**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 42919/8 группы

Соколова Милена Александровна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «07» октября 2024 г. по «19» октября 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 42919/8 группы

Соколова Милена Александровна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «07» октября 2024 г. по «19» октября 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 3**

Задание выдал «07» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «07» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Соколова М.А.(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 42919/8 группы

Соколова Милена Александровна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «07» октября 2024 г. по «19» октября 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** |
| 07.10.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 08.10.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 09.10.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 10.10.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 11.10.2024 | Проектирование интерфейса пользователя. |  |
| 12.10.2024 | Разработка схемы алгоритма программного продукта. |  |
| 14.10.2024 | Разработка модели данных. |  |
| 15.10.2024 | Разработка программного продукта. Создание функциональности ПО. |  |
| 16.10.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 17.10.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 18.10.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. |  |
| 19.10.2024 | Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. Подготовка отчета. |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc180089282)

[Задание 1. Разработка технического задания 7](#_Toc180089283)

[Задание 2. Разработка алгоритмов и диаграмм 12](#_Toc180089284)

[Задание 3. Макеты 20](#_Toc180089285)

[Задание 4. Разработка программы 24](#_Toc180089286)

[Задание 5. Тестирование 32](#_Toc180089287)

[Таблица 1 – Проверка входа администратора с корректными данными 32](#_Toc180089288)

[Задание 6. GitHub. Оценка проекта 38](#_Toc180089289)

[ЗАКЛЮЧЕНИЕ 41](#_Toc180089290)

[ПРИЛОЖЕНИЕ A 42](#_Toc180089291)

[ПРИЛОЖЕНИЕ B 58](#_Toc180089292)

# ВВЕДЕНИЕ

Учебная практика посвящена анализу и разработке программного проекта. Основные цели проекта включают создание технического задания, разработку диаграмм, создание базы данных и разработку приложения.

В результате выполнения данной работы будет создана надежная и эффективная система управления данными, которая улучшит аналитические возможности по учету книг в библиотеке. Эта учебная практика имеет как практическое, так и теоретическое значение, так как дает возможность изучить особенности проектирования и внедрения баз данных, а также приобрести навыки работы с различными инструментами и технологиями, применяемыми в процессе создания баз данных.

# Задание 1. Разработка технического задания

Описание предметной области

Основной целью данного модуля является автоматизация процессов управления фитнес-клубом. Модуль позволяет сотрудникам клуба эффективно управлять записями клиентов на занятия, отслеживать доступные услуги и обеспечивать высокий уровень обслуживания посетителей.

Разработка программного модуля для учета и управления деятельностью фитнес-клуба - это процесс создания программного продукта, который помогает клубам эффективно организовывать график занятий, управлять абонементами и следить за состоянием клиентов.

Задача разработки такого программного модуля состоит в создании удобного и эффективного инструмента для учета, отслеживания и управления услугами клуба, что помогает минимизировать очереди и обеспечивает бесперебойную работу клуба.

Основные функции и возможности модуля включают:

1. Запись на занятия:

* Клиент оставляет новую заявку на участие в тренировке или занятии (например, йога, пилатес, высокоинтенсивные тренировки).
* Заявка может содержать информацию о виде занятия, дате и времени, а также личные данные (ФИО клиента, номер телефона).
* Клиент может редактировать свою заявку.

1. Регистрация заявки:

* Процесс включает приём и регистрацию заявки сотрудником клуба в системе учета.
* Важными аспектами регистрации являются присвоение уникального идентификатора заявке и сохранение информации о ней.

1. Обработка заявки:

* Процесс включает анализ заявки, определение ее приоритетности и назначение тренера или курса в зависимости от доступности.
* В процессе обработки может потребоваться уточнение деталей у клиента (например, уровень подготовки, предпочтения).

1. Исполнение заявки:

* Фактическое проведение занятия или тренировки.
* На этом этапе клиент принимает участие в занятии, получает инструкции от тренера, а также обратную связь о выполнении упражнений.

1. Отчётность и информирование:

* Фиксация и отчёт о выполненных занятиях.
* После завершения занятия клиент может оставить отзыв о тренере и уровне сервиса, а также получить информацию о дополнительных услугах и акциях клуба.

1. Мониторинг и анализ:

* Контроль и анализ процесса учета записей на занятия.
* Важно отслеживать и анализировать посещаемость, качество проведенных занятий, отзывы клиентов и другие параметры, помогающие оптимизировать работу клуба.

Техническое задание: Фитнес-клуб

1. Общие сведения

1.1. Наименование проекта: Разработка программного продукта для автоматизации управления фитнес-клубом.

1.2. Заказчик: Фитнес-клуб "R&M".

1.3. Исполнитель: Разработчик ПО "IT Solutions".

2. Функциональные требования

Программный продукт должен обеспечивать управление деятельностью фитнес-клуба, включая ведение базы данных о клиентах, сотрудниках, абонементах, расписаниях тренировок и продажах дополнительных услуг. Основные роли системы: администратор, сотрудник, клиент.

2.1. Роли пользователей:

- Администратор:

- Управление пользователями (создание, редактирование, удаление).

- Управление расписаниями тренировок.

- Управление абонементами и их состояниями.

- Мониторинг финансовых операций.

- Формирование отчетов.

- Сотрудник:

- Регистрация клиентов.

- Продажа абонементов.

- Управление записью на тренировки.

- Ведение статистики по посещаемости клиентов.

- Клиент:

- Личный кабинет (просмотр и изменение личных данных).

- Просмотр расписания тренировок.

- Запись на тренировки.

- Просмотр состояния абонемента и истории посещений.

2.2. Основные функции системы:

- Управление пользователями:

- Создание профилей пользователей с разделением по ролям (администратор, сотрудник, клиент).

- Управление правами доступа в зависимости от роли.

- Работа с абонементами:

- Продажа абонементов (индивидуальные, групповые).

- Продление и аннулирование абонементов.

- Отслеживание срока действия абонементов и уведомление клиентов о необходимости продления.

- Управление расписанием:

- Создание и редактирование расписания тренировок.

- Назначение тренеров и тренажерных залов.

- Управление записью клиентов на тренировки (просмотр доступных мест, регистрация, отмена записи).

- Ведение базы данных финансовых операций (продажа абонементов, дополнительных услуг).

- Создание отчетов по количеству проданных абонементов, посещаемости тренировок, доходам клуба.

2.3. Операции с данными:

- Добавление, редактирование и удаление данных по основным таблицам:

- Клиенты;

- Сотрудники;

- Абонементы;

- Тренировки;

- Финансовые операции.

- Проверка корректности вводимых данных.

- Вывод сообщений об ошибках или успешности выполнения операций.

3. Нефункциональные требования

3.1. Безопасность:

- Реализация системы авторизации и аутентификации.

- Разделение доступа к функционалу в зависимости от роли пользователя.

- Шифрование паролей пользователей.

3.2. Масштабируемость:

- Возможность добавления новых ролей пользователей и дополнительных функций без необходимости изменения основной архитектуры.

- Возможность увеличения объема данных (расширение клиентской базы, тренировок и т.д.) без снижения производительности системы.

3.3. Производительность:

- Быстрый доступ к данным.

- Минимальное время отклика на запросы пользователя.

3.4. Удобство использования:

- Интуитивно понятный интерфейс.

- Уведомления об успешных или неудачных операциях.

4. Требования к реализации

4.1. Язык программирования: на усмотрение разработчика (C#).

4.2. СУБД: на усмотрение разработчика (MS SQL).

5. Требования к документации

5.1. Техническое задание:

- Должна быть разработана подробная документация по всем функциям и модулям системы, описывающая технические и пользовательские аспекты.

6. Руководство по стилю

6.1. Шрифт: Times New Roman.

6.2. Цветовая схема:

- Основные цвета интерфейса: черный, серые и белые элементы управления.

# Задание 2. Разработка алгоритмов и диаграмм

По составленному техническому заданию необходимо разработать алгоритмы в виде UML-диаграмм (Рисунок 1-3) и ER-диаграмм (Рисунок 4):

Рисунок 1 – Диаграмма вариантов использования



Рисунок 2 – Диаграмма последовательности



Рисунок 3 – Диаграмма активности



Рисунок 4 – ER-диаграмма

Таблица Clients

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение NULL |
| ClientID | int | Запрещено |
| FirstName | nvarchar(50) | Разрешено |
| LastName | nvarchar(50) | Разрешено |
| Email | nvarchar(100) | Разрешено |
| PhoneNumber | nvarchar(15) | Разрешено |
| DateOfBirth | date | Разрешено |
| RegistrationDate | date | Разрешено |

Таблица ClientWorkouts

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение NULL |
| ClientID | int | Запрещено |
| WorkoutID | int | Запрещено |

Таблица Workouts

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение NULL |
| WorkoutID | int | Запрещено |
| EmployeeID | int | Разрешено |
| WorkoutName | nvarchar(50) | Разрешено |
| WorkoutDate | datetime | Разрешено |
| DurationInMinutes | int | Разрешено |

Таблица Employees

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение NULL |
| EmployeeID | int | Запрещено |
| FirstName | nvarchar(50) | Разрешено |
| LastName | nvarchar(50) | Разрешено |
| Position | nvarchar(50) | Разрешено |
| HireDate | date | Разрешено |
| Salary | decimal(10, 2) | Разрешено |

Таблица Transactions

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение NULL |
| TransactionID | int | Запрещено |
| ClientID | int | Разрешено |
| MembershipID | int | Разрешено |
| Amount | decimal(10, 2) | Разрешено |
| TransactionDate | datetime | Разрешено |

Таблица Memberships

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение NULL |
| MembershipID | int | Запрещено |
| ClientID | int | Разрешено |
| Type | nvarchar(50) | Разрешено |
| StartDate | date | Разрешено |
| EndDate | date | Разрешено |
| Price | decimal(10, 2) | Разрешено |

Создание таблиц и заполнение их в SQL сервере.

-- Таблица Clients

CREATE TABLE Clients (

ClientID INT NOT NULL PRIMARY KEY,

FirstName NVARCHAR(50) NULL,

LastName NVARCHAR(50) NULL,

Email NVARCHAR(100) NULL,

PhoneNumber NVARCHAR(15) NULL,

DateOfBirth DATE NULL,

RegistrationDate DATE NULL

);

INSERT INTO Clients (ClientID, FirstName, LastName, Email, PhoneNumber, DateOfBirth, RegistrationDate) VALUES

(1, 'Иван', 'Иванов', 'ivan.ivanov@example.com', '1234567890', '1990-01-15', '2023-01-01'),

(2, 'Мария', 'Петрова', 'maria.petrovna@example.com', '0987654321', '1985-02-20', '2023-01-02'),

(3, 'Сергей', 'Сидоров', 'sergey.sidorov@example.com', '1472583690', '1992-03-30', '2023-01-03');

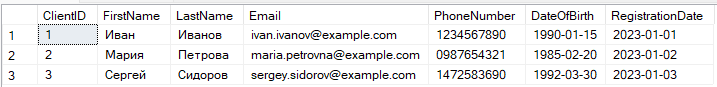


Рисунок 5 – Таблица клиентов

-- Таблица ClientWorkouts

CREATE TABLE ClientWorkouts (

ClientID INT NOT NULL,

WorkoutID INT NOT NULL,

FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),

PRIMARY KEY (ClientID, WorkoutID)

);

-- Заполнение таблицы ClientWorkouts

INSERT INTO ClientWorkouts (ClientID, WorkoutID) VALUES

(1, 1),

(1, 2),

(2, 1),

(3, 3);

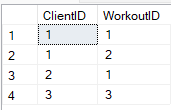


Рисунок 6 – Таблица записей на занятия

-- Таблица Workouts

CREATE TABLE Workouts (

WorkoutID INT NOT NULL PRIMARY KEY,

EmployeeID INT NULL,

WorkoutName NVARCHAR(50) NULL,

WorkoutDate DATETIME NULL,

DurationInMinutes INT NULL

);

-- Заполнение таблицы Workouts

INSERT INTO Workouts (WorkoutID, EmployeeID, WorkoutName, WorkoutDate, DurationInMinutes) VALUES

(1, 1, 'Йога', '2023-10-01 10:00:00', 60),

(2, 2, 'Фитнес', '2023-10-02 11:00:00', 45),

(3, 3, 'Силовая тренировка', '2023-10-03 12:00:00', 75);

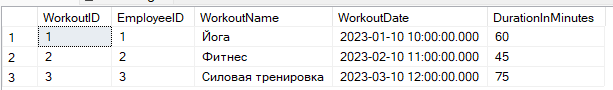


Рисунок 7 – Таблица тренировок

-- Таблица Employees

CREATE TABLE Employees (

EmployeeID INT NOT NULL PRIMARY KEY,

FirstName NVARCHAR(50) NULL,

LastName NVARCHAR(50) NULL,

Position NVARCHAR(50) NULL,

HireDate DATE NULL,

Salary DECIMAL(10, 2) NULL

);

-- Заполнение таблицы Employees

INSERT INTO Employees (EmployeeID, FirstName, LastName, Position, HireDate, Salary) VALUES

(1, 'Анна', 'Кузнецова', 'Тренер', '2020-05-15', 50000.00),

(2, 'Дмитрий', 'Смирнов', 'Администратор', '2021-06-10', 40000.00),

(3, 'Елена', 'Фролова', 'Тренер', '2019-04-22', 55000.00);

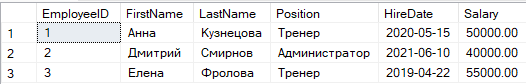


Рисунок 8 – Таблица сотрудников

-- Таблица Transactions

CREATE TABLE Transactions (

TransactionID INT NOT NULL PRIMARY KEY,

ClientID INT NULL,

MembershipID INT NULL,

Amount DECIMAL(10, 2) NULL,

TransactionDate DATETIME NULL

);

-- Заполнение таблицы Transactions

INSERT INTO Transactions (TransactionID, ClientID, MembershipID, Amount, TransactionDate) VALUES

(1, 1, 1, 12000.00, '2023-01-01 09:00:00'),

(2, 2, 2, 6000.00, '2023-01-02 10:00:00'),

(3, 3, 3, 2000.00, '2023-01-03 11:00:00');

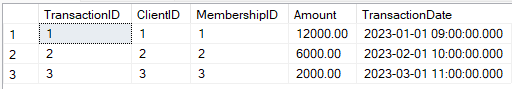


Рисунок 9 – Таблица финансов

-- Таблица Memberships

CREATE TABLE Memberships (

MembershipID INT NOT NULL PRIMARY KEY,

ClientID INT NULL,

Type NVARCHAR(50) NULL,

StartDate DATE NULL,

EndDate DATE NULL,

Price DECIMAL(10, 2) NULL

);

-- Заполнение таблицы Memberships

INSERT INTO Memberships (MembershipID, ClientID, Type, StartDate, EndDate, Price) VALUES

(1, 1, 'Годовой', '2023-01-01', '2024-01-01', 12000.00),

(2, 2, 'Полугодовой', '2023-01-02', '2023-07-01', 6000.00),

(3, 3, 'Месячный', '2023-01-03', '2023-02-01', 2000.00);

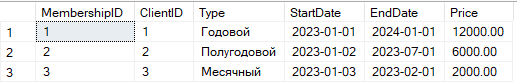


Рисунок 10 – Таблица абонементов

# Задание 3. Макеты

При разработке приложения были разработаны wireframe (Рисунок 11-13) и mockup (Рисунок 14-16) макеты, а также карта навигации (Рисунок 17).

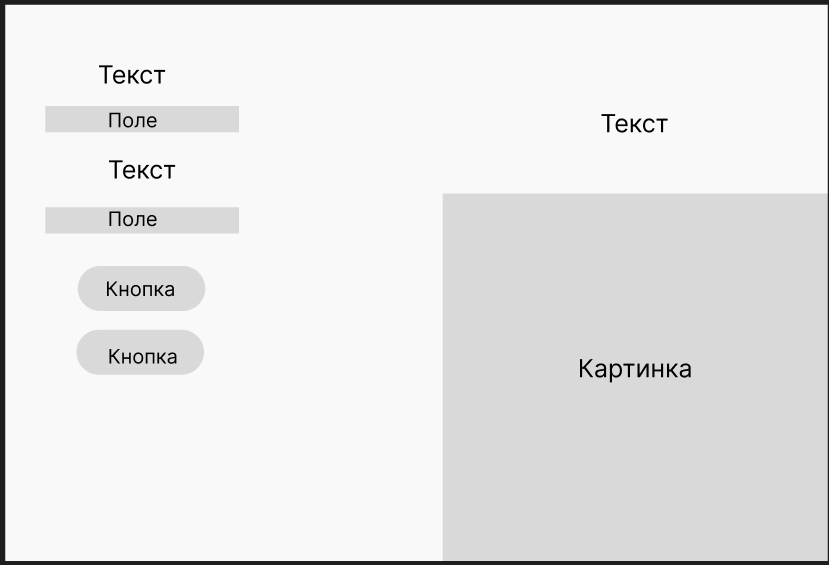


Рисунок 11 - Wireframe макет авторизации/входа



Рисунок 12 - Wireframe макет главного экрана администратора

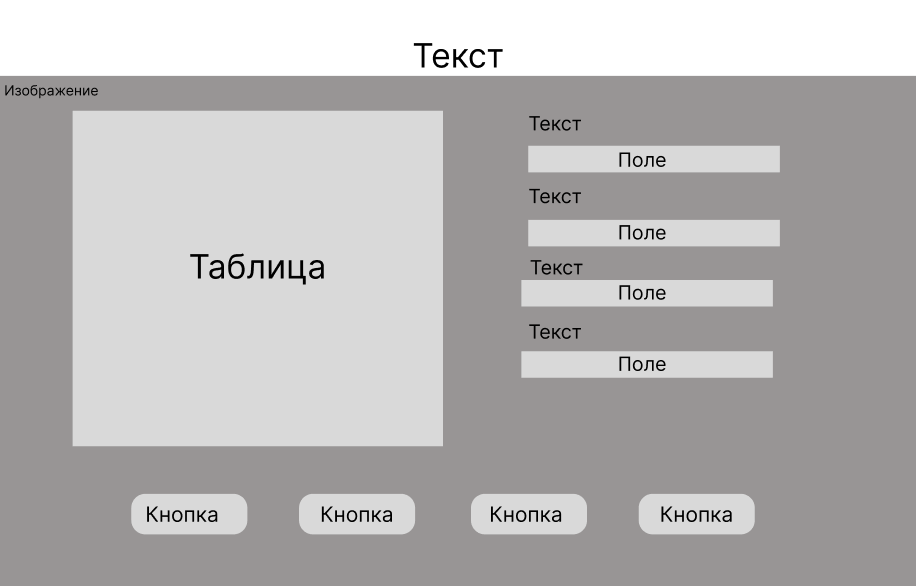


Рисунок 13 - Wireframe макет таблиц

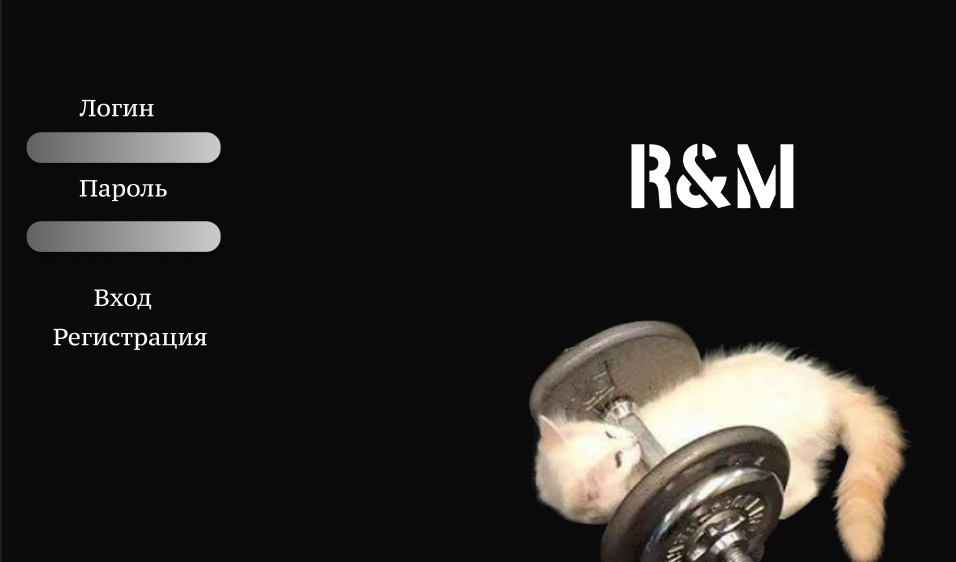


Рисунок 14 – Mockup макет авторизации/входа

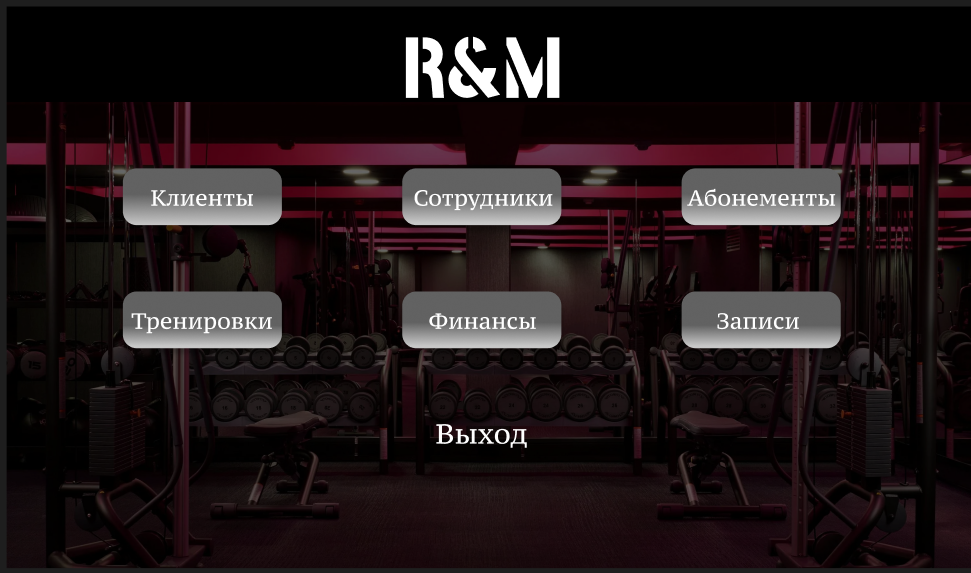


Рисунок 15 – Mockup макет главной страницы администратора

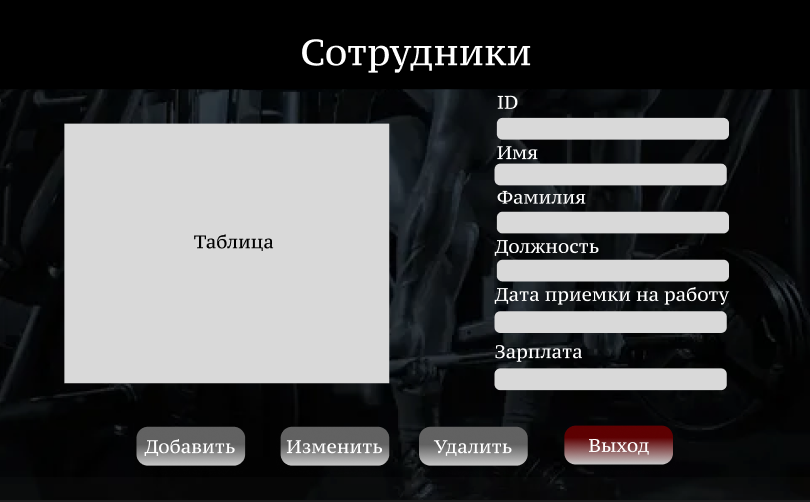


Рисунок 16– Mockup макет таблиц

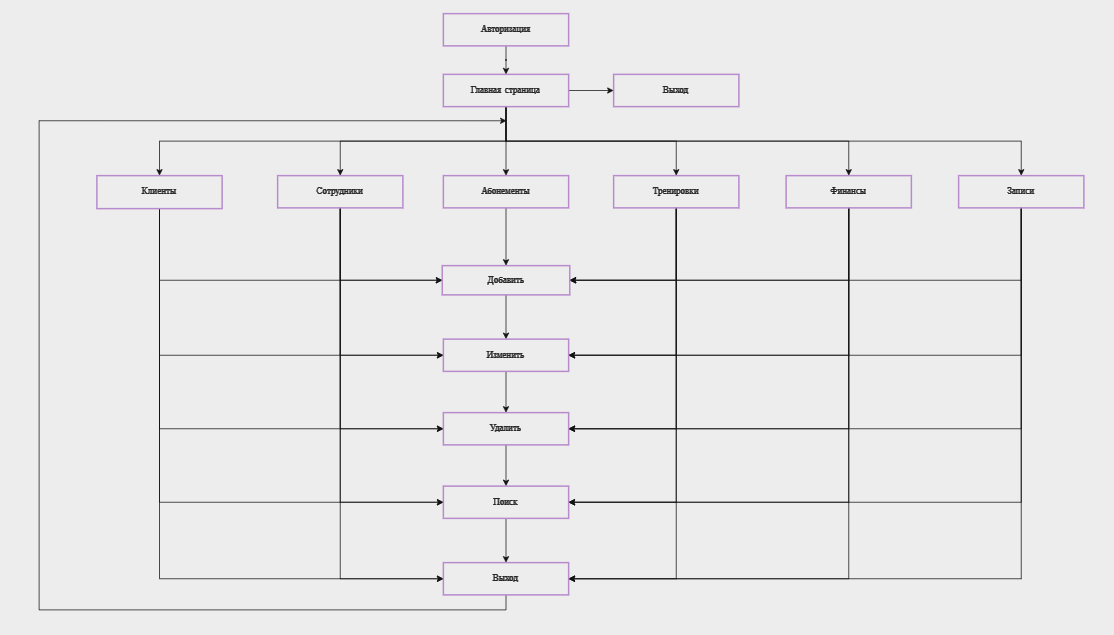


Рисунок 17 – Карта навигаций

Задание 4. Разработка программы

Заходим под Администратором

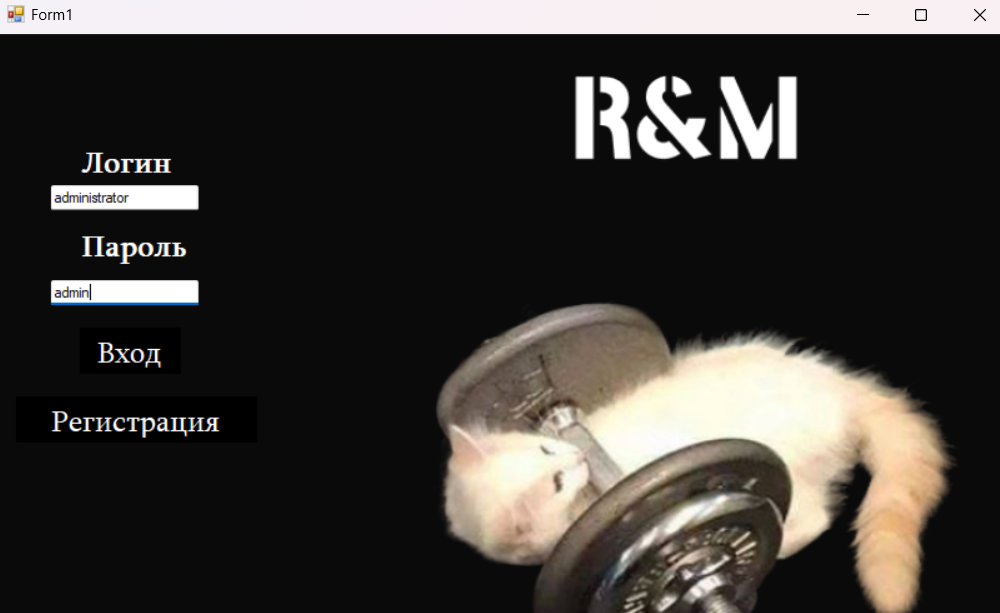


Рисунок 18 – Авторизация администратора

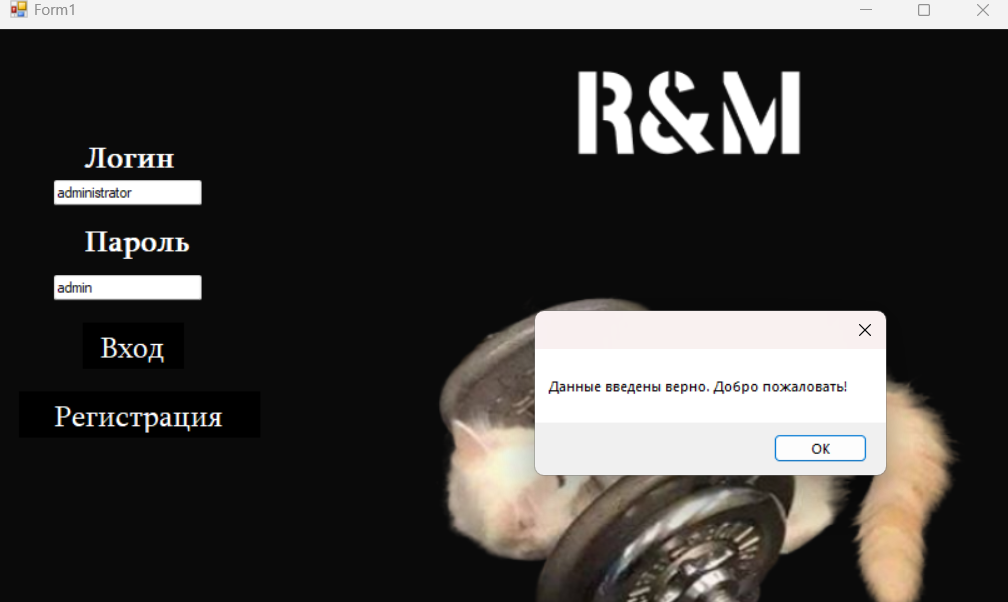


Рисунок 19 – Авторизация администратора

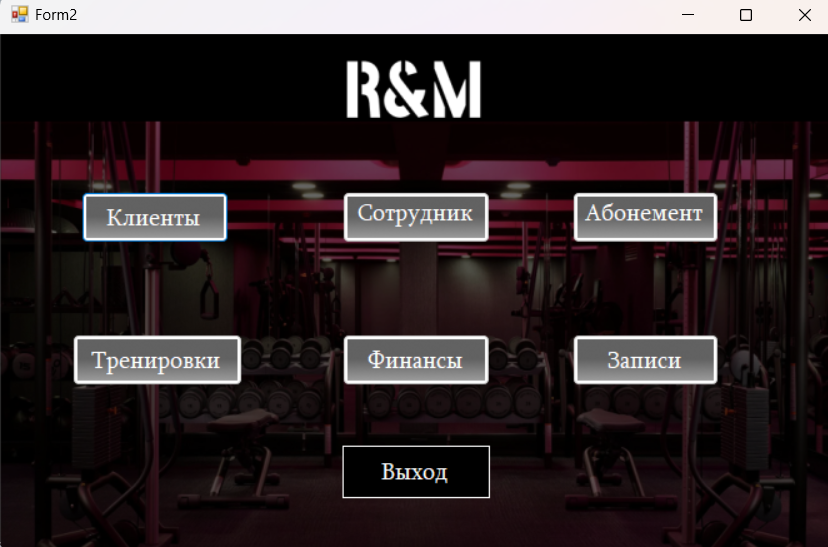


Рисунок 20 – Главная страница администратора

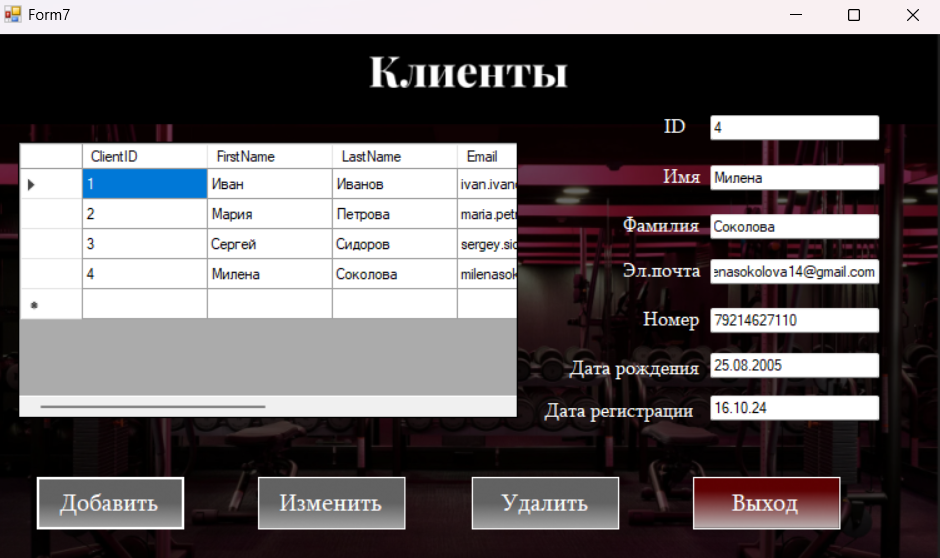


Рисунок 21 – Страница «Клиенты» добавление клиента

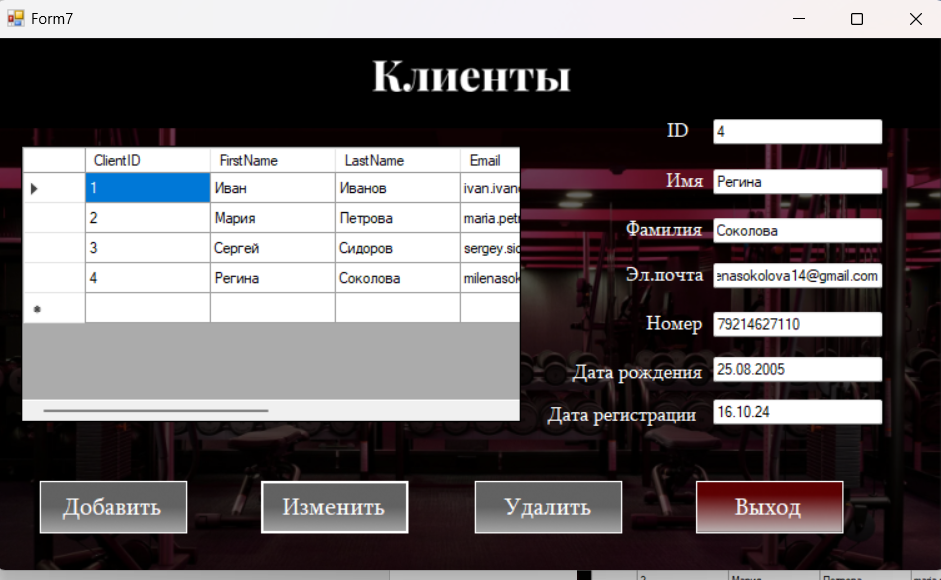


Рисунок 22 – Страница «Клиенты» изменение клиента

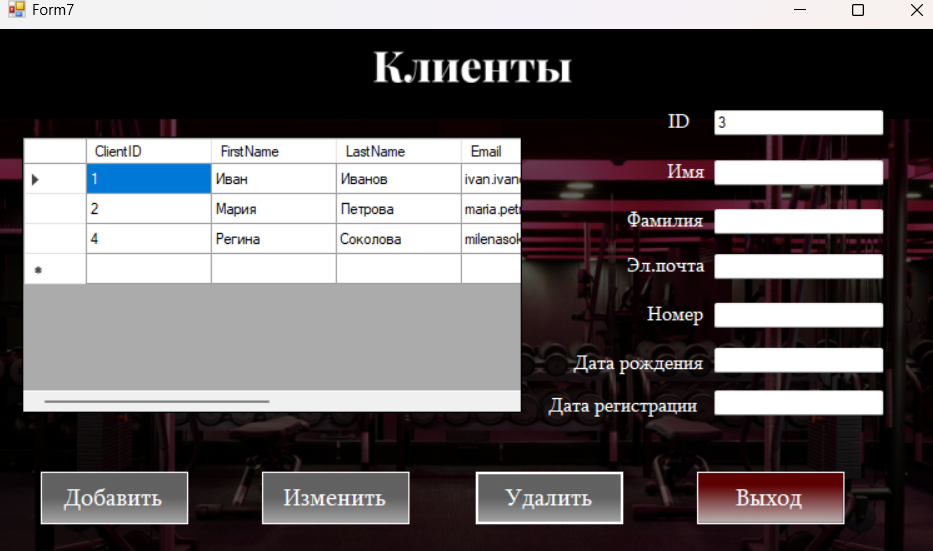


Рисунок 23 – Страница «Клиенты» удаление клиента

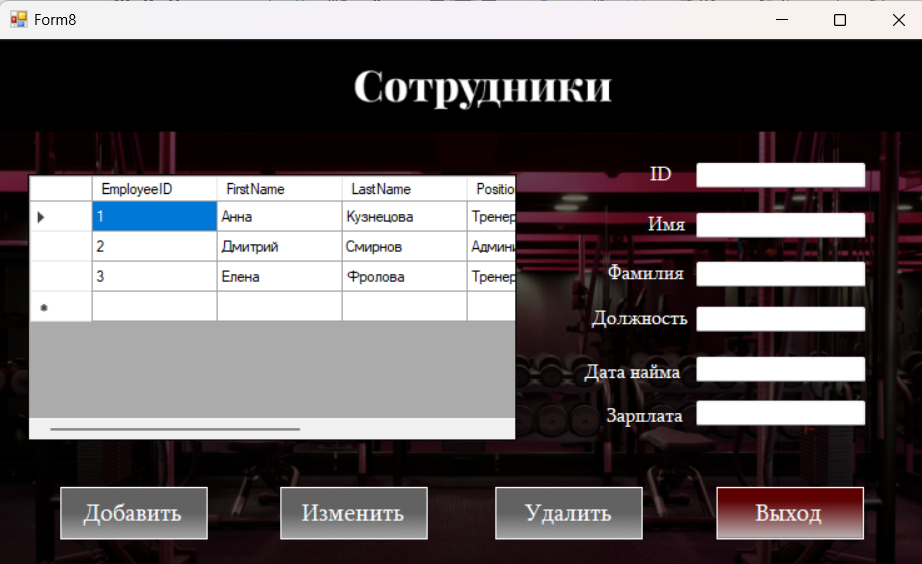


Рисунок 24 – Страница «Сотрудники»

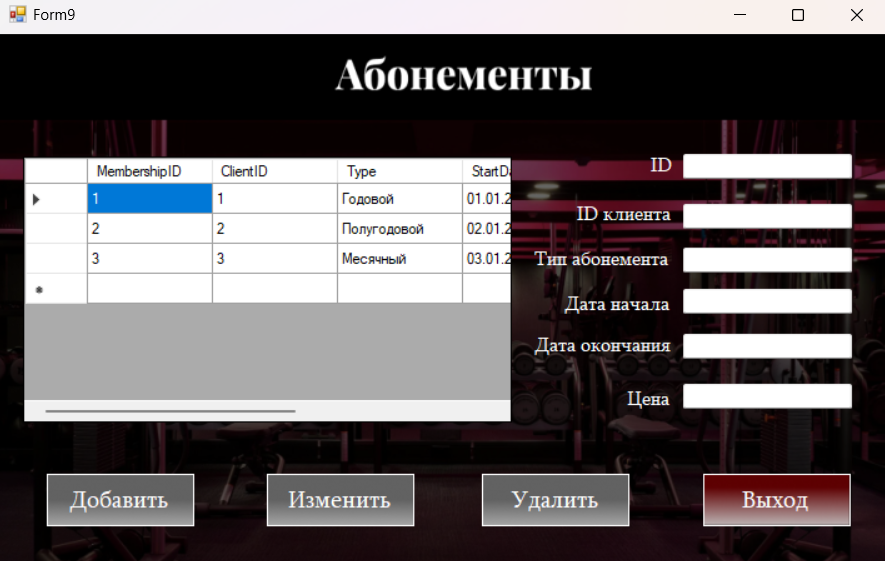


Рисунок 25 – Страница «Абонементы»

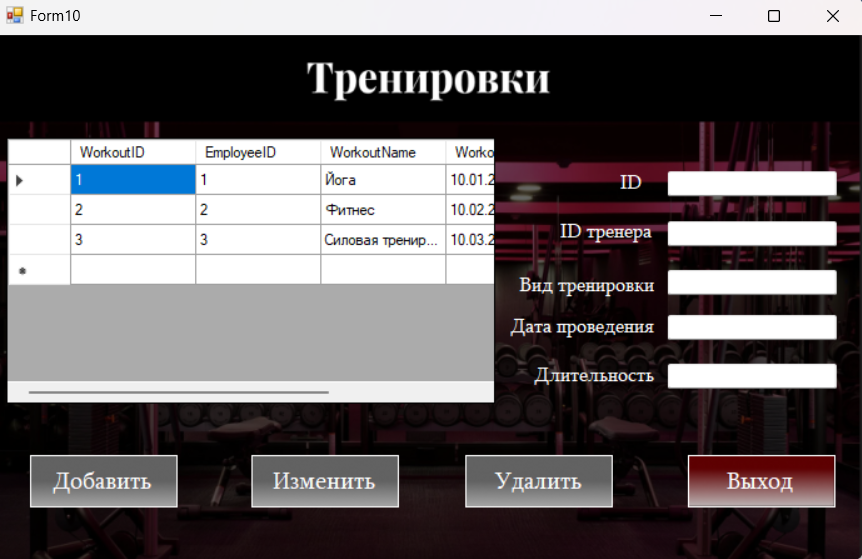


Рисунок 26 – Страница «Тренировки»

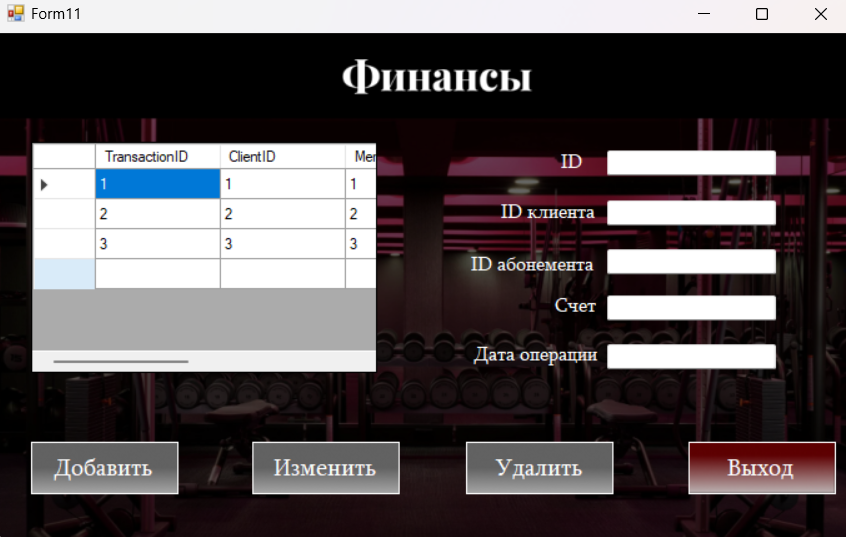


Рисунок 27 – Страница «Финансы»

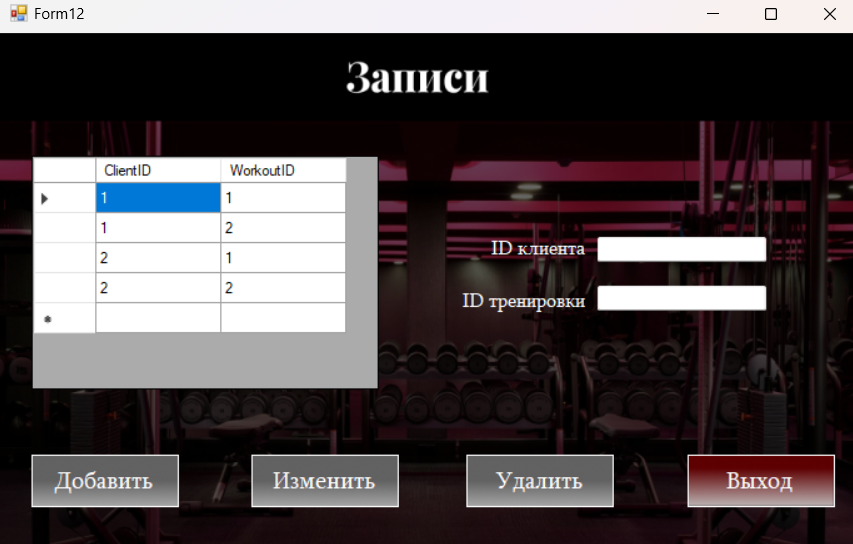


Рисунок 28 – Страница «Записи»

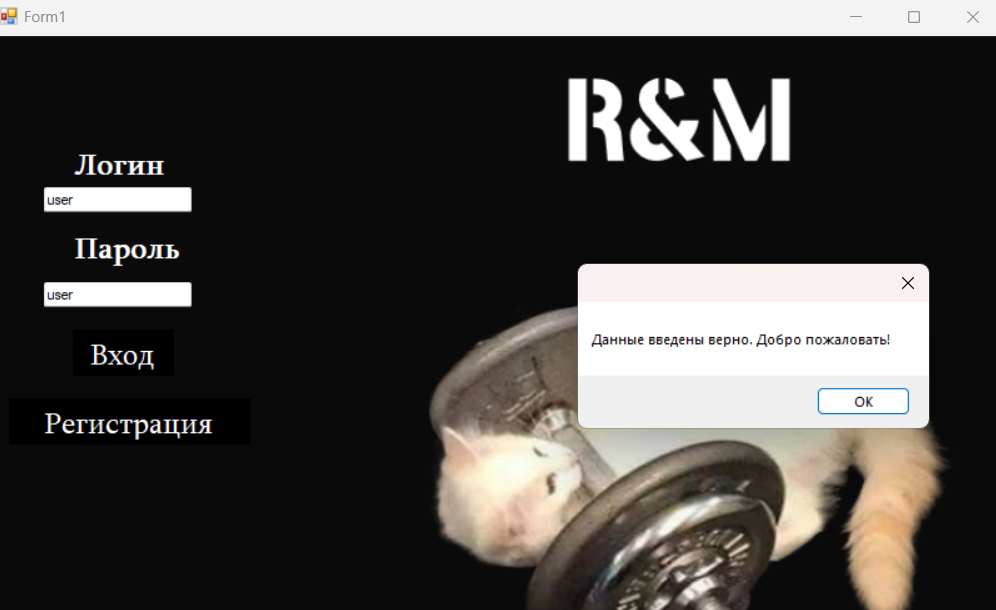


Рисунок 29 – Авторизация пользователя

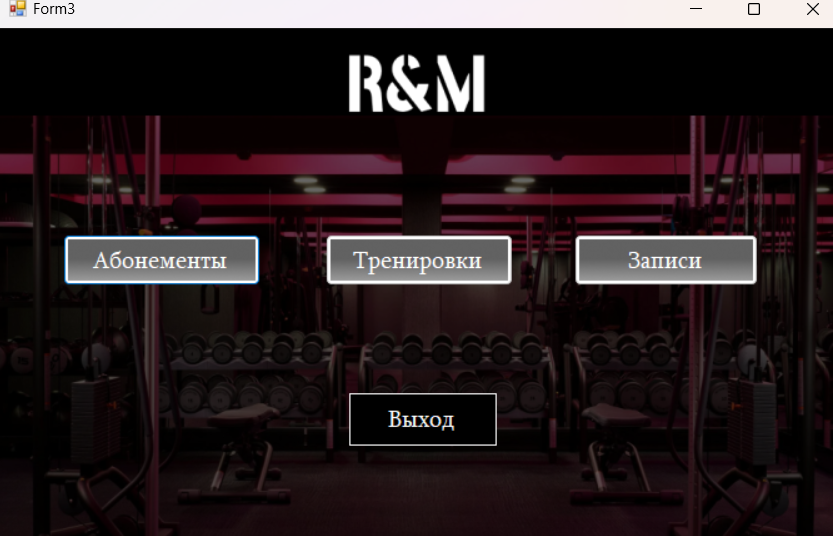


Рисунок 30 – Главная страница пользователи

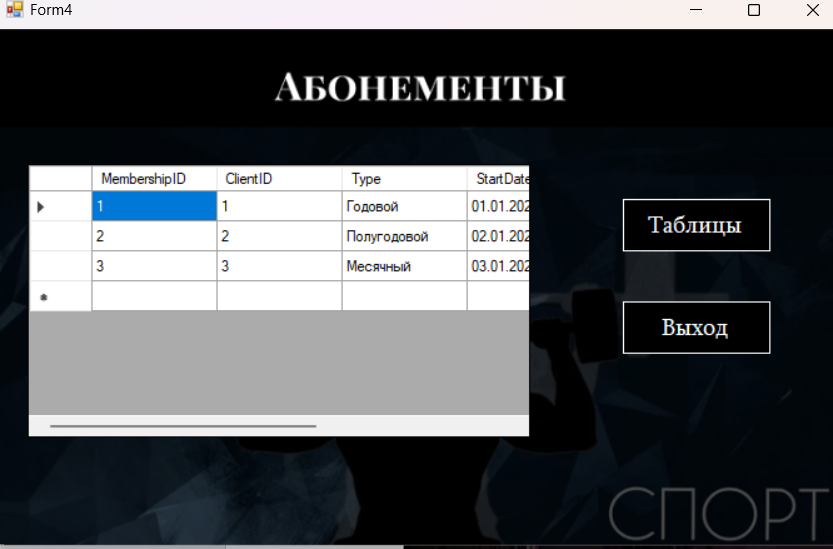


Рисунок 31 – Страница «Абонементы»

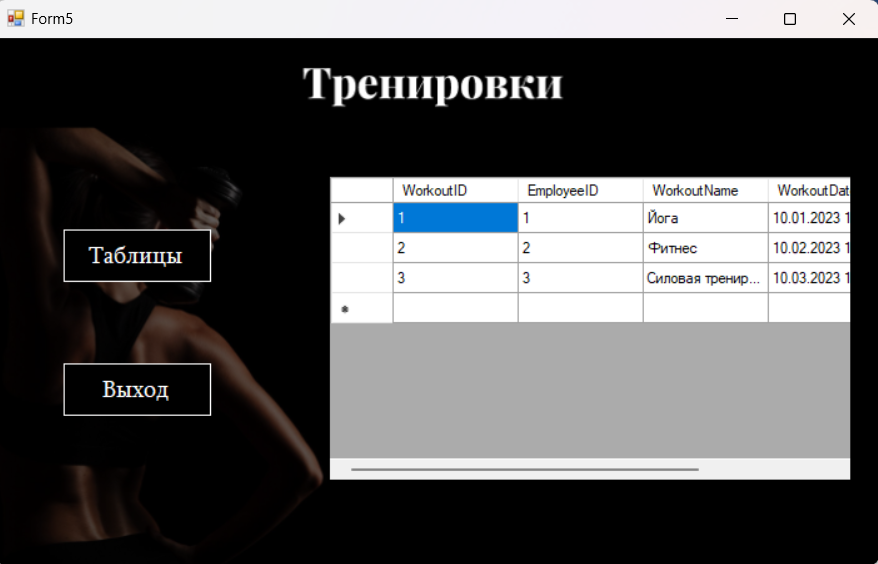


Рисунок 32 – Страница «Тренировка»

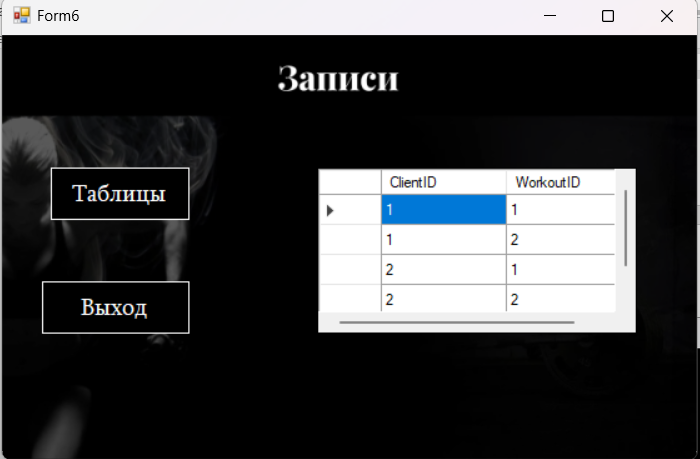


Рисунок 33 – Страница «Записи»

Задание 5. Тестирование

Аннотация теста:

|  |  |
| --- | --- |
| **Название проекта** | «R&M» |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Милена |
| **Дата(ы) теста** | 17.10.2024 |

Таблица 1 – Проверка входа администратора с корректными данными

|  |  |
| --- | --- |
| **Тестовый пример #** | 1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешная аутентификация администратора |
| **Краткое изложение теста** | Проверка успешной аутентификации администратора |
| **Этапы теста** | 1. Ввести логин и пароль.  2. Нажать кнопку "Вход". 3. Проверить, открывается ли соответствующая форма. |
| **Тестовые данные** | Логин: administrator  Пароль: admin |
| **Ожидаемый результат** | Аутентификация успешна. Открытие формы администратора (Form2). |
| **Фактический результат** | Аутентификация успешна. Открытие формы администратора (Form2). |
| **Статус** | Зачет |
| **Предварительное условие** | Приложение запущено. |
| **Постусловие** | Нет |
| **Примечания/комментарии** | - |

Таблица 2 – Проверка входа клиента с корректными данными

|  |  |
| --- | --- |
| **Тестовый пример #** | 2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешная аутентификация клиента |
| **Краткое изложение теста** | Проверка успешной аутентификации клиента |
| **Этапы теста** | 1. Ввести логин и пароль.  2. Нажать кнопку "Вход". 3. Проверить, открывается ли соответствующая форма. |
| **Тестовые данные** | Логин: user Пароль: user |
| **Ожидаемый результат** | Аутентификация успешна. Открытие формы администратора (Form3). |
| **Фактический результат** | Аутентификация успешна. Открытие формы администратора (Form3). |
| **Статус** | Зачет |
| **Предварительное условие** | Приложение запущено. |
| **Постусловие** | Нет |
| **Примечания/комментарии** | - |

Таблица 3 - Проверка входа сотрудника с корректными данными

|  |  |
| --- | --- |
| **Тестовый пример #** | 3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешная аутентификация сотрудника |
| **Краткое изложение теста** | Проверка успешной аутентификации сотрудника |
| **Этапы теста** | 1. Ввести логин и пароль.  2. Нажать кнопку "Вход". 3. Проверить, открывается ли соответствующая форма. |
| **Тестовые данные** | Логин: sotr   Пароль: sotr |
| **Ожидаемый результат** | Аутентификация успешна. Открытие формы сотрудника |
| **Фактический результат** | Аутентификация успешна. Открытие формы сотрудника |
| **Статус** | Зачет |
| **Предварительное условие** | Приложение запущено. |
| **Постусловие** | Нет |
| **Примечания/комментарии** | - |

Таблица 4 - Проверка поведение приложения при вводе некорректного пароля

|  |  |
| --- | --- |
| **Тестовый пример #** | 4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Неуспешная аутентификация из-за неправильного пароля |
| **Краткое изложение теста** | Проверка поведение приложения при вводе некорректного пароля |
| **Этапы теста** | 1. Ввести правильный логин и неправильный пароль. 2. Вызвать метод аутентификации. 3. Нажать кнопку "Вход".  4. Проверить, отображается ли сообщение об ошибке. |
| **Тестовые данные** | Логин: administrator Пароль: administrator |
| **Ожидаемый результат** | Аутентификация неуспешна. Появление окна с надписью: “Вы ввели неверные данные” |
| **Фактический результат** | Аутентификация неуспешна. Появление окна с надписью: “Вы ввели неверные данные” |
| **Статус** | Зачет |
| **Предварительное условие** | Пароль не должен соответствовать с логином в БД |
| **Постусловие** | Нет |
| **Примечания/комментарии** | - |

Таблица 5 - Проверка, что вход не выполняется при пустом логине

|  |  |
| --- | --- |
| **Тестовый пример #** | 5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Неуспешная аутентификация из-за пустого логина |
| **Краткое изложение теста** | Проверка поведение приложения при вводе пустого логина |
| **Этапы теста** | 1. Ввести пустой логин и пароль. 2. Вызвать метод аутентификации. 3. Нажать кнопку "Вход".  4. Проверить, отображается ли сообщение об ошибке. |
| **Тестовые данные** | Логин:  Пароль: admin |
| **Ожидаемый результат** | Вход не выполнен, приложение не может быть запущено. |
| **Фактический результат** | Вход не выполнен, приложение не может быть запущено. |
| **Статус** | Зачет |
| **Предварительное условие** | Логин не должен иметь пустое значение в логине в БД |
| **Постусловие** | Нет |
| **Примечания/комментарии** | - |

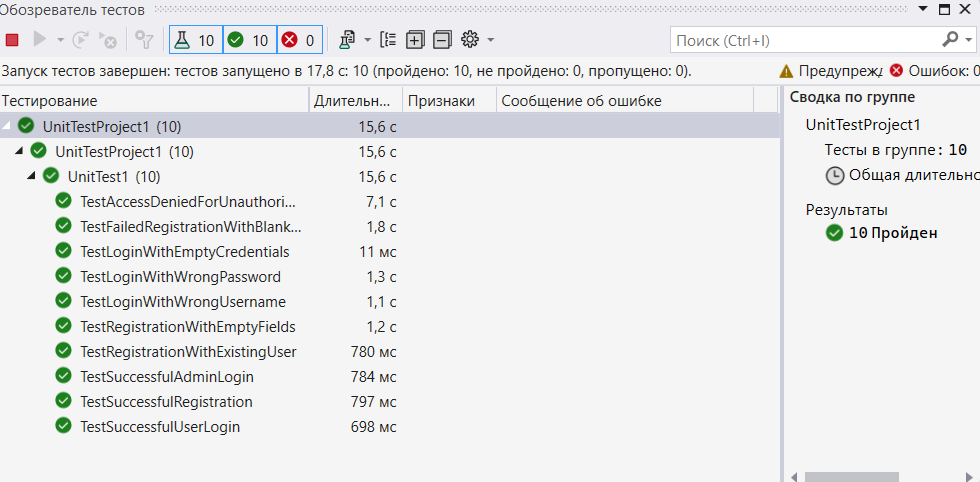


Рисунок 34 – Проверка юнит-тестов

# Задание 6. GitHub. Оценка проекта

Отправление приложения с тестами и отчетом на созданный репозиторий на GitHub (Рисунок 35-41).



Рисунок 35 – Создание нового репозитория

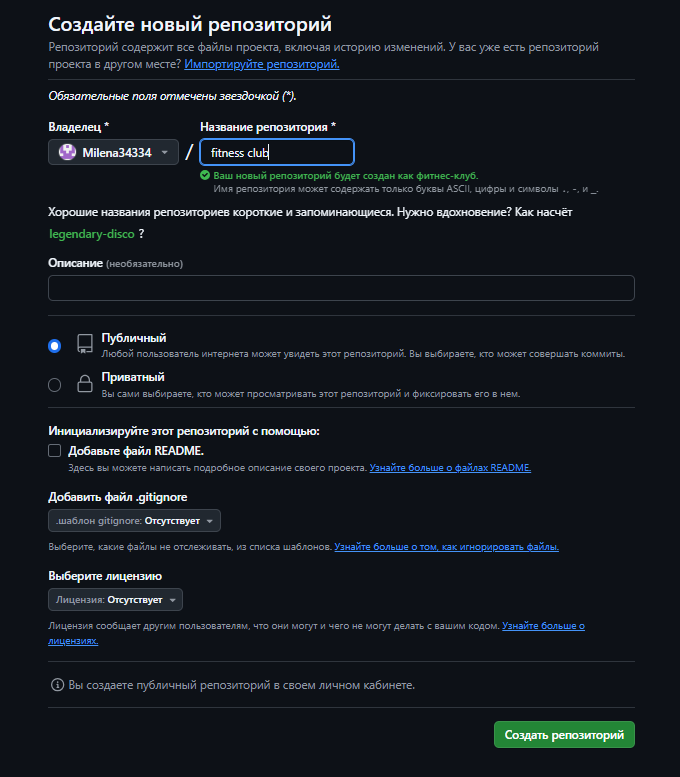


Рисунок 36 – Создание нового репозитория

Ссылка на репозиторий Фитнес-клуб: <https://github.com/Milena34334/fitness-club.git>

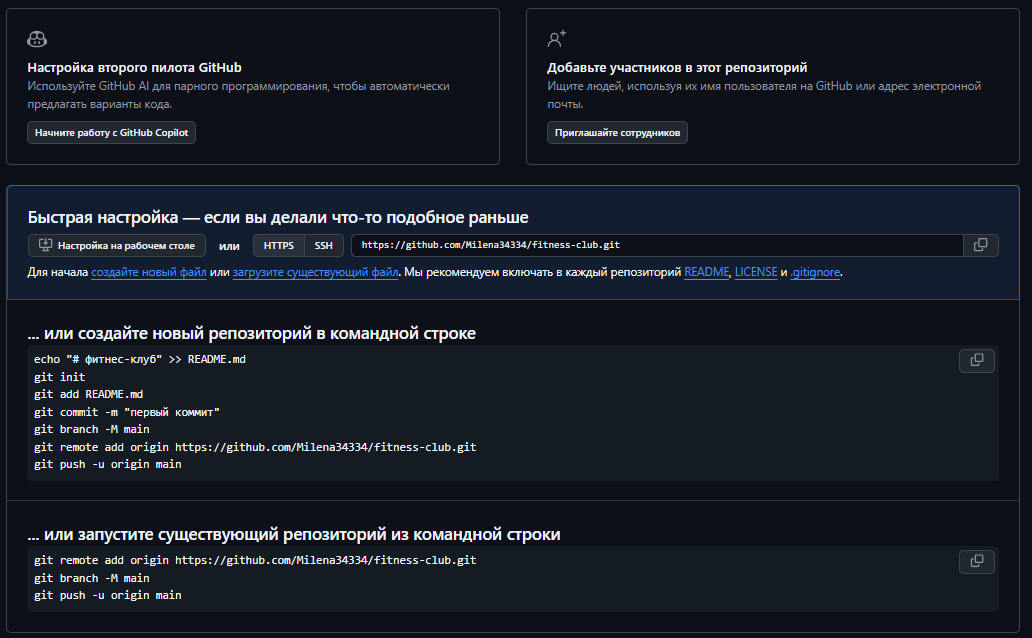


Рисунок 37 – Ссылка на репозиторий

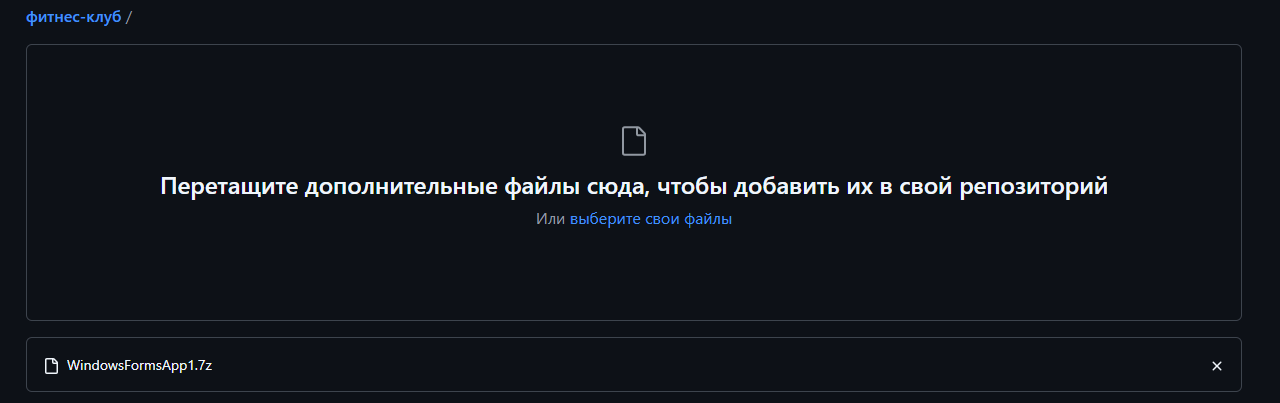


Рисунок 38 – Выгрузка приложения на GitHub

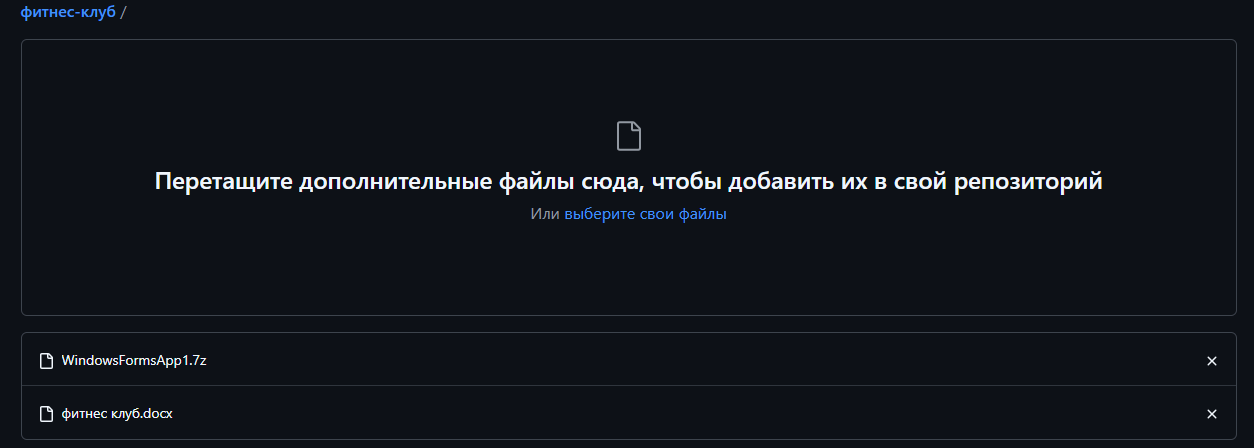


Рисунок 39 – Выгрузка отчета на GitHub

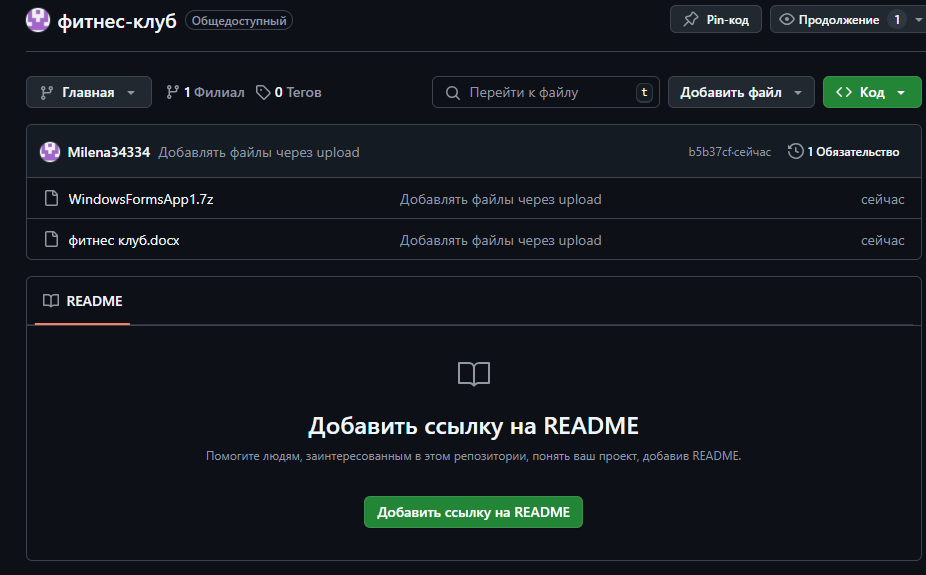


Рисунок 40 – Репозиторий с выгруженными файлами

# ЗАКЛЮЧЕНИЕ

В результате данной учебной практики была разработана база данных для библиотечной предметной области. Созданы формы для взаимодействия с этой базой данных, а также подготовлены техническое задание и диаграммы, описывающие функционал приложения.

В системе управления базами данных (СУБД) реализованы запросы для работы с данными, включая функции добавления, редактирования, удаления и поиска записей. Макеты программного продукта были разработаны с использованием программы Figma, а удобный пользовательский интерфейс был создан на языке C# в среде Visual Studio, что позволяет пользователям выполнять различные действия с базой данных.

# ПРИЛОЖЕНИЕ A

Код страницы авторизации:

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;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

string connectionString = @" Data Source= MILENAAS; Initial catalog=Фитнес клуб; Integrated Security=True";

public Form1()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.FixedSingle; // или FormBorderStyle.Fixed3D;

this.AutoScaleMode = AutoScaleMode.Font;

this.AutoSize = true;

}

public void Form1\_Load(object sender, EventArgs e)

{

}

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

{

string login = textBox1.Text.Trim();

string password = textBox2.Text.Trim(); // Предположим, пароль вводится в textBox3

if (!string.IsNullOrEmpty(login))

{

string query = "SELECT Name, Role FROM register WHERE Login = @Login AND Password = @Password";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

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

command.Parameters.AddWithValue("@Password", password); // Добавляем параметр для пароля

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

string userName = reader["Name"].ToString();

string userRole = reader["Role"].ToString();

if (userRole == "admin")

{

MessageBox.Show("Данные введены верно. Добро пожаловать!");

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

else if (userRole == "user")

{

MessageBox.Show("Данные введены верно. Добро пожаловать!");

Form3 form3 = new Form3();

form3.Show();

this.Hide();

}

else

{

MessageBox.Show("У вас нет доступа к системе.");

}

}

else

{

// Проверка на неверный пароль при верном логине

MessageBox.Show("Вы ввели неверные данные");

}

connection.Close();

}

}

}

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

{

string login = textBox1.Text.Trim();

string password = textBox2.Text.Trim();

string role = "user"; // Предполагая, что регистрируемый пользователь по умолчанию имеет роль "user"

if (!string.IsNullOrEmpty(login) && !string.IsNullOrEmpty(password))

{

string query = "INSERT INTO register (Login, Password, Role) VALUES (@Login, @Password, @Role)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

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

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

command.Parameters.AddWithValue("@Role", role);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Пользователь успешно зарегистрирован!");

}

else

{

MessageBox.Show("Ошибка при регистрации пользователя. Попробуйте еще раз.");

}

connection.Close();

}

}

else

{

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

}

}

}

}

Код главной страницы администратора:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

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

{

Form7 frm = new Form7();

this.Hide();

frm.Show();

}

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

{

Form8 frm = new Form8();

this.Hide();

frm.Show();

}

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

{

Form9 frm = new Form9();

this.Hide();

frm.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

Form10 frm = new Form10();

this.Hide();

frm.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

Form11 frm = new Form11();

this.Hide();

frm.Show();

}

private void button6\_Click(object sender, EventArgs e)

{

Form12 frm = new Form12();

this.Hide();

frm.Show();

}

private void button7\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Form2\_Load(object sender, EventArgs e)

{

}

}

}

Код страницы с таблицами администратора:

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;

namespace WindowsFormsApp1

{

public partial class Form7 : Form

{

string connectionString = @"Data Source= MILENAAS; Initial catalog=Фитнес клуб; Integrated Security=True";

public Form7()

{

InitializeComponent();

LoadData();

this.FormBorderStyle = FormBorderStyle.FixedSingle; // или FormBorderStyle.Fixed3D;

this.AutoScaleMode = AutoScaleMode.Font;

this.AutoSize = true;

}

private void LoadData()

{

string connectionString = @"Data Source= MILENAAS; Initial catalog=Фитнес клуб; Integrated Security=True";

string query = "SELECT \* FROM Clients";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable table = new DataTable();

adapter.Fill(table);

dataGridView1.DataSource = table;

}

}

private void button7\_Click(object sender, EventArgs e)

{

string query = "INSERT INTO Clients (ClientID, FirstName, LastName, Email,PhoneNumber,DateOfBirth,RegistrationDate)" +

"VALUES (@ClientID, @FirstName, @LastName,@Email, @PhoneNumber,@DateOfBirth,@RegistrationDate)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

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

{

command.Parameters.AddWithValue("@ClientID", Int32.Parse(textBox5.Text));

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

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

command.Parameters.AddWithValue("@Email", textBox3.Text);

command.Parameters.AddWithValue("@PhoneNumber", textBox4.Text);

command.Parameters.AddWithValue("@DateOfBirth", textBox6.Text);

command.Parameters.AddWithValue("@RegistrationDate", textBox7.Text);

connection.Open();

command.ExecuteNonQuery();

}

}

LoadData();

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

string query = "UPDATE Clients SET FirstName = @FirstName, LastName = @LastName, Email = @Email,PhoneNumber = @PhoneNumber,DateOfBirth = @DateOfBirth,RegistrationDate = @RegistrationDate WHERE ClientID = @ClientID";

using (SqlConnection connection = new SqlConnection(connectionString))

{

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

{

command.Parameters.AddWithValue("@ClientID", Int32.Parse(textBox5.Text));

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

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

command.Parameters.AddWithValue("@Email", textBox3.Text);

command.Parameters.AddWithValue("@PhoneNumber", textBox4.Text);

command.Parameters.AddWithValue("@DateOfBirth", textBox6.Text);

command.Parameters.AddWithValue("@RegistrationDate", textBox7.Text);

connection.Open();

command.ExecuteNonQuery();

}

}

LoadData();

}

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

{

string query = "DELETE FROM Clients WHERE ClientID = @ClientID";

using (SqlConnection connection = new SqlConnection(connectionString))

{

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

{

command.Parameters.AddWithValue("@ClientID", Int32.Parse(textBox5.Text));

connection.Open();

command.ExecuteNonQuery();

}

}

LoadData();

}

private void button4\_Click(object sender, EventArgs e)

{

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

private void Form7\_Load(object sender, EventArgs e)

{

}

}

Код главной страницы пользователя:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

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

{

Form4 frm = new Form4();

this.Hide();

frm.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

Form5 frm = new Form5();

this.Hide();

frm.Show();

}

private void button6\_Click(object sender, EventArgs e)

{

Form6 frm = new Form6();

this.Hide();

frm.Show();

}

private void button7\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Form3\_Load(object sender, EventArgs e)

{

}

}

}

Код страницы с таблицами пользователя:

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;

namespace WindowsFormsApp1

{

public partial class Form4 : Form

{

public Form4()

{

InitializeComponent();

LoadData();

this.FormBorderStyle = FormBorderStyle.FixedSingle; // или FormBorderStyle.Fixed3D;

this.AutoScaleMode = AutoScaleMode.Font;

this.AutoSize = true;

}

private void LoadData()

{

string connectionString = @"Data Source= MILENAAS; Initial catalog=Фитнес клуб; Integrated Security=True";

string query = "SELECT \* FROM Memberships";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable table = new DataTable();

adapter.Fill(table);

dataGridView1.DataSource = table;

}

}

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

{

Form3 frm = new Form3();

this.Hide();

frm.Show();

}

private void button7\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Form4\_Load(object sender, EventArgs e)

{

}

}

}

# ПРИЛОЖЕНИЕ B

Код юнит-тестов:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

using WindowsFormsApp1;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

private string connectionString = @"Data Source= MILENAAS; Initial catalog=Фитнес клуб; Integrated Security=True";

private Form1 form;

[TestInitialize]

public void Initialize()

{

form = new Form1();

}

[TestMethod]

public void TestSuccessfulAdminLogin()

{

// Arrange

form.textBox1.Text = "adminUser"; // Замените на тестового пользователя с ролью "admin"

form.textBox2.Text = "adminPassword"; // Замените на правильный пароль

// Act

form.button1\_Click(null, null);

// Assert

// Здесь можно было бы использовать механизмы проверки,

// но так как мы не можем проверить MessageBox напрямую,

// нужно либо перехватить его, либо использовать логирование

}

[TestMethod]

public void TestSuccessfulUserLogin()

{

// Arrange

form.textBox1.Text = "normalUser"; // Замените на тестового пользователя с ролью "user"

form.textBox2.Text = "userPassword"; // Замените на правильный пароль

// Act

form.button1\_Click(null, null);

// Assert

// Ожидайте появления Form3 или сообщения "Добро пожаловать!"

}

[TestMethod]

public void TestLoginWithWrongUsername()

{

// Arrange

form.textBox1.Text = "wrongUser";

form.textBox2.Text = "userPassword";

// Act

form.button1\_Click(null, null);

// Assert

// Проверка на сообщение "Вы ввели неверные данные"

}

[TestMethod]

public void TestLoginWithWrongPassword()

{

// Arrange

form.textBox1.Text = "adminUser"; // Замените на тестового пользователя

form.textBox2.Text = "wrongPassword";

// Act

form.button1\_Click(null, null);

// Assert

// Проверка на сообщение "Вы ввели неверные данные"

}

[TestMethod]

public void TestSuccessfulRegistration()

{

// Arrange

form.textBox1.Text = "newUser"; // Убедитесь, что этот пользователь не существует

form.textBox2.Text = "newPassword";

// Act

form.button2\_Click(null, null);

// Assert

// Проверка на сообщение "Пользователь успешно зарегистрирован!"

}

[TestMethod]

public void TestRegistrationWithExistingUser()

{

// Arrange

form.textBox1.Text = "existingUser"; // Замените на существующего пользователя

form.textBox2.Text = "somePassword";

// Act

form.button2\_Click(null, null);

// Assert

// Проверка на сообщение "Ошибка при регистрации пользователя."

}

[TestMethod]

public void TestLoginWithEmptyCredentials()

{

// Act

form.button1\_Click(null, null);

// Assert

// Ожидать появления сообщения о заполнении полей

}

[TestMethod]

public void TestRegistrationWithEmptyFields()

{

// Act

form.button2\_Click(null, null);

// Assert

// Ожидать появления сообщения о заполнении полей для регистрации

}

[TestMethod]

public void TestAccessDeniedForUnauthorizedRole()

{

// Arrange

form.textBox1.Text = "nonAdminUser"; // Пользователь с ролью, не имеющей доступа

form.textBox2.Text = "userPassword";

// Act

form.button1\_Click(null, null);

// Assert

// Проверка на сообщение "У вас нет доступа к системе."

}

[TestMethod]

public void TestFailedRegistrationWithBlankFields()

{

// Act

form.textBox1.Clear();

form.textBox2.Clear();

form.button2\_Click(null, null);

// Assert

// Ожидать сообщение о том, что поля не заполнены

}

}

}

**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 42919/8 группы

Соколова Милена Александровна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «07» октября 2024 г. по «19» октября 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 «19» октября 2024 г.