**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)

Студентка 4 курса 42919/2 группы

Сметанникова Анастасия Алексеевна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» 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)

Студентка 4 курса 42919/2 группы

Сметанникова Анастасия Алексеевна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» 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 16**

Задание выдал «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_ Сметанникова А.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)

Студентка 4 курса 42919/2 группы

Сметанникова Анастасия Алексеевна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» 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** |
| 23.09.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 24.09.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 25.09.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 26.09.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 27.09.2024 | Проектирование интерфейса пользователя. |  |
| 28.09.2024 | Разработка схемы алгоритма программного продукта. |  |
| 30.09.2024 | Разработка модели данных. |  |
| 01.10.2024 | Разработка программного продукта. Создание функциональности ПО. |  |
| 02.10.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 03.10.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 04.10.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. |  |
| 05.10.2024 | Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. Подготовка отчета. |  |

**Cодержание**

[Введение 6](#_Toc178930075)

[Анализ предметной области. Разработка технического задания. 7](#_Toc178930076)

[UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. 12](#_Toc178930077)

[Проектирование инфологической и даталогической модели данных. Создание БД. 15](#_Toc178930078)

[Разработка WireFrame&Mockup. Разработка карты навигации 28](#_Toc178930079)

[Разработка интерфейса приложения 32](#_Toc178930080)

[Тестирование приложения 39](#_Toc178930081)

[Заключение 44](#_Toc178930082)

[Список использованных источников 45](#_Toc178930083)

[Приложение А 46](#_Toc178930084)

# **Введение**

В рамках учебной практики предстоит продемонстрировать теоретические знания в области баз данных и практические навыки работы с ними. Этот процесс включает в себя изучение различных аспектов создания и оптимизации баз данных, разработки SQL-запросов для эффективного извлечения данных из больших объемов информации. В процессе работы необходимо изучить предметную область проекта, адаптировать базу данных под конкретные потребности. Проект выполняется по предметной области «Фитнес-центр».

Одним из этапов проекта является создание диаграмм баз данных, которые помогают визуализировать структуру данных и взаимосвязи между таблицами, что является важным шагом для понимания архитектуры базы данных и оптимизации ее структуры.

Далее предстоит разработать веб-приложение на языке программирования C# для взаимодействия с базой данных, которое будет не только функциональным, но и иметь удобный и интуитивно понятный для пользователей интерфейс.

В заключение работы предстоит реализовать модульные тесты итогового ПО и выгрузить проект на Git.

**Задание 1. Разработка технического задания**

Предметная область для исполнителя (19 вариант):

Основной целью данной информационной системы является автоматизация процесса управления поставками товаров (цветов). Она позволит компании эффективно управлять заказами, отслеживать статус поставок и оптимизировать логистику.

Разработка программы управления поставками товаров направлена на создание удобного и эффективного инструмента, который помогает минимизировать задержки и обеспечивает точность учета. Задача разработки такой программы состоит в создании системы, которая позволяет компаниям эффективно отслеживать и управлять поставками товаров, что способствует бесперебойной работе и оптимизации использования ресурсов. Система позволяет компании сократить время обработки заказов, оптимизировать использование ресурсов и повысить удовлетворенность клиентов.

Основные функции и возможности модуля включают:

1) Создание заказа: Это информация, предоставленная клиентом о товарах, которые требуется поставить. Клиент оставляет новый заказ. Заказ может содержать данные о наименовании товара, количестве, сроках поставки, личной информации (ФИО клиента и контактные данные). Также должны содержаться сведения об идентификаторе заказа

2) Обработка заказа: Процесс, включающий анализ заказа и назначение исполнителя (менеджера по поставкам) для выполнения. В процессе обработки может потребоваться дополнительная информация или уточнение деталей заказа у клиента.

3) Исполнение заказа: Фактическое выполнение поставки товаров. На этом этапе назначенный менеджер по поставкам организует доставку товаров, вносит необходимые изменения или координирует работу с другими специалистами.

4) Отчётность: Фиксация и отчёт о выполненной работе. После завершения поставки менеджер по поставкам должен предоставить отчёт о проделанной работе, включая информацию о затраченных ресурсах (время, транспортные расходы) и оказанной помощи.

5) Мониторинг и анализ: Этот этап предполагает контроль и анализ процесса управления поставками. Важно отслеживать и анализировать время обработки заказов, качество выполненных работ и расходы, которые могут помочь в оптимизации процесса.

Описание предметной области (16 вариант)

Фитнес клуб "SPORTX" представляет собой современное заведение, ориентированное на предоставление высококачественных услуг в области фитнеса и здорового образа жизни. С фокусом на удовлетворение потребностей различных клиентских групп, фитнес-центр предлагает разнообразные зоны и занятия, включая тренажерный зал с современным оборудованием, аэробный зал для кардио-тренировок, бассейн для водных занятий, специализированные залы для единоборств, а также детские секции для развития маленьких спортсменов.

Основной задачей, стоящей перед фитнес-центром, является эффективное управление всеми аспектами его деятельности после открытия. Для успешной работы необходимо вести учет персонала, клиентов, абонементов, расписания занятий, финансовых показателей и других важных данных.

Решением данной задачи является создание программного обеспечения, способного автоматизировать и оптимизировать процессы управления и учета в фитнес-центре. Программа будет обеспечивать возможность эффективного ведения базы данных о клиентах и персонале, формирования расписания занятий, учета абонементов и финансовых операций. Кроме того, она предоставит аналитические инструменты для анализа статистики посещаемости и активности клиентов, что позволит администрации принимать обоснованные решения по управлению центром.

Область применения разработанной программы включает в себя не только внутренние операции фитнес-центра, но и взаимодействие с клиентами. Программа будет использоваться для управления процессами регистрации новых клиентов, контроля посещаемости и оказания услуг, а также для анализа клиентской активности и предоставления персонализированных рекомендаций. Преимущества использования данной программы включают в себя повышение эффективности управления, улучшение качества обслуживания клиентов, сокращение времени на административные процессы и повышение финансовой прозрачности фитнес-центра.

Техническое задание

1. Общие сведения

1.1. Наименование проекта: Разработка эффективной системы управления и учета данных различных аспектов деятельности фитнес-центра «SPORTX»

1.2. Заказчик: Торговый дом «Цветочек».

1.3. Исполнитель: Фитнес-центр «SPORTX».

2. Функциональные требования

2.1. Система должна иметь возможность авторизации пользователей на стартовой странице по следующим данным:

- Имя пользователя

- Пароль

2.2. Система должна иметь возможность закрепить за одним покупателем несколько покупок;

2.3. Система должна автоматически назначать скидку клиентам;

2.4. Система должна иметь возможность сортировки данных в таблицах;

2.5. Система должна иметь возможность фильтровать данные в таблицах;

2.6. Система должна предоставлять администраторам следующие возможности:

- Регистрация новых пользователей;

- Редактирование данных пользователей;

- Удаление пользователей;

- Просмотр и редактирование отображаемых данных;

- Просмотр статистики посещаемости залов

2.7. Система должна предоставлять тренерам следующие возможности:

- Просмотр своего расписания и расписания других тренеров;

- Просмотр и редактирование таблиц «Тренировка», «Упражнения»;

2.8 Система должна предоставлять бухгалтерам следующие возможности:

- Просмотр таблиц «Сотрудники», «Покупатели»;

- Просмотр итоговых сумм в таблицах «Сотрудники», «Покупатели»;

- Генерация статистики прибыли по месяцам;

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Имя пользователя 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;

3.4. Производительность:

- Приложение должно иметь быстрый доступ к данным;

- Минимальное время отклика на запросы пользователя.

4. Требования к реализации

4.1. Язык программирования: С#

4.2. СУБД: MS SQL Server

5. Требования к документации

5.1. Техническое задание на разработку программного модуля.

6. Руководство по стилю

6.1. Шрифт: Comic Sans 12

6.2. Цветовая схема: Windows Standard

**Задание 2. Разработка алгоритмов и диаграмм**

Диаграмма вариантов использования (прецедентов/сценариев поведения) (Рисунок 1)

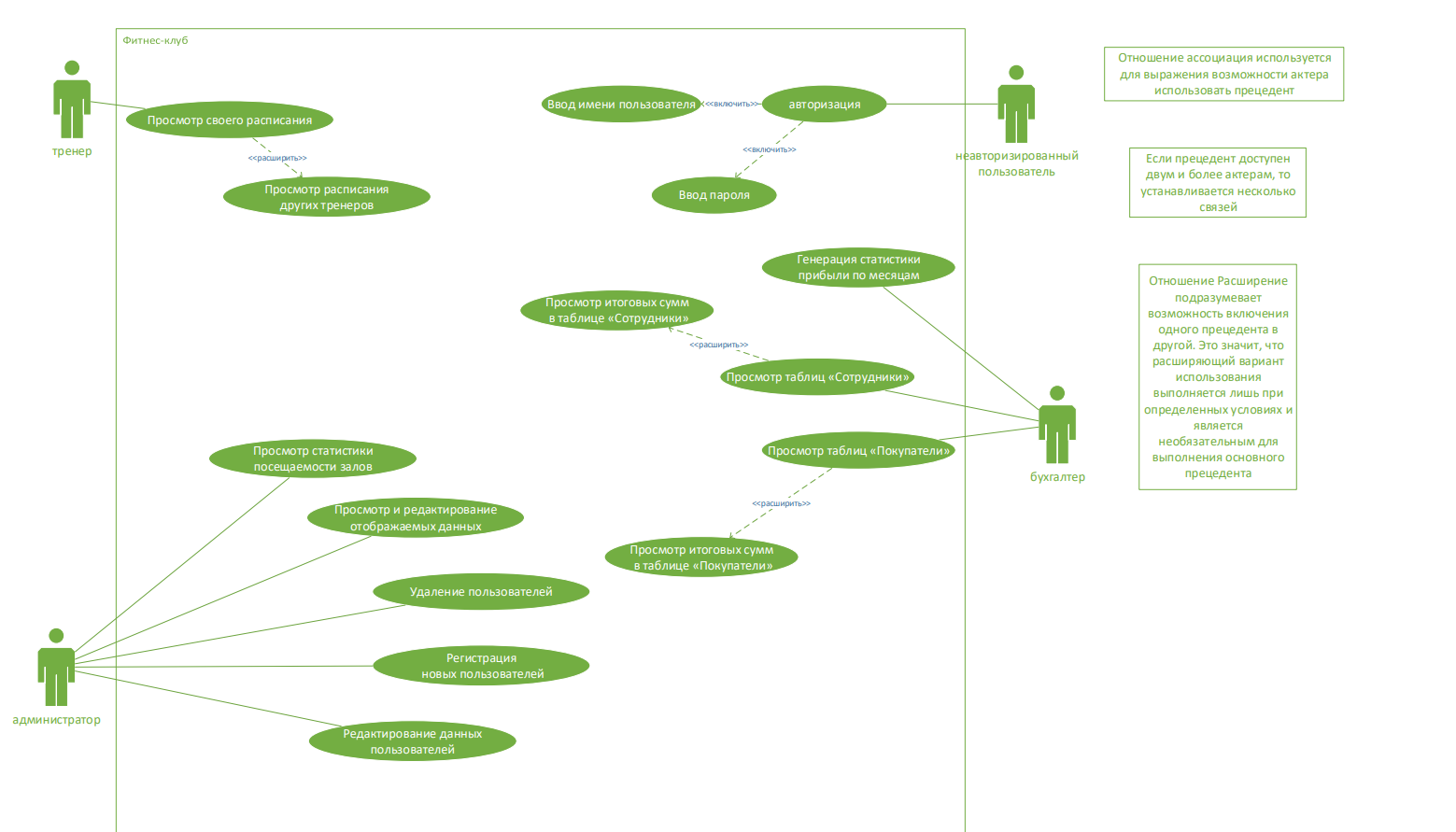


Рисунок 1

Диаграмма последовательностей/сценарий событий. Диаграмма последовательности для прецедента – редактирование данных сотрудника (Рисунок 2)

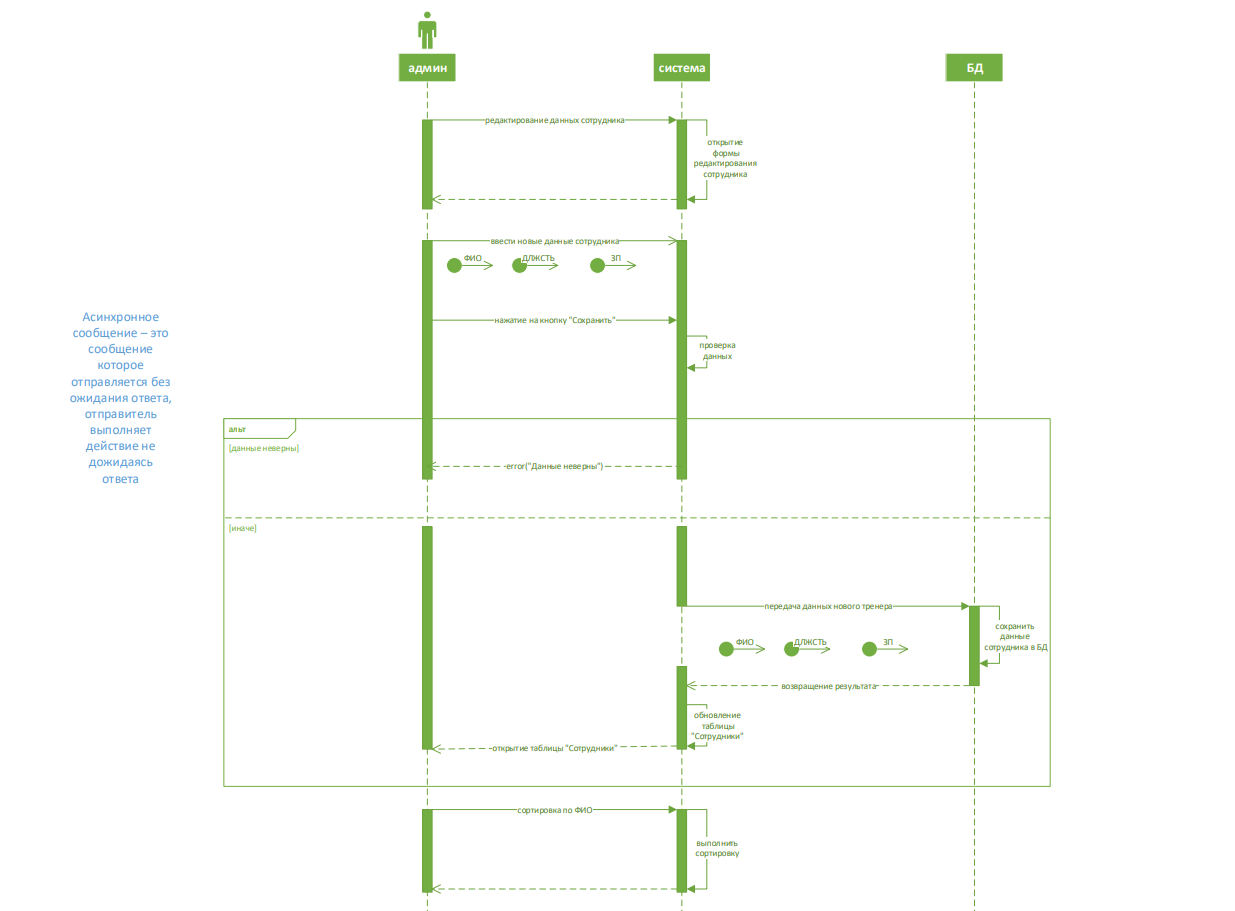


Рисунок 2

Диаграмма последовательностей/сценарий событий. Диаграмма последовательности для прецедента – редактирование данных сотрудника (Рисунок 3)

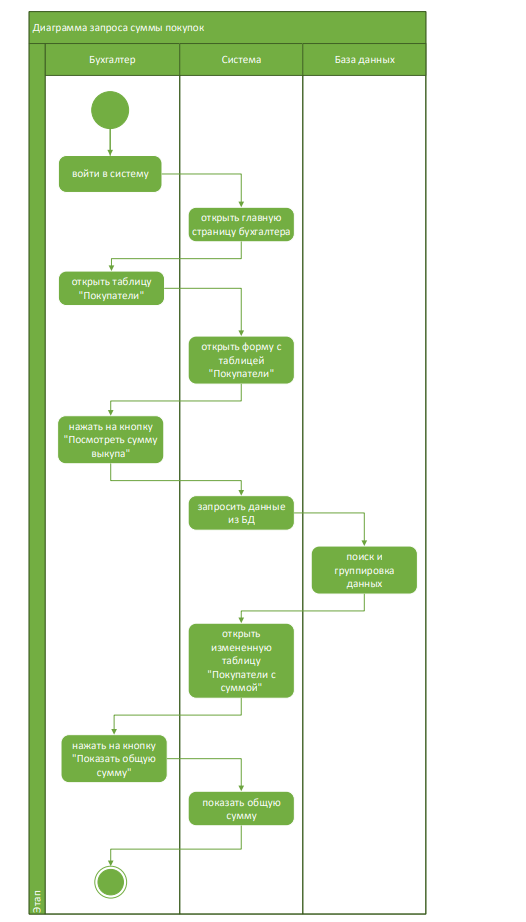


Рисунок 3

ER диаграмма в visio, графическое изображение сущностей предметной области со связи (Рисунок 4)

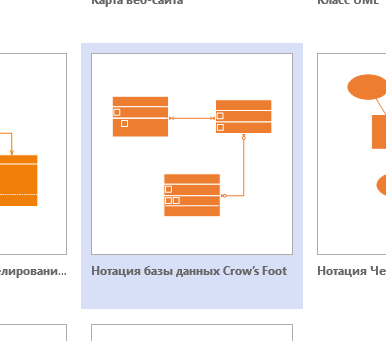


Рисунок 4

Даталогическая и ER модель (инфологическая) (Рисунок 5-6)

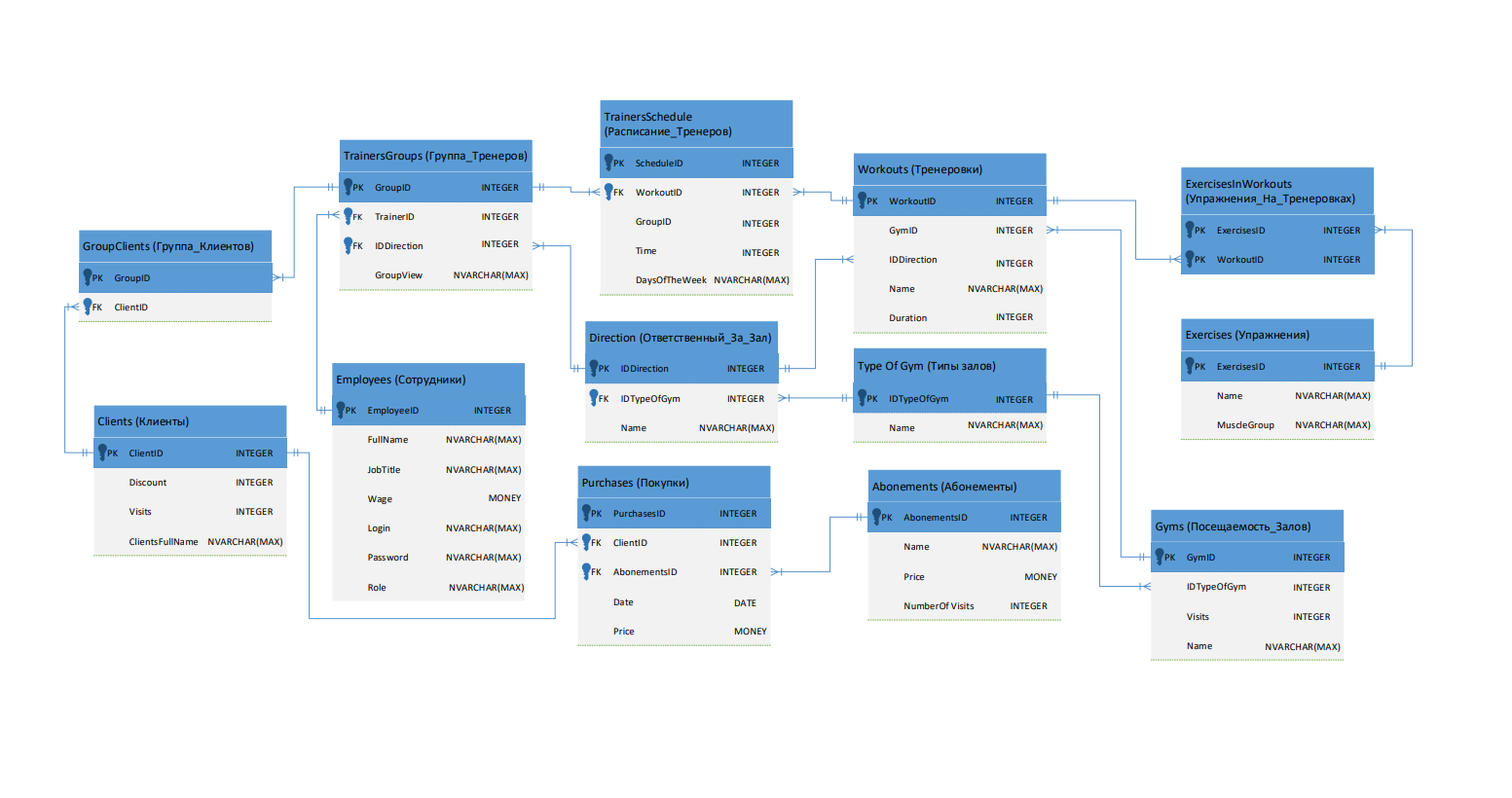


Рисунок 5

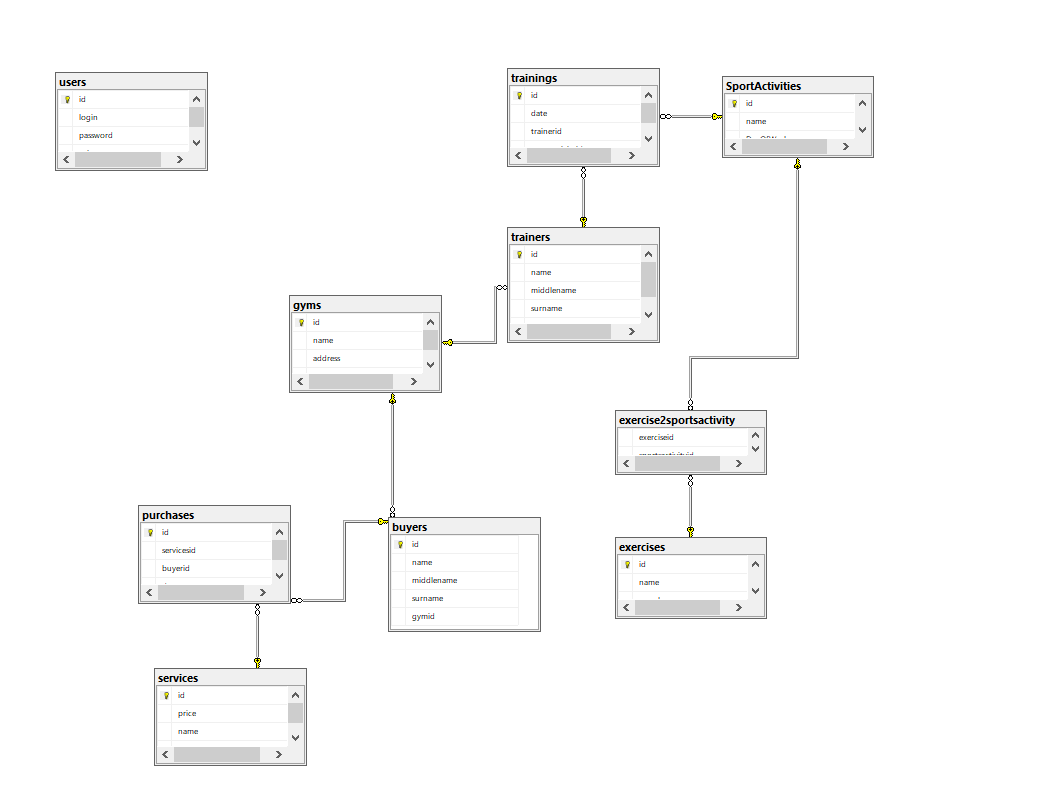


Рисунок 6

Создание и заполнение таблиц БД (Рисунок 7-18)

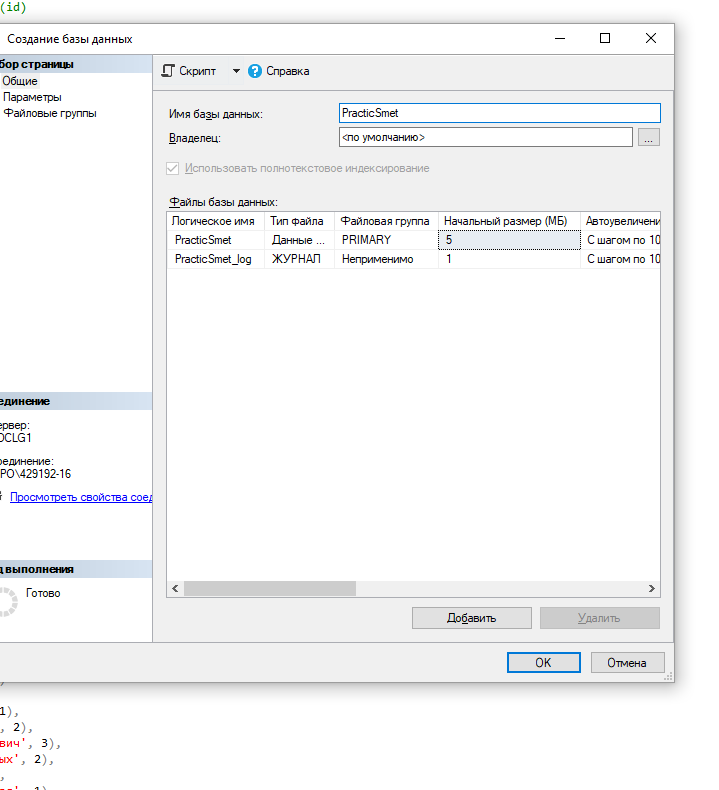


Рисунок 7

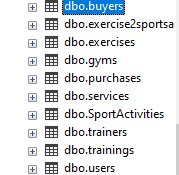


Рисунок 8

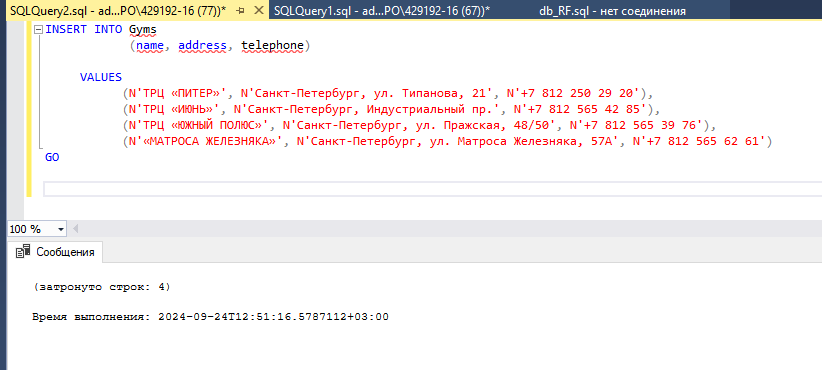


Рисунок 9

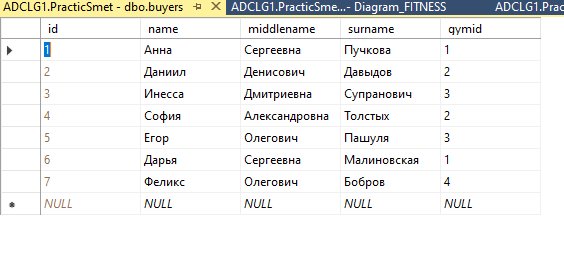


Рисунок 10

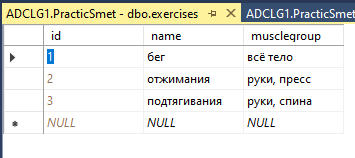


Рисунок 11

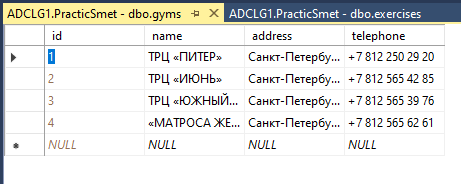


Рисунок 12

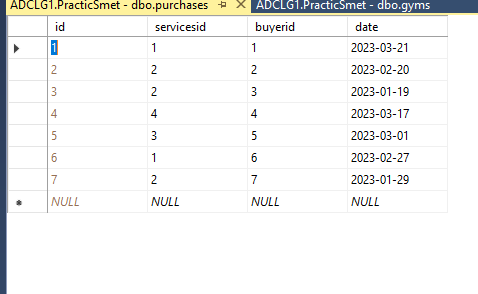


Рисунок 13

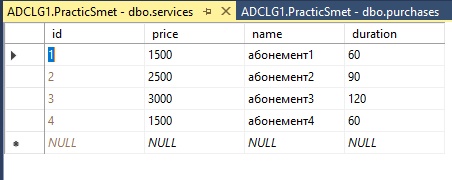


Рисунок 14

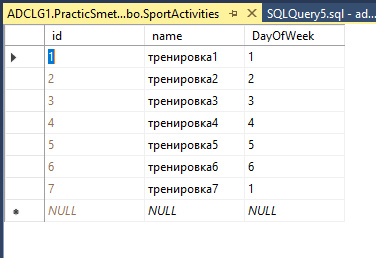


Рисунок 15

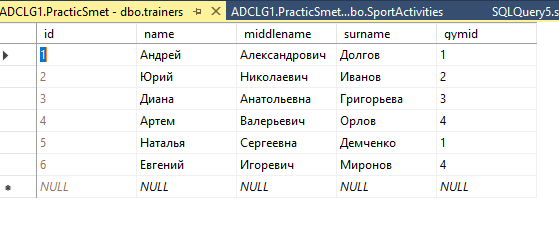


Рисунок 16

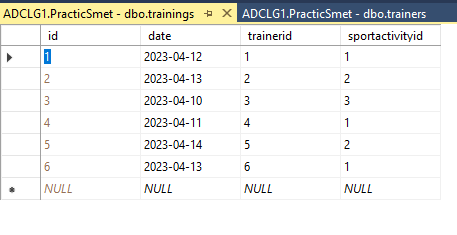


Рисунок 17

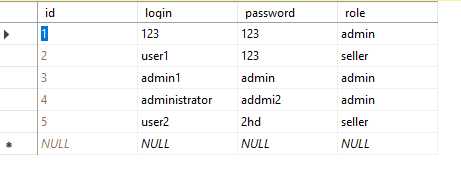


Рисунок 18

Создание диаграммы баз данных с созданными таблицами (Рисунок 19)

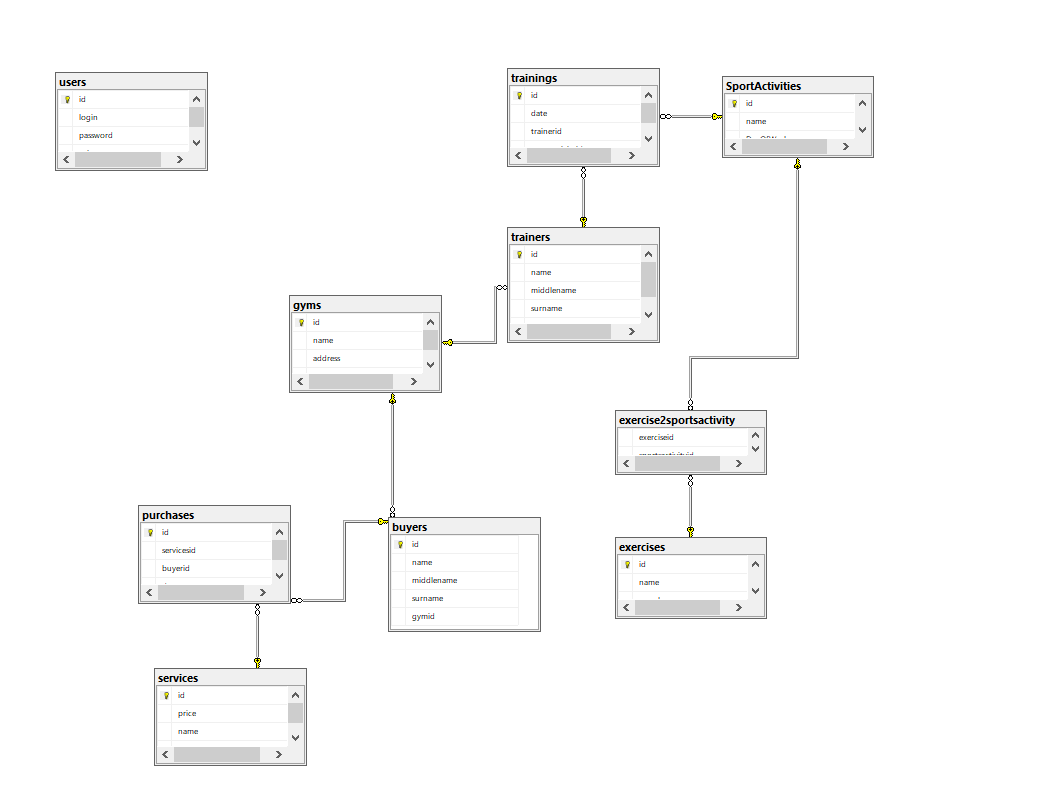


Рисунок 19

СЛОВАРЬ ДАННЫХ:

Таблица GroupClients (Таблица 1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GroupClients | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | GroupID | integer | Y | identity |
| FK | ClientID | integer | N |  |

Таблица 1

Таблица Clients (Таблица 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clients | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | ClientID | integer | Y | identity |
|  | Discount | integer | N |  |
|  | Visits | integer | N |  |
|  | ClientsFullName | nvarchar (MAX) | N |  |

Таблица 2

Таблица TrainersGroups (Таблица 3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TrainersGroups | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | GroupID | integer | Y | identity |
| FK | TrainerID | integer | Y |  |
| FK | IDDirection | integer | Y |  |
|  | GroupView | nvarchar (MAX) | N |  |

Таблица 3

Таблица Employees (Таблица 4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Employees | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | EmployeeID | integer | Y | identity |
|  | FullName | nvarchar (MAX) | N |  |
|  | JobTitle | nvarchar (MAX) | N |  |
|  | Wage | money | N |  |
|  | Login | nvarchar (MAX) | Y |  |
|  | Password | nvarchar (MAX) | Y |  |
|  | Role | nvarchar (MAX) | N |  |

Таблица 4

Таблица Purchases (Таблица 5)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purchases | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | PurchasesID | integer | Y | identity |
| FK | ClientID | integer | Y |  |
| FK | AbonementsID | integer | Y |  |
|  | Date | date | N |  |
|  | Price | money | N |  |

Таблица 5

Таблица Direction (Таблица 6)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Direction | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | IDDirection | integer | Y | identity |
| FK | IDTypeOfGym | integer | Y |  |
|  | Name | nvarchar (MAX) | N |  |

Таблица 6

Таблица TrainersSchedule (Таблица 7)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TrainersSchedule | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | ScheduleID | integer | Y | identity |
| FK | WorkoutID | integer | Y |  |
|  | GroupID | integer | Y |  |
|  | Time | integer | N |  |
|  | DaysOfTheWeek | nvarchar (MAX) | N |  |

Таблица 7

Таблица Workouts (Таблица 8)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Workouts | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | WorkoutID | integer | Y | identity |
|  | GymID | integer | Y |  |
|  | IDDirection | integer | Y |  |
|  | Name | nvarchar (MAX) | N |  |
|  | Duration | integer | N |  |

Таблица 8

Таблица Type Of Gym (Таблица 9)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type Of Gym | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | IDTypeOfGym | integer | Y | identity |
|  | Name | nvarchar (MAX) | N |  |

Таблица 9

Таблица Abonements (Таблица 10)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Abonements | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | AbonementsID | integer | Y | identity |
|  | Name | nvarchar (MAX) | N |  |
|  | Price | money | N |  |
|  | NumberOf Visits | integer | N |  |

Таблица 10

Таблица ExercisesInWorkouts (Таблица 11)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ExercisesInWorkouts | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | ExercisesID | integer | Y | identity |
| PK | WorkoutID | integer | Y | identity |

Таблица 11

Таблица Abonements (Таблица 12)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Abonements | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | Exercises | integer | Y | identity |
|  | Name | nvarchar (MAX) | N |  |
|  | MuscleGroup | nvarchar (MAX) | N |  |

Таблица 12

Таблица Gyms (Таблица 13)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gyms | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **UNUQUE?** | **NOTES** |
| PK | GymID | integer | Y | identity |
|  | IDTypeOfGym | integer | Y |  |
|  | Visits | integer | N |  |
|  | Name | nvarchar (MAX) | N |  |

Таблица 13

**Задание 3. Разработка макетов программы**

Варфрейм (WireFrame) - Черно-белый подробный план страницы сайта. Здесь намечается расположение элементов: кнопок, изображений, текстов. (Рисунок 20-16)

Мокап (Mockup)- Красивый вариант вайрфрейма. Тут уже появляются цвета, подбираются изображения, продумывается типографика. Получается красивая картинка приложения или сайта. (Рисунок 20-26)



Рисунок 20 – WireFrame начальной страницы – разметка элементов (THE START PAGE)

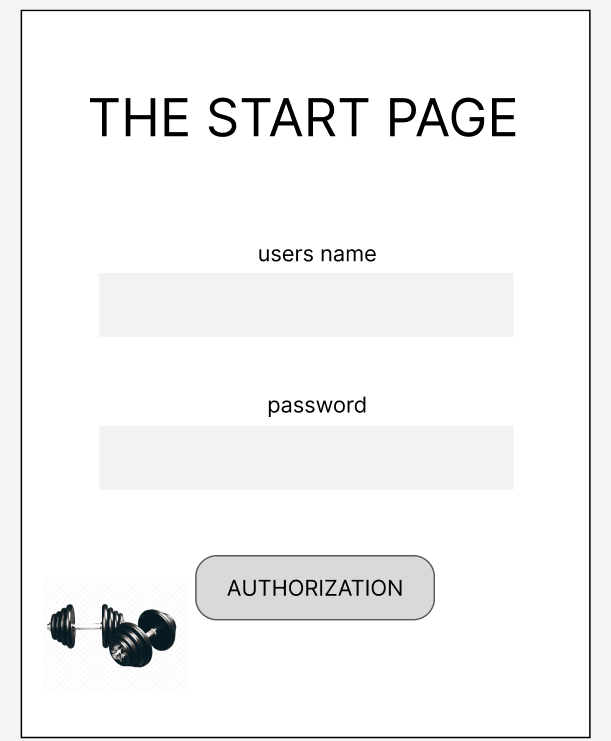


Рисунок 21 - «Раскрашенный» mockup начальной страницы

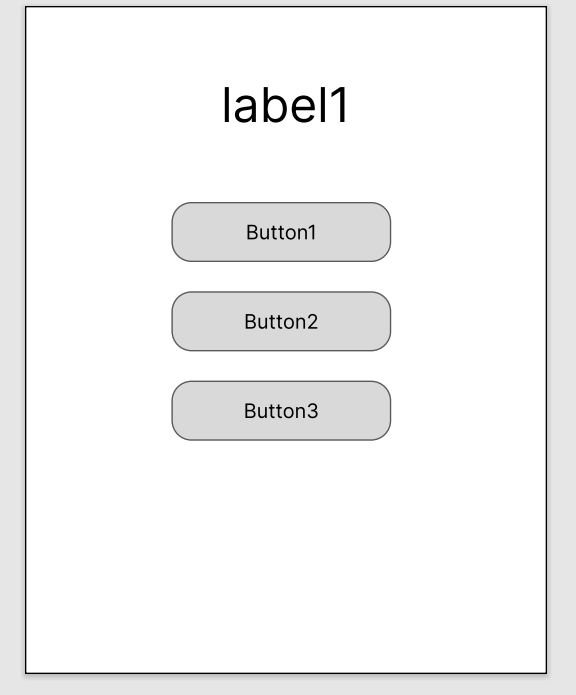


Рисунок 22 - WireFrame главной страницы (THE MAIN PAGE)

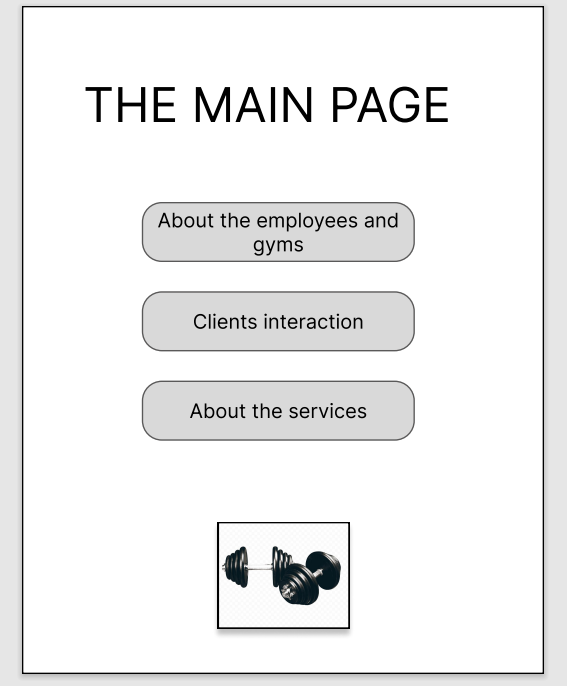


Рисунок 23 - Mockup главной страницы (THE MAIN PAGE)

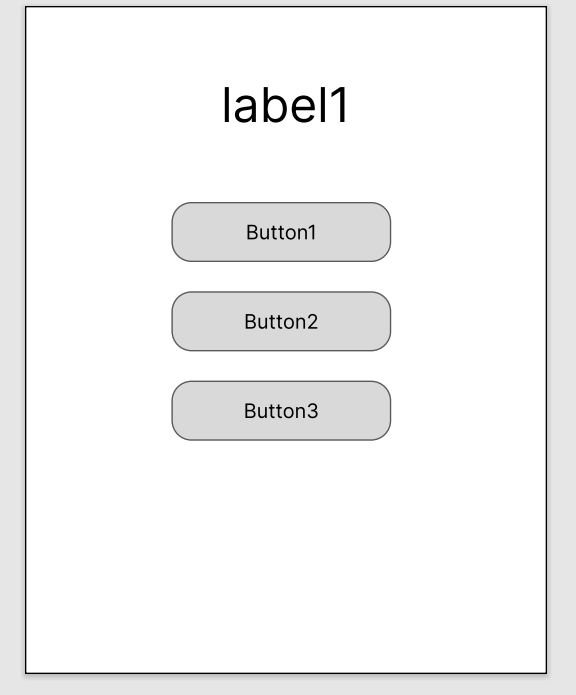


Рисунок 24 - WireFrame страницы об услугах (ABOUT THE SERVICES)

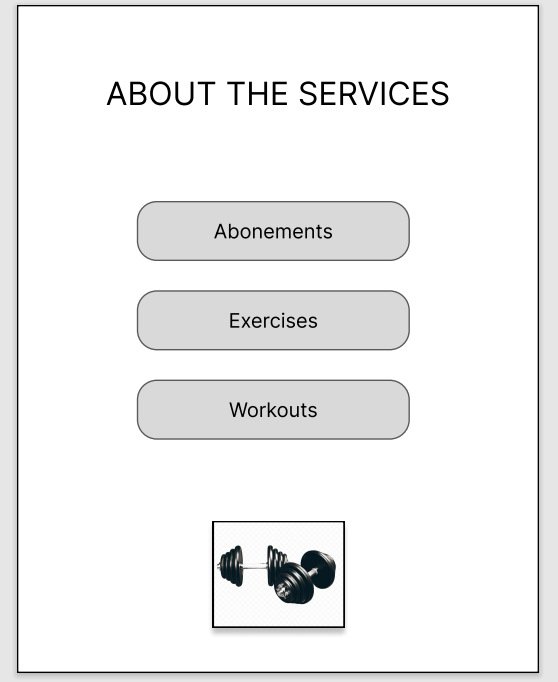


Рисунок 25 - Mockup страницы об услугах(ABOUT THE SERVICES)



Рисунок 26 - Карта навигации по приложению

**Задание 4. Разработка программы**

В ходе разработки информационной системы для фитнес-центра были реализованы формы, рассмотрим несколько из них: авторизации, главная, «о сотрудниках и залах».

Авторизация открывается при запуске приложения, необходимо ввести пароль и имя пользователя, зарегистрированного в системе. Так как ранее мы создали базу данных на сервере колледжа, то связываем проект Visual Studio с database с помощью файла в папке с помощью файла в папке Migrations (Рисунок 27-28)

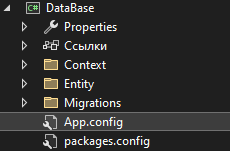


Рисунок 27



Рисунок 28

После данных манипуляций мы получаем проект, связанный с БД (Рисунок 29)

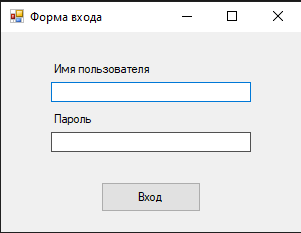


Рисунок 29 – форма входа

Вспоминаем созданную накануне таблицу Users (Рисунок 30)

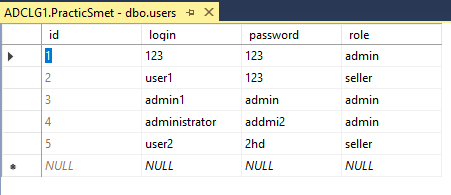


Рисунок 30

Активно пытаемся сломать систему незарегистрированным паролем и логином, на что выводится сообщение «неверное имя пользователя и пароль» (Рисунок 31)

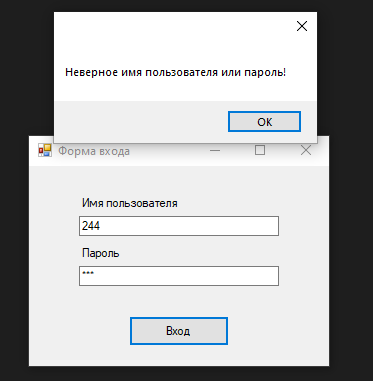


Рисунок 31

Если же ввести верные, зарегистрированные данные, то аутентификация будет пройдена успешно и нашему взору предоставится главная форма, откуда мы можем пойти по трем разным путем, как в былине про Илью Муромца: «на перекрестке камень лежит, а на том камне написано: кто вправо поедет – тому убитым быть, кто влево поедет – тому богатым быть, а кто прямо поедет – тому женатым быть.» (Рисунок 32)

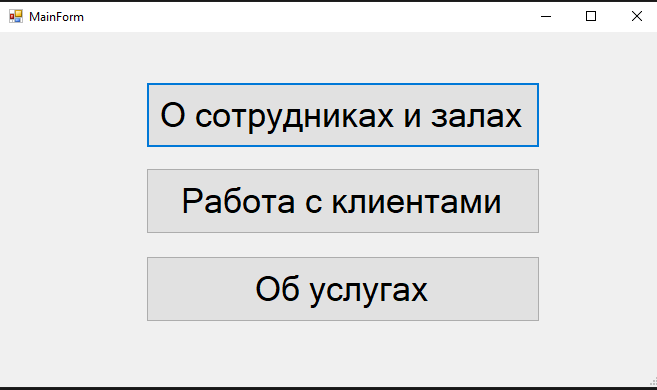


Рисунок 32 – форма MAIN

Как только путник информационной системы фитнес-центра решит почитать «о сотрудниках и залах», то откроется следующая форма (Рисунок 33)

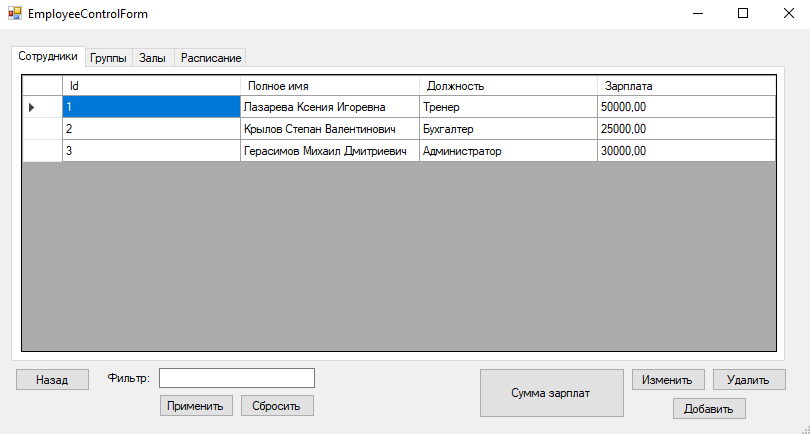


Рисунок 33

В ней реализована функция подсчета суммы зарплат всех немногочисленных сотрудников (Рисунок 34)

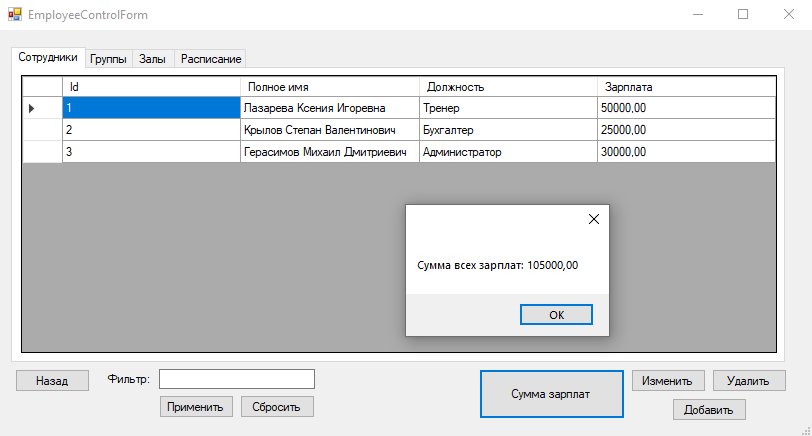


Рисунок 34

Есть возможность добавить/удалить/изменить данные о сотрудниках. Рассмотрим форму добавления бедолаги, устроившегося в контору. При добавлении сотрудника есть выпадающий список ролей (Рисунок 35)

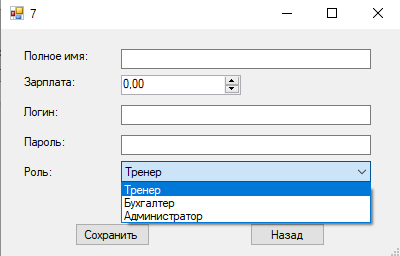


Рисунок 35

Также имеется на форме кнопка фильтрации данных, продемонстрирую ее работу (Рисунок 36-42)

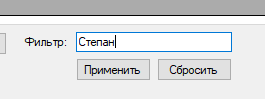


Рисунок 36

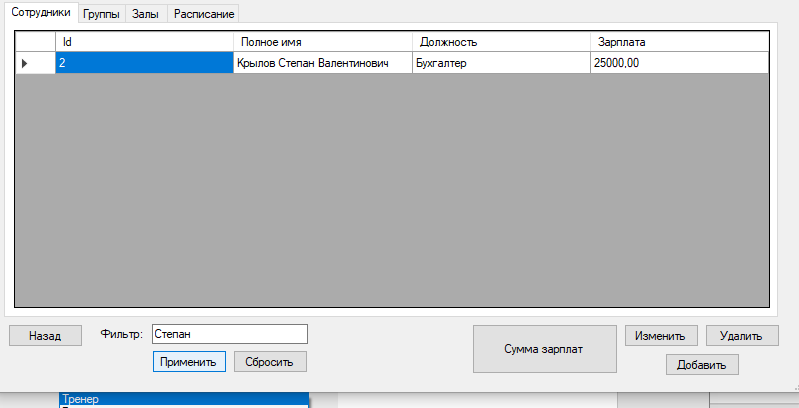


Рисунок 37

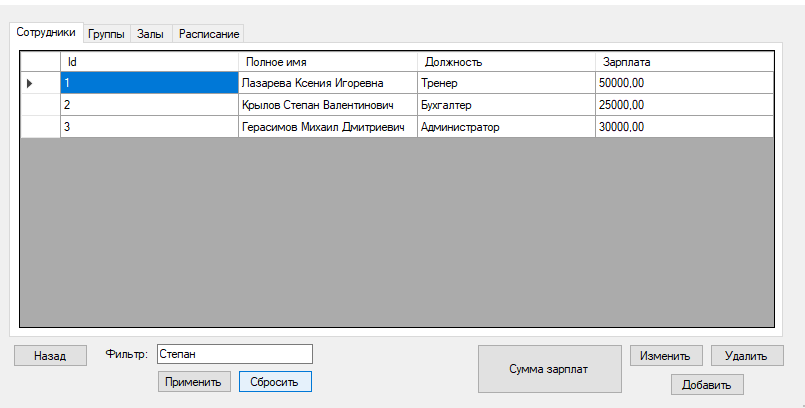


Рисунок 38

Форма «О сотрудниках и залах» содержит несколько вкладок, на которые также можно перейти



Рисунок 39

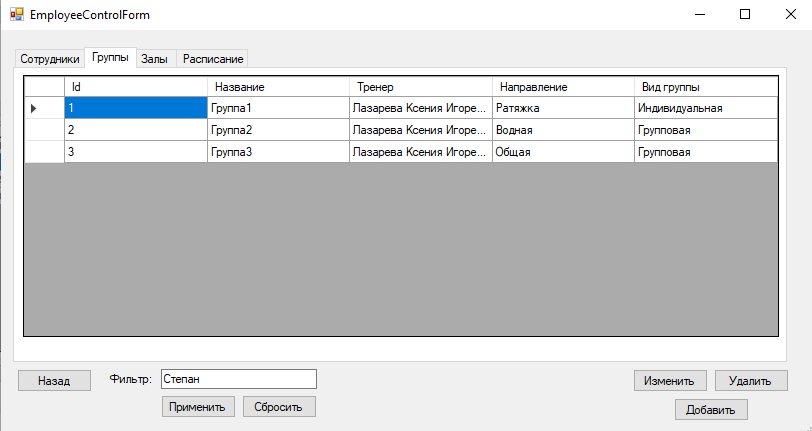


Рисунок 40

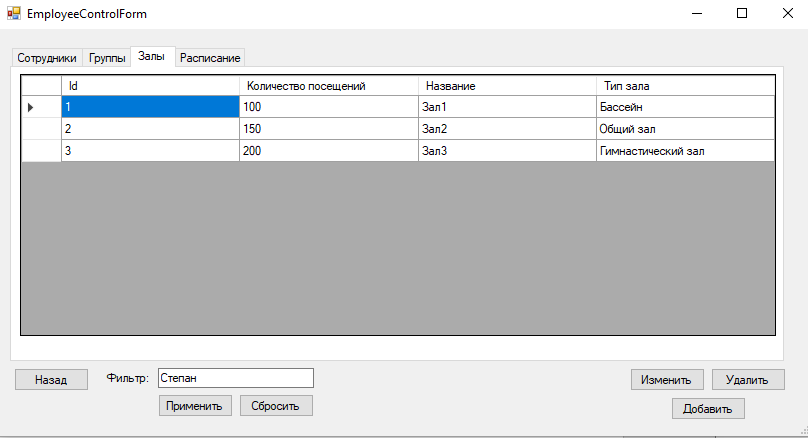


Рисунок 41

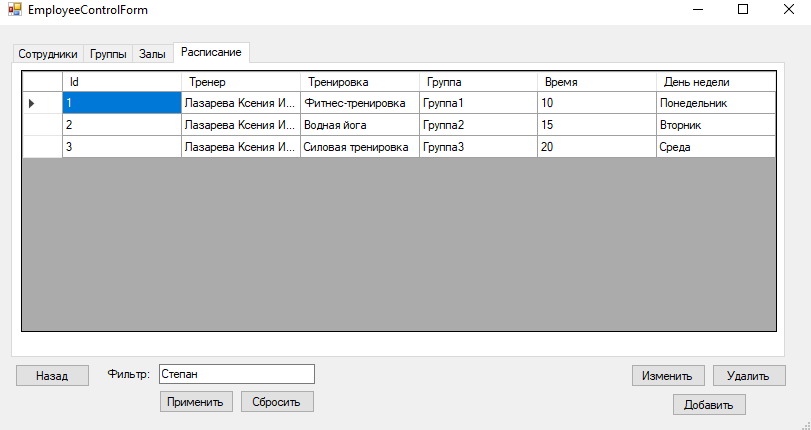


Рисунок 42

**Задание 5. Тестирование**

Таблица 13 – Аннотация теста

|  |  |
| --- | --- |
| **Название проекта** | Фитнес-центр |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Сметанникова Анастасия Алексеевна |
| **Дата(ы) теста** | 03.10.2024 |

Таблица 13

Таблица 14 – Тестовый пример 1

|  |  |
| --- | --- |
| **Тестовый пример №1** | public void CanRetrieveGymsFromDatabase() |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка на корректное добавление зала |
| **Краткое изложение теста** | Совершается попытка извлечения данных о залах из БД, чтобы понять, корректно ли извлекаются данные |
| **Этапы теста** | 1. Создание списка объектов Gym с двумя элементами, у которых есть свойства Id, Name, преобразование. (имитация данных с помощью мок-объекта \_mockGymSet , которые могут быть получены из базы данных) 2. Извлечение списка залов из базы данных 3. Проверка на количество полученных залов из БД, что оно является равным двум. |
| **Тестовые данные** | * new Gym {id= 1, Name = “Gym 1”} * new Gym {id= 2, Name = “Gym 2”} |
| **Ожидаемый результат** | Возвращается список, содержащий два объекта Gym |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Спортивные залы, существующие в БД |
| **Постусловие** | Корректное отображение извлеченных залов из БД |
| **Примечания** | - |

Таблица 14

Таблица 15 – Тестовый пример 2

|  |  |
| --- | --- |
| **Тестовый пример №2** | public void CanAddNewExerciseToDatabase() |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка корректное на добавление упражнения |
| **Краткое изложение теста** | Совершается попытка извлечения данных о залах из БД, чтобы понять, корректно ли извлекаются данные |
| **Этапы теста** | 1. Создание объекта Exercise со значениями свойств Name и MuscleGroup для добавления в базу данных 2. Настройка мока для метода Add 3. Добавление объекта в контекст базы данных 4. Проверка на совпадение созданного объекта и объекта, возвращаемого методом Add. |
| **Тестовые данные** | * Объект Exercise {Name = “Test Exercise”, MuscleGroup = “Test MuscleGroup”} |
| **Ожидаемый результат** | Созданный объект и объект возвращаемый методом Add совпадают |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Упражнения, существующие в БД |
| **Постусловие** | Корректное извлечение упражнения из БД |
| **Примечания** | - |

Таблица 15

Таблица 16 – Тестовый пример 3

|  |  |
| --- | --- |
| **Тестовый пример 3** | public void CanDeleteExerciseFromDatabase() |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка на корректное удаление упражнения. |
| **Краткое изложение теста** | Совершается попытка удаления упражнения из БД |
| **Этапы теста** | 1. Создание объекта Exercise со значениями свойств Name и MuscleGroup для добавления в базу данных 2. Настройка мока для метода Remove 3. Получение контекста базы данных 4. Удаление объекта из контекста БД 5. Проверка на совпадение созданного объекта и объекта, возвращаемого методом Remove. |
| **Тестовые данные** | Объект Exercise {Name = “Test Exercise”, MuscleGroup = “Test MuscleGroup”} |
| **Ожидаемый результат** | Созданный объект и объект возвращаемый методом Remove совпадают |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Упражнения, существующие в БД |
| **Постусловие** | Корректное извлечение упражнения из БД |
| **Примечания** | - |

Таблица 16

Таблица 17 – Тестовый пример 4

|  |  |
| --- | --- |
| **Тестовый пример №4** | public void CanCreateNewWorkoutWithAssignedGym() |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка возможности добавления тренировки |
| **Краткое изложение теста** | Совершается попытка добавление новой тренировки с присвоенным спортзалом |
| **Этапы теста** | 1. Создание объектов Gym, Workout 2. Настройка мока для метода Add. 3. Получение контекста БД 4. Добавление объекта Workout в контекст БД 5. Проверка на совпадение объекта, возвращаемого методом Add и созданным объектом |
| **Тестовые данные** | * Объект Gym {Name = “Test Gym”} * Объект Workout {Name = “Test Workout”, Gym = gym} |
| **Ожидаемый результат** | Добавление тренировки с присвоенным залом |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Наличие залов и тренировок в БД |
| **Постусловие** | Корректное извлечение данных из БД |
| **Примечания** | - |

Таблица 17

Таблица 18 – Тестовый пример 5

|  |  |
| --- | --- |
| **Тестовый пример №5** | public void CanRetrieveClientsWithTheirGroupsFromDatabase() |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Получение клиентов из БД |
| **Краткое изложение теста** | Совершается попытка получения данных клиентов фитнес-центра |
| **Этапы теста** | 1. Создание списка клиентов, добавление объектов со свойствами 2. Настройка мока для контекста базы данных, чтобы возвращал набор данных Clients 3. Получение контекста БД 4. Извлечение данных о клиентах 5. Проверка количества извлеченных данных |
| **Тестовые данные** | * new Client { Id = 1, FullName = "Client 1" }, * new Client { Id = 2, FullName = "Client 2" }, |
| **Ожидаемый результат** | Корректное извлечение данных о клиентах |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Наличие клиентов в БД |
| **Постусловие** | Корректное извлечение данных из БД |
| **Примечания** | - |

Таблица 18

Результаты тестирования продемонстрированы на следующе6м скрине (Рисунок 43)

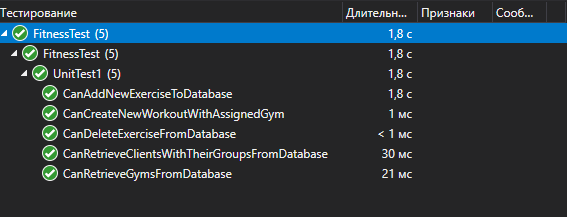


Рисунок 43 – Результаты тестирования

**Задание 6. GitHub. Оценка проекта**

Успешная агрузка файла на гитхаб (Рисунок 44-45)

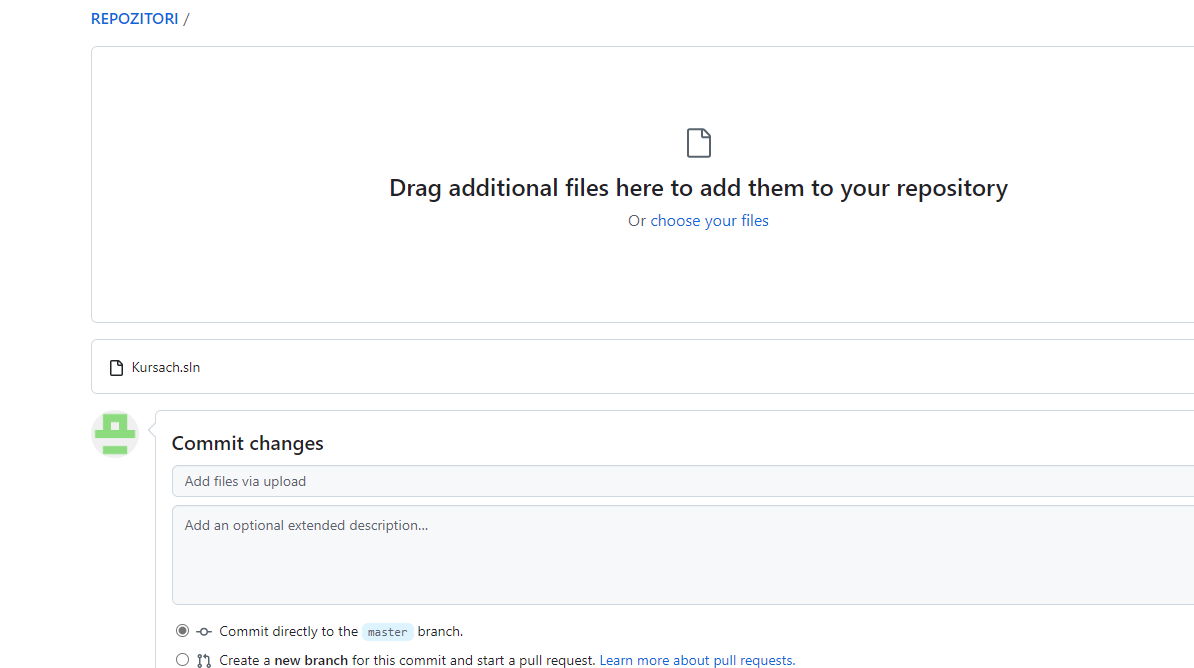


Рисунок 44

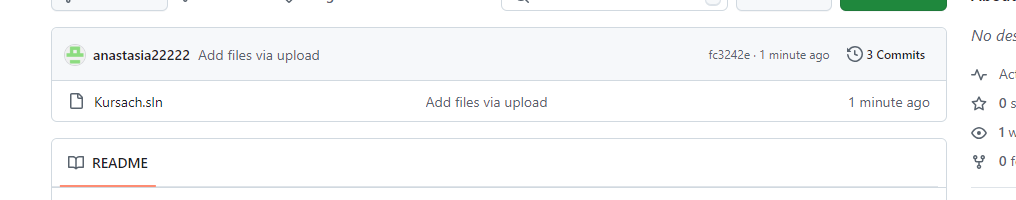


Рисунок 45

# **Заключение**

Результатом выполнения работы стала база данных и приложение, выполняющее с ней операции.

Была изучена предметная область, по которой были рассмотрены и опробованы основные методы работы с базой данных. Получены практические навыки работы с базой данных, а также изучены новые возможности среды программирования С#.

База данных была разработана с использованием СУБД SQL Server. Для разработки пользовательского приложения была использована среда Microsoft Visual Studio 2022 Community, язык программирования C# и библиотека для разработки WindowsForms.

# **Список использованных источников**

1. Кригель, Алекс, Трухнов, Борис, SQL. Библия пользователя, 2-е издание, Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2010. — 752 с.
2. Базы данных: проектирование и использование [Текст]: учебник / С.М. Диго. - М: Финансы и статистика, 2005. - 592 с. - Гриф МО "Рекомендовано". - ISBN 5-279-02571-2: 348-00.
3. Фиайли К. SQL: Пер. с англ. – М.: ДМК Пресс. – 456 с.
4. Шехтман В.Е. Базы данных, SQL и все такое [Текст]: курс лекций / В.Е. Шехтман; НФИ КемГУ. - Новокузнецк, 2006. – 195 с.
5. Работа с базами данных на языке C#. Технология ADO.NET [Текст]: учебное пособие / О.Н. Евсеева, А.Б. Шамшев – Ульяновск: УлГТУ, 2009 – 170 с.
6. Албахари, Джозеф, Албахари, Бен, C# 7.0. Справочник. Полное описание языка.: Пер. с англ. — СпБ.: ООО “Альфакнига”, 2018. — 1024 с

# **Приложение А**

**Код программы**

using DataBase.Entity;

using System.Data.Entity;

namespace DataBase.Context

{

public class FitnessDbContext : DbContext

{

public FitnessDbContext() : base("FC\_Matyagin")

{

}

public virtual DbSet<Gym> Gyms { get; set; }

public virtual DbSet<Exercise> Exercises { get; set; }

public virtual DbSet<ExercisesInWorkouts> ExercisesInWorkouts { get; set; }

public virtual DbSet<Workout> Workouts { get; set; }

public virtual DbSet<TypeOfGym> TypeOfGyms { get; set; }

public virtual DbSet<Abonement> Abonements { get; set; }

public virtual DbSet<TrainerShedule> TrainersShedules { get; set; }

public virtual DbSet<Direction> Directions { get; set; }

public virtual DbSet<Purchase> Purchases { get; set; }

public virtual DbSet<Group> TrainersGroups { get; set; }

public virtual DbSet<Employee> Employees { get; set; }

public virtual DbSet<GroupInClient> GroupInClients { get; set; }

public virtual DbSet<Client> Clients { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataBase.Entity

{

public class Abonement

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public decimal Price { get; set; }

[Required]

public int NumberOfVisits { get; set; }

public ICollection<Purchase> Purchases { get; set;}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataBase.Entity

{

public class Client

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public int Discount { get; set; }

[Required]

public int Visit { get; set; }

[Required]

public string FullName { get; set; }

public ICollection<Purchase> Purchases { get; set; }

public ICollection<GroupInClient> GroupInClients { get; set; }

}

}

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.Collections;

using System.Collections.Generic;

namespace DataBase.Entity

{

public class Direction

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public int TypeOfGymId { get; set; }

public ICollection<Workout> Workouts { get; set; }

public ICollection<Group> Groups { get; set; }

[ForeignKey(nameof(TypeOfGymId))]

public TypeOfGym TypeOfGym { get; set; }

}

}

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.Collections;

using System.Collections.Generic;

namespace DataBase.Entity

{

public class Employee

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public string FullName { get; set; }

[Required]

public string JobTitle { get; set; }

[Required]

public decimal Wage { get; set; }

[Required]

public string Login { get; set; }

[Required]

public string Password { get; set; }

[Required]

public string Role { get; set; }

public ICollection<Group> Groups { get; set; }

}

}

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace DataBase.Entity

{

public class Exercise

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public string MuscleGroup { get; set; }

public IEnumerable<ExercisesInWorkouts> ExercisesInWorkouts { get; set; }

}

}

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace DataBase.Entity

{

public class ExercisesInWorkouts

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public int ExerciseId { get; set; }

[Required]

public int WorkoutId { get; set; }

[ForeignKey(nameof(ExerciseId))]

public Exercise Exercise { get; set; }

[ForeignKey(nameof(WorkoutId))]

public Workout Workout { get; set; }

}

}

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace DataBase.Entity

{

public class Group

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public int TrainerId { get; set;}

[Required]

public int DirectionId { get; set; }

[Required]

public string GroupView { get; set; }

public ICollection<GroupInClient> GroupInClients { get; set; }

public ICollection<TrainerShedule> TrainerShedules { get; set; }

[ForeignKey(nameof(TrainerId))]

public Employee Trainer { get; set; }

[ForeignKey(nameof(DirectionId))]

public Direction Direction { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataBase.Entity

{

public class GroupInClient

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public int GroupId { get; set; }

[Required]

public int ClientId { get; set; }

[ForeignKey(nameof(ClientId))]

public Client Client { get; set; }

[ForeignKey(nameof(GroupId))]

public Group Group { get; set; }

}

}

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace DataBase.Entity

{

public class Gym

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public int Visits { get; set; }

[Required]

public string Name { get; set; }

[Required]

public int TypeOfGymId { get; set; }

[ForeignKey(nameof(TypeOfGymId))]

public TypeOfGym TypeOfGym { get; set; }

public IEnumerable<Workout> Workouts { get; set; }

}

}

using System;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace DataBase.Entity

{

public class Purchase

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public int ClientId { get; set; }

[Required]

public int AbonementId { get; set; }

[Required]

public DateTime Date { get; set; }

[Required]

public decimal Price { get; set; }

[ForeignKey(nameof(AbonementId))]

public Abonement Abonement { get; set; }

[ForeignKey(nameof(ClientId))]

public Client Client { get; set; }

}

}

using System;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace DataBase.Entity

{

public class TrainerShedule

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public int WorkoutId { get; set; }

[Required]

public int GroupId { get; set;}

[Required]

public int Time { get; set; }

[Required]

public DayOfWeek DayOfWeek { get; set; }

[ForeignKey(nameof(WorkoutId))]

public Workout Workout { get; set; }

[ForeignKey(nameof (GroupId))]

public Group Group { get; set; }

}

}

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.Collections.Generic;

namespace DataBase.Entity

{

public class TypeOfGym

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public string Name { get; set; }

public ICollection<Gym> Gyms { get; set; }

public ICollection<Direction> Directions { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace DataBase.Entity

{

public class Workout

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public int GymId { get; set; }

[Required]

public int DirectionId { get; set; }

[Required]

public int Duration { get; set; }

public IEnumerable<ExercisesInWorkouts> ExercisesInWorkouts { get; set; }

public IEnumerable<TrainerShedule> TrainerShedules { get; set; }

[ForeignKey(nameof(GymId))]

public Gym Gym { get; set; }

[ForeignKey(nameof(DirectionId))]

public Direction Direction { get; set; }

}

}

namespace DataBase.Migrations

{

using System;

using System.Data.Entity.Migrations;

public partial class Initial : DbMigration

{

public override void Up()

{

CreateTable(

"dbo.Abonements",

c => new

{

Id = c.Int(nullable: false, identity: true),

Name = c.String(nullable: false),

Price = c.Decimal(nullable: false, precision: 18, scale: 2),

NumberOfVisits = c.Int(nullable: false),

})

.PrimaryKey(t => t.Id);

CreateTable(

"dbo.Purchases",

c => new

{

Id = c.Int(nullable: false, identity: true),

ClientId = c.Int(nullable: false),

AbonementId = c.Int(nullable: false),

Date = c.DateTime(nullable: false),

Price = c.Decimal(nullable: false, precision: 18, scale: 2),

})

.PrimaryKey(t => t.Id)

.ForeignKey("dbo.Abonements", t => t.AbonementId, cascadeDelete: true)

.ForeignKey("dbo.Clients", t => t.ClientId, cascadeDelete: true)

.Index(t => t.ClientId)

.Index(t => t.AbonementId);

CreateTable(

"dbo.Clients",

c => new

{

Id = c.Int(nullable: false, identity: true),

Discount = c.Int(nullable: false),

Visit = c.Int(nullable: false),

FullName = c.String(nullable: false),

})

.PrimaryKey(t => t.Id);

CreateTable(

"dbo.GroupInClients",

c => new

{

Id = c.Int(nullable: false, identity: true),

GroupId = c.Int(nullable: false),

ClientId = c.Int(nullable: false),

})

.PrimaryKey(t => t.Id)

.ForeignKey("dbo.Clients", t => t.ClientId, cascadeDelete: true)

.ForeignKey("dbo.Groups", t => t.GroupId, cascadeDelete: true)

.Index(t => t.GroupId)

.Index(t => t.ClientId);

CreateTable(

"dbo.Groups",

c => new

{

Id = c.Int(nullable: false, identity: true),

Name = c.String(nullable: false),

TrainerId = c.Int(nullable: false),

DirectionId = c.Int(nullable: false),

GroupView = c.String(nullable: false),

})

.PrimaryKey(t => t.Id)

.ForeignKey("dbo.Directions", t => t.DirectionId, cascadeDelete: true)

.ForeignKey("dbo.Employees", t => t.TrainerId, cascadeDelete: true)

.Index(t => t.TrainerId)

.Index(t => t.DirectionId);

CreateTable(

"dbo.Directions",

c => new

{

Id = c.Int(nullable: false, identity: true),

Name = c.String(nullable: false),

TypeOfGymId = c.Int(nullable: false),

})

.PrimaryKey(t => t.Id)

.ForeignKey("dbo.TypeOfGyms", t => t.TypeOfGymId, cascadeDelete: true)

.Index(t => t.TypeOfGymId);

CreateTable(

"dbo.TypeOfGyms",

c => new

{

Id = c.Int(nullable: false, identity: true),

Name = c.String(nullable: false),

})

.PrimaryKey(t => t.Id);

CreateTable(

"dbo.Gyms",

c => new

{

Id = c.Int(nullable: false, identity: true),

Visits = c.Int(nullable: false),

Name = c.String(nullable: false),

TypeOfGymId = c.Int(nullable: false),

})

.PrimaryKey(t => t.Id)

.ForeignKey("dbo.TypeOfGyms", t => t.TypeOfGymId, cascadeDelete: true)

.Index(t => t.TypeOfGymId);

CreateTable(

"dbo.Workouts",

c => new

{

Id = c.Int(nullable: false, identity: true),

Name = c.String(nullable: false),

GymId = c.Int(nullable: false),

DirectionId = c.Int(nullable: false),

Duration = c.Int(nullable: false),

})

.PrimaryKey(t => t.Id)

.ForeignKey("dbo.Directions", t => t.DirectionId, cascadeDelete: true)

.ForeignKey("dbo.Gyms", t => t.GymId, cascadeDelete: false)

.Index(t => t.GymId)

.Index(t => t.DirectionId);

CreateTable(

"dbo.Employees",

c => new

{

Id = c.Int(nullable: false, identity: true),

FullName = c.String(nullable: false),

JobTitle = c.String(nullable: false),

Wage = c.Decimal(nullable: false, precision: 18, scale: 2),

Login = c.String(nullable: false),

Password = c.String(nullable: false),

Role = c.String(nullable: false),

})

.PrimaryKey(t => t.Id);

CreateTable(

"dbo.TrainerShedules",

c => new

{

Id = c.Int(nullable: false, identity: true),

WorkoutId = c.Int(nullable: false),

GroupId = c.Int(nullable: false),

Time = c.Int(nullable: false),

DayOfWeek = c.Int(nullable: false),

})

.PrimaryKey(t => t.Id)

.ForeignKey("dbo.Groups", t => t.GroupId, cascadeDelete: true)

.ForeignKey("dbo.Workouts", t => t.WorkoutId, cascadeDelete: false)

.Index(t => t.WorkoutId)

.Index(t => t.GroupId);

CreateTable(

"dbo.Exercises",

c => new

{

Id = c.Int(nullable: false, identity: true),

Name = c.String(nullable: false),

MuscleGroup = c.String(nullable: false),

})

.PrimaryKey(t => t.Id);

CreateTable(

"dbo.ExercisesInWorkouts",

c => new

{

Id = c.Int(nullable: false, identity: true),

ExerciseId = c.Int(nullable: false),

WorkoutId = c.Int(nullable: false),

})

.PrimaryKey(t => t.Id)

.ForeignKey("dbo.Exercises", t => t.ExerciseId, cascadeDelete: true)

.ForeignKey("dbo.Workouts", t => t.WorkoutId, cascadeDelete: true)

.Index(t => t.ExerciseId)

.Index(t => t.WorkoutId);

}

public override void Down()

{

DropForeignKey("dbo.ExercisesInWorkouts", "WorkoutId", "dbo.Workouts");

DropForeignKey("dbo.ExercisesInWorkouts", "ExerciseId", "dbo.Exercises");

DropForeignKey("dbo.Purchases", "ClientId", "dbo.Clients");

DropForeignKey("dbo.TrainerShedules", "WorkoutId", "dbo.Workouts");

DropForeignKey("dbo.TrainerShedules", "GroupId", "dbo.Groups");

DropForeignKey("dbo.Groups", "TrainerId", "dbo.Employees");

DropForeignKey("dbo.GroupInClients", "GroupId", "dbo.Groups");

DropForeignKey("dbo.Workouts", "GymId", "dbo.Gyms");

DropForeignKey("dbo.Workouts", "DirectionId", "dbo.Directions");

DropForeignKey("dbo.Gyms", "TypeOfGymId", "dbo.TypeOfGyms");

DropForeignKey("dbo.Directions", "TypeOfGymId", "dbo.TypeOfGyms");

DropForeignKey("dbo.Groups", "DirectionId", "dbo.Directions");

DropForeignKey("dbo.GroupInClients", "ClientId", "dbo.Clients");

DropForeignKey("dbo.Purchases", "AbonementId", "dbo.Abonements");

DropIndex("dbo.ExercisesInWorkouts", new[] { "WorkoutId" });

DropIndex("dbo.ExercisesInWorkouts", new[] { "ExerciseId" });

DropIndex("dbo.TrainerShedules", new[] { "GroupId" });

DropIndex("dbo.TrainerShedules", new[] { "WorkoutId" });

DropIndex("dbo.Workouts", new[] { "DirectionId" });

DropIndex("dbo.Workouts", new[] { "GymId" });

DropIndex("dbo.Gyms", new[] { "TypeOfGymId" });

DropIndex("dbo.Directions", new[] { "TypeOfGymId" });

DropIndex("dbo.Groups", new[] { "DirectionId" });

DropIndex("dbo.Groups", new[] { "TrainerId" });

DropIndex("dbo.GroupInClients", new[] { "ClientId" });

DropIndex("dbo.GroupInClients", new[] { "GroupId" });

DropIndex("dbo.Purchases", new[] { "AbonementId" });

DropIndex("dbo.Purchases", new[] { "ClientId" });

DropTable("dbo.ExercisesInWorkouts");

DropTable("dbo.Exercises");

DropTable("dbo.TrainerShedules");

DropTable("dbo.Employees");

DropTable("dbo.Workouts");

DropTable("dbo.Gyms");

DropTable("dbo.TypeOfGyms");

DropTable("dbo.Directions");

DropTable("dbo.Groups");

DropTable("dbo.GroupInClients");

DropTable("dbo.Clients");

DropTable("dbo.Purchases");

DropTable("dbo.Abonements");

}

}

}

namespace DataBase.Migrations

{

using System;

using System.Data.Entity.Migrations;

public partial class adddata : DbMigration

{

public override void Up()

{

Sql(@"INSERT INTO dbo.TypeOfGyms (Name)

VALUES

(N'Бассейн'),

(N'Общий зал'),

(N'Гимнастический зал');

INSERT INTO dbo.Directions (Name, TypeOfGymId)

VALUES (N'Общая', 2),

(N'Ратяжка', 3),

(N'Водная', 1),

(N'Силовая', 2),

(N'Танцы', 3),

(N'Атлетика', 3);

INSERT INTO dbo.Gyms (Visits, Name, TypeOfGymId)

VALUES (100, N'Зал1', 1),

(150, N'Зал2', 2),

(200, N'Зал3', 3);

INSERT INTO dbo.Employees (FullName, JobTitle, Wage, Login, Password, Role)

VALUES (N'Лазарева Ксения Игоревна', N'Тренер', 50000, N'111', N'111', N'trainer'),

(N'Крылов Степан Валентинович', N'Бухгалтер', 25000, N'222', N'222', N'accountan'),

(N'Герасимов Михаил Дмитриевич', N'Администратор', 30000, N'123', N'123', N'admin');

INSERT INTO dbo.Clients (Discount, Visit, FullName)

VALUES (10, 5, N'Васильев Виктор Сергеевич'),

(15, 10, N'Захарова Ирина Васильевна'),

(20, 15, N'Григорьев Илья Петрович');

INSERT INTO dbo.Groups (Name, TrainerId, DirectionId, GroupView)

VALUES (N'Группа1', 1, 2, N'Индивидуальная'),

(N'Группа2', 1, 3, N'Групповая'),

(N'Группа3', 1, 1, N'Групповая');

INSERT INTO dbo.Exercises (Name, MuscleGroup)

VALUES (N'Жим лежа', N'Грудные мышцы'),

(N'Приседания со штангой', N'Ноги и ягодицы'),

(N'Подтягивания', N'Спина и бицепс');

INSERT INTO dbo.Abonements (Name, Price, NumberOfVisits)

VALUES (N'Абонемент на месяц', 1000, 12),

(N'Абонемент на год', 10000, 150),

(N'Разовый визит', 200, 1);

INSERT INTO dbo.GroupInClients (GroupId, ClientId)

VALUES (1, 1),

(2, 2),

(3, 3);

INSERT INTO dbo.Workouts (Name, GymId, DirectionId, Duration)

VALUES (N'Фитнес-тренировка', 2, 1, 90),

(N'Водная йога', 1, 3, 90),

(N'Силовая тренировка', 2, 4, 90),

(N'Танцы', 3, 3, 60);

INSERT INTO dbo.TrainerShedules (WorkoutId, GroupId, Time, DayOfWeek)

VALUES (1, 1, 10, 1),

(2, 2, 15, 2),

(3, 3, 20, 3);

INSERT INTO dbo.ExercisesInWorkouts (ExerciseId, WorkoutId)

VALUES (1, 1),

(2, 2),

(3, 1);

");

}

public override void Down()

{

}

}

}

namespace DataBase.Migrations

{

using System;

using System.Data.Entity;

using System.Data.Entity.Migrations;

using System.Linq;

internal sealed class Configuration : DbMigrationsConfiguration<DataBase.Context.FitnessDbContext>

{

public Configuration()

{

AutomaticMigrationsEnabled = true;

}

protected override void Seed(DataBase.Context.FitnessDbContext context)

{

// This method will be called after migrating to the latest version.

// You can use the DbSet<T>.AddOrUpdate() helper extension method

// to avoid creating duplicate seed data.

}

}

}

using DataBase.Context;

using DataBase.Entity;

using System;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach.EntityForm

{

public partial class AbonementForm : Form

{

private Abonement abonement;

private FitnessDbContext dbContext;

public AbonementForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

}

public AbonementForm(int id) : this()

{

abonement = dbContext.Abonements.FirstOrDefault(x => x.Id == id);

textBoxName.Text = abonement.Name;

numericUpDownPrice.Value = abonement.Price;

numericUpDownVisits.Value = abonement.NumberOfVisits;

}

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

{

if (string.IsNullOrEmpty(textBoxName.Text))

{

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

return;

}

if (abonement == null)

{

var newAbonement = new Abonement

{

Name = textBoxName.Text,

Price = numericUpDownPrice.Value,

NumberOfVisits = (int)numericUpDownVisits.Value

};

dbContext.Abonements.Add(newAbonement);

dbContext.SaveChanges();

MessageBox.Show("Новый абонемент успешно добавлен!");

}

else{

abonement.Name = textBoxName.Text;

abonement.Price = numericUpDownPrice.Value;

abonement.NumberOfVisits = (int)numericUpDownVisits.Value;

dbContext.SaveChanges();

MessageBox.Show("Данные о покупке успешно обновлены!");

}

Close();

}

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

{

Close();

}

}

}

using DataBase.Context;

using DataBase.Entity;

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 FitnessKursach.EntityForm

{

public partial class ClientForm : Form

{

private FitnessDbContext dbContext;

private Client currentClient;

public ClientForm()

{

InitializeComponent();

// Инициализация контекста базы данных

dbContext = new FitnessDbContext();

}

public ClientForm(int clientId) : this()

{

currentClient = dbContext.Clients.FirstOrDefault(c => c.Id == clientId);

if (currentClient != null)

{

// Заполнение элементов управления данными о клиенте для изменения

txtFullName.Text = currentClient.FullName;

numDiscount.Value = currentClient.Discount;

numVisit.Value = currentClient.Visit;

}

}

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

{

if (currentClient == null)

{

// Добавление нового клиента

string fullName = txtFullName.Text;

int discount = (int)numDiscount.Value;

int visit = (int)numVisit.Value;

var newClient = new Client

{

FullName = fullName,

Discount = discount,

Visit = visit

};

dbContext.Clients.Add(newClient);

dbContext.SaveChanges();

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

}

else

{

// Изменение существующего клиента

currentClient.FullName = txtFullName.Text;

currentClient.Visit = (int)numVisit.Value;

var discount = currentClient.Visit / 30;

if(discount > 50)

discount = 50;

discount = Math.Max(discount, (int)numDiscount.Value);

currentClient.Discount = discount;

dbContext.SaveChanges();

MessageBox.Show("Данные о клиенте успешно обновлены!");

}

this.Close();

}

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

{

this.Close();

}

}

}

using DataBase.Context;

using DataBase.Entity;

using System;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach.EntityForm

{

public partial class EmployeeForm : Form

{

private FitnessDbContext dbContext;

private Employee currentEmployee;

public EmployeeForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

comboBoxRole.DataSource = HelpData.RussiaRoles;

}

public EmployeeForm(int employeeId) : this()

{

currentEmployee = dbContext.Employees.FirstOrDefault(e => e.Id == employeeId);

if (currentEmployee != null)

{

// Заполнение элементов управления данными о сотруднике для изменения

txtFullName.Text = currentEmployee.FullName;

numWage.Value = currentEmployee.Wage;

txtLogin.Text = currentEmployee.Login;

txtPassword.Text = currentEmployee.Password;

comboBoxRole.Text = HelpData.RolesToRussia[currentEmployee.Role];

}

}

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

{

if (currentEmployee == null)

{

// Добавление нового сотрудника

var login = txtLogin.Text;

var password = txtPassword.Text;

var fullName = txtFullName.Text;

if (string.IsNullOrEmpty(password))

{

MessageBox.Show("Невозможно создать сотрудника с пустым паролем");

return;

}

if (string.IsNullOrEmpty(fullName))

{

MessageBox.Show("Невозможно создать сотрудника с пустым имеенем");

return;

}

var existEmployee = dbContext.Employees.FirstOrDefault(x => x.Login == login);

if (existEmployee != null)

{

MessageBox.Show("Сотрудник с таким логином уже существует");

return;

}

var newEmployee = new Employee

{

FullName = fullName,

JobTitle =comboBoxRole.Text,

Wage = numWage.Value,

Login = login,

Password = password,

Role = HelpData.RolesToSystem[comboBoxRole.Text]

};

dbContext.Employees.Add(newEmployee);

dbContext.SaveChanges();

MessageBox.Show("Новый сотрудник успешно добавлен!");

}

else

{

// Изменение существующего сотрудника

currentEmployee.FullName = txtFullName.Text;

currentEmployee.JobTitle =comboBoxRole.Text;

currentEmployee.Wage = numWage.Value;

currentEmployee.Login = txtLogin.Text;

currentEmployee.Password = txtPassword.Text;

currentEmployee.Role = HelpData.RolesToSystem[comboBoxRole.Text];

dbContext.SaveChanges();

MessageBox.Show("Данные о сотруднике успешно обновлены!");

}

this.Close();

}

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

{

this.Close();

}

}

}

using DataBase.Context;

using DataBase.Entity;

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 FitnessKursach.EntityForm

{

public partial class ExerciseForm : Form

{

private Exercise exercise;

private FitnessDbContext dbContext;

public ExerciseForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

}

public ExerciseForm(int id) : this()

{

exercise = dbContext.Exercises.FirstOrDefault(x => x.Id == id);

textBoxName.Text = exercise.Name;

textBoxMuscleGroup.Text = exercise.MuscleGroup;

}

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

{

if (string.IsNullOrEmpty(textBoxName.Text) || string.IsNullOrEmpty(textBoxMuscleGroup.Text))

{

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

return;

}

if (exercise == null)

{

var newExercise = new Exercise

{

Name = textBoxName.Text,

MuscleGroup = textBoxMuscleGroup.Text

};

dbContext.Exercises.Add(newExercise);

dbContext.SaveChanges();

MessageBox.Show("Новое упражнение успешно добавлено!");

}

else

{

exercise.Name = textBoxName.Text;

exercise.MuscleGroup = textBoxMuscleGroup.Text;

dbContext.SaveChanges();

MessageBox.Show("Данные об упражнении успешно обновлены!");

}

Close();

}

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

{

Close();

}

}

}

using DataBase.Context;

using DataBase.Entity;

using System;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach.EntityForm

{

public partial class GroupForm : Form

{

private FitnessDbContext dbContext;

private Group currentGroup;

public GroupForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

PopulateComboBoxes(); // Заполняем комбобоксы при инициализации формы

}

public GroupForm(int groupId) : this()

{

currentGroup = dbContext.TrainersGroups.FirstOrDefault(g => g.Id == groupId);

if (currentGroup != null)

{

// Заполнение элементов управления данными о группе для изменения

txtName.Text = currentGroup.Name;

comboBoxTrainer.SelectedValue = currentGroup.TrainerId;

comboBoxDirection.SelectedValue = currentGroup.DirectionId;

txtGroupView.Text = currentGroup.GroupView;

}

}

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

{

if (currentGroup == null)

{

// Добавление новой группы

var newGroup = new Group

{

Name = txtName.Text,

TrainerId = (int)comboBoxTrainer.SelectedValue,

DirectionId = (int)comboBoxDirection.SelectedValue,

GroupView = txtGroupView.Text

};

dbContext.TrainersGroups.Add(newGroup);

dbContext.SaveChanges();

MessageBox.Show("Новая группа успешно добавлена!");

}

else

{

// Изменение существующей группы

currentGroup.Name = txtName.Text;

currentGroup.TrainerId = ((HelperForForms)comboBoxTrainer.SelectedItem).Id;

currentGroup.DirectionId = ((HelperForForms)comboBoxDirection.SelectedItem).Id;

currentGroup.GroupView = txtGroupView.Text;

dbContext.SaveChanges();

MessageBox.Show("Данные о группе успешно обновлены!");

}

this.Close();

}

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

{

this.Close();

}

private void PopulateComboBoxes()

{

// Здесь добавьте код для заполнения комбобоксов значениями для TrainerId и DirectionId

// Например:

comboBoxTrainer.DataSource = dbContext.Employees.Where(x => x.Role == "trainer").Select(x => new HelperForForms { Id = x.Id, Name = x.FullName }).ToList();

comboBoxTrainer.DisplayMember = "Name";

comboBoxTrainer.ValueMember = "Id";

comboBoxDirection.DataSource = dbContext.Directions.Select(x => new HelperForForms { Id = x.Id, Name = x.Name }).ToList();

comboBoxDirection.DisplayMember = "Name";

comboBoxDirection.ValueMember = "Id";

}

}

}

using DataBase.Context;

using DataBase.Entity;

using System;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach.EntityForm

{

public partial class GymForm : Form

{

private FitnessDbContext dbContext;

private Gym currentGym;

public GymForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

comboBoxTypeOfGym.DataSource = dbContext.TypeOfGyms.Select(x => new HelperForForms { Id = x.Id, Name = x.Name }).ToList();

comboBoxTypeOfGym.DisplayMember = "Name";

comboBoxTypeOfGym.ValueMember = "Id";

}

public GymForm(int gymId) : this()

{

currentGym = dbContext.Gyms.FirstOrDefault(g => g.Id == gymId);

if (currentGym != null)

{

// Заполнение элементов управления данными о зале для изменения

txtName.Text = currentGym.Name;

numVisits.Value = currentGym.Visits;

comboBoxTypeOfGym.SelectedValue = currentGym.TypeOfGymId;

}

}

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

{

if (currentGym == null)

{

// Добавление нового зала

var newGym = new Gym

{

Name = txtName.Text,

Visits = (int)numVisits.Value,

TypeOfGymId = ((HelperForForms)comboBoxTypeOfGym.SelectedItem).Id,

// Здесь добавьте соответствующий код для выбора значения TypeOfGymId

};

dbContext.Gyms.Add(newGym);

dbContext.SaveChanges();

MessageBox.Show("Новый зал успешно добавлен!");

}

else

{

// Изменение существующего зала

currentGym.Name = txtName.Text;

currentGym.Visits = (int)numVisits.Value;

currentGym.TypeOfGymId = ((HelperForForms)comboBoxTypeOfGym.SelectedItem).Id;

dbContext.SaveChanges();

MessageBox.Show("Данные о зале успешно обновлены!");

}

this.Close();

}

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

{

this.Close();

}

}

}

namespace FitnessKursach.EntityForm

{

public class HelperForForms

{

public int Id { get; set; }

public string Name { get; set; }

}

}

using DataBase.Context;

using DataBase.Entity;

using System;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach.EntityForm

{

public partial class PurchaseForm : Form

{

private FitnessDbContext dbContext;

private Purchase currentPurchase;

public PurchaseForm()

{

InitializeComponent();

// Инициализация контекста базы данных

dbContext = new FitnessDbContext();

comboBoxAbonement.DataSource = dbContext.Abonements.Select(x => new HelperForForms { Id = x.Id, Name = x.Name }).ToArray();

comboBoxAbonement.ValueMember = "Id";

comboBoxAbonement.DisplayMember = "Name";

comboBoxClient.DataSource = dbContext.Clients.Select(x => new HelperForForms { Id = x.Id, Name = x.FullName }).ToArray();

comboBoxClient.ValueMember = "Id";

comboBoxClient.DisplayMember = "Name";

}

public PurchaseForm(int purchaseId) : this()

{

var purchase = dbContext.Purchases.FirstOrDefault(x => x.Id == purchaseId);

currentPurchase = purchase;

if (currentPurchase != null)

{

// Заполнение элементов управления данными о покупке для изменения

comboBoxAbonement.SelectedItem = currentPurchase.AbonementId;

dateTimePickerDate.Value = currentPurchase.Date;

numericUpDownPrice.Value = currentPurchase.Price;

}

}

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

{

if (currentPurchase == null)

{

// Добавление новой покупки

var clientId = ((HelperForForms)comboBoxClient.SelectedItem).Id;

var abonementId = ((HelperForForms)comboBoxAbonement.SelectedItem).Id;

var date = dateTimePickerDate.Value;

var price = numericUpDownPrice.Value;

var newPurchase = new Purchase

{

ClientId = clientId,

AbonementId = abonementId,

Date = date,

Price = price

};

dbContext.Purchases.Add(newPurchase);

dbContext.SaveChanges();

MessageBox.Show("Новая покупка успешно добавлена!");

}

else

{

// Изменение существующей покупки

currentPurchase.ClientId = ((HelperForForms)comboBoxClient.SelectedItem).Id;

currentPurchase.AbonementId = ((HelperForForms)comboBoxAbonement.SelectedItem).Id;

currentPurchase.Date = dateTimePickerDate.Value;

currentPurchase.Price = numericUpDownPrice.Value;

dbContext.SaveChanges();

MessageBox.Show("Данные о покупке успешно обновлены!");

}

this.Close();

}

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

{

this.Close();

}

}

}

using DataBase.Context;

using DataBase.Entity;

using System;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach.EntityForm

{

public partial class TrainerSheduleForm : Form

{

private FitnessDbContext dbContext;

private TrainerShedule currentTrainerShedule;

public TrainerSheduleForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

PopulateComboBoxes(); // Заполняем комбобоксы при инициализации формы

}

public TrainerSheduleForm(int trainerSheduleId) : this()

{

currentTrainerShedule = dbContext.TrainersShedules.FirstOrDefault(t => t.Id == trainerSheduleId);

if (currentTrainerShedule != null)

{

// Заполнение элементов управления данными о расписании для изменения

comboBoxWorkout.SelectedValue = currentTrainerShedule.WorkoutId;

comboBoxGroup.SelectedValue = currentTrainerShedule.GroupId;

comboBoxDayOfWeek.SelectedItem = HelpData.DayOfWeekToRussia[currentTrainerShedule.DayOfWeek];

numTime.Value = currentTrainerShedule.Time;

}

}

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

{

if (currentTrainerShedule == null)

{

// Добавление нового расписания

var newTrainerShedule = new TrainerShedule

{

WorkoutId = ((HelperForForms)comboBoxWorkout.SelectedItem).Id,

GroupId = ((HelperForForms)comboBoxGroup.SelectedItem).Id,

DayOfWeek = HelpData.DayofWeekToSystem[comboBoxDayOfWeek.SelectedItem.ToString()],

Time = (int)numTime.Value

};

dbContext.TrainersShedules.Add(newTrainerShedule);

dbContext.SaveChanges();

MessageBox.Show("Новое расписание успешно добавлено!");

}

else

{

// Изменение существующего расписания

currentTrainerShedule.WorkoutId = ((HelperForForms)comboBoxWorkout.SelectedItem).Id;

currentTrainerShedule.GroupId = ((HelperForForms)comboBoxGroup.SelectedItem).Id;

currentTrainerShedule.DayOfWeek = HelpData.DayofWeekToSystem[comboBoxDayOfWeek.SelectedItem.ToString()];

currentTrainerShedule.Time = (int)numTime.Value;

dbContext.SaveChanges();

MessageBox.Show("Данные о расписании успешно обновлены!");

}

this.Close();

}

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

{

this.Close();

}

private void PopulateComboBoxes()

{

// Здесь добавьте код для заполнения комбобоксов значениями для WorkoutId, GroupId и DayOfWeek

// Например:

comboBoxWorkout.DataSource = dbContext.Workouts.Select(x => new HelperForForms { Id = x.Id, Name = x.Name }).ToList();

comboBoxWorkout.DisplayMember = "Name";

comboBoxWorkout.ValueMember = "Id";

comboBoxGroup.DataSource = dbContext.TrainersGroups.Select(x => new HelperForForms { Id = x.Id, Name = x.Name }).ToList();

comboBoxGroup.DisplayMember = "Name";

comboBoxGroup.ValueMember = "Id";

comboBoxDayOfWeek.DataSource = HelpData.RussiaDayOfWeek;

}

}

}

using DataBase.Context;

using DataBase.Entity;

using System;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach.EntityForm

{

public partial class WorkoutForm : Form

{

private Workout workout;

private FitnessDbContext dbContext;

public WorkoutForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

comboBoxGym.DataSource = dbContext.Gyms.Select(x => new HelperForForms { Id = x.Id, Name = x.Name }).ToArray();

comboBoxGym.ValueMember = "Id";

comboBoxGym.DisplayMember = "Name";

comboBoxDirection.DataSource = dbContext.Directions.Select(x => new HelperForForms { Id = x.Id, Name = x.Name }).ToArray();

comboBoxDirection.ValueMember = "Id";

comboBoxDirection.DisplayMember = "Name";

}

public WorkoutForm(int id) : this()

{

workout = dbContext.Workouts.FirstOrDefault(x => x.Id == id);

textBoxName.Text = workout.Name;

comboBoxGym.SelectedItem = workout.GymId;

comboBoxDirection.SelectedItem = workout.DirectionId;

numericUpDownDuration.Value = workout.Duration;

}

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

{

if (string.IsNullOrEmpty(textBoxName.Text))

{

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

return;

}

var gymId = ((HelperForForms)comboBoxGym.SelectedItem).Id;

var directionId = ((HelperForForms)comboBoxDirection.SelectedItem).Id;

var gymDirections = dbContext.Gyms

.Include(x => x.TypeOfGym)

.Include(y => y.TypeOfGym.Directions)

.FirstOrDefault(x => x.Id == gymId)

.TypeOfGym.Directions

.Select(x => x.Id)

.ToArray();

if (!gymDirections.Contains(directionId))

{

MessageBox.Show("Пожалуйста, выберите зал, который подходит для направления.");

return;

}

if (workout == null)

{

var newWorkout = new Workout

{

Name = textBoxName.Text,

GymId = gymId,

DirectionId = directionId,

Duration = (int)numericUpDownDuration.Value

};

dbContext.Workouts.Add(newWorkout);

dbContext.SaveChanges();

MessageBox.Show("Новая тренировка успешно добавлена!");

}

else

{

workout.Name = textBoxName.Text;

workout.GymId = gymId;

workout.DirectionId = directionId;

workout.Duration = (int)numericUpDownDuration.Value;

dbContext.SaveChanges();

MessageBox.Show("Данные о тренировке успешно обновлены!");

}

Close();

}

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

{

Close();

}

}

}

using DataBase.Context;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Wordprocessing;

using DocumentFormat.OpenXml;

using FitnessKursach.EntityForm;

using System;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Windows.Forms;

using System.Diagnostics;

using System.Globalization;

namespace FitnessKursach

{

public partial class ClientWorkForm : Form

{

private FitnessDbContext dbContext;

public ClientWorkForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

if (HelpData.UserRole != "admin")

{

tabControl.TabPages.Remove(tabPageClients);

}

}

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

{

// Загрузка данных о клиентах и покупках при загрузке формы

LoadClientsData();

LoadPurchasesData();

}

private void LoadClientsData()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

try

{

// Запрос к базе данных для получения списка клиентов

var clients = dbContext.Clients.ToList();

// Создание DataTable для отображения данных о клиентах

DataTable clientsDataTable = new DataTable();

clientsDataTable.Columns.Add("ID", typeof(int));

clientsDataTable.Columns.Add("Скидка", typeof(int));

clientsDataTable.Columns.Add("Посещения", typeof(int));

clientsDataTable.Columns.Add("Полное имя", typeof(string));

// Заполнение DataTable данными о клиентах

foreach (var client in clients)

{

clientsDataTable.Rows.Add(client.Id, client.Discount, client.Visit, client.FullName);

}

// Привязка DataTable к DataGridView для отображения данных о клиентах

dataGridViewClients.DataSource = clientsDataTable;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка загрузки данных о клиентах: " + ex.Message);

}

}

private void LoadPurchasesData()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

try

{

// Запрос к базе данных для получения списка покупок

var purchases = dbContext.Purchases

.Include(x => x.Client)

.Include(x => x.Abonement)

.ToList();

// Создание DataTable для отображения данных о покупках

DataTable purchasesDataTable = new DataTable();

purchasesDataTable.Columns.Add("ID", typeof(int));

purchasesDataTable.Columns.Add("Покупатель", typeof(string));

purchasesDataTable.Columns.Add("Абонемент", typeof(string));

purchasesDataTable.Columns.Add("Дата", typeof(DateTime));

purchasesDataTable.Columns.Add("Цена", typeof(decimal));

// Заполнение DataTable данными о покупках

foreach (var purchase in purchases)

{

purchasesDataTable.Rows.Add(purchase.Id, purchase.Client.FullName, purchase.Abonement.Name, purchase.Date, purchase.Price);

}

// Привязка DataTable к DataGridView для отображения данных о покупках

dataGridViewPurchases.DataSource = purchasesDataTable;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка загрузки данных о покупках: " + ex.Message);

}

}

private void ClientsForm\_FormClosing(object sender, FormClosingEventArgs e)

{

// Закрытие контекста базы данных при закрытии формы

dbContext.Dispose();

}

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

{

if (tabControl.SelectedTab == tabPageClients)

{

var clientForm = new ClientForm();

clientForm.ShowDialog();

LoadClientsData(); // Обновление данных о клиентах после добавления нового клиента

}

else if (tabControl.SelectedTab == tabPagePurchases)

{

var purchaseForm = new PurchaseForm();

purchaseForm.ShowDialog();

LoadPurchasesData(); // Обновление данных о покупках после добавления новой покупки

}

}

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

{

if (tabControl.SelectedTab == tabPageClients)

{

if (dataGridViewClients.SelectedRows.Count == 0)

{

MessageBox.Show("Клиент не выбран!");

return;

}

if (dataGridViewClients.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите однго клиента!");

return;

}

int clientId = (int)dataGridViewClients.SelectedRows[0].Cells["ID"].Value;

var clientForm = new ClientForm(clientId);

clientForm.ShowDialog();

LoadClientsData(); // Обновление данных о клиентах после изменения

}

else if (tabControl.SelectedTab == tabPagePurchases)

{

if (dataGridViewPurchases.SelectedRows.Count == 0)

{

MessageBox.Show("Покупка не выбрана!");

return;

}

if (dataGridViewPurchases.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите одну покупку!");

return;

}

int purchaseId = (int)dataGridViewPurchases.SelectedRows[0].Cells["ID"].Value;

var purchaseForm = new PurchaseForm(purchaseId);

purchaseForm.ShowDialog();

LoadPurchasesData(); // Обновление данных о покупках после изменения

}

}

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

{

if (tabControl.SelectedTab == tabPageClients)

{

var clientsDataTable = (DataTable)dataGridViewClients.DataSource;

var dv = clientsDataTable.DefaultView;

dv.RowFilter = $"[Полное имя] LIKE '%{txtFilter.Text}%'";

dataGridViewClients.DataSource = dv.ToTable();

}

else if (tabControl.SelectedTab == tabPagePurchases)

{

var purchasesDataTable = (DataTable)dataGridViewPurchases.DataSource;

var dv = purchasesDataTable.DefaultView;

dv.RowFilter = $"[Покупатель] LIKE '%{txtFilter.Text}%'";

dataGridViewPurchases.DataSource = dv.ToTable();

}

}

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

{

if (tabControl.SelectedTab == tabPageClients)

{

// Удаление клиента

if (dataGridViewClients.SelectedRows.Count <= 0)

{

MessageBox.Show("Клиент не выбран!");

return;

}

for (int i = 0; i < dataGridViewClients.SelectedRows.Count; i++)

{

int clientId = (int)dataGridViewClients.SelectedRows[i].Cells["ID"].Value;

var clientToDelete = dbContext.Clients.FirstOrDefault(c => c.Id == clientId);

if (clientToDelete != null)

{

dbContext.Clients.Remove(clientToDelete);

dbContext.SaveChanges();

}

}

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

LoadClientsData(); // Обновление данных о клиентах после удаления

LoadPurchasesData(); // Обновление данных о покупках после удаления

}

else if (tabControl.SelectedTab == tabPagePurchases)

{

// Удаление покупки

if (dataGridViewPurchases.SelectedRows.Count <= 0)

{

MessageBox.Show("Покупка не выбрана!");

return;

}

for (int i = 0; i < dataGridViewPurchases.SelectedRows.Count; i++)

{

int purchaseId = (int)dataGridViewPurchases.SelectedRows[0].Cells["ID"].Value;

var purchaseToDelete = dbContext.Purchases.FirstOrDefault(p => p.Id == purchaseId);

if (purchaseToDelete != null)

{

dbContext.Purchases.Remove(purchaseToDelete);

dbContext.SaveChanges();

}

}

MessageBox.Show("Покупка успешно удалена!");

LoadPurchasesData(); // Обновление данных о покупках после удаления

}

}

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

{

if (tabControl.SelectedTab == tabPageClients)

{

LoadClientsData();

}

else if (tabControl.SelectedTab == tabPagePurchases)

{

LoadPurchasesData();

}

}

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

{

Close();

}

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

{

var sum = dbContext.Purchases.Sum(p => p.Price);

MessageBox.Show($"Сумма всех покупок: {sum}");

}

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

{

var purchasesByMont = dbContext.Purchases

.GroupBy(

p => p.Date.Month,

x => x.Price,

(key, price) => new { Month = key, Sum = price.Sum() })

.ToList();

string fileName = "Statistics.docx";

using (WordprocessingDocument wordDocument = WordprocessingDocument.Create(fileName, WordprocessingDocumentType.Document))

{

// Создание основного содержимого документа

MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = mainPart.Document.AppendChild(new Body());

// Добавление заголовка

Paragraph heading = new Paragraph(new Run(new Text("Статистика покупок")));

body.AppendChild(heading);

// Добавление таблицы с данными о покупках

Table table = new Table();

TableProperties tableProperties = new TableProperties(

new TableBorders(

new TopBorder { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new BottomBorder { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new LeftBorder { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new RightBorder { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new InsideHorizontalBorder { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new InsideVerticalBorder { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 }

)

);

table.AppendChild(tableProperties);

// Заголовки столбцов таблицы

TableRow headerRow = new TableRow();

headerRow.AppendChild(CreateTableCell("Месяц"));

headerRow.AppendChild(CreateTableCell("Сумма"));

table.AppendChild(headerRow);

// Добавление данных о покупках

foreach (var purchase in purchasesByMont)

{

TableRow row = new TableRow();

row.AppendChild(CreateTableCell(new DateTime(2015, purchase.Month, 1).ToString("MMMM", CultureInfo.CreateSpecificCulture("ru"))));

row.AppendChild(CreateTableCell(purchase.Sum.ToString()));

table.AppendChild(row);

}

body.AppendChild(table);

}

Process.Start(fileName);

MessageBox.Show("Статистика успешно создана и сохранена в файле " + fileName, "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private TableCell CreateTableCell(string text)

{

return new TableCell(new Paragraph(new Run(new Text(text))));

}

}

}

using DataBase.Context;

using DocumentFormat.OpenXml;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Wordprocessing;

using FitnessKursach.EntityForm;

using System;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach

{

public partial class EmployeeControlForm : Form

{

private FitnessDbContext dbContext;

public EmployeeControlForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

if (HelpData.UserRole != "admin")

{

tabControl.TabPages.Remove(tabPageGyms);

}

if (HelpData.UserRole != "admin" && HelpData.UserRole != "trainer")

{

tabControl.TabPages.Remove(tabPageTrainerSchedules);

tabControl.TabPages.Remove(tabPageGroups);

}

if (HelpData.UserRole != "admin" && HelpData.UserRole != "accountant")

{

button2.Visible = false;

}

if (tabControl.SelectedTab == tabPageEmployees && HelpData.UserRole != "admin")

{

btnDelete.Visible = false;

AddBtn.Visible = false;

btnUpdate.Visible = false;

}

if (tabControl.SelectedTab != tabPageEmployees )

{

button2.Visible = false;

}

}

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

{

LoadGymsData();

LoadGroupsData();

LoadTrainerSchedulesData();

LoadEmployeesData();

}

private void LoadGymsData()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

var query = dbContext.Gyms

.Include(x => x.TypeOfGym)

.ToList();

var dataTable = new DataTable();

dataTable.Columns.Add("Id", typeof(int));

dataTable.Columns.Add("Количество посещений", typeof(int));

dataTable.Columns.Add("Название", typeof(string));

dataTable.Columns.Add("Тип зала", typeof(string));

foreach (var item in query)

{

dataTable.Rows.Add(item.Id, item.Visits, item.Name, item.TypeOfGym.Name);

}

dataGridViewGyms.DataSource = dataTable;

}

private void LoadGroupsData()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

var query = dbContext.TrainersGroups

.Include(x => x.Trainer)

.Include(x => x.Direction)

.ToList();

var dataTable = new DataTable();

dataTable.Columns.Add("Id", typeof(int));

dataTable.Columns.Add("Название", typeof(string));

dataTable.Columns.Add("Тренер", typeof(string));

dataTable.Columns.Add("Направление", typeof(string));

dataTable.Columns.Add("Вид группы", typeof(string));

foreach (var item in query)

{

dataTable.Rows.Add(item.Id, item.Name, item.Trainer.FullName, item.Direction.Name, item.GroupView);

}

dataGridViewGroups.DataSource = dataTable;

}

private void LoadTrainerSchedulesData()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

var query = dbContext.TrainersShedules

.Include(x => x.Workout)

.Include(x => x.Group)

.Include(x => x.Group.Trainer)

.ToList();

var dataTable = new DataTable();

dataTable.Columns.Add("Id", typeof(int));

dataTable.Columns.Add("Тренер", typeof(string));

dataTable.Columns.Add("Тренировка", typeof(string));

dataTable.Columns.Add("Группа", typeof(string));

dataTable.Columns.Add("Время", typeof(int));

dataTable.Columns.Add("День недели", typeof(string));

foreach (var item in query)

{

dataTable.Rows.Add(item.Id, item.Group.Trainer.FullName, item.Workout.Name, item.Group.Name, item.Time, HelpData.DayOfWeekToRussia[item.DayOfWeek]);

}

dataGridViewTrainerSchedules.DataSource = dataTable;

}

private void LoadEmployeesData()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

var query = dbContext.Employees.ToList();

var dataTable = new DataTable();

dataTable.Columns.Add("Id", typeof(int));

dataTable.Columns.Add("Полное имя", typeof(string));

dataTable.Columns.Add("Должность", typeof(string));

dataTable.Columns.Add("Зарплата", typeof(decimal));

foreach (var item in query)

{

dataTable.Rows.Add(item.Id, item.FullName, item.JobTitle, item.Wage);

}

dataGridViewEmployees.DataSource = dataTable;

}

private void EmployeeControlForm\_FormClosing(object sender, FormClosingEventArgs e)

{

dbContext.Dispose();

}

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

{

if (tabControl.SelectedTab == tabPageGyms)

{

var form = new GymForm();

form.ShowDialog();

LoadGymsData();

}

else if (tabControl.SelectedTab == tabPageGroups)

{

var form = new GroupForm();

form.ShowDialog();

LoadGroupsData();

}

else if (tabControl.SelectedTab == tabPageTrainerSchedules)

{

var form = new TrainerSheduleForm();

form.ShowDialog();

LoadTrainerSchedulesData();

}

else if (tabControl.SelectedTab == tabPageEmployees)

{

var form = new EmployeeForm();

form.ShowDialog();

LoadEmployeesData();

}

}

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

{

if (tabControl.SelectedTab == tabPageGyms)

{

if (dataGridViewGyms.SelectedRows.Count == 0)

{

MessageBox.Show("Зал не выбран!");

return;

}

if (dataGridViewGyms.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите один зал!");

return;

}

int id = (int)dataGridViewGyms.SelectedRows[0].Cells["Id"].Value;

var form = new GymForm(id);

form.ShowDialog();

LoadGymsData();

}

else if (tabControl.SelectedTab == tabPageGroups)

{

if (dataGridViewGroups.SelectedRows.Count == 0)

{

MessageBox.Show("Группа не выбрана!");

return;

}

if (dataGridViewGroups.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите одну группу!");

return;

}

int id = (int)dataGridViewGroups.SelectedRows[0].Cells["Id"].Value;

var form = new GroupForm(id);

form.ShowDialog();

LoadGroupsData();

}

else if (tabControl.SelectedTab == tabPageTrainerSchedules)

{

if (dataGridViewTrainerSchedules.SelectedRows.Count == 0)

{

MessageBox.Show("Расписание не выбрано!");

return;

}

if (dataGridViewTrainerSchedules.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите одно расписание!");

return;

}

int id = (int)dataGridViewTrainerSchedules.SelectedRows[0].Cells["Id"].Value;

var form = new TrainerSheduleForm(id);

form.ShowDialog();

LoadTrainerSchedulesData();

}

else if (tabControl.SelectedTab == tabPageEmployees)

{

if (dataGridViewEmployees.SelectedRows.Count == 0)

{

MessageBox.Show("Сотрудник не выбран!");

return;

}

if (dataGridViewEmployees.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите однго сотрудника!");

return;

}

int id = (int)dataGridViewEmployees.SelectedRows[0].Cells["Id"].Value;

var form = new EmployeeForm(id);

form.ShowDialog();

LoadEmployeesData();

}

}

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

{

if (tabControl.SelectedTab == tabPageGyms)

{

var dataTable = (DataTable)dataGridViewGyms.DataSource;

var dv = dataTable.DefaultView;

dv.RowFilter = $"[Название] LIKE '%{txtFilter.Text}%'";

dataGridViewGyms.DataSource = dv.ToTable();

}

else if (tabControl.SelectedTab == tabPageGroups)

{

var dataTable = (DataTable)dataGridViewGroups.DataSource;

var dv = dataTable.DefaultView;

dv.RowFilter = $"[Название] LIKE '%{txtFilter.Text}%'";

dataGridViewGroups.DataSource = dv.ToTable();

}

else if (tabControl.SelectedTab == tabPageTrainerSchedules)

{

var dataTable = (DataTable)dataGridViewTrainerSchedules.DataSource;

var dv = dataTable.DefaultView;

dv.RowFilter = $"[Тренер] LIKE '%{txtFilter.Text}%'";

dataGridViewTrainerSchedules.DataSource = dv.ToTable();

}

else if (tabControl.SelectedTab == tabPageEmployees)

{

var dataTable = (DataTable)dataGridViewEmployees.DataSource;

var dv = dataTable.DefaultView;

dv.RowFilter = $"[Полное имя] LIKE '%{txtFilter.Text}%'";

dataGridViewEmployees.DataSource = dv.ToTable();

}

}

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

{

if (tabControl.SelectedTab == tabPageGyms)

{

if (dataGridViewGyms.SelectedRows.Count <= 0)

{

MessageBox.Show("Зал не выбран!");

return;

}

for (int i = 0; i < dataGridViewGyms.SelectedRows.Count; i++)

{

int clientId = (int)dataGridViewGyms.SelectedRows[i].Cells["ID"].Value;

var clientToDelete = dbContext.Clients.FirstOrDefault(c => c.Id == clientId);

if (clientToDelete != null)

{

dbContext.Clients.Remove(clientToDelete);

dbContext.SaveChanges();

}

}

MessageBox.Show("Зал успешно удален!");

LoadGymsData();

}

else if (tabControl.SelectedTab == tabPageGroups)

{

if (dataGridViewGroups.SelectedRows.Count <= 0)

{

MessageBox.Show("Группа не выбрана!");

return;

}

for (int i = 0; i < dataGridViewGroups.SelectedRows.Count; i++)

{

int purchaseId = (int)dataGridViewGroups.SelectedRows[0].Cells["ID"].Value;

var purchaseToDelete = dbContext.Purchases.FirstOrDefault(p => p.Id == purchaseId);

if (purchaseToDelete != null)

{

dbContext.Purchases.Remove(purchaseToDelete);

dbContext.SaveChanges();

}

}

MessageBox.Show("Группа успешно удалена!");

LoadGroupsData();

}

else if (tabControl.SelectedTab == tabPageTrainerSchedules)

{

if (dataGridViewTrainerSchedules.SelectedRows.Count <= 0)

{

MessageBox.Show("Раснисание не выбрано!");

return;

}

for (int i = 0; i < dataGridViewTrainerSchedules.SelectedRows.Count; i++)

{

int purchaseId = (int)dataGridViewTrainerSchedules.SelectedRows[0].Cells["ID"].Value;

var purchaseToDelete = dbContext.Purchases.FirstOrDefault(p => p.Id == purchaseId);

if (purchaseToDelete != null)

{

dbContext.Purchases.Remove(purchaseToDelete);

dbContext.SaveChanges();

}

}

MessageBox.Show("Раснисание успешно удалено!");

LoadTrainerSchedulesData();

}

else if (tabControl.SelectedTab == tabPageEmployees)

{

if (dataGridViewEmployees.SelectedRows.Count <= 0)

{

MessageBox.Show("Сотрудник не выбран!");

return;

}

for (int i = 0; i < dataGridViewEmployees.SelectedRows.Count; i++)

{

int purchaseId = (int)dataGridViewEmployees.SelectedRows[0].Cells["ID"].Value;

var purchaseToDelete = dbContext.Purchases.FirstOrDefault(p => p.Id == purchaseId);

if (purchaseToDelete != null)

{

dbContext.Purchases.Remove(purchaseToDelete);

dbContext.SaveChanges();

}

}

MessageBox.Show("Сотрудник успешно удален!");

LoadEmployeesData();

}

}

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

{

if (tabControl.SelectedTab == tabPageGyms)

{

LoadGymsData();

}

else if (tabControl.SelectedTab == tabPageGroups)

{

LoadGroupsData();

}

else if (tabControl.SelectedTab == tabPageTrainerSchedules)

{

LoadTrainerSchedulesData();

}

else if (tabControl.SelectedTab == tabPageEmployees)

{

LoadEmployeesData();

}

}

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

{

Close();

}

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

{

var sum = dbContext.Employees.Sum(p => p.Wage);

MessageBox.Show($"Сумма всех зарплат: {sum}");

}

private void tabControl\_SelectedIndexChanged(object sender, EventArgs e)

{

if (tabControl.SelectedTab == tabPageEmployees && HelpData.UserRole != "admin")

{

btnDelete.Visible = false;

AddBtn.Visible = false;

btnUpdate.Visible = false;

}

else

{

btnDelete.Visible = true;

AddBtn.Visible = true;

btnUpdate.Visible = true;

}

if (tabControl.SelectedTab == tabPageEmployees)

{

button2.Visible = true;

}

else

{

button2.Visible = false;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FitnessKursach

{

public static class HelpData

{

public static string UserRole { get; set; }

public static string[] RussiaRoles = new string[] { "Тренер", "Бухгалтер", "Администратор" };

public static string[] RussiaDayOfWeek = new string[] { "Понедельник", "Вторник", "Среда", "Четверг", "Пятница", "Суббота", "Воскресенье" };

public static Dictionary<string, string> RolesToSystem { get; } = new Dictionary<string, string>()

{

{"Тренер", "trainer" },

{"Бухгалтер", "accountant" },

{"Администратор", "admin" },

};

public static Dictionary<string, string> RolesToRussia { get; } = new Dictionary<string, string>()

{

{ "trainer", "Тренер" },

{ "accountant", "Бухгалтер" },

{ "admin", "Администратор" },

};

public static Dictionary<DayOfWeek, string> DayOfWeekToRussia { get; } = new Dictionary<DayOfWeek, string>()

{

{ DayOfWeek.Sunday, "Воскресенье" },

{ DayOfWeek.Monday, "Понедельник" },

{ DayOfWeek.Tuesday, "Вторник" },

{ DayOfWeek.Wednesday, "Среда" },

{ DayOfWeek.Thursday, "Четверг" },

{ DayOfWeek.Friday, "Пятница" },

{ DayOfWeek.Saturday, "Суббота" }

};

public static Dictionary<string, DayOfWeek> DayofWeekToSystem { get; } = new Dictionary<string, DayOfWeek>()

{

{ "Воскресенье", DayOfWeek.Sunday },

{ "Понедельник" , DayOfWeek.Monday },

{ "Вторник" , DayOfWeek.Tuesday },

{ "Среда" , DayOfWeek.Wednesday },

{ "Четверг" , DayOfWeek.Thursday },

{ "Пятница" , DayOfWeek.Friday },

{ "Суббота" , DayOfWeek.Saturday }

};

}

}

using DataBase.Context;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach

{

public partial class Login : Form

{

private FitnessDbContext dbContext;

public Login()

{

InitializeComponent();

// Инициализация контекста базы данных

dbContext = new FitnessDbContext();

}

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

{

string login = txtUsername.Text;

string password = txtPassword.Text;

// Поиск пользователя в базе данных

var user = dbContext.Employees.FirstOrDefault(emp => emp.Login == login && emp.Password == password);

if (user != null)

{

HelpData.UserRole = user.Role;

Hide();

var form = new MainForm();

form.ShowDialog();

Close();

}

else

{

MessageBox.Show("Неверное имя пользователя или пароль!");

}

}

private void LoginForm\_FormClosing(object sender, FormClosingEventArgs e)

{

// Закрытие контекста базы данных при закрытии формы

dbContext.Dispose();

}

}

}

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 FitnessKursach

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

if (HelpData.UserRole != "admin" && HelpData.UserRole != "accountant")

{

ClientWork.Visible = false;

}

}

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

{

Hide();

var form = new ClientWorkForm();

form.ShowDialog();

Show();

}

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

{

Hide();

var form = new EmployeeControlForm();

form.ShowDialog();

Show();

}

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

{

Hide();

var form = new ServiceForm();

form.ShowDialog();

Show();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FitnessKursach

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Login());

}

}

}

using DataBase.Context;

using FitnessKursach.EntityForm;

using System;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Windows.Forms;

namespace FitnessKursach

{

public partial class ServiceForm : Form

{

private FitnessDbContext dbContext;

public ServiceForm()

{

InitializeComponent();

dbContext = new FitnessDbContext();

if (HelpData.UserRole != "admin" && HelpData.UserRole != "trainer")

{

tabControl.TabPages.Remove(tabPageExercises);

tabControl.TabPages.Remove(tabPageWorkouts);

}

}

private void ServiceForm\_FormClosing(object sender, FormClosingEventArgs e)

{

dbContext.Dispose();

}

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

{

if (tabControl.SelectedTab == tabPageAbonement)

{

var form = new AbonementForm();

form.ShowDialog();

LoadAbonement();

}

else if (tabControl.SelectedTab == tabPageExercises)

{

var form = new ExerciseForm();

form.ShowDialog();

LoadExercises();

}

else if (tabControl.SelectedTab == tabPageWorkouts)

{

var form = new WorkoutForm();

form.ShowDialog();

LoadWorkouts();

}

}

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

{

if (tabControl.SelectedTab == tabPageAbonement)

{

if (dataGridViewAbonement.SelectedRows.Count == 0)

{

MessageBox.Show("Абонемент не выбран!");

return;

}

if (dataGridViewAbonement.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите однин абонемент!");

return;

}

int id = (int)dataGridViewAbonement.SelectedRows[0].Cells["Id"].Value;

var form = new AbonementForm(id);

form.ShowDialog();

LoadAbonement();

}

else if (tabControl.SelectedTab == tabPageExercises)

{

if (dataGridViewExercises.SelectedRows.Count == 0)

{

MessageBox.Show("Упражение не выбрана!");

return;

}

if (dataGridViewExercises.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите одно упражнение!");

return;

}

int id = (int)dataGridViewExercises.SelectedRows[0].Cells["Id"].Value;

var form = new ExerciseForm(id);

form.ShowDialog();

LoadExercises();

}

else if (tabControl.SelectedTab == tabPageWorkouts)

{

if (dataGridViewExercises.SelectedRows.Count == 0)

{

MessageBox.Show("Тренировка не выбрана!");

return;

}

if (dataGridViewExercises.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите одну тренировку!");

return;

}

int id = (int)dataGridViewExercises.SelectedRows[0].Cells["Id"].Value;

var form = new WorkoutForm(id);

form.ShowDialog();

LoadWorkouts(); // Обновление данных о покупках после изменения

}

}

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

{

if (tabControl.SelectedTab == tabPageAbonement)

{

var clientsDataTable = (DataTable)dataGridViewAbonement.DataSource;

var dv = clientsDataTable.DefaultView;

dv.RowFilter = $"[Название] LIKE '%{txtFilter.Text}%'";

dataGridViewAbonement.DataSource = dv.ToTable();

}

else if (tabControl.SelectedTab == tabPageExercises)

{

var purchasesDataTable = (DataTable)dataGridViewExercises.DataSource;

var dv = purchasesDataTable.DefaultView;

dv.RowFilter = $"[Название] LIKE '%{txtFilter.Text}%'";

dataGridViewExercises.DataSource = dv.ToTable();

}

else if (tabControl.SelectedTab == tabPageWorkouts)

{

var purchasesDataTable = (DataTable)dataGridViewExercises.DataSource;

var dv = purchasesDataTable.DefaultView;

dv.RowFilter = $"[Название] LIKE '%{txtFilter.Text}%'";

dataGridViewExercises.DataSource = dv.ToTable();

}

}

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

{

if (tabControl.SelectedTab == tabPageAbonement)

{

if (dataGridViewAbonement.SelectedRows.Count <= 0)

{

MessageBox.Show("Абонемент не выбран!");

return;

}

for (int i = 0; i < dataGridViewAbonement.SelectedRows.Count; i++)

{

int id = (int)dataGridViewAbonement.SelectedRows[i].Cells["Id"].Value;

var abonements = dbContext.Abonements.FirstOrDefault(c => c.Id == id);

if (abonements != null)

{

dbContext.Abonements.Remove(abonements);

dbContext.SaveChanges();

}

}

MessageBox.Show("Абонемент успешно удален!");

LoadAbonement();

}

else if (tabControl.SelectedTab == tabPageExercises)

{

if (dataGridViewExercises.SelectedRows.Count <= 0)

{

MessageBox.Show("Упражнение не выбрано!");

return;

}

for (int i = 0; i < dataGridViewExercises.SelectedRows.Count; i++)

{

int id = (int)dataGridViewExercises.SelectedRows[0].Cells["Id"].Value;

var exercises = dbContext.Exercises.FirstOrDefault(p => p.Id == id);

if (exercises != null)

{

dbContext.Exercises.Remove(exercises);

dbContext.SaveChanges();

}

}

MessageBox.Show("Упражнение успешно удалено!");

LoadExercises();

}

else if (tabControl.SelectedTab == tabPageWorkouts)

{

if (dataGridViewWorkouts.SelectedRows.Count <= 0)

{

MessageBox.Show("Тренировка не выбрана!");

return;

}

for (int i = 0; i < dataGridViewWorkouts.SelectedRows.Count; i++)

{

int id = (int)dataGridViewWorkouts.SelectedRows[0].Cells["Id"].Value;

var workout = dbContext.Workouts.FirstOrDefault(p => p.Id == id);

if (workout != null)

{

dbContext.Workouts.Remove(workout);

dbContext.SaveChanges();

}

}

MessageBox.Show("Тренировка успешно удалена!");

LoadWorkouts();

}

}

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

{

if (tabControl.SelectedTab == tabPageAbonement)

{

LoadAbonement();

}

else if (tabControl.SelectedTab == tabPageExercises)

{

LoadExercises();

}

else if (tabControl.SelectedTab == tabPageWorkouts)

{

LoadWorkouts();

}

}

private void LoadExercises()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

var exercises = dbContext.Exercises.ToList();

DataTable dt = new DataTable();

dt.Columns.Add("Id", typeof(int));

dt.Columns.Add("Название");

dt.Columns.Add("Группа мышц");

foreach (var exercise in exercises)

{

dt.Rows.Add(exercise.Id, exercise.Name, exercise.MuscleGroup);

}

dataGridViewExercises.DataSource = dt;

}

private void LoadWorkouts()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

var workouts = dbContext.Workouts

.Include(x => x.Gym)

.Include(x => x.Direction)

.ToList();

DataTable dt = new DataTable();

dt.Columns.Add("Id", typeof(int));

dt.Columns.Add("Название");

dt.Columns.Add("Зал");

dt.Columns.Add("Направление");

dt.Columns.Add("Длительность");

foreach (var workout in workouts)

{

dt.Rows.Add(workout.Id, workout.Name, workout.Gym.Name, workout.Direction.Name, workout.Duration);

}

dataGridViewWorkouts.DataSource = dt;

}

private void LoadAbonement()

{

dbContext.Dispose();

dbContext = new FitnessDbContext();

var subscriptions = dbContext.Abonements.ToList();

DataTable dt = new DataTable();

dt.Columns.Add("Id", typeof(int));

dt.Columns.Add("Название");

dt.Columns.Add("Цена");

dt.Columns.Add("Количество посещений");

foreach (var subscription in subscriptions)

{

dt.Rows.Add(subscription.Id, subscription.Name, subscription.Price, subscription.NumberOfVisits);

}

dataGridViewAbonement.DataSource = dt;

}

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

{

Close();

}

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

{

LoadAbonement();

LoadExercises();

LoadWorkouts();

}

**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 42919/2 группы

Сметанникова Анастасия Алексеевна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» 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 «05» октября 2024 г.