**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Фоминых Валентина Романовна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Фоминых Валентина Романовна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ):*

Участие в выработке требований к программному обеспечению;

Стадии проектирования программного обеспечения. Проектирование UML-диаграмм;

Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения;

Разработка рабочего проекта и технологической документации.

**Индивидуальное задание: ВАРИАНТ 20**

Задание выдал «28» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «28» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Фоминых В.Р. (подпись) (Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Фоминых Валентина Романовна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 28.10.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 29.10.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 30.10.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 31.10.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 01.11.2024 | Проектирование интерфейса пользователя. |  |
| 02.11.2024 | Разработка схемы алгоритма программного продукта. |  |
| 05.11.2024 | Разработка модели данных. Разработка программного продукта. Создание функциональности ПО. |  |
| 06.11.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 07.11.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 08.11.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. |  |
| 09.11.2024 | Подготовка отчета. |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc181969038)

[Задание 1. Техническое задание. 6](#_Toc181969039)

[Задание 2. Разработка алгоритмов и диаграмм 10](#_Toc181969040)

[Задание 3. Разработка макетов программы 18](#_Toc181969041)

[Задание 4. Разработка программы 25](#_Toc181969042)

[Задание 5. Тестирование 32](#_Toc181969043)

[Задание 6. GitHub. Оценка проекта 40](#_Toc181969044)

[ЗАКЛЮЧЕНИЕ 41](#_Toc181969045)

[СПИСОК ЛИТЕРАТУРЫ 42](#_Toc181969046)

[ПРИЛОЖЕНИЯ 43](#_Toc181969047)

[Приложение А 43](#_Toc181969048)

[Приложение Б 55](#_Toc181969049)

# ВВЕДЕНИЕ

Целью данной работы является разработка настольного приложения для парка развлечений с использованием технологии Windows Forms. Приложение включает в себя интерфейс, который предоставляет пользователям доступ к различным услугам, клиенты могут покупать билеты онлайн, сотрудники сравнивать уникальные коды на заказах клиентов, руководители вести учет заказов, билетов и т.д. Основное внимание уделено созданию удобного и функционального пользовательского интерфейса (UI).

Результатом проделанной работы стало создание рабочего интерфейса, который может служить основой для дальнейшего развития функционала приложения, включая интеграцию с базой данных для управления заказами и пользователями.

# Задание 1. Техническое задание.

Описание предметной области

Целью разработки является создание программного решения для автоматизации учета и управления посетителями, аттракционами и событиями в парке развлечений. Модуль обеспечивает удобное управление нагрузкой аттракционов, контролем безопасности, продажей билетов, а также упрощает взаимодействие с клиентами и анализ эффективности работы парка.

Разработка этого модуля направлена на улучшение работы парка путем оптимизации времени ожидания на вход в парк, обеспечения безопасности и рационального использования всех ресурсов.

Основные функции и возможности модуля включают:

1. Управление событиями и аттракционами (Доступ: только руководитель):

- Функционал: добавление и обновление расписания событий (например, праздников и специальных программ), настройка времени работы аттракционов, установка ограничений на количество посетителей.

- Права доступа: Права доступа: руководитель может добавлять новые события, изменять информацию о дате, времени, описании событий и ограничениях по возрасту и количеству посетителей. Также может изменять статус аттракционов, указывая, например, «Открыт» или «Закрыт на реставрацию».

2. Управление билетами (Доступ: только для руководителя):

- Функционал: управление ценами на билеты, настройка скидок, акций и льготных программ.

- Права доступа: Руководитель может устанавливать цены категории билетов и периоды действия скидок. Также может настраивать параметры доступных аттракционов и событий по билетам.

3. Проверка билетов (Доступ: сотрудник):

- Функционал: проверка билетов через систему, проверка данных клиента его уникального номера заказа и аттракционов, входящих в тариф.

- Права доступа: Сотрудник оформляет билеты посетителям, проверяет их актуальные данные и применяет заказ.

4. Покупка билетов онлайн (Доступ: клиент):

- Функционал: просмотр доступных событий и аттракционов, выбор времени, покупка билета онлайн.

- Права доступа: Клиент может выбрать количество билетов, указать дату посещения, оплатить покупку и получить уникальный код для предоставления сотруднику парка.

5. Уведомления и мониторинг безопасности (Доступ: руководитель и сотрудник):

- Функционал: уведомления о возможных неисправностях аттракционов, предупреждения об обновлении оборудования и соблюдении стандартов безопасности.

- Права доступа: Руководитель получает уведомления о критических неисправностях и изменениях в статусе аттракционов, тогда как сотрудник отслеживает статус и передает информацию посетителям.

Техническое задание

1. Общие сведения

1.1. Наименование проекта: Информационная система для управления парком развлечений.

1.2. Заказчик: Федоров Глеб Олегович.

1.3. Исполнитель: Фоминых Валентина Романовна.

2. Функциональные требования

2.1. Управление событиями и аттракционами:

- Название события.

- Дата и время проведения.

- Описание программы.

- Ограничения по количеству посетителей.

- Возрастные и другие условия допуска.

2.2. Управление билетами:

- Цена билета.

- Периоды действия билетов.

- Входящие в билет аттракционы.

- Входящие в билет события.

2.3. Проверка билетов:

- Уникальный номер билета.

- Статус билета (действителен или нет).

- Доступные аттракционы по тарифу.

- Срок действия билета.

2.4. Покупка билетов онлайн:

- Выбор даты посещения.

- Выбор количества билетов.

- Выбор тарифа.

- Оформление заказа (Цена).

- Получение уникального кода.

2.5. Уведомления и мониторинг безопасности:

- Состояние аттракционов.

- Необходимость проведения технического обслуживания.

- Уведомления о нарушениях стандартов безопасности.

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Логин и пароль для доступа к приложению;

- Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:

- Простой и интуитивный интерфейс;

- Информативные уведомления и подсказки.

3.4. Производительность:

- Приложение должно иметь быстрый доступ к данным;

- Минимальное время отклика на запросы пользователя.

4. Требования к реализации

4.1. Язык программирования: C#.

4.2. СУБД: MS Sql.

5. Требования к документации

5.1. Техническое задание на разработку программного модуля.

6. Руководство по стилю

6.1. Шрифт: Обычный текст: Ebrima. Заголовки: Gost Type BU.

6.2. Цветовая схема: Цвет фона: #b5838d. Цвет текста: #001219. Цвет других элементов можно выбирать из: #e5989b, #ffcdb2, #ffb4a2, #6d6875.

# Задание 2. Разработка алгоритмов и диаграмм

Создание UML-диаграмм

Диаграмма вариантов использования (Рисунок 1).



Рисунок 1 – Диаграмма вариантов использования

Диаграмма последовательности (Рисунок 2).



Рисунок 2 – Диаграмма последовательности

Диаграмма активности (Рисунок 3).



Рисунок 3 – Диаграмма активности

ER-диаграмма (Рисунок 4).



Рисунок 4 - ER-диаграмма

Создание справочных таблиц

Таблица 1 - Таблица Событий

|  |  |  |
| --- | --- | --- |
| Таблица Event | | |
| EventID | INT (PK) | Уникальный идентификатор события. |
| Date | DATE | Дата проведения события. |
| Name | NVARCHAR(255) | Название события. |
| StartDate | DATETIME | Время начала события. |
| EndDate | DATETIME | Время окончания собятия. |
| Description | NVARCHAR(255) | Описание программы события. |
| VisitorLimit | INT | Ограничение на количество посетителей. |
| AgeRestriction | INT | Возрастное ограничение. |

Таблица 2 - Таблица Билетов.

|  |  |  |
| --- | --- | --- |
| Таблица Ticket | | |
| TicketID | INT(PK) | Уникальный идентификатор билета. |
| Status | NVARCHAR(50) | Статус билета (действителен, недействителен). |
| ExpirationDate | DATE | Срок действия билета. |
| AvailableAttractions | NVARCHAR(255) | Список аттракционов, доступных по тарифу. |
| EventID | INT(FK) | Вторичный ключ для связи с событями. |
| AttractionID | INT(FK) | Вторичный ключ для аттракционов. |
| OrderID | INT(FK) | Вторичный ключ для заказа. |

Таблица 3 - Таблица Аттракционов.

|  |  |  |
| --- | --- | --- |
| Таблица Attraction | | |
| AttractionID | INT (PK) | Уникальный идентификатор аттракциона. |
| Name | NVARCHAR(255) | Название аттракциона. |
| SafetyStatus | NVARCHAR(50) | Состояние безопасности аттракциона. |

Таблица 4 - Таблица руководителей.

|  |  |  |
| --- | --- | --- |
| Таблица Supervisor | | |
| SupervisorID | INT(PK) | Уникальный идентификатор руководителя. |
| Name | NVARCHAR(255) | ФИО руководителя. |
| Email | NVARCHAR(255) | Почта руководителя. |
| Phone | NVARCHAR(20) | Телефон руководителя. |
| Department | NVARCHAR(255) | Отдел руководителя. |

Таблица 5 - Таблица Сотрудников.

|  |  |  |
| --- | --- | --- |
| Таблица Employee | | |
| EmployeeID | INT(PK) | Уникальный идентификатор сотрудника. |
| Name | NVARCHAR(255) | ФИО сотрудника. |
| Position | NVARCHAR(50) | Должность сотрудника. |
| HireDate | DATE | Дата приема на работу. |
| SupervisorID | NVARCHAR(255) | Ссылка на руководителя (если есть) . |
| Phone | NVARCHAR(20) | номер телефона сотрудника. |

Таблица 6 - Таблица Заказов.

|  |  |  |
| --- | --- | --- |
| Таблица Order | | |
| OrderID | INT(PK) | Уникальный идентификатор заказа. |
| VisitDate | DATE | Выбранная дата посещения. |
| TicketQuantity | INT | Количество заказанных билетов. |
| TotalPrice | DECIMAL(10,2) | Итоговая сумма заказа. |
| UniqueCode | NVARCHAR(255) | Уникальный код заказа. |
| ClientID | INT(FK) | Ссылка на клиента, оформившего заказ. |
| EmployeeID | INT(FK) | Ссылка на сотрудника, обработавшего заказ. |

Таблица 7 - Таблица Клиентов.

|  |  |  |
| --- | --- | --- |
| Таблица Client | | |
| ClientID | INT(PK) | Уникальный идентификатор клиента. |
| Name | NVARCHAR(50) | ФИО клиента. |
| Email | NVARCHAR(50) | Электронная почта клиента. |
| DateOfBirth | DATE | Дата рождения клиента. |
| Address | NVARCHAR(255) | Адрес клиента. |
| Phone | NVARCHAR(20) | Номер телефона клиента. |

Создаю базу данных, таблицы и вношу (Рисунки 5-11):

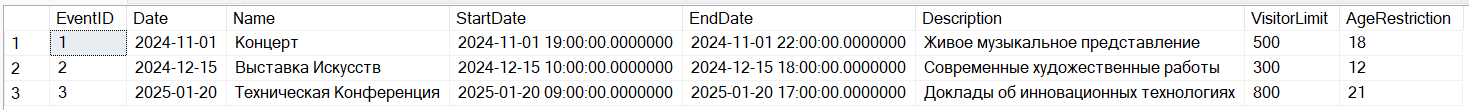


Рисунок 5 - Заполнение таблицы Событий

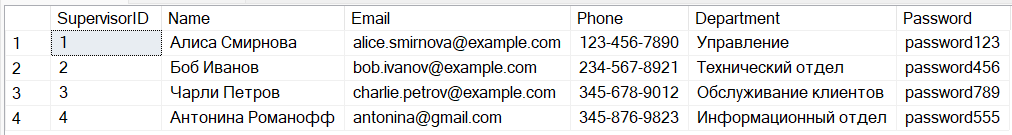


Рисунок 6 - Заполнение таблицы Руководители

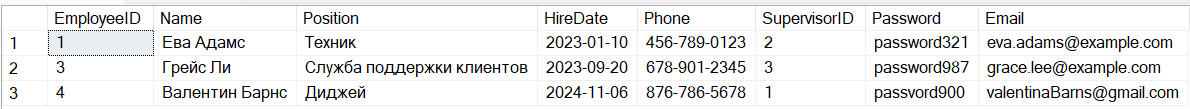


Рисунок 7 - Заполнение таблицы Сотрудники

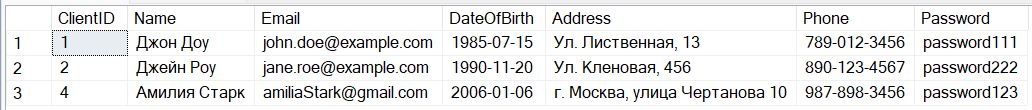


Рисунок 8 - Заполнение таблицы Клиенты

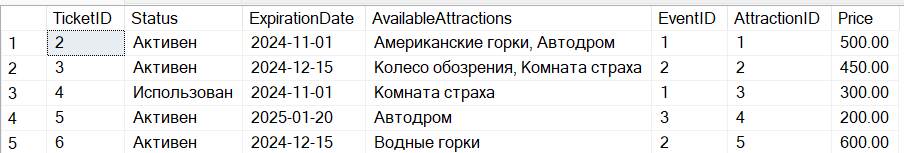


Рисунок 9 - Заполнение таблицы Билеты



Рисунок 10 - Заполнение таблицы Аттракционы

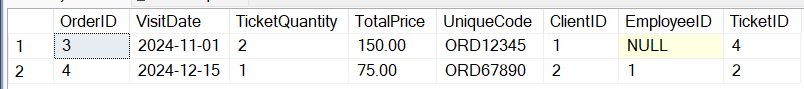


Рисунок 11 - Заполнение таблицы Заказы

# Задание 3. Разработка макетов программы

Разработка макетов приложения вариант Wireframe (черно-белые) (Рисунки 12-17):

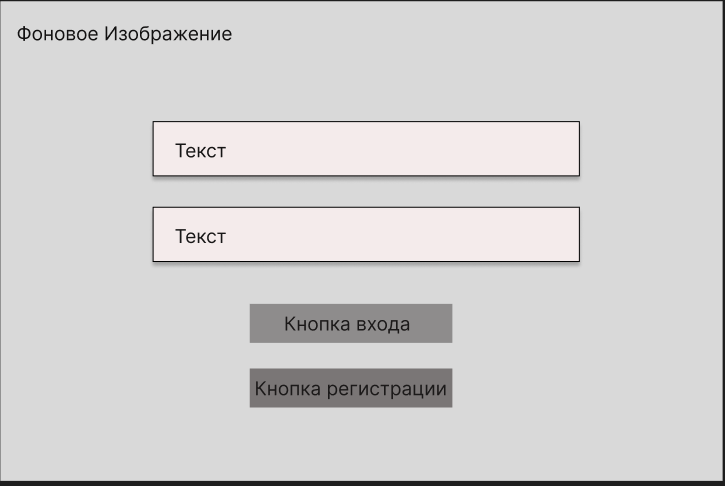


Рисунок 12 - Панель входа

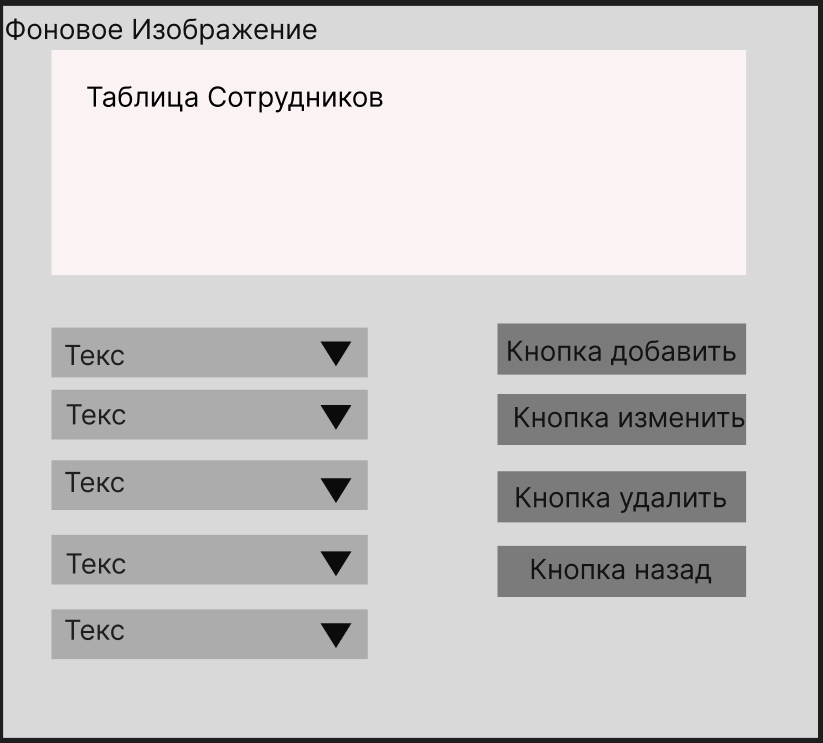


Рисунок 13 - Панель сотрудника

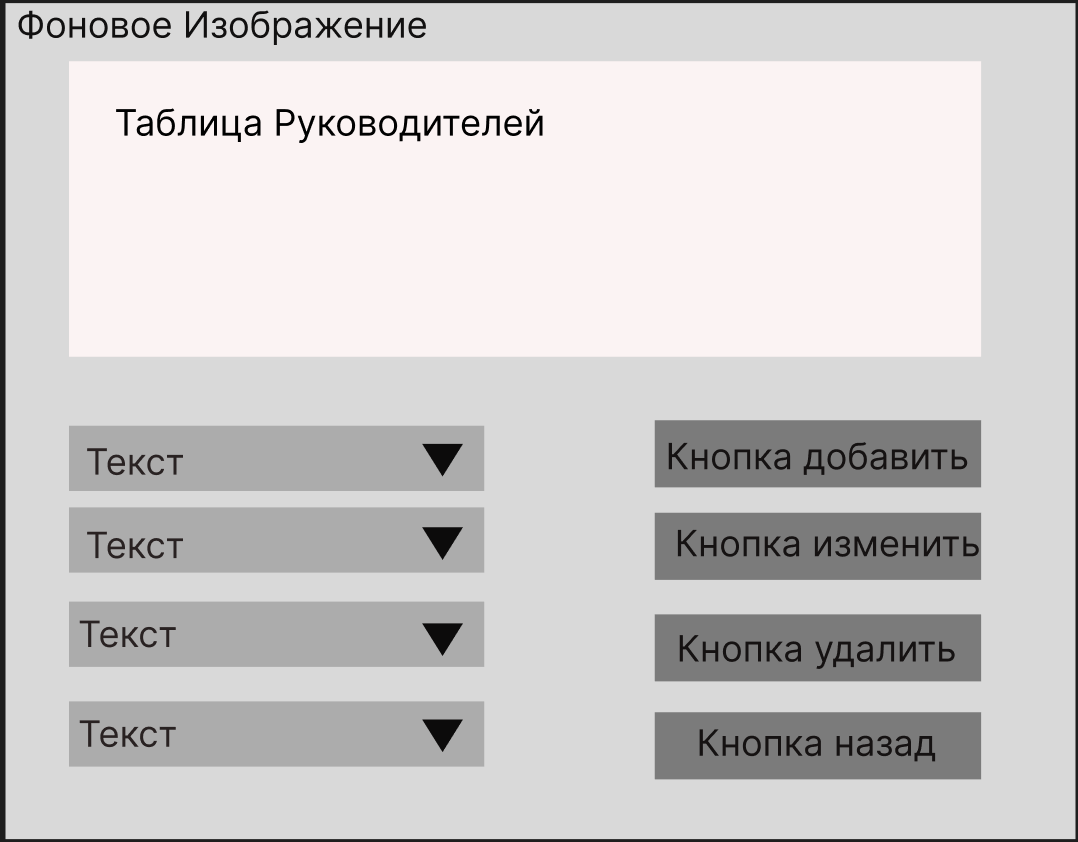


Рисунок 14 - Панель руководителя

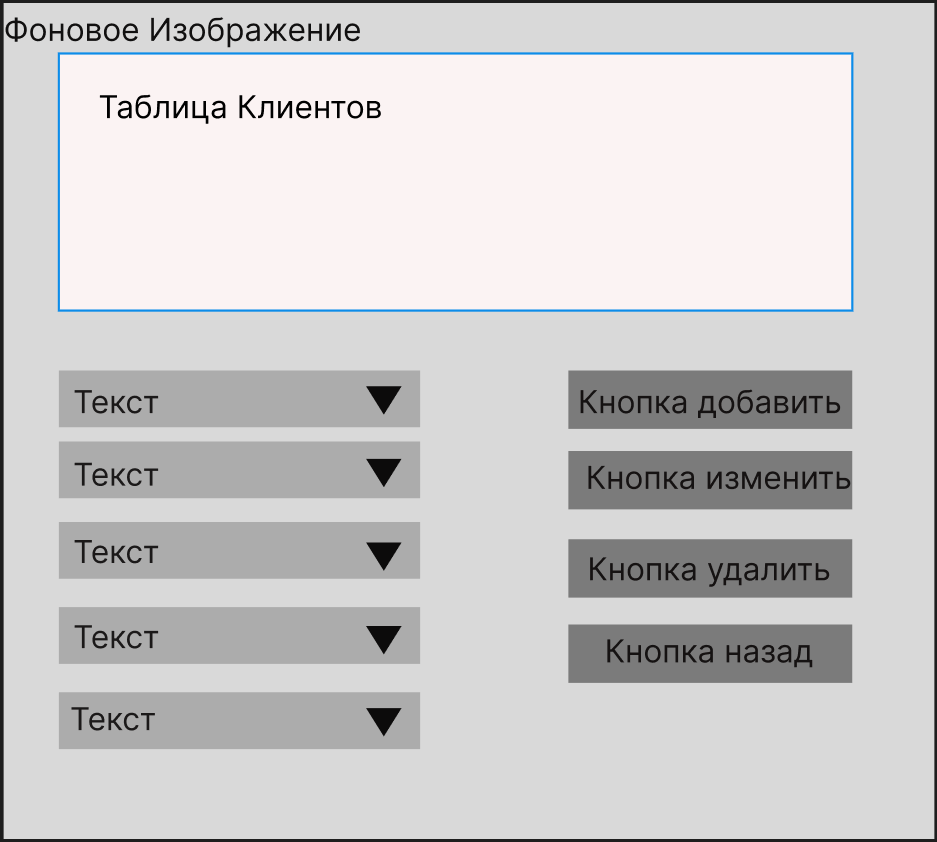


Рисунок 15 - Панель клиента

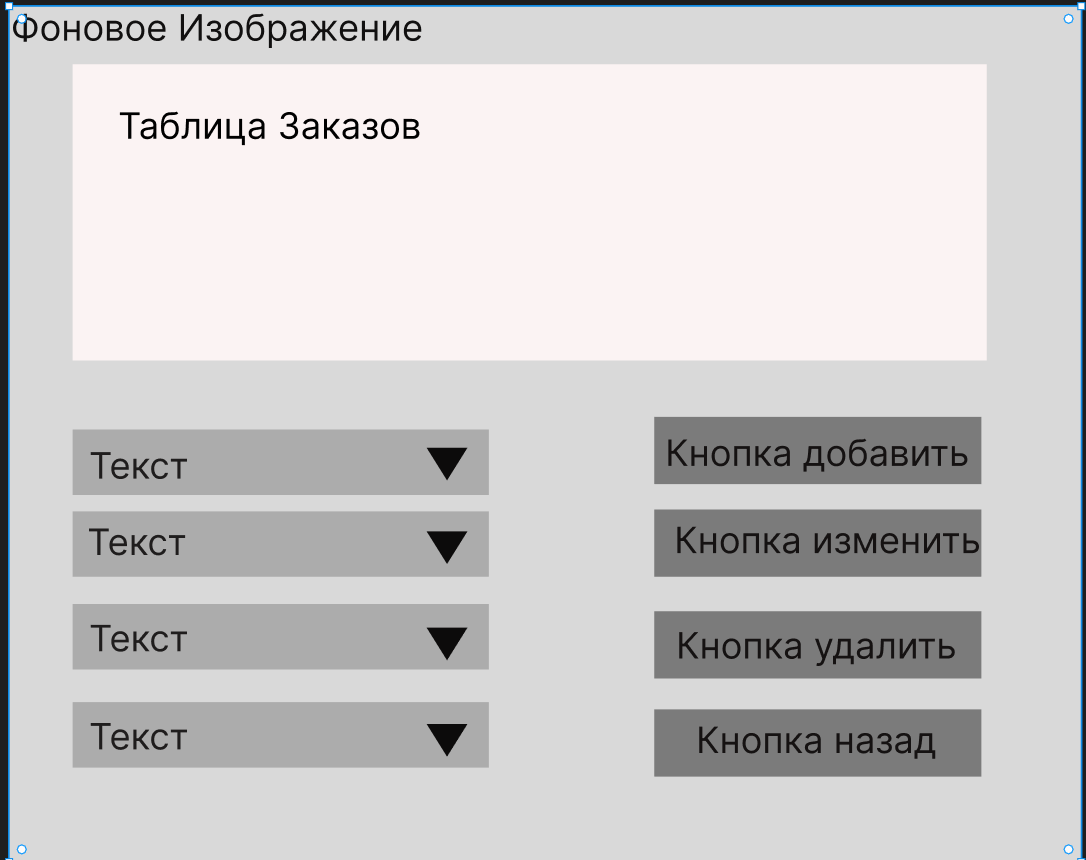


Рисунок 16 - Панель заказов

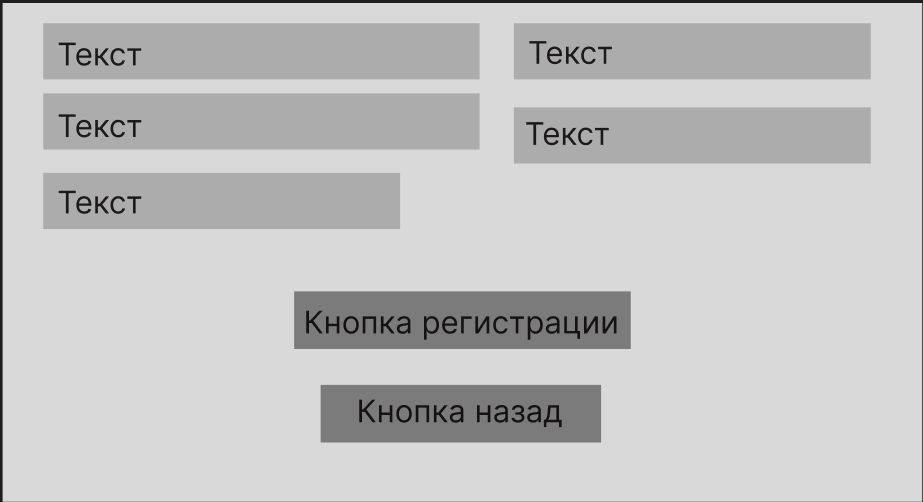


Рисунок 17 - Панель регистрации

Разработка макетов приложения вариантMockup (цветные) (Рисунки 18-23):



Рисунок 18 - Панель входа

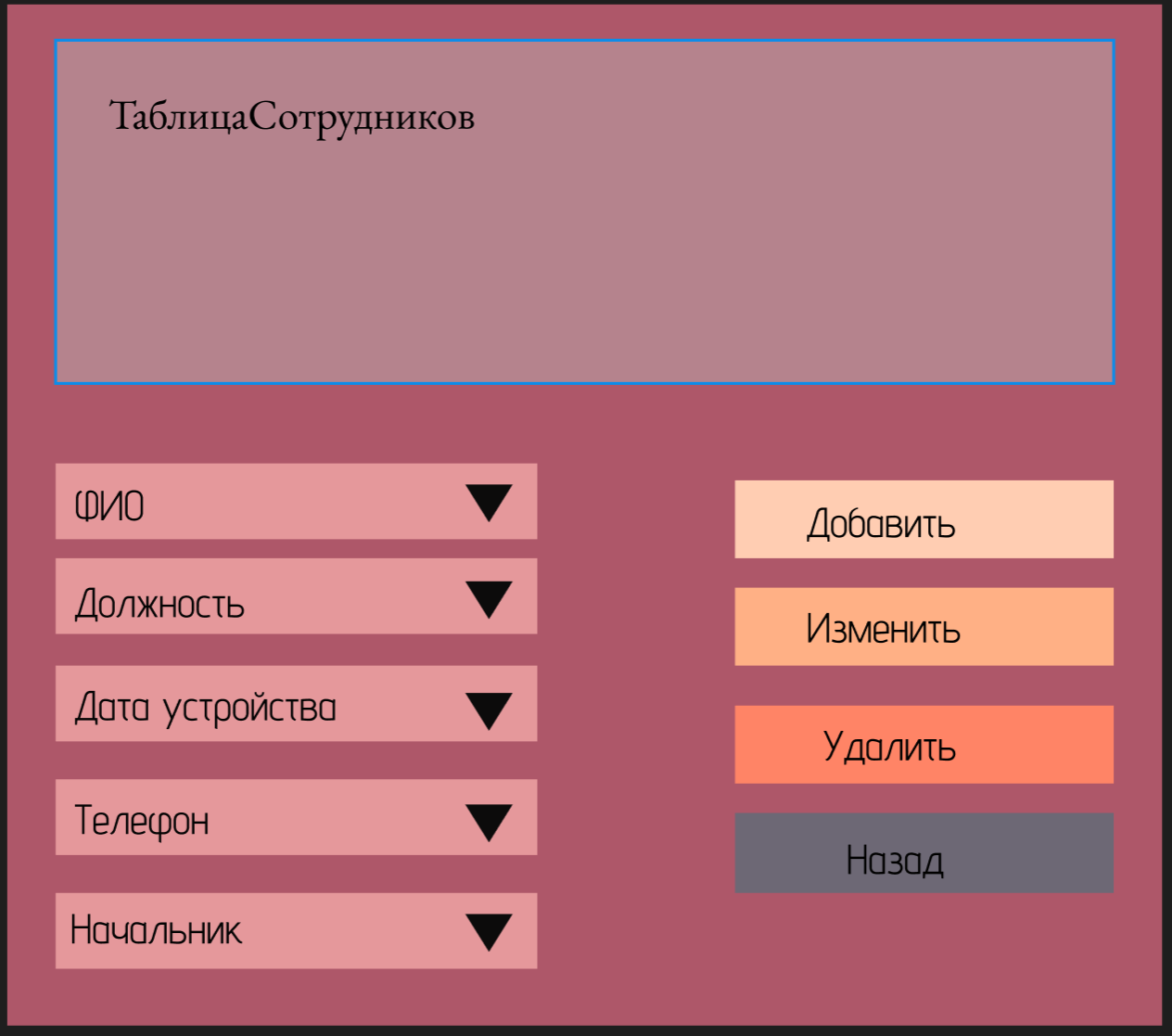


Рисунок 19 - Панель сотрудников

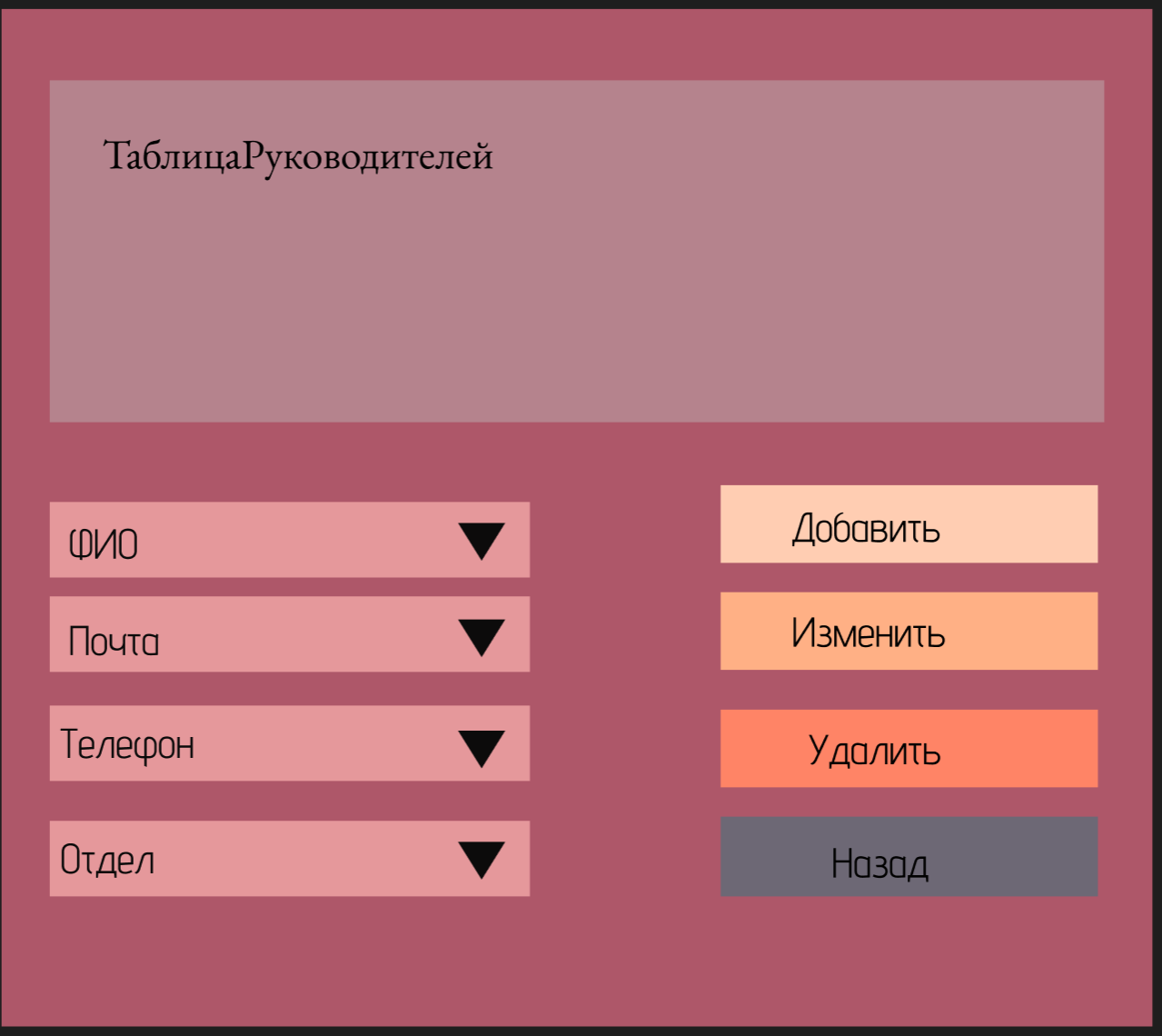


Рисунок 20 - Панель руководителей

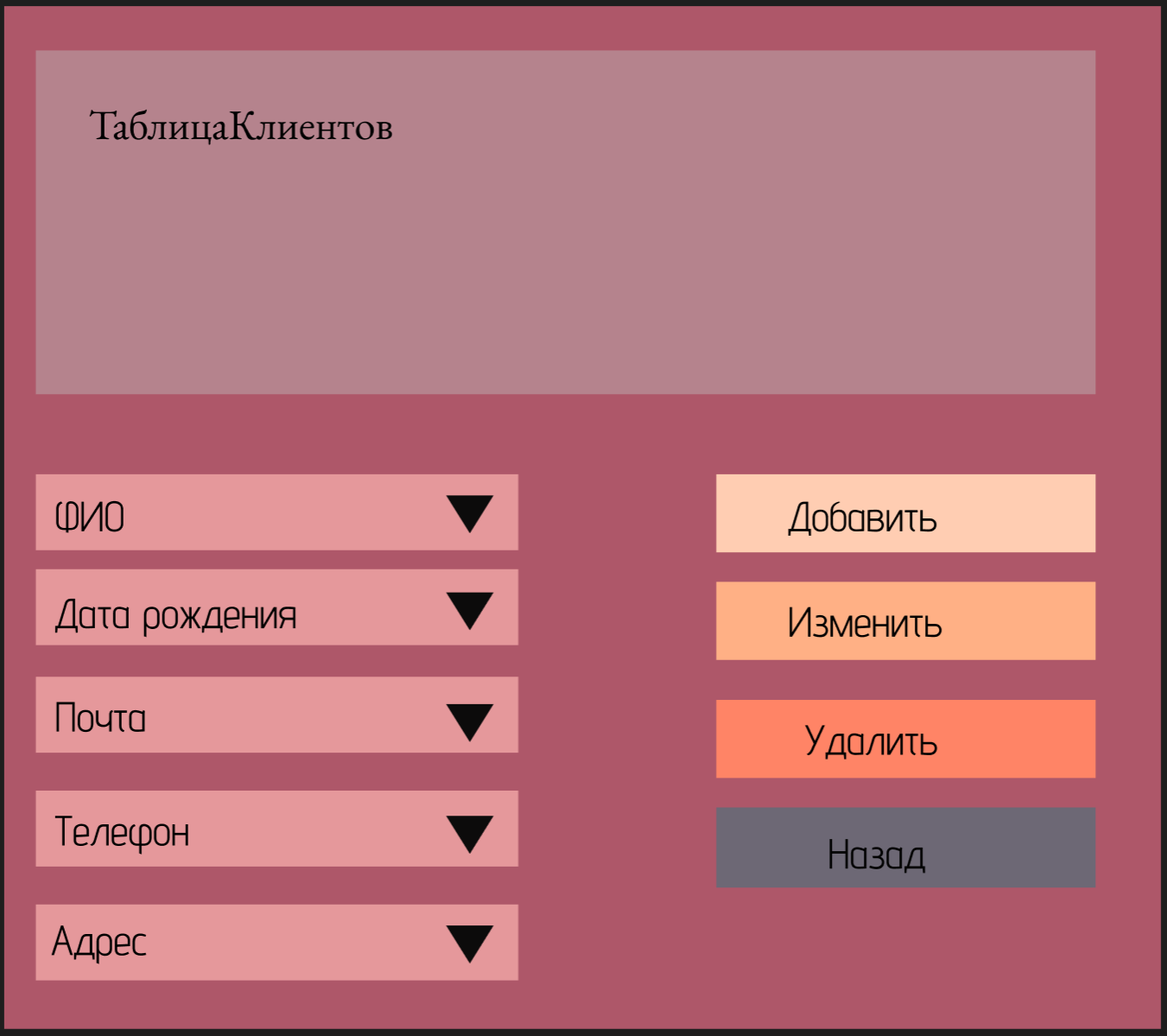


Рисунок 21 - Панель клиентов

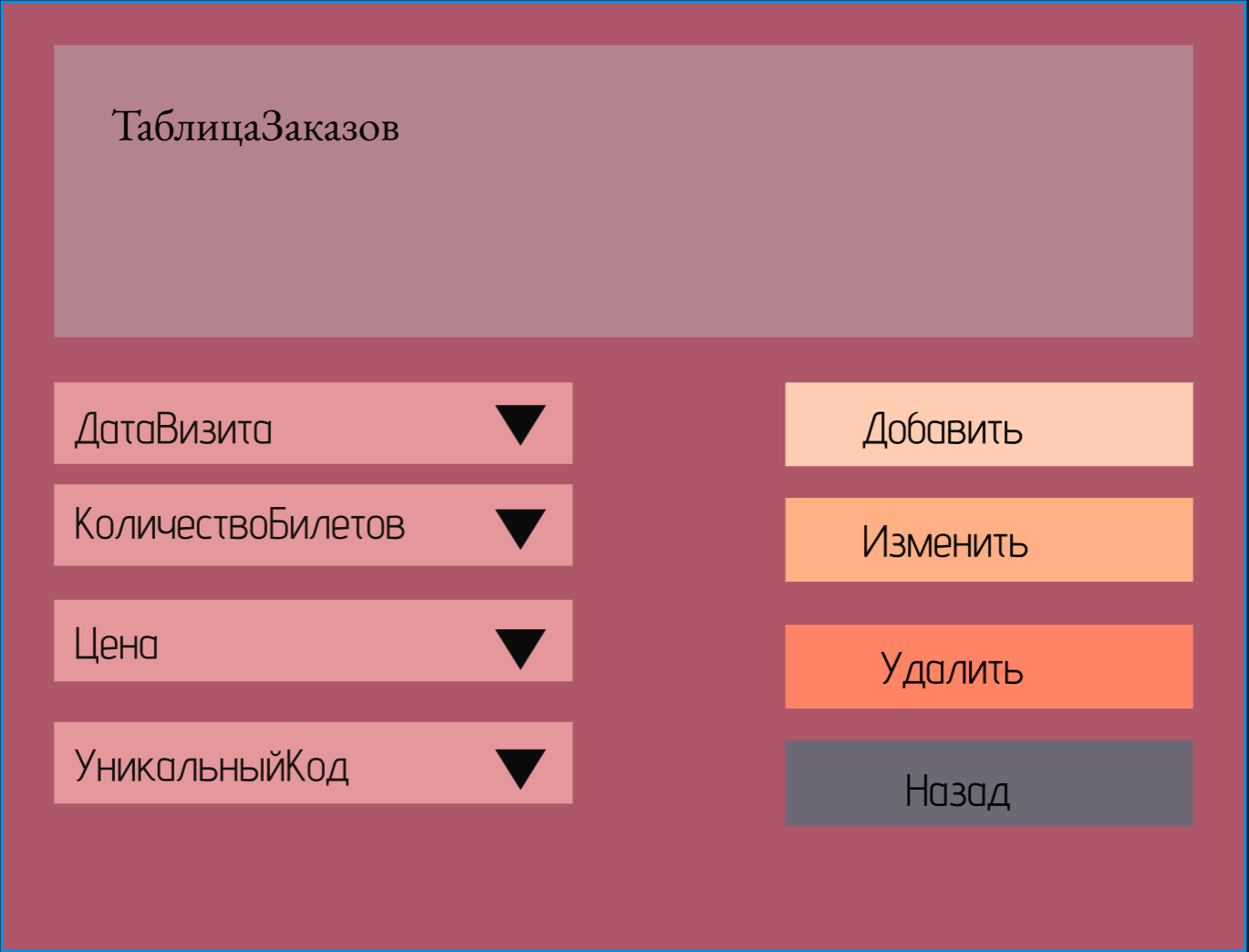


Рисунок 22 - Панель заказов

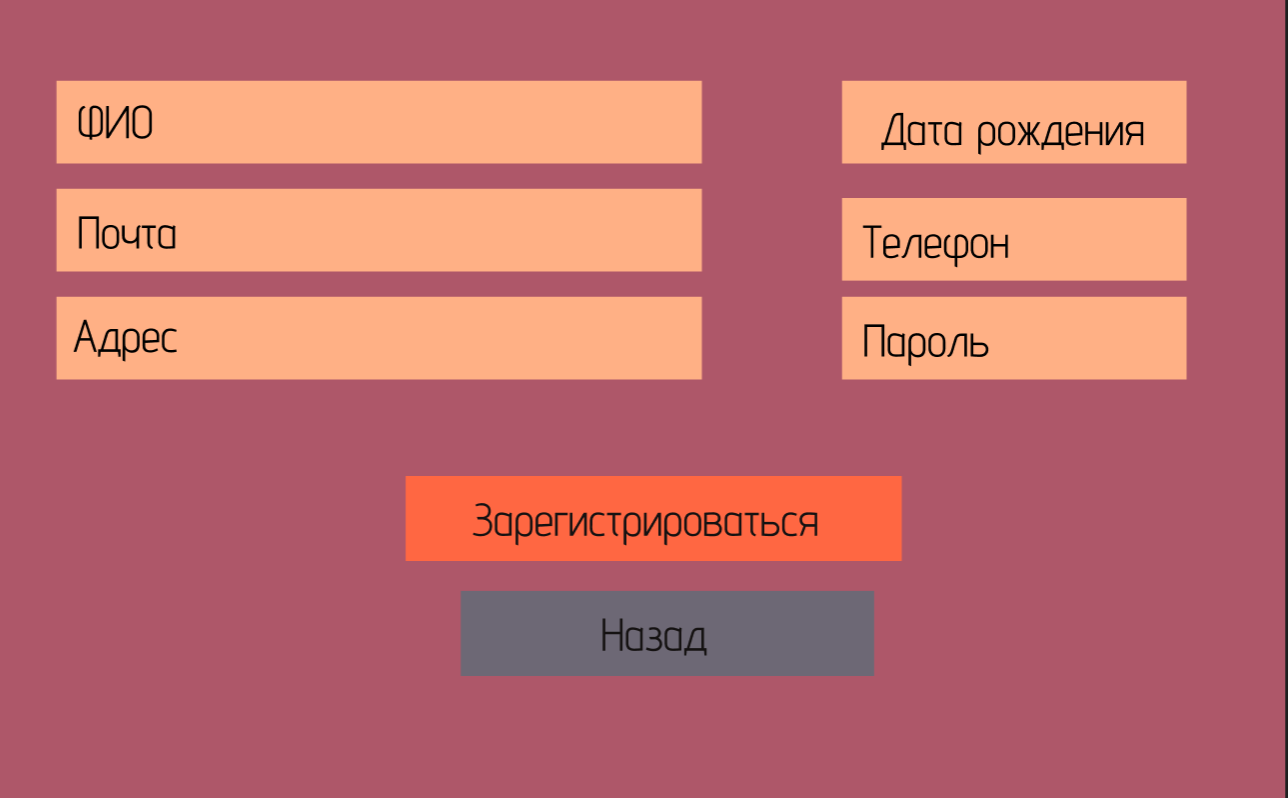


Рисунок 23 - Панель регистрации

Создание карты навигации (Рисунок 24):



Рисунок 24 - Схема навигации

# Задание 4. Разработка программы

Разработала программный продукт, опираясь на составленные алгоритмы, диаграммы, макеты и техническое задание.

При запуске приложения открывается форма входа для пользователей (Рисунок 25).

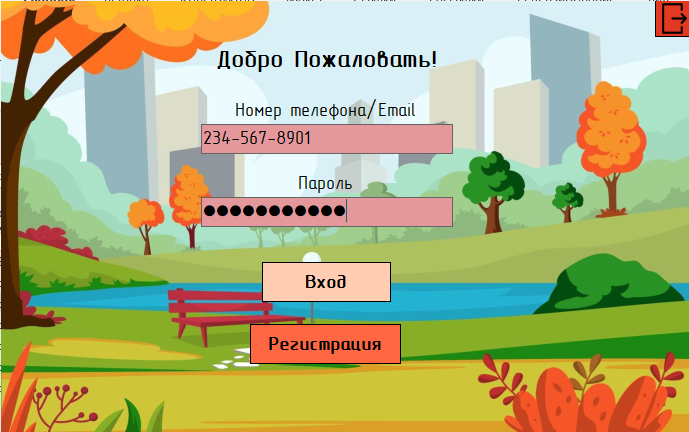


Рисунок 25 - Форма входа

Если пользователь введет не те данные ему выведется сообщение об ошибке (Рисунок 26).

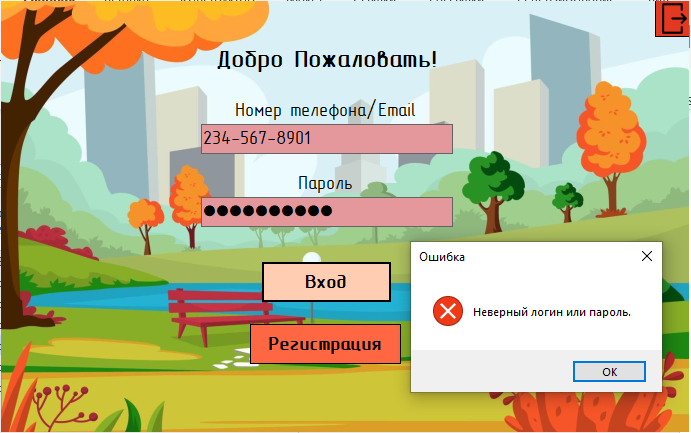


Рисунок 26 - Сообщение в случае ошибочных данных

Так же если появляется новый пользователь он может зарегистрироваться, стоит заметить, что все новые пользователи по умолчанию клиенты (Рисунок 27).

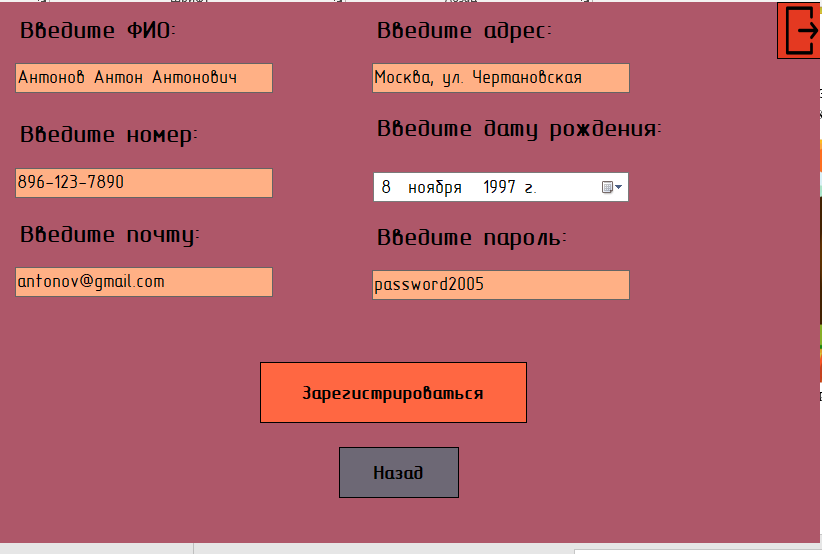


Рисунок 27 - Регистрация нового пользователя

В случае если пользователь введет не свою почту или номер (который уже существует в базе данных) выведется исключение и если пользователь не заполнит одно из полей тоже будет исключение (Рисунки 28-29).

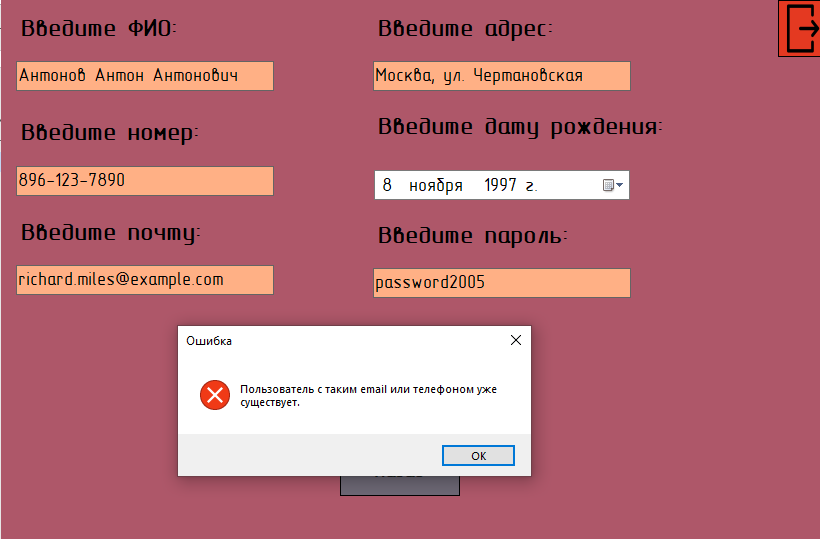


Рисунок 28 - Сообщение в случае ввода существующей почты или номера

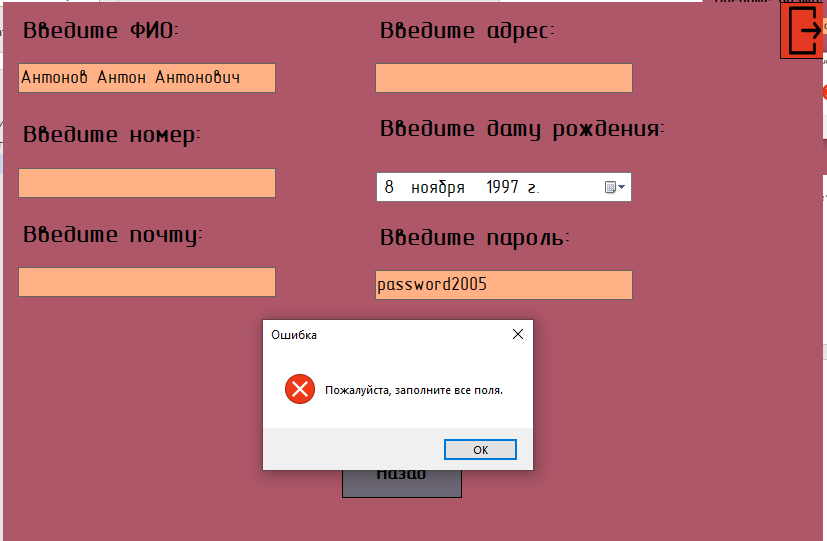


Рисунок 29 - Сообщение в случае незаполненных полей.

Так же когда мы входим под клиентом у нас отображаются только его данные и только его заказы (Рисунки 30-31).

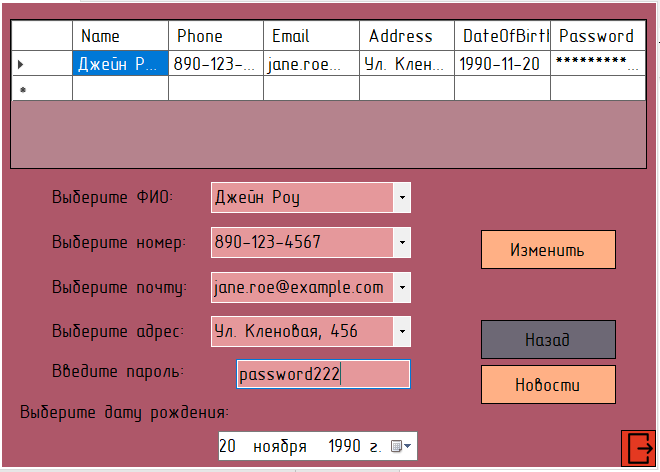


Рисунок 30 - Форма клиента

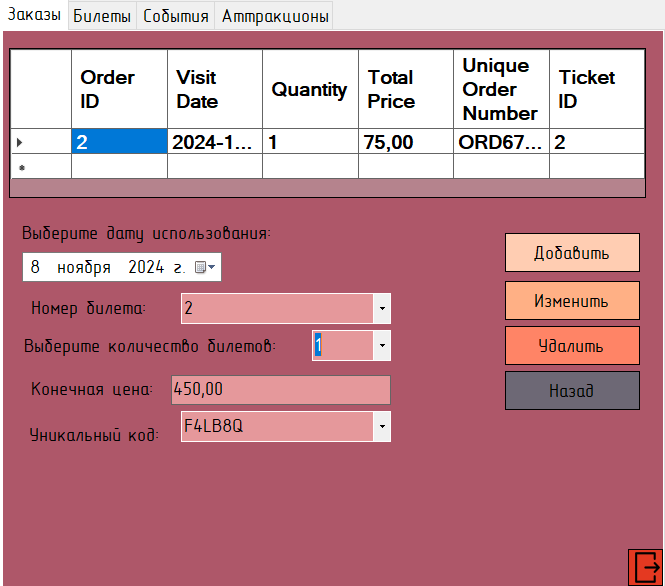


Рисунок 31 - Форма заказов и возможность добавления новых

В случае если клиент захочет изменить заказ ему надо выбрать в выпадающем списке его уникальный код иначе выведет ошибка (Рисунок 32).

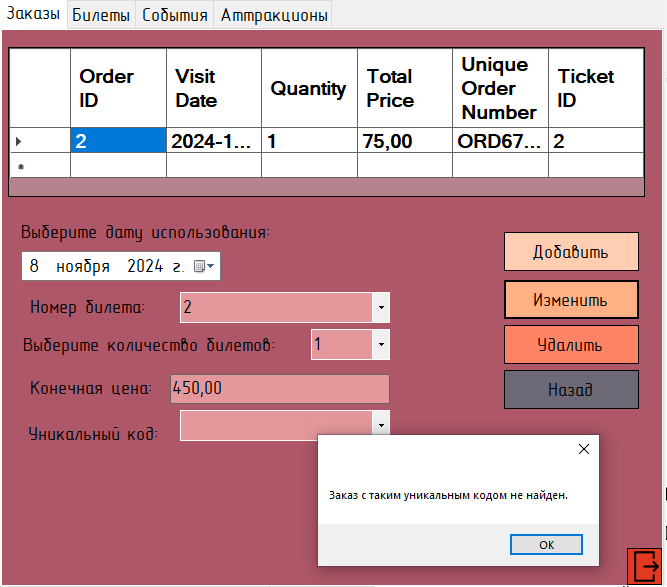


Рисунок 32 - Сообщения в случае попытки изменения без указания кода

Если мы заходим под сотрудником, то у нас отображаются все клиенты и заказы только те, за которые он отвечает, на форме сотрудника отображаются только его данные (Рисунки 33-35).

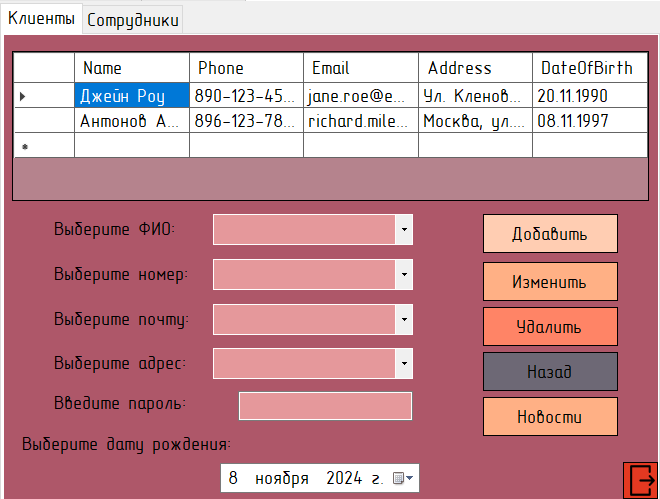


Рисунок 33 - Форма клиентов прикрепленных к сотруднику

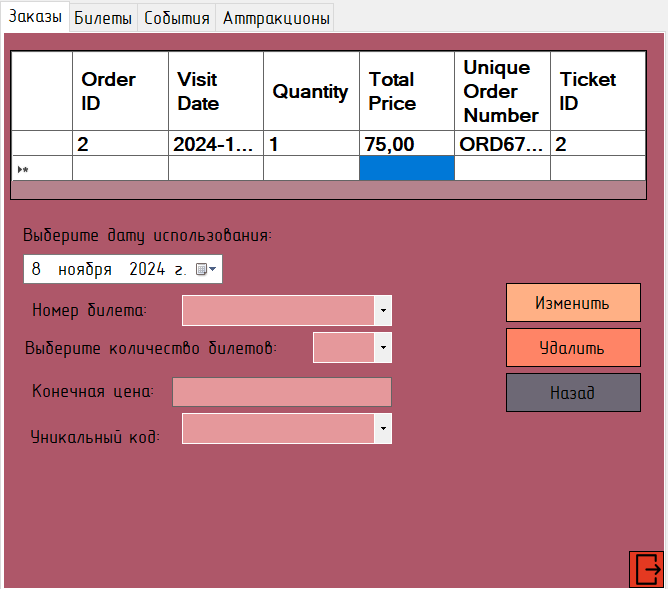


Рисунок 34 - Отображение заказов прикрепленных к сотруднику

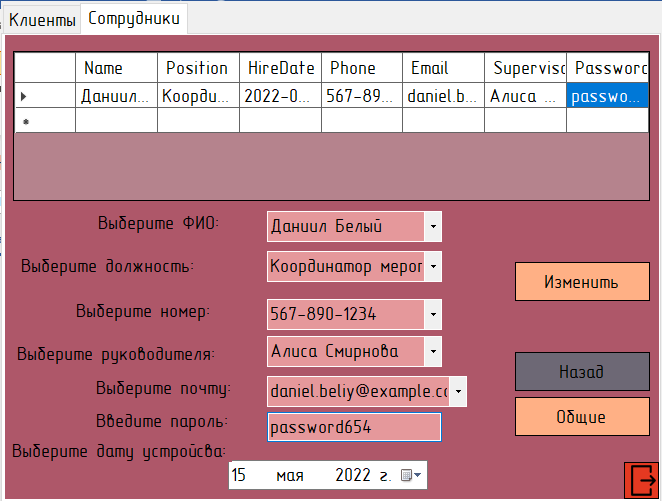


Рисунок 35 - Форма сотрудника

Если входит руководитель, то у него отображаются все клиенты, прикреплённые к его отделу сотруднику, все заказы, в форме руководителя только данные конкретного пользователя (Рисунки 36-39).

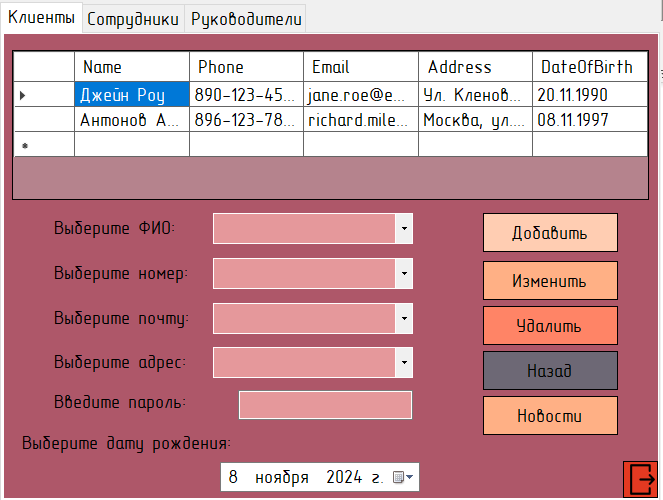


Рисунок 36 - Отображение клиентов у руководителей

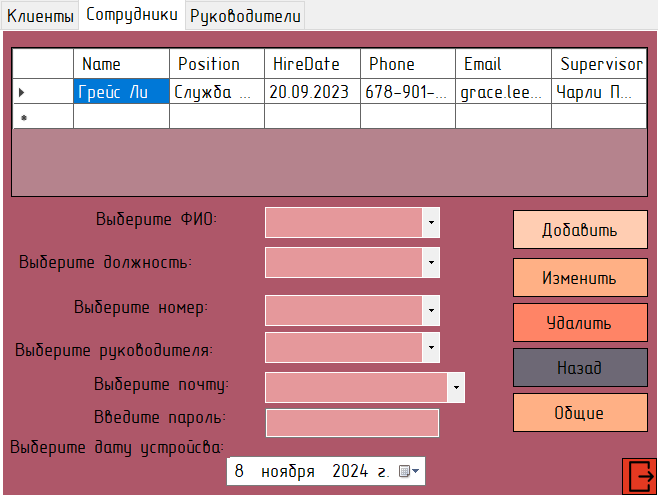


Рисунок 37 - Форма сотрудников работающих у руководителя

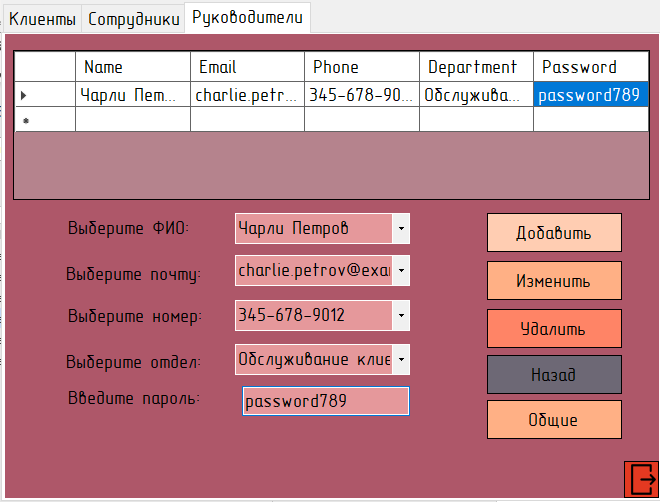


Рисунок 38 - Форма руководителя

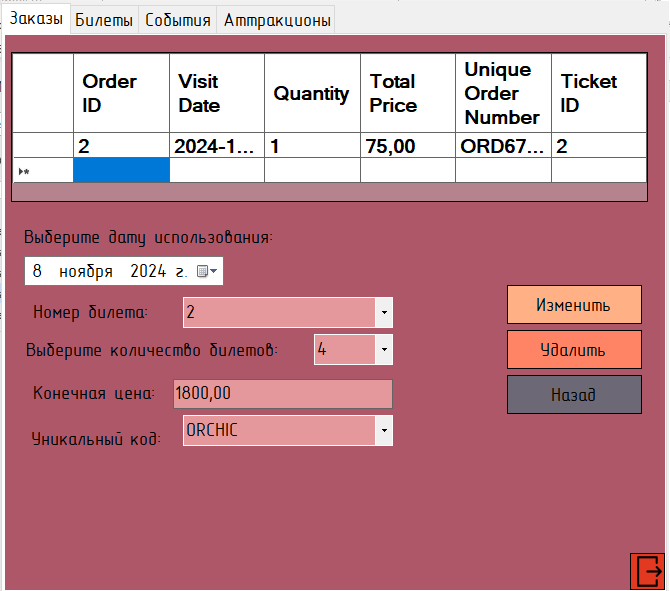


Рисунок 39 - Отображение формы заказов у руководителя

# Задание 5. Тестирование

Пишу 5 Test-Case, проверяющие один наиболее важный процесс в системе. Тестовые данные предусматривают различные ситуации в системе (Таблицы 8-12).

Таблица 8 - Удалить событие

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | Парк развлечений |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Валентина |
| **Дата(ы) теста** | 06.11.2024 |
| **Тестовый пример #** | TC\_UI\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Удаление события по названию и дате |
| **Краткое изложение теста** | Проверить возможность удаления события по его названию и дате |
| **Этапы теста** | 1. Зайти как руководитель.  2. Открыть форму событий. 3. Выбрать событие по названию и дате. 4. Нажать кнопку "Удалить". |
| **Тестовые данные** | Название события: Летний концерт Дата события: 20.10.2024 |
| **Ожидаемый результат** | Событие успешно удалено из базы данных |
| **Фактический результат** | Событие успешно удалено из базы данных |
| **Предварительное условие** | Событие существует в базе данных |
| **Постусловие** | Событие удалено из базы данных |
| **Статус**  *(Зачет/Незачет)* | Зачет |
| **Примечания/комментарии** | – |

Таблица 9 - Регистрация нового клиента

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | Парк развлечений |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Валентина |
| **Дата(ы) теста** | 06.11.2024 |
| **Тестовый пример #** | TC\_UI\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешная регистрация нового клиента |
| **Краткое изложение теста** | Проверить возможность регистрации нового клиента с корректными данными |
| **Этапы теста** | 1.Открыть форму регистрации.  2.Заполнить поля: полное имя, email, пароль, телефон, адрес, дата рождения.  3.Нажать кнопку "Зарегистрироваться". |
| **Тестовые данные** | Полное имя: Антипова Маша Сергеевна, email: marisia@example.com, пароль: зфыыцщкв123, телефон: 123-456-7890, адрес: Москва, дата рождения: 01.01.2000 |
| **Ожидаемый результат** | Клиент успешно регистрируется, отображается сообщение о регистрации, пользователь перенаправляется на форму входа. |
| **Фактический результат** | Клиент успешно регистрируется, отображается сообщение о регистрации, пользователь перенаправляется на форму входа. |
| **Предварительное условие** | Клиент не существует в базе данных. |
| **Постусловие** | Новый клиент добавлен в базу данных. |
| **Статус**  *(Зачет/Незачет)* | Зачет |
| **Примечания/комментарии** | – |

Таблица 10 - Добавление заказа

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | Парк развлечений |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Валентина |
| **Дата(ы) теста** | 06.11.2024 |
| **Тестовый пример #** | TC\_UI\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешное добавление нового заказа |
| **Краткое изложение теста** | Проверить возможность добавления нового заказа с корректными данными |
| **Этапы теста** | 1.Зайти как клиент  2. Открыть форму заказа . 3. Выбрать билет, количество, дату. 4. Нажать кнопку "Добавить". |
| **Тестовые данные** | Билет: 6  Количество: 2  Дата: 15.11.2024 |
| **Ожидаемый результат** | Новый заказ добавлен в базу данных, отображается сообщение об успешном создании заказа. |
| **Фактический результат** | Новый заказ добавлен в базу данных, отображается сообщение об успешном создании заказа. |
| **Предварительное условие** | Клиент существует в базе данных. |
| **Постусловие** | Новый заказ добавлен в таблицу заказов. |
| **Статус**  *(Зачет/Незачет)* | Зачет |
| **Примечания/комментарии** | – |

Таблица 11 - Изменение данных аттракциона

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | Парк развлечений |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Валентина |
| **Дата(ы) теста** | 06.11.2024 |
| **Тестовый пример #** | TC\_UI\_4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Успешное изменение данных аттракциона |
| **Краткое изложение теста** | Проверить возможность изменения данных аттракциона по его названию |
| **Этапы теста** | 1.Зайти как руководитель  2. Открыть форму аттракциона. 3. Выбрать аттракцион по названию. 4. Изменить его описание и статус безопасности. 5. Нажать кнопку "Изменить". |
| **Тестовые данные** | Название аттракциона: Колесо обозрения Описание: Новый аттракцион Статус безопасности: Open |
| **Ожидаемый результат** | Изменения успешно сохранены в базе данных. |
| **Фактический результат** | Изменения успешно сохранены в базе данных. |
| **Предварительное условие** | Аттракцион существует в базе данных. |
| **Постусловие** | Аттракцион обновлен в базе данных. |
| **Статус**  *(Зачет/Незачет)* | Зачет |
| **Примечания/комментарии** | – |

Таблица 12 - Добавление сотрудника

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | Парк развлечений |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Валентина |
| **Дата(ы) теста** | 06.11.2024 |
| **Тестовый пример #** | TC\_UI\_5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешное добавление нового сотрудника |
| **Краткое изложение теста** | Проверить возможность добавления нового сотрудника с корректными данными |
| **Этапы теста** | 1.Зайти как руководитель  2. Открыть форму добавления сотрудника. 3. Ввести данные сотрудника (Имя, Должность, Телефон, Email, Дата принятия, Руководитель). 4. Нажать кнопку "Добавить". |
| **Тестовые данные** | Имя: Иванов Иван Иванович Должность: Менеджер Телефон: 790-0123-4567 Email: ivan@example.com Дата принятия: 06.11.2024 Руководитель: Петров Петр |
| **Ожидаемый результат** | Новый сотрудник добавлен в базу данных. |
| **Фактический результат** | Новый сотрудник добавлен в базу данных. |
| **Предварительное условие** | Сотрудник с такими Email и телефоном не существует в базе данных. |
| **Постусловие** | Новый сотрудник добавлен в таблицу сотрудников. |
| **Статус**  *(Зачет/Незачет)* | Зачет |
| **Примечания/комментарии** | – |

Разработка модульных тестов для проверки корректной работы программы и исключений.

Пишу несколько тестов для проверки авторизации пользователей (Рисунки 1-5).

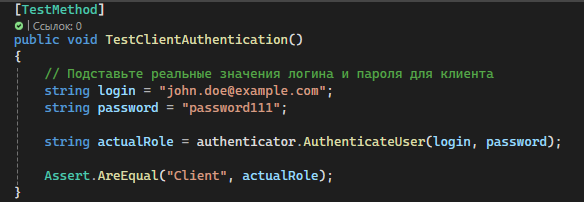


Рисунок 40 - Проверяем истинные значения клиента.

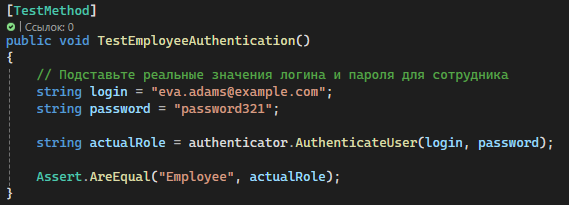


Рисунок 41 - Подставляем существующего сотрудника

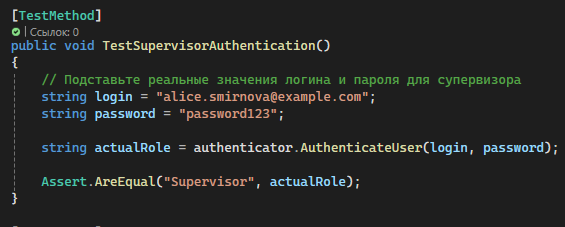


Рисунок 42 - Подставляем существующего руководителя

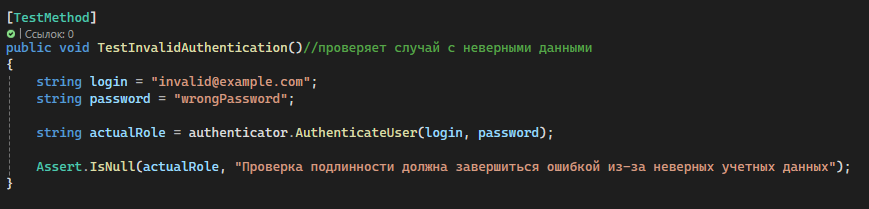


Рисунок 43 - Негативный тест для проверки неверных данных

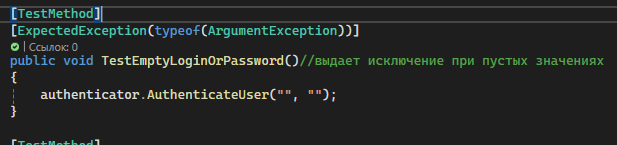


Рисунок 44 - Проверяет на исключение в случае пустых значений

Пишу тесты для проверки корректной работы формы клиента, с возможностью добавления новых клиентов, изменением их данных и удалением (Рисунки 6-10).

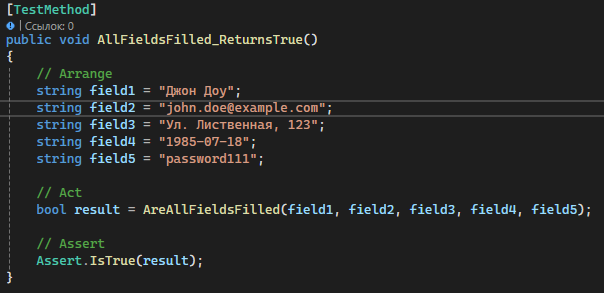


Рисунок 45 - Добавляем клиента

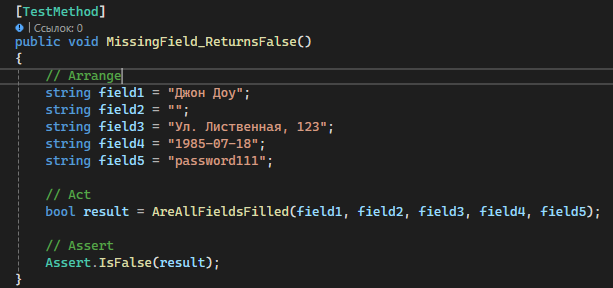


Рисунок 46 - Проверяем что при пустом поле будет исключение

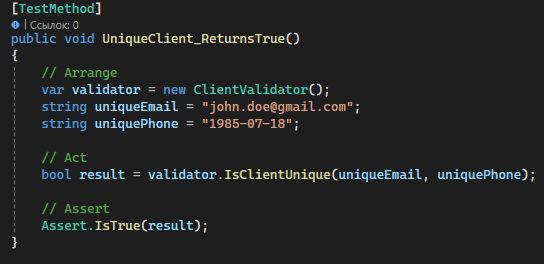


Рисунок 47 - Проверяет уникальность почты и номера

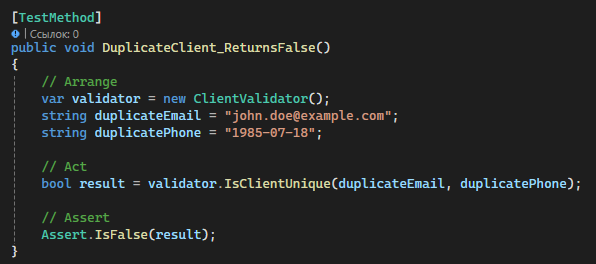


Рисунок 48 - Проверяет на дубликаты почты и номера

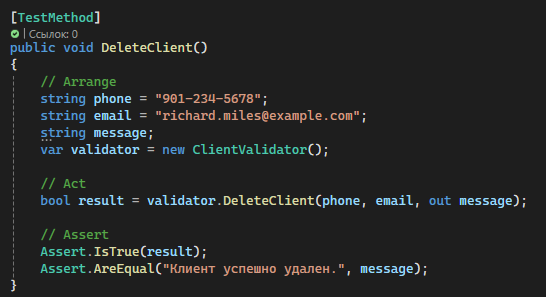


Рисунок 49 - Удаление клиента, если он существует

Запуск и проверка отработки UnitTests (Рисунок 11).

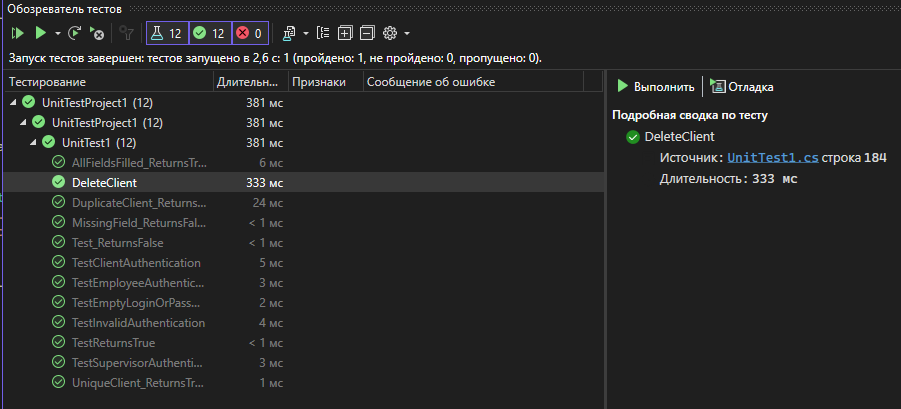


Рисунок 50 - Выполнение тестов

# Задание 6. GitHub. Оценка проекта

В репозиторий GIT выгружаю получившееся клиентское приложение, приложение-тестировщик unit-test и отчет (Рисунок 51).

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы было успешно разработано настольное приложение для парка развлечений на платформе Windows Forms. Приложение предоставляет пользователям интуитивно понятный интерфейс, позволяющий легко навигировать между различными услугами и получать необходимую информацию.

Данное приложение может служить основой для дальнейшей разработки и улучшения функционала. В будущем планируется интеграция с базой данных для управления клиентскими заказами и билетами, а также добавление новых функций, таких как обработка заказов и управление событиями, аттракционными.

Таким образом, выполненная работа не только продемонстрировала навыки разработки программного обеспечения, но и создала платформу для дальнейших улучшений, способствующих повышению эффективности работы страховой компании и улучшению клиентского опыта.

# СПИСОК ЛИТЕРАТУРЫ

1. Павловская, Татьяна Александровна. C#. Программирование на языке высокого уровня: учебник для вузов по направлению подготовки дипломированных специалистов "Информатика и вычислительная техника" / Т. А. Павловская.— М. [и др.] : Питер, 2016.— 432 с. : ил. ; — (Учебник для вузов).
2. Орлов, Сергей Александрович. Технологии разработки программного обеспечения. Учебное пособие. 2-ое издание/С.А.Орлов. – СПб.:Питер, 2016. – 480 с.: ил.
3. Кумскова, И. А. Базы данных: учебник для СПО / И. А. Кумскова.- М. : КНОРУС, 2016.- 488 с.
4. Интуит. Качесто ПО и методы его контроля [Офиц.сайт]. URL: <http://www.intuit.ru/studies/courses/64/64/lecture/1874?page=1>

# ПРИЛОЖЕНИЯ

(обязательное)

## Приложение А

Программный код для формы входа:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace Pracrica02

{

public partial class Form1 : Form

{

//string connectionString = @"Data Source=ADCLG1;Initial catalog=!Park;Integrated Security=True";

string connectionString = @"Data Source=ADCLG1;Initial catalog=!1!Park;Integrated Security=True";

public Form1()

{

InitializeComponent();

}

private void button3\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

string login = textBox1.Text.Trim();

string password = textBox2.Text.Trim();

if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))

{

MessageBox.Show("Пожалуйста, введите логин и пароль.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

// Проверка в таблице Client

string clientQuery = "SELECT COUNT(\*) FROM Client WHERE (Email = @Login OR Phone = @Login) AND Password = @Password";

SqlCommand clientCommand = new SqlCommand(clientQuery, connection);

clientCommand.Parameters.AddWithValue("@Login", login);

clientCommand.Parameters.AddWithValue("@Password", password);

if ((int)clientCommand.ExecuteScalar() > 0)

{

UserManager.SetCurrentUser(login, "Client");

ShowMainForm();

return;

}

// Проверка в таблице Employee

string employeeQuery = "SELECT COUNT(\*) FROM Employee WHERE (Email = @Login OR Phone = @Login) AND Password = @Password";

SqlCommand employeeCommand = new SqlCommand(employeeQuery, connection);

employeeCommand.Parameters.AddWithValue("@Login", login);

employeeCommand.Parameters.AddWithValue("@Password", password);

if ((int)employeeCommand.ExecuteScalar() > 0)

{

UserManager.SetCurrentUser(login, "Employee");

ShowMainForm();

return;

}

// Проверка в таблице Supervisor

string supervisorQuery = "SELECT COUNT(\*) FROM Supervisor WHERE (Email = @Login OR Phone = @Login) AND Password = @Password";

SqlCommand supervisorCommand = new SqlCommand(supervisorQuery, connection);

supervisorCommand.Parameters.AddWithValue("@Login", login);

supervisorCommand.Parameters.AddWithValue("@Password", password);

if ((int)supervisorCommand.ExecuteScalar() > 0)

{

UserManager.SetCurrentUser(login, "Supervisor");

ShowMainForm();

return;

}

// Если ни один из вариантов не совпал, выводим сообщение об ошибке

MessageBox.Show("Неверный логин или пароль.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ShowMainForm()

{

Form3 form = new Form3();

form.Show();

this.Hide();

}

private void button2\_Click(object sender, EventArgs e)

{

Form2 secondForm = new Form2();

secondForm.Show(); // Для открытия формы

this.Hide(); // Если хотите скрыть первую форму

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

textBox2.UseSystemPasswordChar = true;

}

}

}

Программный код для регистрации нового пользователя:

namespace Pracrica02

{

public partial class Form2 : Form

{

//string connectionString = @"Data Source=ADCLG1;Initial catalog=!Park;Integrated Security=True";

string connectionString = @"Data Source=ADCLG1;Initial catalog=!1!Park;Integrated Security=True";

public Form2()

{

InitializeComponent();

}

private void LoadData()

{

string query = "SELECT \* FROM Client"; // Предполагается, что вы хотите загружать данные из таблицы Client

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable table = new DataTable();

adapter.Fill(table);

// dataGridView1.DataSource = table; // Раскомментируйте, если хотите отображать данные

}

}

private void button2\_Click(object sender, EventArgs e)

{

// Считываем введенные данные

string email = textBox3.Text.Trim();

string phone = textBox2.Text.Trim();

string password = textBox5.Text.Trim();

// Проверяем, заполнены ли все поля

if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox4.Text) ||

string.IsNullOrEmpty(email) || string.IsNullOrEmpty(phone) ||

string.IsNullOrEmpty(password))

{

MessageBox.Show("Пожалуйста, заполните все поля.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

// Проверка, существует ли клиент с таким email или телефоном

string checkUserQuery = "SELECT COUNT(\*) FROM Client WHERE Email = @Email OR Phone = @Phone";

using (SqlCommand checkUserCommand = new SqlCommand(checkUserQuery, connection))

{

checkUserCommand.Parameters.AddWithValue("@Email", email);

checkUserCommand.Parameters.AddWithValue("@Phone", phone);

int userCount = (int)checkUserCommand.ExecuteScalar();

if (userCount > 0)

{

MessageBox.Show("Пользователь с таким email или телефоном уже существует.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

// Вставка нового клиента

string query = "INSERT INTO Client (Name, Email, DateOfBirth, Address, Phone, Password) " +

"VALUES (@Name, @Email, @DateOfBirth, @Address, @Phone, @Password)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", textBox1.Text.Trim());

command.Parameters.AddWithValue("@Email", email);

command.Parameters.AddWithValue("@DateOfBirth", dateTimePicker1.Value); // Замените на реальную дату рождения, если есть

command.Parameters.AddWithValue("@Address", textBox4.Text.Trim()); // Замените на реальный адрес, если есть

command.Parameters.AddWithValue("@Phone", phone);

command.Parameters.AddWithValue("@Password", password); // Не забудьте использовать хеширование для пароля

command.ExecuteNonQuery();

}

}

MessageBox.Show("Регистрация прошла успешно!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadData();

UserManager.SetCurrentUser(email, "Client");

Form3 form = new Form3();

form.Show();

this.Hide();

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 secondForm = new Form1();

secondForm.Show(); // Для открытия формы

this.Hide(); // Если хотите скрыть первую форму

}

private void button3\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

Программный код для отображения данных в зависимости от роли пользователя:

private void LoadData()

{

string userTable = UserManager.CurrentUser.Table;

switch (userTable)

{

case "Client":

tabControl1.TabPages.Remove(tabPage2);

tabControl1.TabPages.Remove(tabPage3);

button1.Visible = false;

button3.Visible = false;

LoadClientData();

break;

case "Employee":

tabPage1.Visible = true;

tabPage2.Visible = true;

button12.Visible = false;

button10.Visible = false;

tabControl1.TabPages.Remove(tabPage3);

LoadEmployeeData();

break;

case "Supervisor":

tabPage1.Visible = true;

tabPage2.Visible = true;

tabPage3.Visible = true;

LoadSupervisorData();

break;

default:

tabControl1.TabPages.Remove(tabPage1);

tabControl1.TabPages.Remove(tabPage2);

tabControl1.TabPages.Remove(tabPage3);

MessageBox.Show("Ошибка: Пользователь не привязан к допустимой таблице.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

private void LoadClientData()

{

string query = "SELECT Name, Phone, Email, Address, DateOfBirth, Password FROM Client WHERE Email = @Login OR Phone = @Login";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Login", UserManager.CurrentUser.Login);

connection.Open();

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

dataGridView1.Rows.Clear();

dataGridView1.Columns.Clear();

// Добавляем только нужные столбцы

dataGridView1.Columns.Add("Name", "Name");

dataGridView1.Columns.Add("Phone", "Phone");

dataGridView1.Columns.Add("Email", "Email");

dataGridView1.Columns.Add("Address", "Address");

dataGridView1.Columns.Add("DateOfBirth", "DateOfBirth");

var passwordColumn = new DataGridViewTextBoxColumn

{

Name = "Password",

HeaderText = "Password"

};

dataGridView1.Columns.Add(passwordColumn);

// Заполнение строки данными клиента

dataGridView1.Rows.Add(

reader["Name"].ToString(),

reader["Phone"].ToString(),

reader["Email"].ToString(),

reader["Address"].ToString(),

Convert.ToDateTime(reader["DateOfBirth"]).ToString("yyyy-MM-dd"),

new string('\*', reader["Password"].ToString().Length)

);

// Настройка ComboBox и других элементов интерфейса

comboBox1.Items.Clear();

comboBox1.Items.Add(reader["Name"].ToString());

comboBox2.Items.Clear();

comboBox2.Items.Add(reader["Phone"].ToString());

comboBox3.Items.Clear();

comboBox3.Items.Add(reader["Email"].ToString());

comboBox4.Items.Clear();

comboBox4.Items.Add(reader["Address"].ToString());

dateTimePicker1.Value = Convert.ToDateTime(reader["DateOfBirth"]);

}

else

{

MessageBox.Show("Ошибка: Данные клиента не найдены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

// Метод для загрузки данных сотрудника в dataGridView2

private void LoadEmployeeData()

{

string employeeQuery = @"

SELECT e.EmployeeID, e.Name, e.Position, e.HireDate, e.Phone, e.Email, e.Password,

COALESCE(s.Name, 'Нет руководителя') AS SupervisorName

FROM Employee e

LEFT JOIN Supervisor s ON e.SupervisorID = s.SupervisorID

WHERE e.Email = @Login OR e.Phone = @Login";

string clientQuery = "SELECT Name, Phone, Email, Address, DateOfBirth FROM Client";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand employeeCommand = new SqlCommand(employeeQuery, connection);

employeeCommand.Parameters.AddWithValue("@Login", UserManager.CurrentUser.Login);

connection.Open();

using (SqlDataReader reader = employeeCommand.ExecuteReader())

{

if (reader.Read())

{

dataGridView2.Rows.Clear();

dataGridView2.Columns.Clear();

dataGridView2.Columns.Add("Name", "Name");

dataGridView2.Columns.Add("Position", "Position");

dataGridView2.Columns.Add("HireDate", "HireDate");

dataGridView2.Columns.Add("Phone", "Phone");

dataGridView2.Columns.Add("Email", "Email");

dataGridView2.Columns.Add("SupervisorName", "Supervisor");

var passwordColumn = new DataGridViewTextBoxColumn

{

Name = "Password",

HeaderText = "Password"

};

dataGridView2.Columns.Add(passwordColumn);

string supervisorName = reader["SupervisorName"].ToString();

dataGridView2.Rows.Add(

reader["Name"].ToString(),

reader["Position"].ToString(),

Convert.ToDateTime(reader["HireDate"]).ToString("yyyy-MM-dd"),

reader["Phone"].ToString(),

reader["Email"].ToString(),

supervisorName,

reader["Password"].ToString()

);

comboBox5.Items.Clear();

comboBox5.Items.Add(supervisorName);

comboBox5.SelectedIndex = 0;

comboBox5.Enabled = supervisorName != "Нет руководителя";

comboBox8.Items.Clear();

comboBox8.Items.Add(reader["Name"].ToString());

comboBox6.Items.Clear();

comboBox6.Items.Add(reader["Phone"].ToString());

comboBox13.Items.Clear();

comboBox13.Items.Add(reader["Email"].ToString());

comboBox7.Items.Clear();

comboBox7.Items.Add(reader["Position"].ToString());

dateTimePicker2.Value = Convert.ToDateTime(reader["HireDate"]);

}

else

{

MessageBox.Show("Ошибка: Данные сотрудника не найдены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

SqlDataAdapter clientAdapter = new SqlDataAdapter(clientQuery, connection);

DataTable clientTable = new DataTable();

clientAdapter.Fill(clientTable);

dataGridView1.DataSource = clientTable;

comboBox1.Items.Clear();

comboBox2.Items.Clear();

comboBox3.Items.Clear();

comboBox4.Items.Clear();

foreach (DataRow row in clientTable.Rows)

{

comboBox1.Items.Add(row["Name"].ToString());

comboBox2.Items.Add(row["Phone"].ToString());

comboBox3.Items.Add(row["Email"].ToString());

comboBox4.Items.Add(row["Address"].ToString());

}

}

}

private void LoadSupervisorData()

{

// Запрос для загрузки данных руководителя

string supervisorQuery = "SELECT Name, Email, Phone, Department, Password FROM Supervisor WHERE Email = @Login OR Phone = @Login";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(supervisorQuery, connection);

command.Parameters.AddWithValue("@Login", UserManager.CurrentUser.Login);

connection.Open();

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

// Отобразить данные руководителя в dataGridView3

dataGridView3.Rows.Clear();

dataGridView3.Columns.Clear();

dataGridView3.Columns.Add("Name", "Name");

dataGridView3.Columns.Add("Email", "Email");

dataGridView3.Columns.Add("Phone", "Phone");

dataGridView3.Columns.Add("Department", "Department");

dataGridView3.Columns.Add("Password", "Password");

dataGridView3.Rows.Add(

reader["Name"].ToString(),

reader["Email"].ToString(),

reader["Phone"].ToString(),

reader["Department"].ToString(),

reader["Password"].ToString() // Пароль в открытом виде

);

comboBox12.Items.Clear();

comboBox12.Items.Add(reader["Name"].ToString());

comboBox10.Items.Clear();

comboBox10.Items.Add(reader["Phone"].ToString());

comboBox11.Items.Clear();

comboBox11.Items.Add(reader["Email"].ToString());

comboBox9.Items.Clear();

comboBox9.Items.Add(reader["Department"].ToString());

}

else

{

MessageBox.Show("Ошибка: Данные руководителя не найдены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

}

// Запрос для загрузки данных сотрудников в отделе руководителя

string employeesQuery = @"

SELECT e.Name, e.Position, e.HireDate, e.Phone, e.Email,

CASE WHEN e.SupervisorID IS NULL THEN 'Без руководителя' ELSE s.Name END AS SupervisorName

FROM Employee e

LEFT JOIN Supervisor s ON e.SupervisorID = s.SupervisorID

WHERE s.Email = @Login OR s.Phone = @Login OR e.SupervisorID IS NULL";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(employeesQuery, connection);

command.Parameters.AddWithValue("@Login", UserManager.CurrentUser.Login);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable employeeTable = new DataTable();

adapter.Fill(employeeTable);

dataGridView2.DataSource = employeeTable;

// Заполнение комбобоксов данными о сотрудниках

comboBox5.Items.Clear();

comboBox8.Items.Clear();

comboBox6.Items.Clear();

comboBox13.Items.Clear();

comboBox7.Items.Clear();

foreach (DataRow row in employeeTable.Rows)

{

comboBox5.Items.Add(row["SupervisorName"].ToString()); // Имя руководителя или "Без руководителя"

comboBox8.Items.Add(row["Name"].ToString()); // Имя сотрудника

comboBox6.Items.Add(row["Phone"].ToString()); // Телефон сотрудника

comboBox13.Items.Add(row["Email"].ToString()); // Email сотрудника

comboBox7.Items.Add(row["Position"].ToString()); // Должность сотрудника

}

}

// Загрузка данных всех клиентов для отображения в dataGridView1 и ComboBox

string clientsQuery = "SELECT Name, Phone, Email, Address, DateOfBirth FROM Client";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(clientsQuery, connection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable clientTable = new DataTable();

adapter.Fill(clientTable);

dataGridView1.DataSource = clientTable; // Отображаем всех клиентов в dataGridView1

// Настройка ComboBox для выбора данных по всем клиентам

comboBox1.Items.Clear();

comboBox2.Items.Clear();

comboBox3.Items.Clear();

comboBox4.Items.Clear();

foreach (DataRow row in clientTable.Rows)

{

comboBox1.Items.Add(row["Name"].ToString()); // ФИО клиента

comboBox2.Items.Add(row["Phone"].ToString()); // Номер телефона

comboBox3.Items.Add(row["Email"].ToString()); // Email

comboBox4.Items.Add(row["Address"].ToString()); // Адрес

}

}

}

Пример на программном коде клиента для добавления/удаления/изменения данных:

private void button1\_Click(object sender, EventArgs e)

{

// Проверка на заполненность всех обязательных полей

if (string.IsNullOrWhiteSpace(comboBox1.Text) || string.IsNullOrWhiteSpace(comboBox2.Text) || string.IsNullOrWhiteSpace(comboBox3.Text) || string.IsNullOrWhiteSpace(comboBox4.Text) || string.IsNullOrWhiteSpace(textBox2.Text))

{

MessageBox.Show("Пожалуйста, заполните все поля.");

return;

}

// Проверка на уникальность номера телефона и email

string checkQuery = @"

SELECT COUNT(\*)

FROM Client

WHERE Email = @Email OR Phone = @Phone";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(checkQuery, connection);

command.Parameters.AddWithValue("@Email", comboBox3.Text); // Используем Text вместо SelectedItem

command.Parameters.AddWithValue("@Phone", comboBox2.Text); // Используем Text вместо SelectedItem

connection.Open();

int count = (int)command.ExecuteScalar();

if (count > 0)

{

MessageBox.Show("Клиент с таким Email или номером телефона уже существует.");

return;

}

}

// Добавление нового клиента в базу данных

string insertQuery = @"

INSERT INTO Client (Name, Email, DateOfBirth, Address, Phone, Password)

VALUES (@Name, @Email, @DateOfBirth, @Address, @Phone, @Password)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(insertQuery, connection);

command.Parameters.AddWithValue("@Name", comboBox1.Text); // Используем Text вместо SelectedItem

command.Parameters.AddWithValue("@Email", comboBox3.Text); // Используем Text вместо SelectedItem

command.Parameters.AddWithValue("@DateOfBirth", dateTimePicker1.Value.Date);

command.Parameters.AddWithValue("@Address", comboBox4.Text); // Используем Text вместо SelectedItem

command.Parameters.AddWithValue("@Phone", comboBox2.Text); // Используем Text вместо SelectedItem

command.Parameters.AddWithValue("@Password", textBox2.Text);

connection.Open();

command.ExecuteNonQuery();

}

MessageBox.Show("Новый клиент успешно добавлен.");

LoadData();

}

private void button2\_Click(object sender, EventArgs e)

{

// Проверка, что все поля заполнены

if (string.IsNullOrWhiteSpace(comboBox1.Text) ||

string.IsNullOrWhiteSpace(comboBox3.Text) ||

string.IsNullOrWhiteSpace(comboBox4.Text) ||

string.IsNullOrWhiteSpace(comboBox2.Text) ||

string.IsNullOrWhiteSpace(textBox2.Text))

{

MessageBox.Show("Пожалуйста, заполните все поля.");

return;

}

// Обновление данных клиента по номеру телефона или email

string updateQuery = @"

UPDATE Client

SET Name = @Name, DateOfBirth = @DateOfBirth, Address = @Address, Password = @Password

WHERE Phone = @Phone OR Email = @Email";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(updateQuery, connection);

// Заполнение параметров

command.Parameters.AddWithValue("@Name", comboBox1.Text);

command.Parameters.AddWithValue("@DateOfBirth", dateTimePicker1.Value.Date);

command.Parameters.AddWithValue("@Address", comboBox4.Text);

command.Parameters.AddWithValue("@Password", textBox2.Text);

command.Parameters.AddWithValue("@Phone", comboBox2.Text);

command.Parameters.AddWithValue("@Email", comboBox3.Text);

connection.Open();

try

{

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Данные клиента успешно обновлены.");

LoadData();

}

else

{

MessageBox.Show("Клиент с указанным номером телефона или email не найден.");

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при обновлении данных: " + ex.Message);

}

}

}

private void button3\_Click(object sender, EventArgs e)

{

// Проверка, что номер телефона или email заполнены

if (string.IsNullOrWhiteSpace(comboBox2.Text) && string.IsNullOrWhiteSpace(comboBox3.Text))

{

MessageBox.Show("Пожалуйста, укажите номер телефона или email для удаления клиента.");

return;

}

// Удаление заказов, связанных с клиентом

string deleteOrdersQuery = @"

DELETE FROM [Order]

WHERE ClientID = (SELECT ClientID FROM Client WHERE Phone = @Phone OR Email = @Email)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

// Удаление связанных заказов

SqlCommand deleteOrdersCommand = new SqlCommand(deleteOrdersQuery, connection);

deleteOrdersCommand.Parameters.AddWithValue("@Phone", comboBox2.Text);

deleteOrdersCommand.Parameters.AddWithValue("@Email", comboBox3.Text);

try

{

deleteOrdersCommand.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении связанных заказов: " + ex.Message);

return;

}

// Удаление клиента

string deleteClientQuery = @"

DELETE FROM Client

WHERE Phone = @Phone OR Email = @Email";

SqlCommand deleteClientCommand = new SqlCommand(deleteClientQuery, connection);

deleteClientCommand.Parameters.AddWithValue("@Phone", comboBox2.Text);

deleteClientCommand.Parameters.AddWithValue("@Email", comboBox3.Text);

try

{

int rowsAffected = deleteClientCommand.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Клиент успешно удален.");

LoadData();

}

else

{

MessageBox.Show("Клиент с указанным номером телефона или email не найден.");

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении клиента: " + ex.Message);

}

}

}

Программный код для отображения заказов в зависимости от роли пользователя:

private void LoadClientData()

{

string userTable = UserManager.CurrentUser.Table; // Получаем тип пользователя (Клиент, Сотрудник, Руководитель)

// Запрос на получение заказов клиента

string queryOrders = @"

SELECT o.OrderID, o.VisitDate, o.TicketQuantity, o.TotalPrice, o.UniqueCode, o.TicketID

FROM [Order] o

JOIN Client c ON o.ClientID = c.ClientID

WHERE c.Email = @Login OR c.Phone = @Login";

if (userTable == "Employee" || userTable == "Supervisor")

{

// Если пользователь - сотрудник или руководитель, получаем все заказы

queryOrders = @"

SELECT o.OrderID, o.VisitDate, o.TicketQuantity, o.TotalPrice, o.UniqueCode, o.TicketID

FROM [Order] o";

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(queryOrders, connection);

command.Parameters.AddWithValue("@Login", UserManager.CurrentUser.Login);

connection.Open();

using (SqlDataReader reader = command.ExecuteReader())

{

dataGridView3.Rows.Clear();

dataGridView3.Columns.Clear();

// Добавляем столбцы для отображения данных заказов

dataGridView3.Columns.Add("OrderID", "Order ID");

dataGridView3.Columns.Add("VisitDate", "Visit Date");

dataGridView3.Columns.Add("TicketQuantity", "Quantity");

dataGridView3.Columns.Add("TotalPrice", "Total Price");

dataGridView3.Columns.Add("UniqueCode", "Unique Order Number");

dataGridView3.Columns.Add("TicketID", "Ticket ID"); // Добавляем TicketID

while (reader.Read())

{

// Заполнение строками данными заказов клиента

dataGridView3.Rows.Add(

reader["OrderID"].ToString(),

Convert.ToDateTime(reader["VisitDate"]).ToString("yyyy-MM-dd"),

reader["TicketQuantity"].ToString(),

reader["TotalPrice"].ToString(), // Форматируем для отображения

reader["UniqueCode"].ToString(),

reader["TicketID"].ToString() // Заполняем TicketID

);

}

}

// Заполнение comboBox5 с TicketID

string queryTickets = "SELECT TicketID FROM Ticket"; // Получаем все ID билетов

using (SqlCommand commands = new SqlCommand(queryTickets, connection))

{

using (SqlDataReader reader = commands.ExecuteReader())

{

comboBox5.Items.Clear();

while (reader.Read())

{

comboBox5.Items.Add(reader["TicketID"].ToString());

}

}

}

// Заполнение comboBox6 в зависимости от типа пользователя

if (userTable == "Client")

{

// Уникальные номера заказов клиента

string queryUniqueCodes = "SELECT DISTINCT UniqueCode FROM [Order] WHERE ClientID = (SELECT ClientID FROM Client WHERE Email = @Login OR Phone = @Login)";

using (SqlCommand commandUniqueCodes = new SqlCommand(queryUniqueCodes, connection))

{

commandUniqueCodes.Parameters.AddWithValue("@Login", UserManager.CurrentUser.Login);

using (SqlDataReader reader = commandUniqueCodes.ExecuteReader())

{

comboBox6.Items.Clear();

while (reader.Read())

{

comboBox6.Items.Add(reader["UniqueCode"].ToString());

}

}

}

}

else // Если сотрудник или руководитель

{

// Все уникальные номера заказов

string queryUniqueCodes = "SELECT DISTINCT UniqueCode FROM [Order]";

using (SqlCommand commandUniqueCodes = new SqlCommand(queryUniqueCodes, connection))

{

using (SqlDataReader reader = commandUniqueCodes.ExecuteReader())

{

comboBox6.Items.Clear();

while (reader.Read())

{

comboBox6.Items.Add(reader["UniqueCode"].ToString());

}

}

}

}

}

// Заполнение comboBox11 с числами от 1 до 15

comboBox11.Items.Clear();

for (int i = 1; i <= 15; i++)

{

comboBox11.Items.Add(i.ToString());

}

// Подсчет суммы при выборе

comboBox5.SelectedIndexChanged += (s, e) => CalculateTotalPrice();

comboBox11.SelectedIndexChanged += (s, e) => CalculateTotalPrice();

comboBox5.SelectedIndexChanged += (s, e) => UpdateOrderNumber();

comboBox11.SelectedIndexChanged += (s, e) => UpdateOrderNumber();

textBox1.TextChanged += (s, e) => UpdateOrderNumber();

}

private void UpdateOrderNumber()

{

// Проверяем, заполнены ли все необходимые поля

if (comboBox5.SelectedItem != null && comboBox11.SelectedItem != null && !string.IsNullOrWhiteSpace(textBox1.Text))

{

comboBox6.Text = GenerateUniqueOrderNumber();

}

}

private string GenerateUniqueOrderNumber()

{

// Генерация уникального номера заказа из 6 символов

const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

Random random = new Random();

char[] orderNumber = new char[6];

for (int i = 0; i < orderNumber.Length; i++)

{

orderNumber[i] = chars[random.Next(chars.Length)];

}

return new string(orderNumber);

}

private void CalculateTotalPrice()

{

if (comboBox11.SelectedItem != null && comboBox5.SelectedItem != null)

{

int ticketID = int.Parse(comboBox5.SelectedItem.ToString());

int quantity = int.Parse(comboBox11.SelectedItem.ToString());

// Получение цены билета по TicketID

string priceQuery = "SELECT Price FROM Ticket WHERE TicketID = @TicketID";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(priceQuery, connection);

command.Parameters.AddWithValue("@TicketID", ticketID);

connection.Open();

object priceResult = command.ExecuteScalar();

if (priceResult != null)

{

decimal price = (decimal)priceResult;

decimal totalPrice = price \* quantity;

textBox1.Text = totalPrice.ToString("F2"); // Устанавливаем формат для отображения

}

else

{

textBox1.Text = "0.00"; // Если цена не найдена

}

}

}

}

## Приложение Б

(обязательное)

Листинг SQL-листа

Создаю таблицы:

-- Таблица Событий

CREATE TABLE Event (

EventID INT IDENTITY PRIMARY KEY,

Date DATE,

Name NVARCHAR(255),

StartDate DATETIME2,

EndDate DATETIME2,

Description NVARCHAR(255),

VisitorLimit INT,

AgeRestriction INT

);

-- Таблица Руководителей

CREATE TABLE Supervisor (

SupervisorID INT IDENTITY PRIMARY KEY,

Name NVARCHAR(255),

Email NVARCHAR(255) UNIQUE, -- Уникальный Email

Phone NVARCHAR(20) UNIQUE, -- Уникальный Phone

Department NVARCHAR(255),

Password NVARCHAR(255)

);

-- Обновленная таблица Аттракционов

CREATE TABLE Attraction (

AttractionID INT IDENTITY PRIMARY KEY,

Name NVARCHAR(255) NOT NULL,

SafetyStatus NVARCHAR(50) CHECK (SafetyStatus IN ('Open', 'Closed for Restoration')),

Description NVARCHAR(500) -- Добавлено поле для краткого описания

);

-- Обновленная таблица Билетов

CREATE TABLE Ticket (

TicketID INT IDENTITY PRIMARY KEY,

Status NVARCHAR(50),

ExpirationDate DATE,

AvailableAttractions NVARCHAR(255),

EventID INT, -- Ссылка на событие

AttractionID INT,

Price DECIMAL(10, 2), -- Добавлено поле для цены билета

FOREIGN KEY (EventID) REFERENCES Event(EventID),

FOREIGN KEY (AttractionID) REFERENCES Attraction(AttractionID) -- Связь с аттракционами

);

-- Таблица Сотрудников

CREATE TABLE Employee (

EmployeeID INT IDENTITY PRIMARY KEY,

Name NVARCHAR(255),

Position NVARCHAR(50),

HireDate DATE,

Phone NVARCHAR(20) UNIQUE, -- Уникальный Phone

SupervisorID INT, -- Ссылка на руководителя

Password NVARCHAR(255),

Email NVARCHAR(255) UNIQUE, -- Уникальный Email

FOREIGN KEY (SupervisorID) REFERENCES Supervisor(SupervisorID) -- Связь с руководителем

);

-- Таблица Клиентов

CREATE TABLE Client (

ClientID INT IDENTITY PRIMARY KEY,

Name NVARCHAR(50),

Email NVARCHAR(50) UNIQUE, -- Уникальный Email

DateOfBirth DATE,

Address NVARCHAR(255),

Phone NVARCHAR(20) UNIQUE, -- Уникальный Phone

Password NVARCHAR(255)

);

-- Таблица Заказов

CREATE TABLE [Order] (

OrderID INT IDENTITY PRIMARY KEY,

VisitDate DATE,

TicketQuantity INT,

TotalPrice DECIMAL(10, 2),

UniqueCode NVARCHAR(255),

ClientID INT, -- Ссылка на клиента

EmployeeID INT, -- Ссылка на сотрудника, обработавшего заказ

TicketID INT, -- Ссылка на билет

FOREIGN KEY (ClientID) REFERENCES Client(ClientID),

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),

FOREIGN KEY (TicketID) REFERENCES Ticket(TicketID) -- Связь с билетами

);

-- Заполнение таблицы Event (События)

INSERT INTO Event (Date, Name, StartDate, EndDate, Description, VisitorLimit, AgeRestriction)

VALUES

('2024-11-01', 'Концерт', '2024-11-01 19:00:00', '2024-11-01 22:00:00', 'Живое музыкальное представление', 500, 18),

('2024-12-15', 'Выставка Искусств', '2024-12-15 10:00:00', '2024-12-15 18:00:00', 'Современные художественные работы', 300, 12),

('2025-01-20', 'Техническая Конференция', '2025-01-20 09:00:00', '2025-01-20 17:00:00', 'Доклады об инновационных технологиях', 800, 21);

-- Заполнение таблицы Supervisor (Руководители)

INSERT INTO Supervisor (Name, Email, Phone, Department, Password)

VALUES

('Алиса Смирнова', 'alice.smirnova@example.com', '123-456-7890', 'Управление', 'password123'),

('Боб Иванов', 'bob.ivanov@example.com', '234-567-8901', 'Технический отдел', 'password456'),

('Чарли Петров', 'charlie.petrov@example.com', '345-678-9012', 'Обслуживание клиентов', 'password789');

-- Заполнение таблицы Employee (Сотрудники)

INSERT INTO Employee (Name, Position, HireDate, Phone, SupervisorID, Password, Email)

VALUES

('Ева Адамс', 'Техник', '2023-01-10', '456-789-0123', 2, 'password321', 'eva.adams@example.com'),

('Даниил Белый', 'Координатор мероприятий', '2022-05-15', '567-890-1234', 1, 'password654', 'daniel.beliy@example.com'),

('Грейс Ли', 'Служба поддержки клиентов', '2023-09-20', '678-901-2345', 3, 'password987', 'grace.lee@example.com');

-- Заполнение таблицы Client (Клиенты)

INSERT INTO Client (Name, Email, DateOfBirth, Address, Phone, Password)

VALUES

('Джон Доу', 'john.doe@example.com', '1985-07-15', 'Ул. Лиственная, 123', '789-012-3456', 'password111'),

('Джейн Роу', 'jane.roe@example.com', '1990-11-20', 'Ул. Кленовая, 456', '890-123-4567', 'password222'),

('Ричард Майлз', 'richard.miles@example.com', '1978-02-05', 'Ул. Дубовая, 789', '901-234-5678', 'password333');

-- Заполнение таблицы Ticket (Билеты)

INSERT INTO Ticket (Status, ExpirationDate, AvailableAttractions, EventID, AttractionID, Price)

VALUES

('Активен', '2024-11-01', 'Американские горки, Автодром', 1, 1, 500.00),

('Активен', '2024-12-15', 'Колесо обозрения, Комната страха', 2, 2, 450.00),

('Использован', '2024-11-01', 'Комната страха', 1, 3, 300.00),

('Активен', '2025-01-20', 'Автодром', 3, 4, 200.00),

('Активен', '2024-12-15', 'Водные горки', 2, 5, 600.00);

-- Заполнение таблицы Attraction (Аттракционы)

INSERT INTO Attraction (Name, SafetyStatus, Description)

VALUES

('Американские горки', 'Open', 'Экстремальные горки с крутыми спусками и подъёмами'),

('Колесо обозрения', 'Closed for Restoration', 'Высокое колесо обозрения с панорамным видом на парк'),

('Комната страха', 'Open', 'Мистическая комната с пугающими эффектами'),

('Автодром', 'Open', 'Катание на машинках с возможностью столкновений'),

('Водные горки', 'Closed for Restoration', 'Аквапарк с горками разной высоты и сложности');

INSERT INTO [Order] (VisitDate, TicketQuantity, TotalPrice, UniqueCode, ClientID, EmployeeID, TicketID)

VALUES

('2024-11-01', 2, 150.00, 'ORD12345', 1, 2, 4), -- TicketID=1

('2024-12-15', 1, 75.00, 'ORD67890', 2, 1, 2), -- TicketID=2

('2025-01-20', 3, 225.00, 'ORD54321', 3, 3, 3); -- TicketID=3

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Фоминых Валентина Романовна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| **Тема 1.** Участие в выработке требований к программному обеспечению. | 12 |  |
| **Тема 2.** Стадии проектирования программного обеспечения. Проектирование UML-диаграмм. | 15 |  |
| **Тема 3.** Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения. | 25 |  |
| **Тема 4.** Тестирование и отладка программных модулей. Модульное тестирование. | 12 |  |
| **Тема 5.** Разработка программной документации. | 9 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены **/** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Хисамутдинова А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «09» ноября 2024 г.