-” отмечены занятые места).

Далее можно перейти в корзину нажав на одноименную кнопку. Перед пользователем откроется окно (Рис.10) где он может посмотреть текущие билеты, общую сумму, а также выбрать способ оплаты и возможного сотрудника который оказывал помощь в покупке.

После нажатия на кнопку “Оплатить” производится запись билетов в БД и в PDF файл, который сохраняется в выбранное пользователем место (Рис.11).

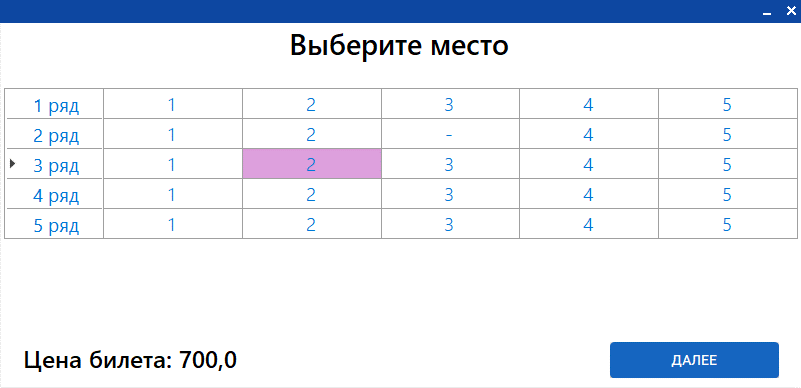


Рисунок 9 – Окно выбора места в зале

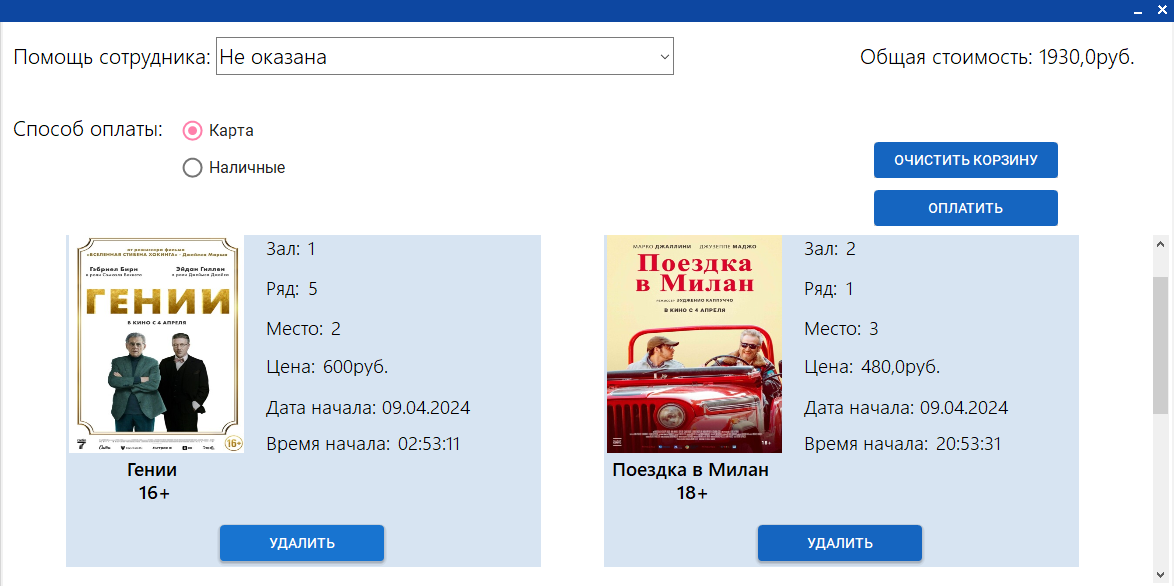


Рисунок 10 – Корзина



Рисунок 11 – PDF файл билета

Третья вкладка “Билеты” (Рис.12) предоставляет возможность просматривать все билеты из БД, а также удалять (только Администратору).

При нажатии на кнопку посмотреть информацию открывается окно “Информация о билете” (Рис.13) в котором присутствует вся информация о билете и составляющих в нем.

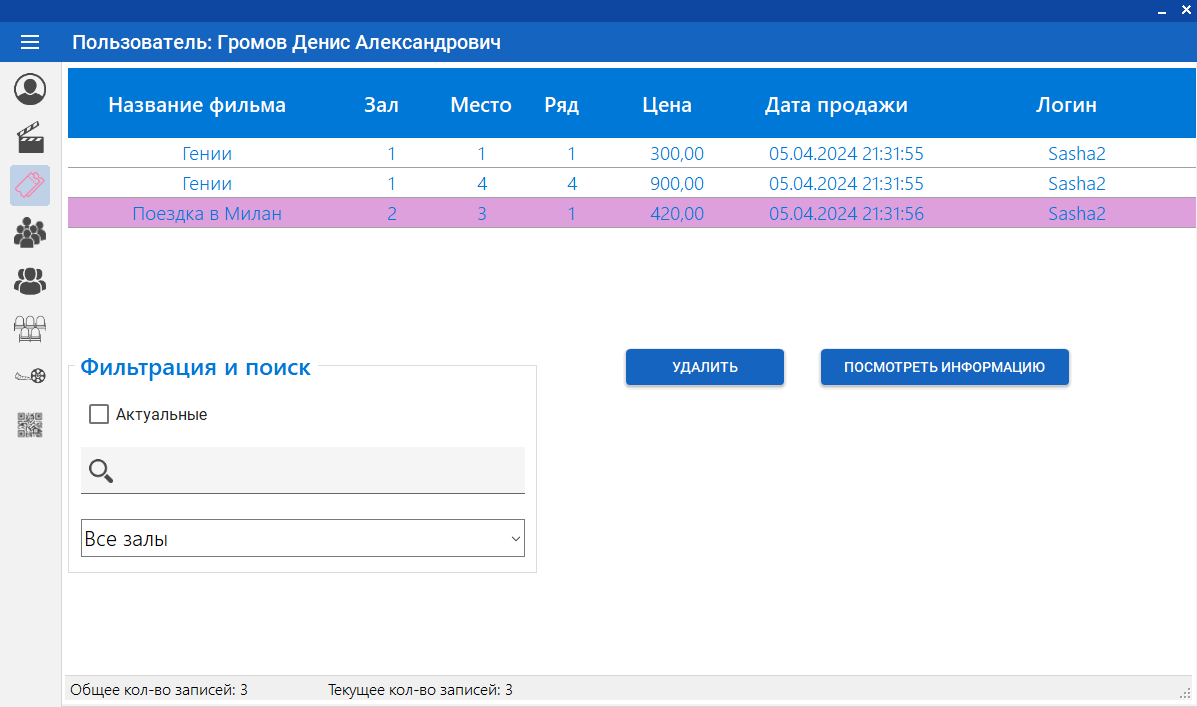


Рисунок 12 – Вкладка “Билеты”

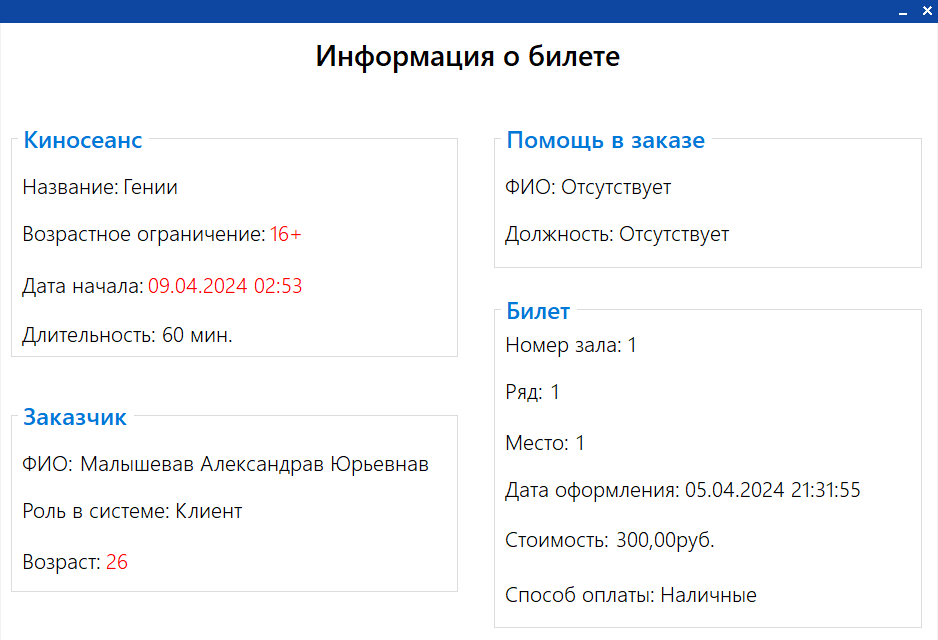


Рисунок 13 – Окно “Информация о билете”

Вкладка под номером 4 называется “Пользователи” (Рис.14). С помощью администратор или менеджер имеют возможность просмотреть всех пользователей в системе, а также управлять этими записями.

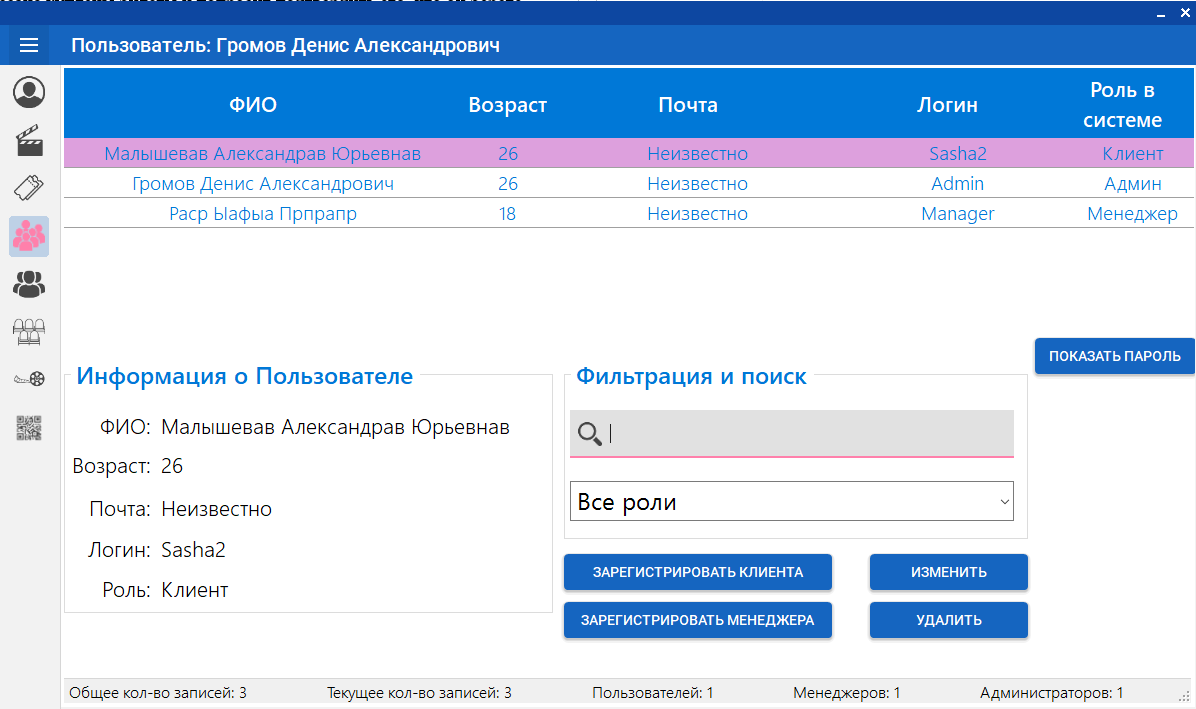


Рисунок 14 – Вкладка “Пользователи”

Кнопка “Зарегистрировать клиента” переносит менеджера на форму регистрации (Рис.2) после чего при успешном заполнении полей пользователь добавляется в систему.

Кнопка “Зарегистрировать менеджера” доступна только админу, она позволяет регистрировать пользователей с такими ролями как менеджер и администратор. После нажатии на кнопку приложение переносит администратора на форму регистрации менеджеров (Рис.15) после чего при успешном заполнении полей пользователь добавляется в систему.

Кнопка “Изменить” открывает окно “Редактирование профиля” (Рис.5) после чего при успешном заполнении полей информация о пользователе обновляется.

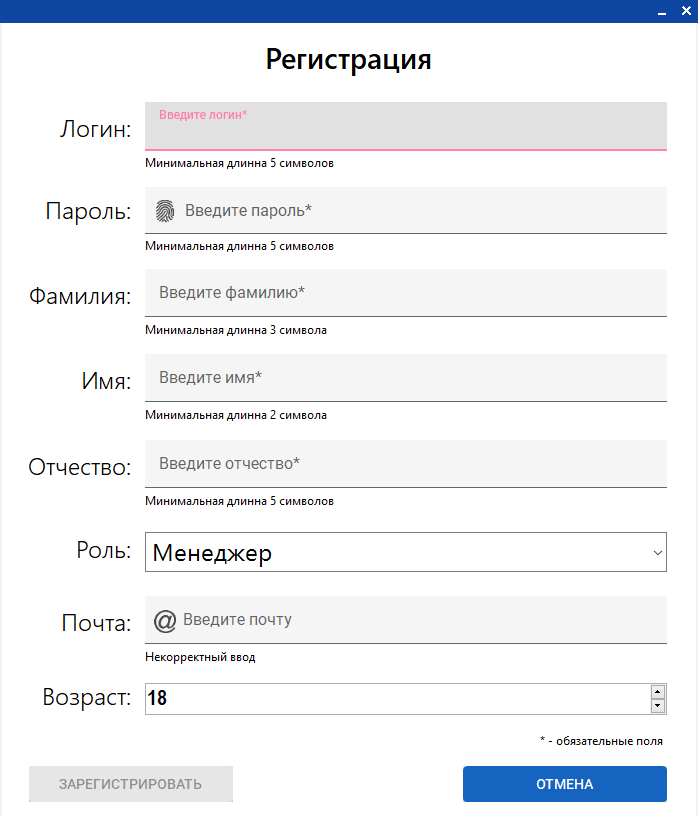


Рисунок 15 – Форма регистрации менеджеров и администраторов

Далее идет вкладка “Сотрудники” (Рис.16), она позволяет получать и редактировать информацию о Сотрудниках кинотеатра. Менеджер может только просматривать и фильтровать данные в таблице.

Кнопка “Удалить” открывает окно подтверждения удаления и при положительном решении удаляет сотрудника из БД.

Кнопка “Добавить” открывает окно “Добавить сотрудника” (Рис.17).

Кнопка “Изменить” открывает окно “Изменить сотрудника” (Рис.18) с уже заполненными данными.

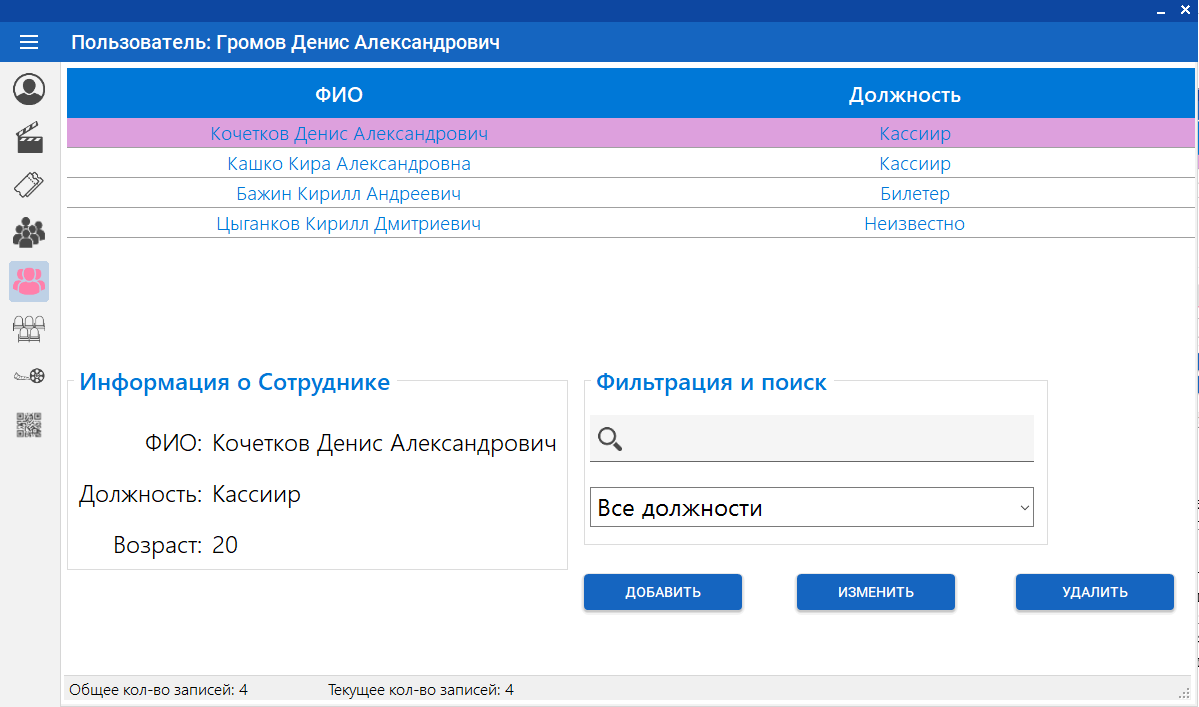


Рисунок 16 – “Вкладка сотрудники”

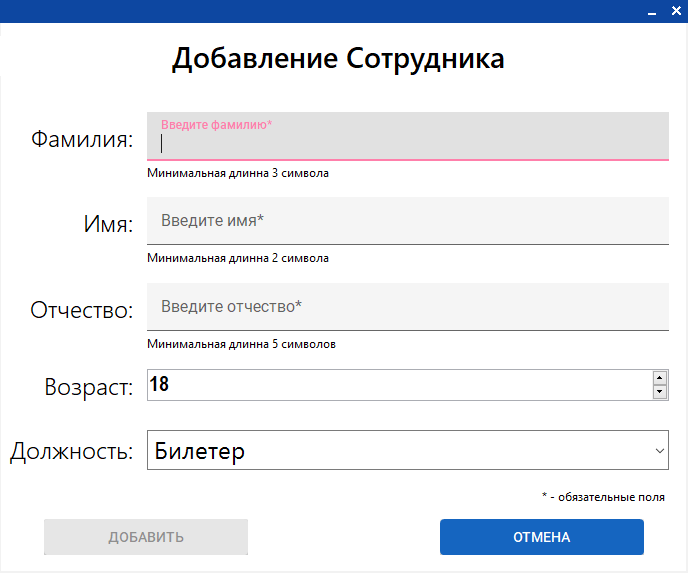


Рисунок 17 – Окно “Добавить сотрудника”

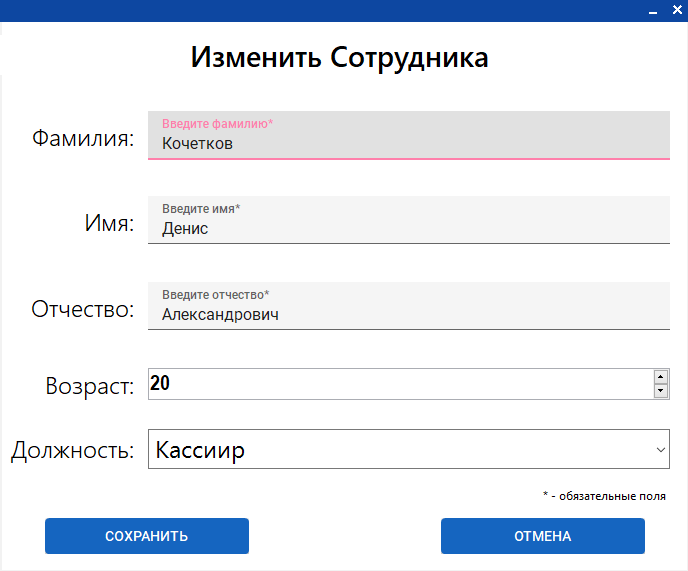


Рисунок 18 – Окно “Изменить сотрудника”

Следующая вкладка “Залы” (Рис.19) доступна только администраторам, она позволяет получать и изменять информацию связную с залами в кинотеатре.

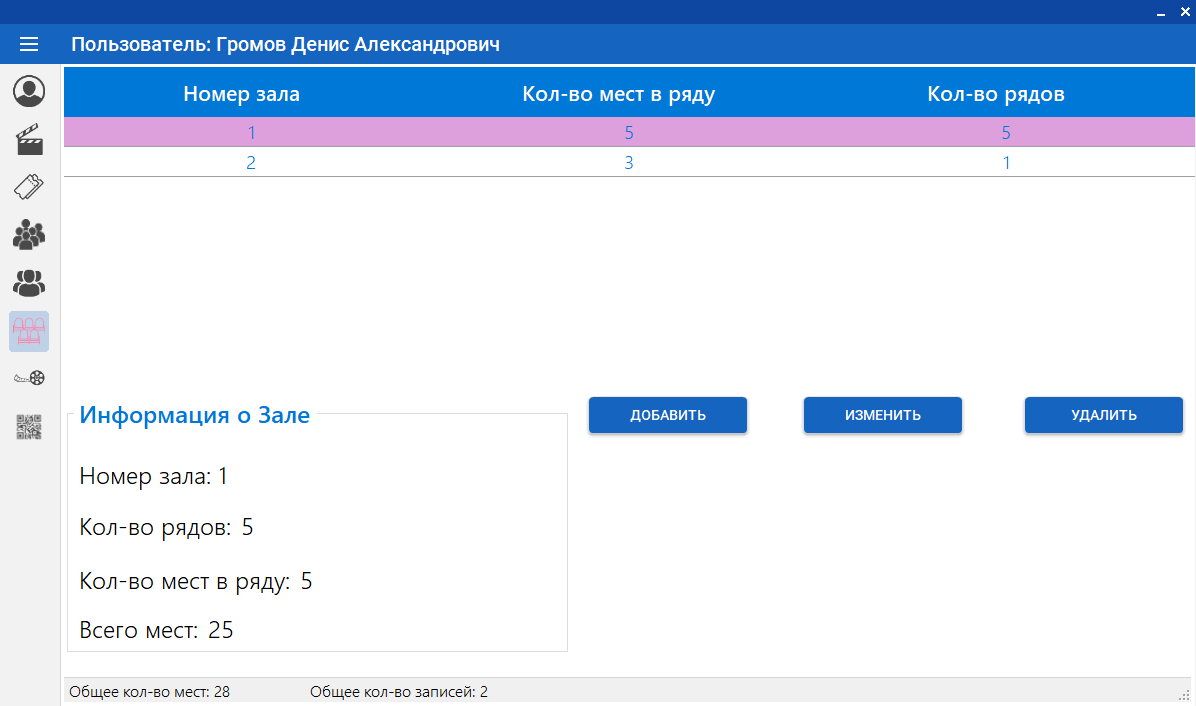


Рисунок 19 – Вкладка “Залы”

На данной вкладке доступны кнопки управления залами.

Кнопка “Удалить” открывает окно подтверждения удаления и при положительном ответе удаляет из БД не только зал, но также билеты и сеансы, связанные с ним, это сделано для исключения купле продажи билетов на отменённые сеансы.

Кнопка “Добавить” открывает окно “Добавление зала” (Рис.20) в котором нужно заполнит необходимые поля и нажать на кнопку “Добавить”.

Кнопка “Изменить” открывает окно “Изменение зала” (Рис.21) в котором заранее заполнены текущие данные. Администратору остается лишь внести изменения и нажать на кнопку “Сохранить”.

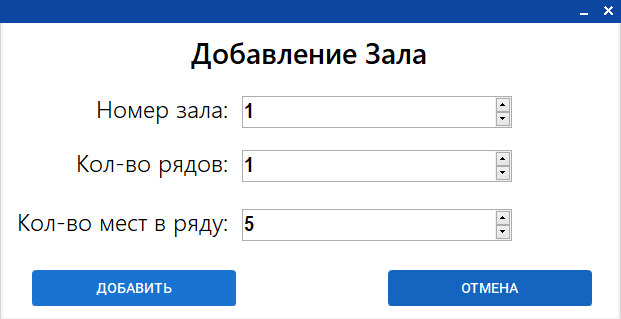


Рисунок 20 – Окно “Добавление зала”

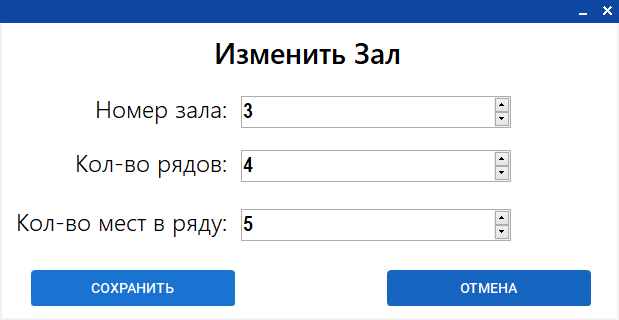


Рисунок 21 – Окно “Изменение зала”

Предпоследняя вкладка “Фильмы” (Рис.22) позволяет администраторам просматривать и редактировать информацию о текущих фильмах в кинотеатре.

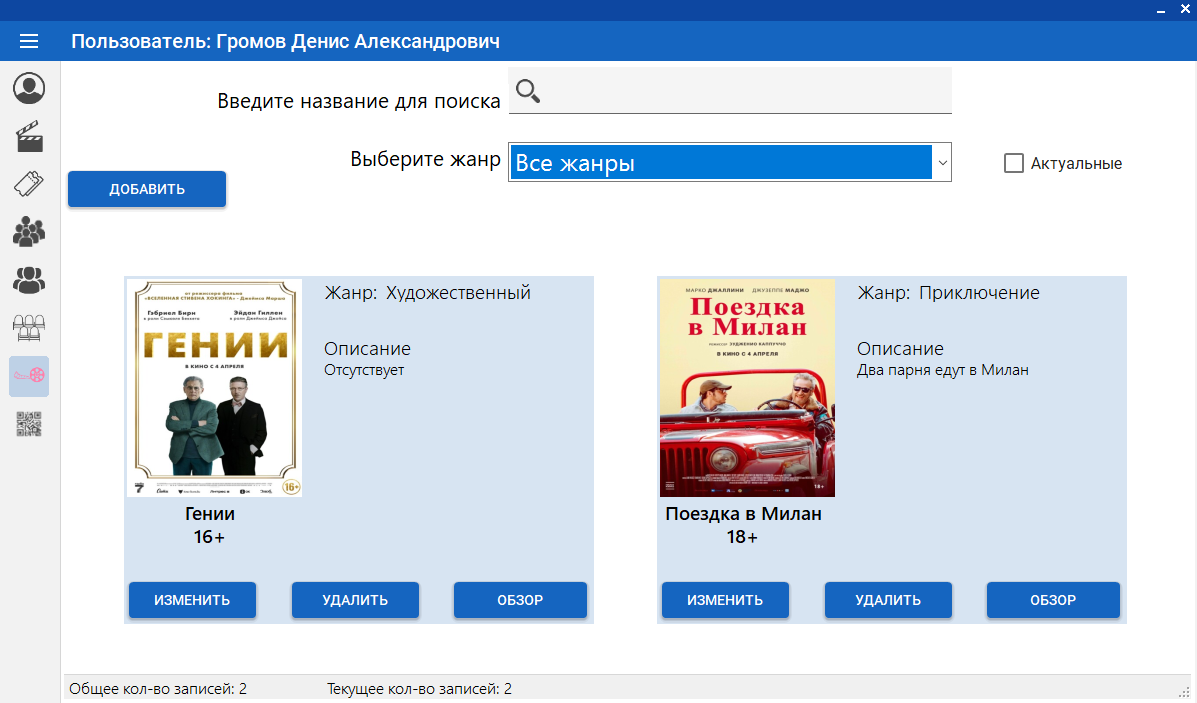


Рисунок 22 – Вкладка “Фильмы”

Кнопка “Удалить” открывает окно подтверждения удаления и при положительном ответе удаляет из БД не только фильм, но также билеты и сеансы, связанные с ним, это сделано для исключения купле продажи билетов на отменённые фильмы и проведение сеансов на фильмы без лицензии.

Кнопка “Обзор” открывает окно выбора изображения с последующей установкой его на превью фильма.

Кнопка “Добавить” открывает окно “Добавления фильма” (Рис.23) в котором нужно заполнить необходимые поля и нажать на кнопку “Добавить”, после чего фильм появится в списке.

Кнопка “Изменить” открывает окно “Изменение фильма” (Рис.24) при открытие которого все необходимые поля уже заполнены. Администратору остаётся лишь поменять необходимые поля и нажать на кнопку “Сохранить”. В случае успешного изменения, изменяться и сессии, связанные с этим фильмом.

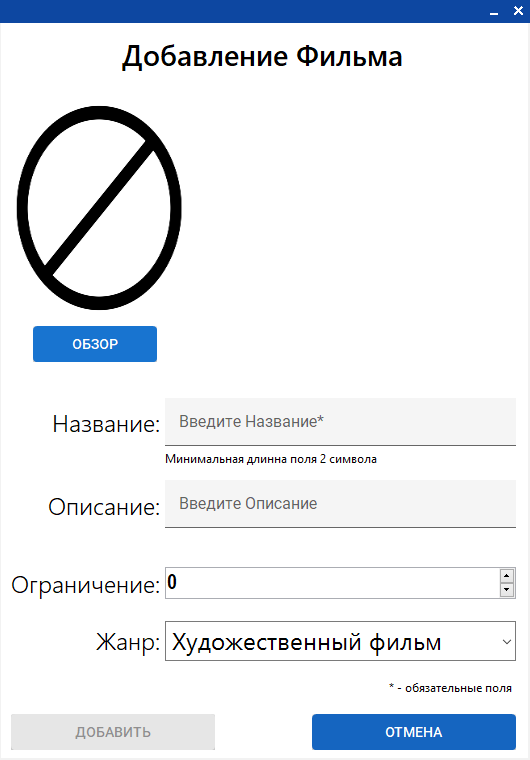


Рисунок 23 – Окно “Добавление фильма”

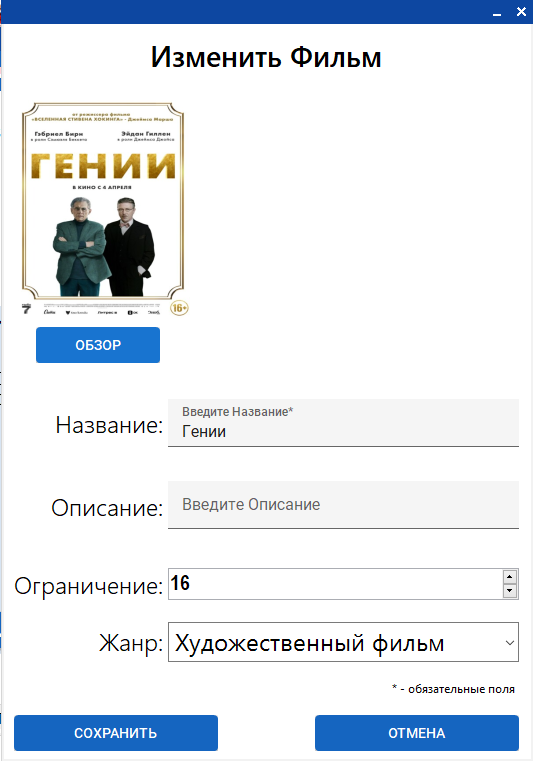


Рисунок 24 – Окно “Изменение фильма”

Последняя вкладка “Скан QR-кода” позволяет перейти в окно “Сканирование QR-кода” (Рис.25) нажав на кнопку “Начать сканирование” (в случае имеющихся в системе видеокамер).

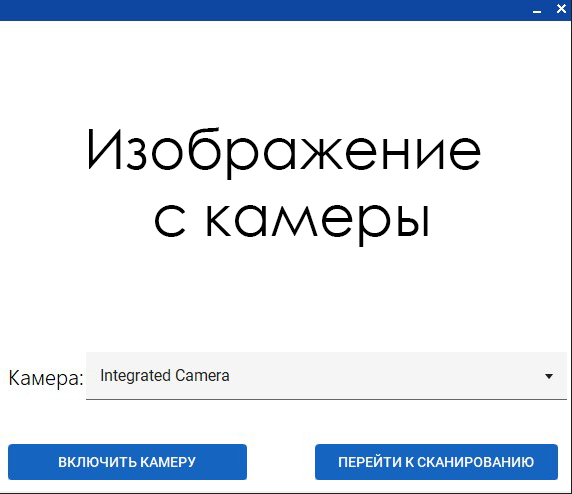


Рисунок 25 – Окно “Сканирование QR-кода”

После нажатия на кнопку “Включить камеру” появляется изображение с той камеры которая выбрана из списка. При нажатии на кнопку “Перейти к сканированию” появляется уведомление что сканирование активно, после чего можно сканировать QR-коды. При успешном нахождении билета пользователь получит сообщение о самой новой информации (в случае экстренных изменений) о его билете в виде окна “Информация о билете” (Рис.13), а в случае отсутствия билета в БД пользователь получит уведомление об этом. При отсутствии билета следует обратиться к менеджеру.

### **5.2 Сообщение пользователю**

В случае безуспешной авторизации пользователю будет показана ошибка авторизации (Рис.26). После ее прочтения пользователь может попробовать авторизоваться еще раз.

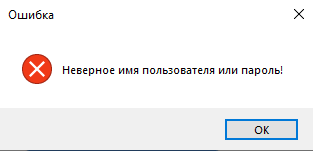


Рисунок 26 – Ошибка авторизации

В случае успешной авторизации пользователю будет показано окно приветствия (Рис.27). После его закрытия пользователь попадет на главную страницу приложения.

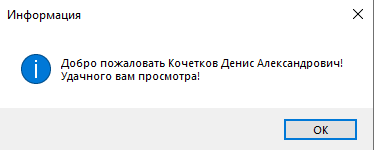


Рисунок 27 – Окно приветствия

В случае успешной регистрации пользователю будет показано окно, сообщающее об успешной авторизации (Рис.28). После его закрытия пользователь попадет на главную страницу приложения.

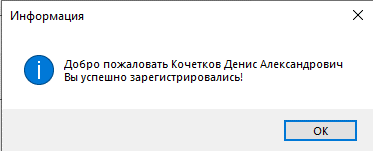
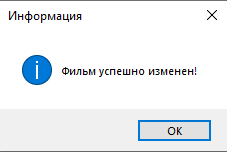


Рисунок 28 – Окно успешной авторизации

В случае добавления либо изменения данных в таблицах пользователю будут показаны окна сообщающее о проведенной операции, в качестве примера взято окно при работе с фильмами (Рис. 29).



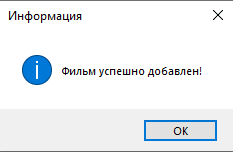


Рисунок 29 – Информационное окно

В случае удаления данных из таблиц пользователю будет показано диалоговое окно, требующее подтверждение операции, в качестве примера взято окно при удалении фильма (Рис.30). После подтверждения операции, данные удалятся из БД.

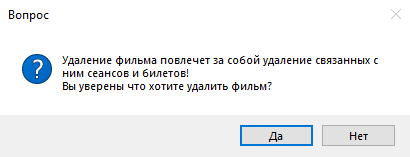


Рисунок 30 – Окно подтверждения удаления

При добавлении билета в корзину пользователю будет показано сообщение об успешном добавлении (Рис.31).

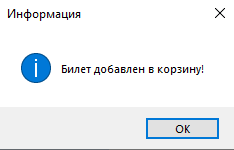


Рисунок 31 – Информативное окно добавления товара в корзину

При попытке удаления товара из корзины пользователю будет показано диалоговое окно подтверждения (Рис. 32).

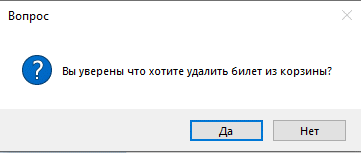


Рисунок 32 – Окно подтверждения удаления товара из корзины

После успешной оплаты заказа пользователь увидит окно подтверждающее успешность операции (Рис.33).

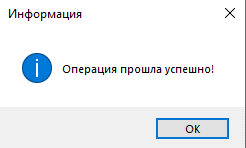


Рисунок 33 – Окно успешной операции

При попытке активировать сканер QR-кода без подключения камеры, пользователь получит уведомление (Рис. 34).

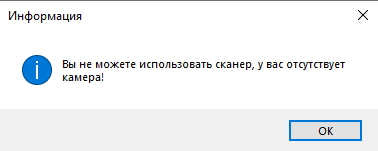


Рисунок 34 – Уведомление об отсутствии камеры

При попытке заказать билет с нарушением возрастного ограничения, пользователь получит уведомлении о нарушении и невозможности заказать билет на данный сеанс (Рис. 35).

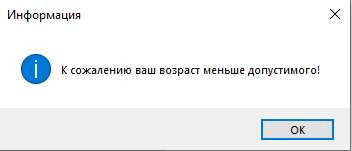


Рисунок 35 – Уведомление о нарушении возрастного лимита

При попытке изменить данные на уже существующие, пользователю будет показано уведомление о невозможности операции. В качестве примера приведено изменение профиля (Рис. 36).

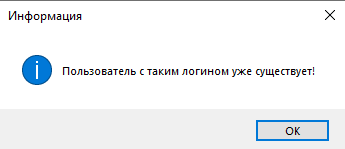


Рисунок 36 – Уведомление об уже существующих данных

При попытке добавления пользователем киносеанса при отсутствии фильма либо зала, пользователь увидит уведомление о невозможности добавления киносеанса (Рис.37).

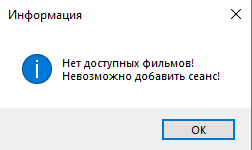


Рисунок 37 - Уведомление о невозможности добавления киносеанса

### **6 Мероприятия по информационной безопасности**

### **6.1 Защита данных:**

Для обеспечения безопасности информации в программе "Синематограф" будут применены следующие мероприятия:

* Хеширование паролей с использованием алгоритмов безопасности.
* Настройка приватности для контроля доступа к сообщениям.
* Применение паттерна Anti-Corruption Layer

### **6.2 Защита от несанкционированного доступа:**

Принимаются меры для предотвращения несанкционированного доступа к программе:

* Механизм аутентификации: реализуется надежный механизм аутентификации пользователей с применением паролей.
* Ограничение прав: Пользователям назначаются минимально необходимые права для выполнения операций, что уменьшает риски несанкционированного доступа.

При проектировании приложения были определены группы пользователей и их права представленные в модели разграничения доступа (Рис. 38).



Рисунок 38 - Модель разграничения доступа

### **Заключение**

В дипломном проекте была разработана автоматизированная система продажи билетов в кинотеатр на основе API контроллеров. Система включает в себя веб-приложение для бронирования и покупки билетов, а также API для интеграции с существующей системой кинотеатра.

В процессе разработки были проведены анализ предметной области, проектирование базы данных, разработка алгоритмов обработки данных и логики диалога с пользователем. Система обеспечивает безопасность и целостность данных, а также удобство использования для клиентов и сотрудников кинотеатра.

Применение автоматизированной системы продажи билетов позволяет повысить эффективность работы кинотеатра, сократить время обслуживания клиентов и уменьшить вероятность ошибок персонала. Кроме того, использование API контроллеров обеспечивает гибкость и масштабируемость системы, что делает её подходящей для внедрения в других организациях сферы услуг.

Автоматизированная система продажи билетов в кинотеатр на основе API-контроллеров представляет собой инновационное решение, которое повышает эффективность работы кинотеатра, сокращает время обслуживания клиентов и уменьшает вероятность ошибок персонала. Интеграция с другими системами и сервисами, такими как социальные сети, платёжные системы и системы аналитики, обеспечивает гибкость и масштабируемость системы, делая её подходящей для внедрения в других организациях сферы услуг.

### **ПРИЛОЖЕНИЕ А**

**Код класса хеширования:**

public static class SecurePasswordHasher

{

private const int SaltSize = 16;

private const int HashSize = 20;

private static string Hash(string password, int iterations)

{

byte[] salt;

new RNGCryptoServiceProvider().GetBytes(salt = new byte[SaltSize]);

var pbkdf2 = new Rfc2898DeriveBytes(password, salt, iterations);

var hash = pbkdf2.GetBytes(HashSize);

var hashBytes = new byte[SaltSize + HashSize];

Array.Copy(salt, 0, hashBytes, 0, SaltSize);

Array.Copy(hash, 0, hashBytes, SaltSize, HashSize);

var base64Hash = Convert.ToBase64String(hashBytes);

return string.Format("$MYHASH$V1${0}${1}", iterations, base64Hash);

}

public static string Hash(string password)

{

return Hash(password, 10000);

}

public static bool IsHashSupported(string hashString)

{

return hashString.Contains("$MYHASH$V1$");

}

public static bool Verify(string password, string hashedPassword)

{

if (!IsHashSupported(hashedPassword))

{

throw new NotSupportedException("The hashtype is not supported");

}

var splittedHashString = hashedPassword.Replace("$MYHASH$V1$", "").Split('$');

var iterations = int.Parse(splittedHashString[0]);

var base64Hash = splittedHashString[1];

var hashBytes = Convert.FromBase64String(base64Hash);

var salt = new byte[SaltSize];

Array.Copy(hashBytes, 0, salt, 0, SaltSize);

var pbkdf2 = new Rfc2898DeriveBytes(password, salt, iterations);

byte[] hash = pbkdf2.GetBytes(HashSize);

for (var i = 0; i < HashSize; i++)

{

if (hashBytes[i + SaltSize] != hash[i])

{

return false;

}

}

return true;

}

}

**Код класса работы с заказом:**

public static class WorkToOrder

{

private static List<CreateTicketRequest> order;

public static List<CreateTicketRequest> Order

{

get

{

if(order == null)

{

order = new List<CreateTicketRequest>();

}

return order;

}

set { order = value; }

}

public static void AddToOrder(CreateTicketRequest request)

{

order.Add(request);

}

public static void RemoveAllOrder()

{

order.Clear();

}

public static void RemoveTicket(CreateTicketRequest request)

{

order.Remove(request);

}

}

**Код класса контекста:**

public class TicketSellingContext : DbContext, ITicketSellingContext, IDbRead, IDbWriter, IUnitOfWork

{

public DbSet<Session> Sessions { get; set; }

public DbSet<User> Users { get; set; }

public DbSet<Film> Films { get; set; }

public DbSet<Hall> Halls { get; set; }

public DbSet<Staff> Staffs { get; set; }

public DbSet<Ticket> Tickets { get; set; }

public TicketSellingContext(DbContextOptions<TicketSellingContext> options) : base(options)

{

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.ApplyConfigurationsFromAssembly(typeof(FilmEntityTypeConfiguration).Assembly);

}

/// <summary>

/// Сохранение изменений в БД

/// </summary>

async Task<int> IUnitOfWork.SaveChangesAsync(CancellationToken cancellationToken)

{

var count = await base.SaveChangesAsync(cancellationToken);

foreach (var entry in base.ChangeTracker.Entries().ToArray())

{

entry.State = EntityState.Detached;

}

return count;

}

/// <summary>

/// Чтение сущностей из БД

/// </summary>

IQueryable<TEntity> IDbRead.Read<TEntity>()

=> base.Set<TEntity>()

.AsNoTracking()

.AsQueryable();

/// <summary>

/// Запись сущности в БД

/// </summary>

void IDbWriter.Add<TEntity>(TEntity entity)

=> base.Entry(entity).State = EntityState.Added;

/// <summary>

/// Обновление сущностей

/// </summary>

void IDbWriter.Update<TEntity>(TEntity entity)

=> base.Entry(entity).State = EntityState.Modified;

/// <summary>

/// Удаление сущности из БД

/// </summary>

void IDbWriter.Delete<TEntity>(TEntity entity)

=> base.Entry(entity).State = EntityState.Deleted;

}

### **Источники, использованные при разработке**

1. ГОСТ Р ИСО\_МЭК 25051-2017 Требования к качеству готового к использованию программного продукта (RUSP) и инструкции по тестированию.
2. ГОСТ Р 2.105–2019 Общие требования к текстовым документам.
3. ГОСТ 34.602–2020 Техническое задание на создание автоматизированной системы.
4. ГОСТ Р 59795–2021 Комплекс стандартов на автоматизированные системы.
5. ЕСПД Единая система программной документации.

**Internet – ресурсы**

1. Metanit [Электронный ресурс] - https://metanit.com/sharp/tutorial
2. Stackoverflow [Электронный ресурс] - https://stackoverflow.com
3. YouTube [Электронный ресурс] - https://www.youtube.com