**Оглавление**

[**ВВЕДЕНИЕ** 2](#_Toc196437408)

[**ГЛАВА 1. ПРЕДМЕТНАЯ ОБЛАСТЬ** 3](#_Toc196437409)

[**1.1** **Исходная постановка задачи** 3](#_Toc196437410)

[**1.2** **Среда разработки** 4](#_Toc196437411)

[**1.3** **Техническое задание** 4](#_Toc196437412)

[**1.4** **Руководство по стилю** 9](#_Toc196437413)

[**1.5** **ER – Диаграмма** 10](#_Toc196437414)

[**1.6** **UML – Диаграммы** 10](#_Toc196437415)

[**1.6.1** **UML – Диаграмма прецендентов** 10](#_Toc196437416)

[**1.6.2** **UML – Диаграмма деятельности** 10](#_Toc196437417)

[**1.6.3** **UML – Диаграмма взаимодействия** 11](#_Toc196437418)

[**1.6.4 UML – Диаграмма классов** 11](#_Toc196437419)

[**1.6.5 Бизнес – процессы** 12](#_Toc196437420)

[**ГЛАВА 2. РАЗРАБОТКА БАЗЫ ДАННЫХ** 14](#_Toc196437421)

[**2.1 Разработка логической модели данных** 14](#_Toc196437422)

[**2.2 Добавление таблиц к базе данных** 14](#_Toc196437423)

[**2.3 Связи на диаграмме базе данных** 15](#_Toc196437424)

[**2.4 Определение ограничений налагаемые на данные** 15](#_Toc196437425)

[**2.5 Разработка физической структуры данных** 15](#_Toc196437426)

[**2.6 Создание БД и таблиц** 19](#_Toc196437427)

[**2.7 Данные в таблицах** 21](#_Toc196437428)

[**2.8 Создание запросов** 25](#_Toc196437429)

[**ГЛАВА 3. РАЗРАБОТКА ПРИЛОЖЕНИЯ** 28](#_Toc196437430)

[**3.1 Разработка пользовательского интерфейса** 28](#_Toc196437431)

[**3.2 Разработка библиотек класса** 34](#_Toc196437432)

[**3.3 Листинг программы** 38](#_Toc196437433)

[**ЗАКЛЮЧЕНИЕ** 73](#_Toc196437434)

[**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ** 74](#_Toc196437435)

# **ВВЕДЕНИЕ**

Объект

«Фитнес клуб - Красивые&Бодрые»

Актуальность

В современном обществе забота о здоровье и физической форме приобретает всё большую значимость, становясь неотъемлемой частью образа жизни. Увеличение сидячего образа жизни, неблагоприятная экологическая обстановка и повышенный уровень стресса способствуют росту заболеваний, связанных с недостаточной физической активностью. В связи с этим, открытие современного и доступного фитнес-клуба, ориентированного на индивидуальные потребности клиентов и предлагающего широкий спектр услуг, является крайне актуальным. Такой проект позволит удовлетворить растущий спрос на качественные фитнес-услуги, способствуя повышению уровня здоровья населения, формированию культуры здорового образа жизни и, как следствие, улучшению качества жизни в целом. Кроме того, проект "Фитнес-клуб" обладает значительным потенциалом для успешного развития бизнеса, обеспечивая стабильный доход и создавая новые рабочие места.

# **ГЛАВА 1. ПРЕДМЕТНАЯ ОБЛАСТЬ**

* 1. **Исходная постановка задачи**

Цель

Разработать приложение «Фитнес клуб – Красивые & Бодрые»

Задачи

1. Описать функциональные возможности приложения «Фитнес клуб К&Б»

2. Проанализировать удобство использования приложения для различных категории пользователей.

3. Оценить влияние приложения формирования предпочтений и развития работоспособности фитнес клуба.

4. Разработать рекомендации по улучшению приложения и его дальнейшему развитию.

Исходные данные

1. Клиенты
2. Абонементы
3. Абонементы клиентов
4. Тренеры
5. Групповые занятия
6. Расписание занятий
7. Регистрация на занятия
8. Платежи
9. Оборудование
10. Оборудование для занятий

## **Среда разработки**

ОС – Windows 7/10/11

Среда разработки и язык программирования – Visual Studio, C#, СУБД: SQL Server.

Требования к среде выполнения

1. Поддержка современных версий операционных систем;
2. Обеспечение плавной работы приложения;
3. Поддержка хранения структурированных данных;
4. Возможность интеграции с внешними сервисами;
   1. **Техническое задание**

УТВЕРЖДЕНО

-022А 01.М-ЛУ к Договору № 2 от 02.04.06

Разработка информационной системы «Фитнес Клуб»

Техническое задание

02068120.508100.027-02 2А 01.М

Листов 5

2024

**ОГЛАВЛЕНИЕ**

1.ВВЕДЕНИЕ 6

2.ВХОДНЫЕ И ИСХОДНЫЕ ДАННЫЕ 7

3.АЛГОРИТМ РАБОТЫ ПРОГРАММЫ 8

**ВВЕДЕНИЕ**

Информационная система «Фитнес клуб К&Б» – объект разработки, предназначенный для автоматизации производственных и управленческих процессов фитнес клуба. Система обеспечивает учет абонементов и клиентов, планирование занятий и формирование аналитической отчетности. Приложение применимо на персональных компьютерах с ОС Windows 10/11

**2.ВХОДНЫЕ И ИСХОДНЫЕ ДАННЫЕ**

Входные данные содержат в себе:

- Данные о клиентах.

- Данные о абонементах.

- Данные о тренерах.

- Данные о платежах.

- Данные оборудования.

- Данные о групповых занятиях.

- Данные о расписаний занятий.

Исходные данные:

- Отзывы клиентов.

- Данные о приобретенных абонементах.

- Данные о записи клиентов на занятия.

**3.АЛГОРИТМ РАБОТЫ ПРОГРАММЫ**

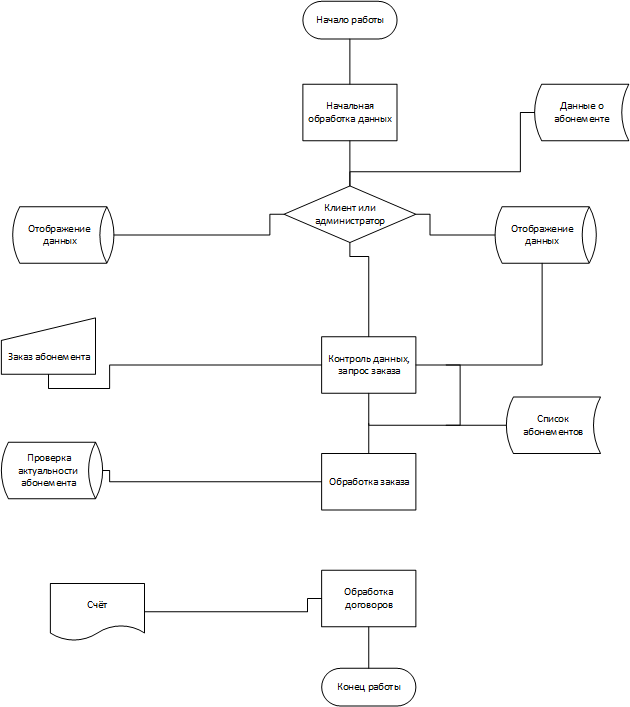


Рис. 1 Блок-схема (алгоритм программы)

* 1. **Руководство по стилю**

Общие требования:

Не допускать орфографических и грамматических ошибок.

Размер окон (форм) и кнопок

Титульная форма, форма авторизации, администратора и пользователя размер – 515x350, формы редактирования, просмотра и табличные формы – 850x450, Основной размер кнопок: 75x23 (По умолчанию), размер может зависеть от содержания. (Content)

Использование логотипа

Логотип используется в каждой форме слева в верхнем углу экрана, также используется как икона.

Шрифт

Шрифт Times New Roman. Размер шрифтов

14 Пт.

Персонализация элементов приложения

|  |  |
| --- | --- |
| 11#5FCEF7 | Основной/Акцентный |
| 11#00c0c0 | Дополнительный/Дочерний |

## **ER – Диаграмма**

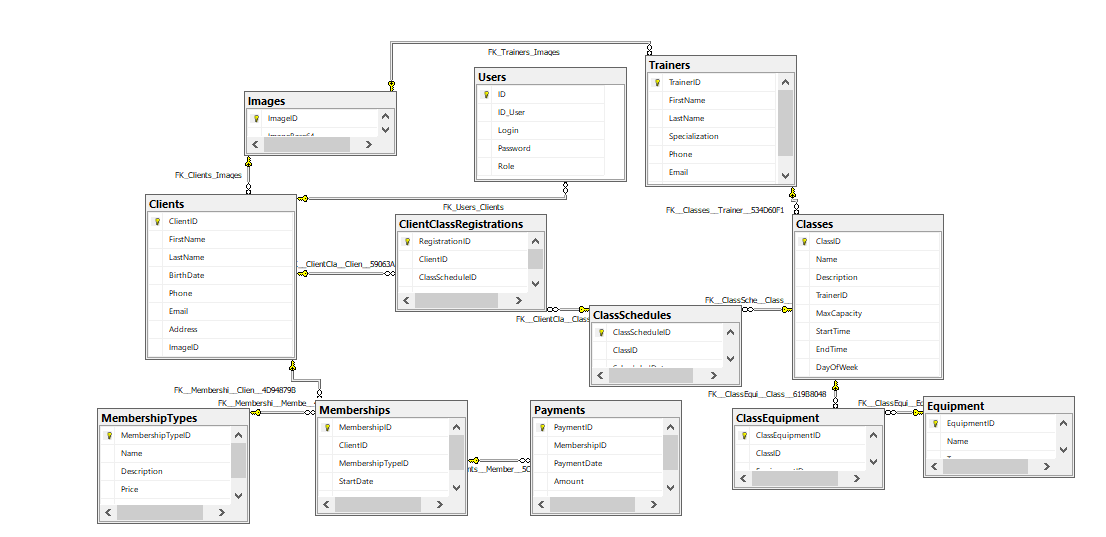


Рис. 2 ER – Диаграмма

* 1. **UML – Диаграммы**

### **UML – Диаграмма прецендентов**

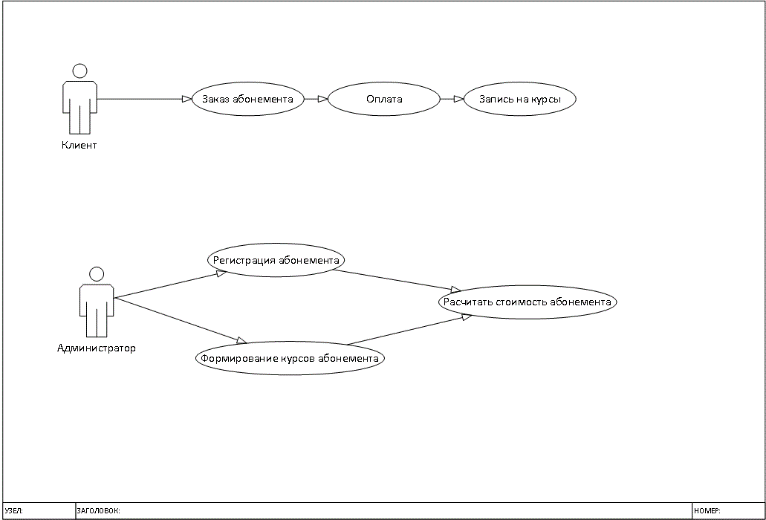


Рис. 3 UML – Диаграмма прецендентов

* + 1. **UML – Диаграмма деятельности**

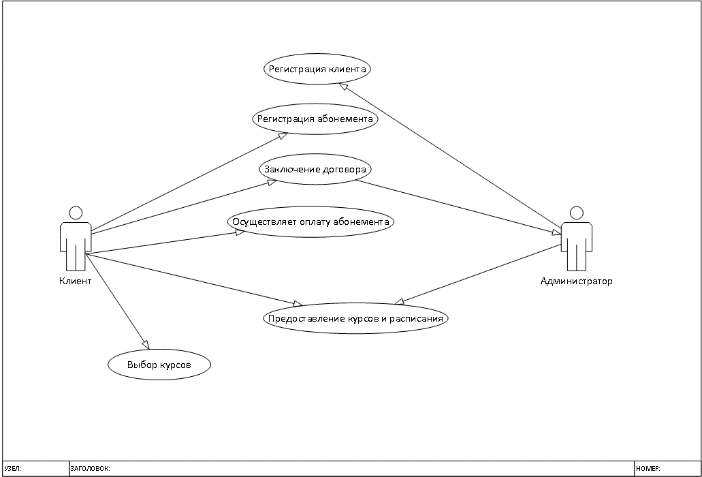


Рис. 4 UML – Диаграмма деятельности

* + 1. **UML – Диаграмма взаимодействия**

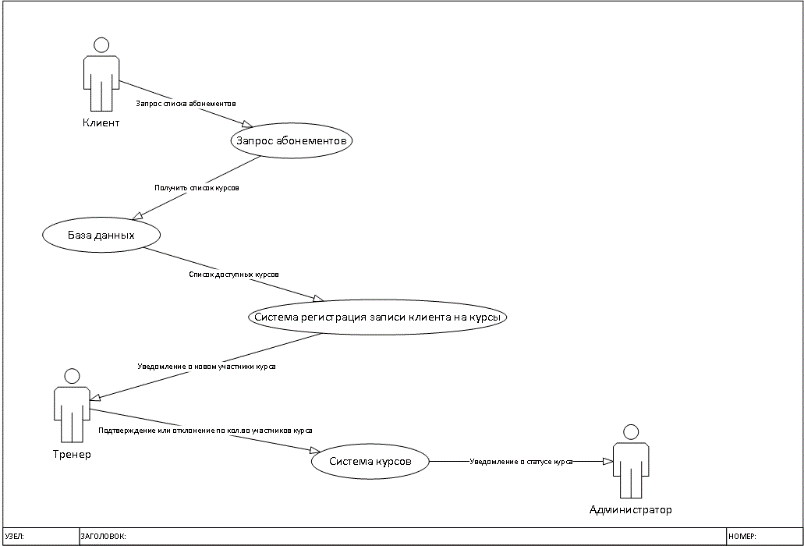


Рис. 5 UML – Диаграмма взаимодействия

**1.6.4 UML – Диаграмма классов**

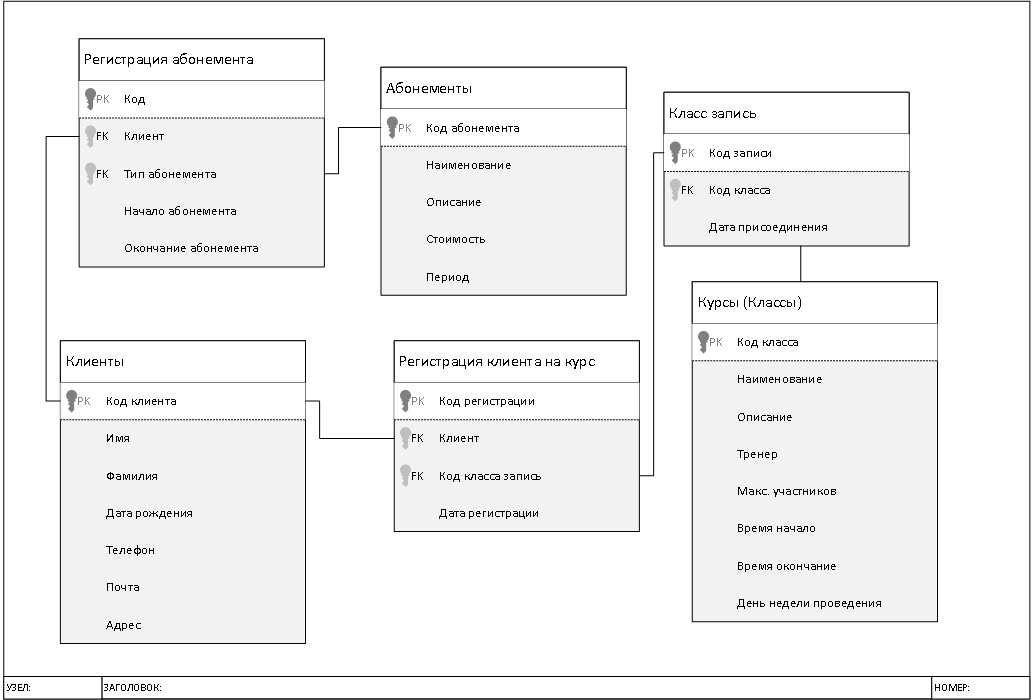


Рис. 6 UML – Диаграмма классов

## **1.6.5 Бизнес – процессы**

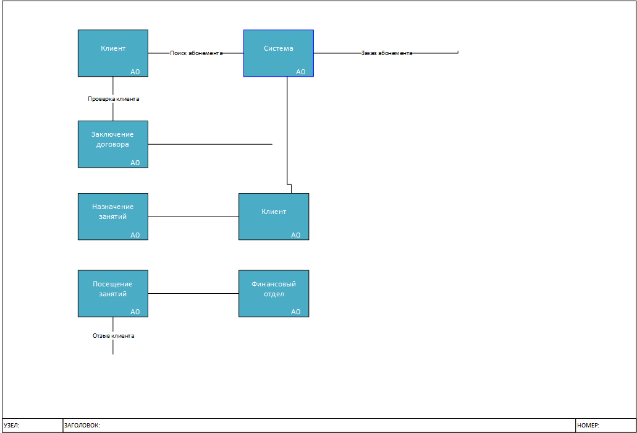


Рис. 7 Диаграмма бизнес процессы

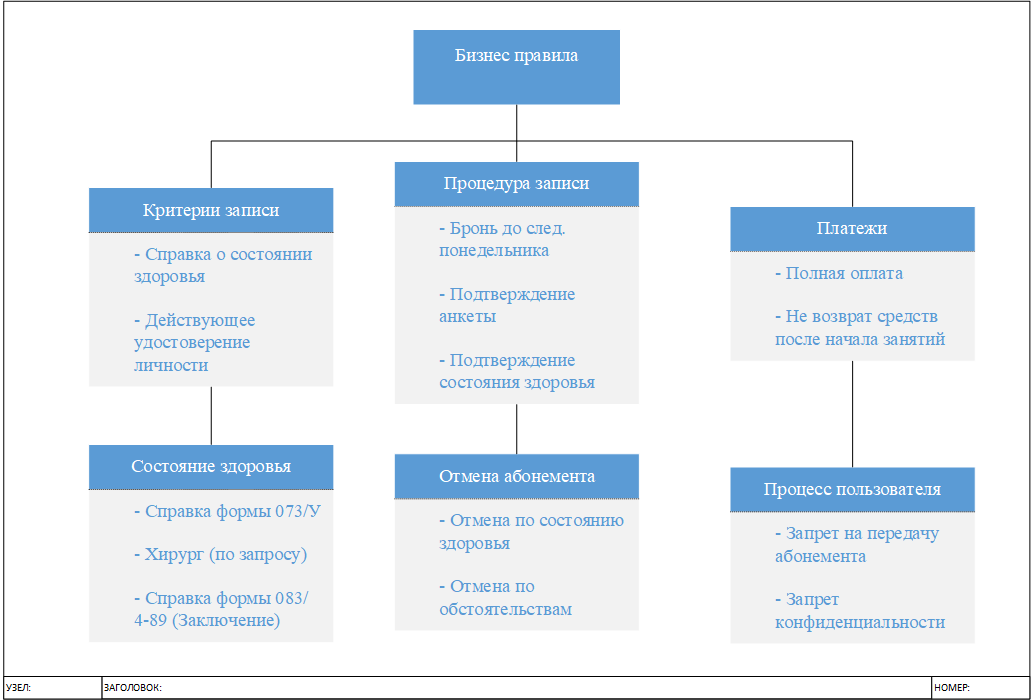


Рис. 8 Диаграмма бизнес правила

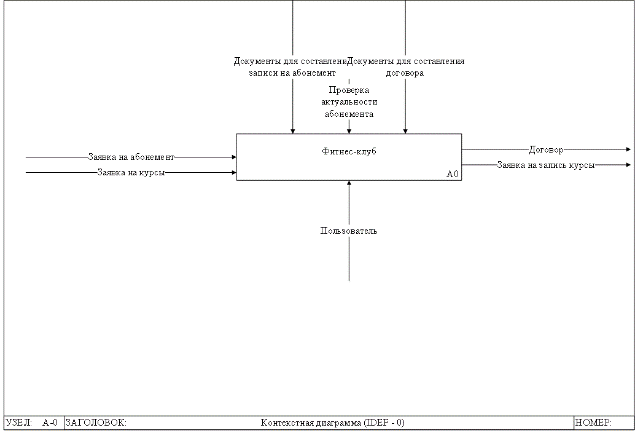


Рис. 9 IDEF-0 Контекстная диаграмма

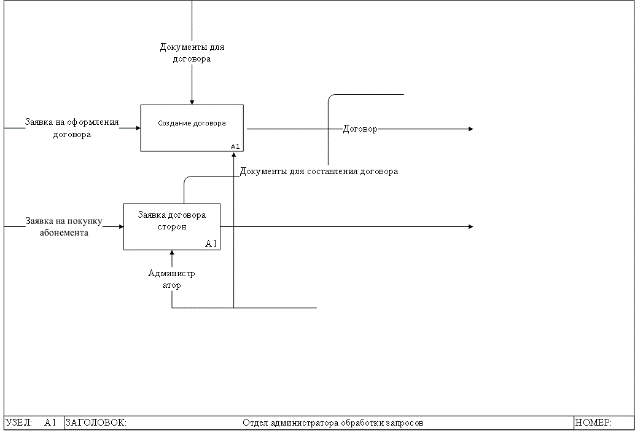


Рис. 10 IDEF-1 Диаграмма Отдел обработки

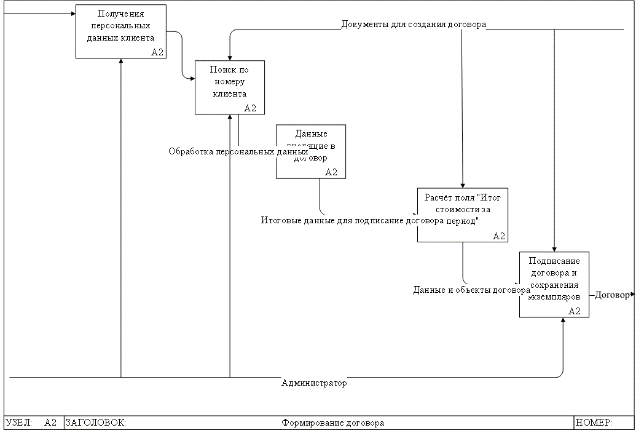


Рис. 11 IDEF-2 Диаграмма Формирование договора

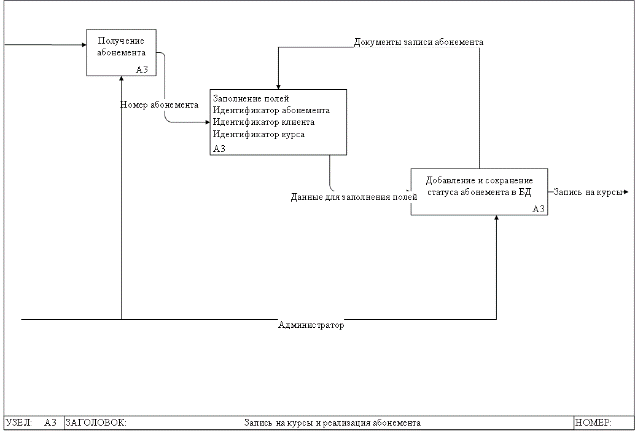


Рис. 12 IDEF-3 Диаграмма записи и реализации

# **ГЛАВА 2. РАЗРАБОТКА БАЗЫ ДАННЫХ**

**2.1 Разработка логической модели данных**

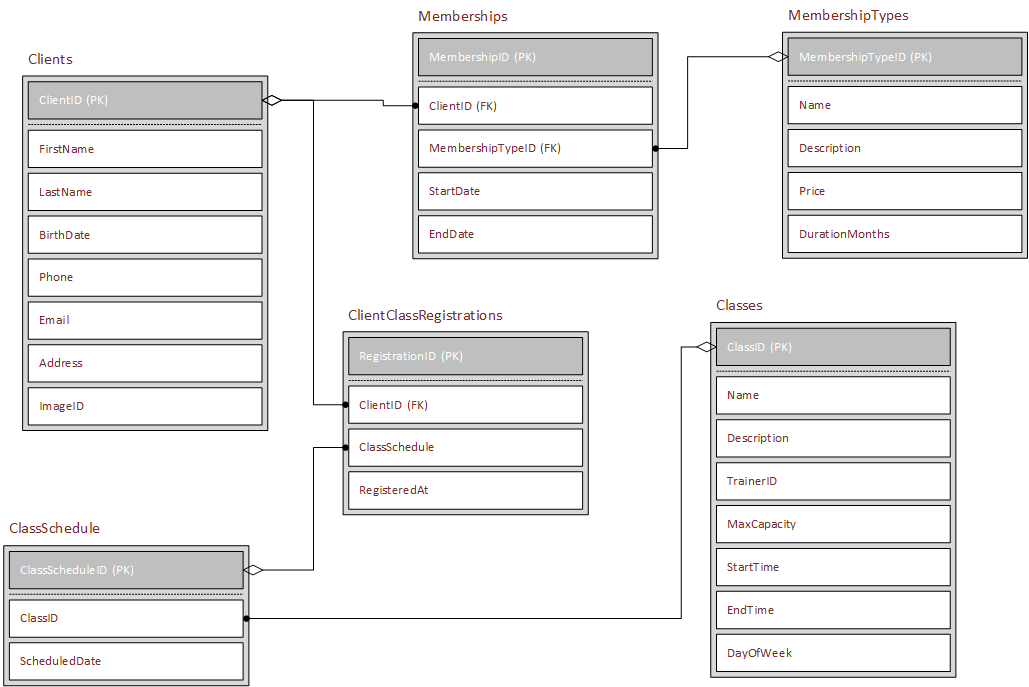
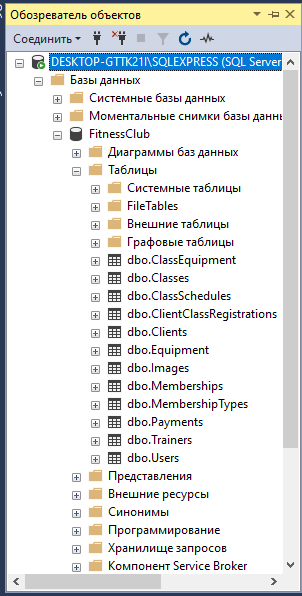


Рис. 13 Логическая модель данных

**2.2 Добавление таблиц к базе данных**



**Рис. 14 Добавление таблиц к базе данных**

**2.3 Связи на диаграмме базе данных**

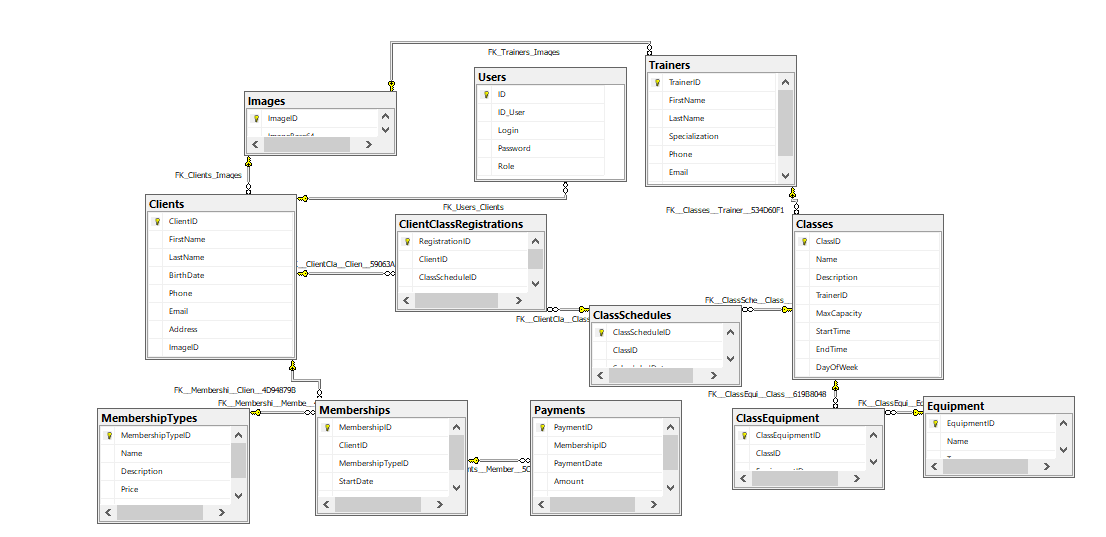


Рис. 15 Связи в базе данных

**2.4 Определение ограничений налагаемые на данные**

- Текст ограничен до 255 символов в запись.  
- Дата ограничена от 0L до 3155378975999999999L (от 01.01.0001 0:00:00 до 31.12.9999 23:59:59)

- Числа от -2147483648 до 2147483647

- Не целые числа от -3,4028235E+38 до 3,4028235E+38

**2.5 Разработка физической структуры данных**

USE [FitnessClub]

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (1, N'Иван', N'Иванов', CAST(N'1985-03-15' AS Date), N'+79161234567', N'ivan.ivanov@example.com', N'ул. Ленина, д. 1')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (2, N'Мария', N'Петрова', CAST(N'1990-11-20' AS Date), N'+79255678901', N'maria.petrova@example.com', N'пр. Мира, д. 10')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (3, N'Дмитрий', N'Сидоров', CAST(N'1978-06-10' AS Date), N'+79179012345', N'dmitry.sidorov@example.com', N'ул. Пушкина, д. 5')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (4, N'Анастасия', N'Смирнова', CAST(N'1988-02-28' AS Date), N'+79263456789', N'anastasia.smirnova@example.com', N'ул. Толстого, д. 15')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (5, N'Михаил', N'Кузнецов', CAST(N'1992-09-05' AS Date), N'+79157890123', N'mikhail.kuznetsov@example.com', N'ул. Горького, д. 20')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (6, N'Елена', N'Попова', CAST(N'1986-05-18' AS Date), N'+79212345678', N'elena.popova@example.com', N'ул. Чехова, д. 25')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (7, N'Алексей', N'Соколов', CAST(N'1975-08-01' AS Date), N'+79126789012', N'alexey.sokolov@example.com', N'ул. Лермонтова, д. 30')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (8, N'Ольга', N'Васильева', CAST(N'1991-04-12' AS Date), N'+79230123456', N'olga.vasilyeva@example.com', N'ул. Островского, д. 35')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (9, N'Сергей', N'Морозов', CAST(N'1989-07-25' AS Date), N'+79184567890', N'sergey.morozov@example.com', N'ул. Тургенева, д. 40')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (10, N'Светлана', N'Николаева', CAST(N'1993-12-30' AS Date), N'+79298901234', N'svetlana.nikolaeva@example.com', N'ул. Некрасова, д. 45')

INSERT [dbo].[Clients] ([ClientID], [FirstName], [LastName], [BirthDate], [Phone], [Email], [Address]) VALUES (11, N'Кирилл', N'Сухинин', CAST(N'2005-03-10' AS Date), N'+79011229960', N'SuxininKiril@yandex.ru', N'Бейшлотская наб. 93')

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (1, N'Базовый', N'Базовый абонемент', 10000, 1)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (2, N'Премиум', N'Премиум абонемент', 20000, 3)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (3, N'Элитный', N'Элитный абонемент', 40000, 6)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (4, N'Золотой', N'Золотой абонемент', 50000, 12)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (5, N'Серебряный', N'Серебряный абонемент', 40000, 6)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (6, N'Бронзовый', N'Бронзовый абонемент', 55000, 3)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (7, N'Платиновый', N'Платиновый абонемент', 80000, 12)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (8, N'Семейный', N'Семейный абонемент', 65000, 12)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (9, N'Студенческий', N'Студенческий абонемент', 15000, 3)

INSERT [dbo].[MembershipTypes] ([MembershipTypeID], [Name], [Description], [Price], [DurationMonths]) VALUES (10, N'Пенсионный', N'Пенсионный абонемент', 13000, 6)

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (1, 1, 1, CAST(N'2024-01-15' AS Date), CAST(N'2024-02-15' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (2, 2, 2, CAST(N'2024-02-20' AS Date), CAST(N'2024-05-20' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (3, 3, 3, CAST(N'2024-03-10' AS Date), CAST(N'2024-09-10' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (4, 4, 4, CAST(N'2024-01-28' AS Date), CAST(N'2025-01-28' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (5, 5, 5, CAST(N'2024-04-05' AS Date), CAST(N'2024-10-05' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (6, 6, 6, CAST(N'2024-05-18' AS Date), CAST(N'2024-08-18' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (7, 7, 7, CAST(N'2024-06-01' AS Date), CAST(N'2025-06-01' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (8, 8, 8, CAST(N'2024-07-12' AS Date), CAST(N'2025-07-12' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (9, 9, 9, CAST(N'2024-08-25' AS Date), CAST(N'2024-11-25' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (10, 10, 10, CAST(N'2024-09-30' AS Date), CAST(N'2025-03-30' AS Date))

INSERT [dbo].[Memberships] ([MembershipID], [ClientID], [MembershipTypeID], [StartDate], [EndDate]) VALUES (11, 11, 3, CAST(N'2025-02-05' AS Date), CAST(N'2025-03-05' AS Date))

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (1, N'Андрей', N'Васильев', N'Силовые тренировки', N'+79161111111', N'andrey.vasilyev@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (2, N'Елена', N'Смирнова', N'Йога', N'+79252222222', N'elena.smirnova@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (3, N'Дмитрий', N'Петров', N'Кардио', N'+79173333333', N'dmitry.petrov@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (4, N'Ольга', N'Иванова', N'Пилатес', N'+79264444444', N'olga.ivanova@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (5, N'Сергей', N'Кузнецов', N'Зумба', N'+79155555555', N'sergey.kuznetsov@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (6, N'Анастасия', N'Морозова', N'Спининг', N'+79216666666', N'anastasia.morozova@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (7, N'Михаил', N'Соколов', N'Кроссфит', N'+79127777777', N'mikhail.sokolov@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (8, N'Мария', N'Попова', N'Функциональный тренинг', N'+79238888888', N'maria.popova@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (9, N'Алексей', N'Николаев', N'Бодибилдинг', N'+79189999999', N'alexey.nikolaev@example.com')

INSERT [dbo].[Trainers] ([TrainerID], [FirstName], [LastName], [Specialization], [Phone], [Email]) VALUES (10, N'Светлана', N'Васильева', N'Стретчинг', N'+79290000000', N'svetlana.vasilyeva@example.com')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (1, N'Йога для начинающих', N'Йога для начинающих', 2, 20, CAST(N'07:00:00' AS Time), CAST(N'08:00:00' AS Time), N'Понедельник')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (2, N'Силовые тренировки', N'Тренировки с отягощениями', 1, 15, CAST(N'18:00:00' AS Time), CAST(N'19:00:00' AS Time), N'Вторник')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (3, N'Зумба', N'Занятия по Зумбе', 5, 25, CAST(N'19:30:00' AS Time), CAST(N'20:30:00' AS Time), N'Среда')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (4, N'Пилатес', N'Занятия по Пилатесу', 4, 10, CAST(N'09:00:00' AS Time), CAST(N'10:00:00' AS Time), N'Четверг')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (5, N'Спининг', N'Занятия на велотренажерах', 6, 12, CAST(N'17:00:00' AS Time), CAST(N'18:00:00' AS Time), N'Пятница')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (6, N'Кроссфит', N'Высокоинтенсивные тренировки', 7, 18, CAST(N'10:00:00' AS Time), CAST(N'11:00:00' AS Time), N'Суббота')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (7, N'Функциональный тренинг', N'Тренировки с собственным весом', 8, 20, CAST(N'08:00:00' AS Time), CAST(N'09:00:00' AS Time), N'Четверг')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (8, N'Кардио', N'Кардиотренировки', 3, 22, CAST(N'16:00:00' AS Time), CAST(N'17:00:00' AS Time), N'Понедельник')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (9, N'Силовой тренинг', N'Силовые тренировки', 1, 15, CAST(N'17:30:00' AS Time), CAST(N'18:30:00' AS Time), N'Вторник')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (10, N'Бодибилдинг', N'Тренировки на массу', 9, 10, CAST(N'19:00:00' AS Time), CAST(N'20:00:00' AS Time), N'Среда')

INSERT [dbo].[Classes] ([ClassID], [Name], [Description], [TrainerID], [MaxCapacity], [StartTime], [EndTime], [DayOfWeek]) VALUES (11, N'Стретчинг', N'Стретчинг', 10, 10, CAST(N'12:00:00' AS Time), CAST(N'13:00:00' AS Time), N'Четверг')

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (1, 1, CAST(N'2024-03-04' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (2, 2, CAST(N'2024-03-05' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (3, 3, CAST(N'2024-03-06' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (4, 4, CAST(N'2024-03-07' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (5, 5, CAST(N'2024-03-08' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (6, 6, CAST(N'2024-03-09' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (7, 7, CAST(N'2024-03-10' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (8, 8, CAST(N'2024-03-11' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (9, 9, CAST(N'2024-03-12' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (10, 10, CAST(N'2024-03-13' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (11, 2, CAST(N'2025-02-05' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (13, 7, CAST(N'2025-02-05' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (14, 1, CAST(N'2025-02-14' AS Date))

INSERT [dbo].[ClassSchedules] ([ClassScheduleID], [ClassID], [ScheduledDate]) VALUES (15, 3, CAST(N'2025-02-14' AS Date))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (1, 1, 1, CAST(N'2024-02-28T10:00:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (2, 2, 2, CAST(N'2024-02-29T14:30:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (3, 3, 3, CAST(N'2024-03-01T09:15:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (4, 4, 4, CAST(N'2024-03-02T11:45:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (5, 5, 5, CAST(N'2024-03-03T16:20:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (6, 6, 6, CAST(N'2024-02-27T18:00:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (7, 7, 7, CAST(N'2024-02-26T13:45:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (8, 8, 8, CAST(N'2024-03-01T10:00:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (9, 9, 9, CAST(N'2024-02-25T15:30:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (10, 10, 10, CAST(N'2024-03-03T12:15:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (11, 11, 11, CAST(N'2025-02-05T18:41:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (13, 11, 13, CAST(N'2025-02-05T18:41:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (14, 11, 14, CAST(N'2025-02-14T00:37:00.000' AS DateTime))

INSERT [dbo].[ClientClassRegistrations] ([RegistrationID], [ClientID], [ClassScheduleID], [RegisteredAt]) VALUES (15, 11, 15, CAST(N'2025-02-14T00:37:00.000' AS DateTime))

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (1, 1, CAST(N'2024-01-15' AS Date), 25, N'Карта')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (2, 2, CAST(N'2024-02-20' AS Date), 50, N'Наличные')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (3, 3, CAST(N'2024-03-10' AS Date), 100, N'Онлайн')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (4, 4, CAST(N'2024-01-28' AS Date), 75, N'Карта')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (5, 5, CAST(N'2024-04-05' AS Date), 50, N'Наличные')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (6, 6, CAST(N'2024-05-18' AS Date), 30, N'Онлайн')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (7, 7, CAST(N'2024-06-01' AS Date), 150, N'Карта')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (8, 8, CAST(N'2024-07-12' AS Date), 120, N'Наличные')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (9, 9, CAST(N'2024-08-25' AS Date), 15, N'Онлайн')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (10, 10, CAST(N'2024-09-30' AS Date), 35, N'Карта')

INSERT [dbo].[Payments] ([PaymentID], [MembershipID], [PaymentDate], [Amount], [PaymentMethod]) VALUES (11, 11, CAST(N'2025-02-05' AS Date), 25, N'Онлайн')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (1, N'Гантели', N'Оборудование')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (2, N'Штанга', N'Оборудование')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (3, N'Блинчики', N'Оборудование')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (4, N'Коврики для йоги', N'Инвентарь')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (5, N'Велотренажёр', N'Кардио')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (6, N'Беговая дорожка', N'Кардио')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (7, N'Эллиптический тренажёр', N'Кардио')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (8, N'Гребной тренажёр', N'Кардио')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (9, N'Реформер для пилатеса', N'Оборудование')

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [Type]) VALUES (10, N'Велотренажёры для спиннинга', N'Кардио')

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (1, 1, 4)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (2, 2, 1)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (3, 2, 2)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (4, 2, 3)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (5, 3, 5)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (6, 4, 9)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (7, 5, 10)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (8, 6, 6)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (9, 7, 7)

INSERT [dbo].[ClassEquipment] ([ClassEquipmentID], [ClassID], [EquipmentID]) VALUES (10, 8, 5)

INSERT [dbo].[Users] ([ID], [ID\_User], [Login], [Password], [Role]) VALUES (0, 1, N'Admin', N'Admin', N'Admin')

INSERT [dbo].[Users] ([ID], [ID\_User], [Login], [Password], [Role]) VALUES (1, 11, N'SuxininKiril', N'7%;\*/(poh/}%', N'User')

**2.6 Создание БД и таблиц**

use FitnessClub;

CREATE TABLE Images(

ImageID INT PRIMARY KEY NOT NULL,

ImageBase64 VARCHAR(MAX) NOT NULL

);

--1. Clients: Эта таблица хранит информацию о клиентах фитнес-клуба.

CREATE TABLE Clients (

ClientID INT PRIMARY KEY,

FirstName VARCHAR(MAX) NOT NULL,

LastName VARCHAR(MAX) NOT NULL,

BirthDate DATE NULL,

Phone VARCHAR(MAX) NULL,

Email VARCHAR(MAX) NULL,

[Address] VARCHAR(MAX) NULL,

ImageID INT NOT NULL,

FOREIGN KEY (ImageID) REFERENCES Images(ImageID)

);

--2. MembershipTypes: Эта таблица описывает различные типы абонементов, предлагаемых фитнес-клубом.

CREATE TABLE MembershipTypes (

MembershipTypeID INT PRIMARY KEY,

Name VARCHAR(MAX) NOT NULL,

[Description] VARCHAR(MAX) NULL,

Price FLOAT NOT NULL,

DurationMonths INT NOT NULL

);

--3. Memberships: Эта таблица связывает клиентов с конкретными абонементами.

CREATE TABLE Memberships (

MembershipID INT PRIMARY KEY,

ClientID INT NOT NULL,

MembershipTypeID INT NOT NULL,

StartDate DATE NOT NULL,

EndDate DATE NULL,

FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),

FOREIGN KEY (MembershipTypeID) REFERENCES MembershipTypes (MembershipTypeID),

);

--4. Trainers: Эта таблица содержит информацию о тренерах, работающих в фитнес-клубе.

CREATE TABLE Trainers (

TrainerID INT PRIMARY KEY,

FirstName VARCHAR(MAX) NOT NULL,

LastName VARCHAR(MAX) NOT NULL,

Specialization VARCHAR(MAX) NULL,

Phone VARCHAR(MAX) NULL,

Email VARCHAR(MAX) NULL,

ImageID INT NOT NULL,

FOREIGN KEY (ImageID) REFERENCES Images(ImageID)

);

--5. Classes: Эта таблица описывает групповые занятия, предлагаемые фитнес-клубом.

CREATE TABLE Classes (

ClassID INT PRIMARY KEY,

[Name] VARCHAR(MAX) NOT NULL,

[Description] VARCHAR(MAX) NULL,

TrainerID INT NOT NULL,

MaxCapacity INT NOT NULL,

StartTime TIME NOT NULL,

EndTime TIME NOT NULL,

[DayOfWeek] VARCHAR(MAX) NOT NULL,

FOREIGN KEY (TrainerID) REFERENCES Trainers(TrainerID)

);

--6. ClassSchedules: Эта таблица хранит расписание занятий.

CREATE TABLE ClassSchedules (

ClassScheduleID INT PRIMARY KEY,

ClassID INT NOT NULL,

ScheduledDate DATE NOT NULL, --More descriptive name

FOREIGN KEY (ClassID) REFERENCES Classes(ClassID)

);

--7. ClientClassRegistrations: Эта таблица отслеживает регистрацию клиентов на групповые занятия.

CREATE TABLE ClientClassRegistrations (

RegistrationID INT PRIMARY KEY,

ClientID INT NOT NULL,

ClassScheduleID INT NOT NULL,

RegisteredAt DATETIME,

FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),

FOREIGN KEY (ClassScheduleID) REFERENCES ClassSchedules(ClassScheduleID)

);

--8. Payments: Эта таблица хранит информацию о платежах, совершенных клиентами.

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY,

MembershipID INT NOT NULL,

PaymentDate DATE NOT NULL,

Amount FLOAT NOT NULL,

PaymentMethod VARCHAR(MAX) NOT NULL, -- Use VARCHAR instead of ENUM

FOREIGN KEY (MembershipID) REFERENCES Memberships(MembershipID)

);

--9. Equipment: Эта таблица перечисляет оборудование, имеющееся в фитнес-клубе.

CREATE TABLE Equipment (

EquipmentID INT PRIMARY KEY,

[Name] VARCHAR(MAX) NOT NULL,

[Type] VARCHAR(MAX) NULL

);

--10. ClassEquipment: Эта таблица устанавливает связь между занятиями и используемым для них оборудованием (связывающая таблица).

CREATE TABLE ClassEquipment (

ClassEquipmentID INT PRIMARY KEY,

ClassID INT NOT NULL,

EquipmentID INT NOT NULL,

FOREIGN KEY (ClassID) REFERENCES Classes(ClassID) ON DELETE CASCADE,

FOREIGN KEY (EquipmentID) REFERENCES Equipment(EquipmentID) ON DELETE CASCADE

);

**2.7 Данные в таблицах**

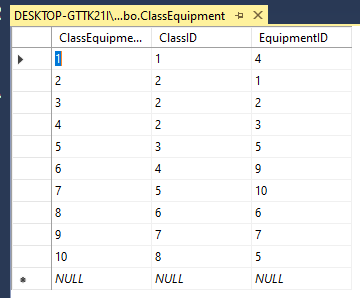


Рис. 16 Таблица «ClassEquipment»

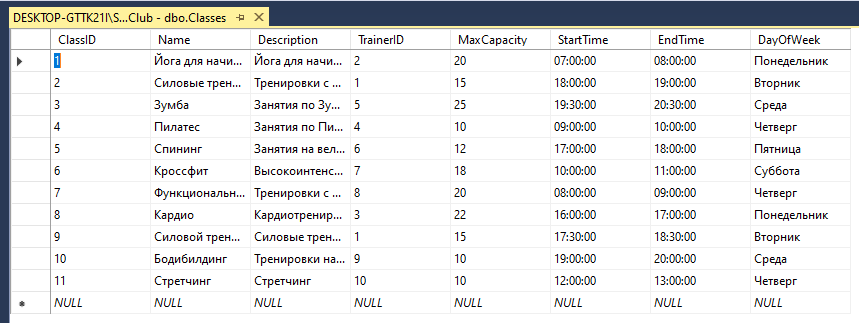


Рис. 17 Таблица «Classes»

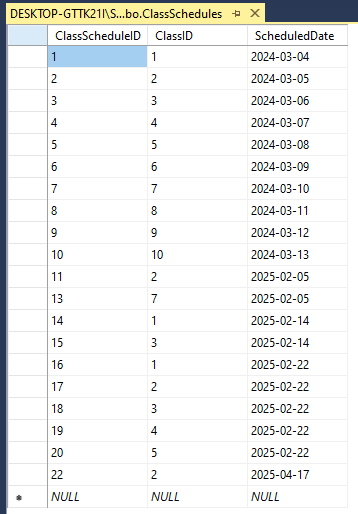


Рис. 18 Таблица «ClassSchedule»

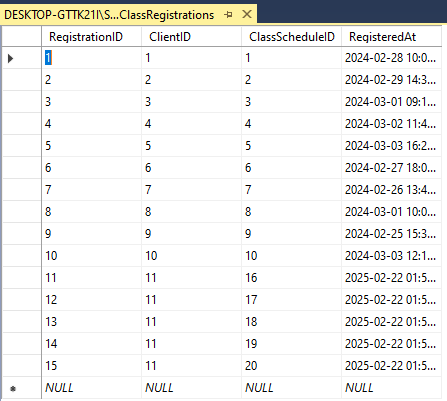


Рис. 19 Таблица «ClientClassRegistrations»

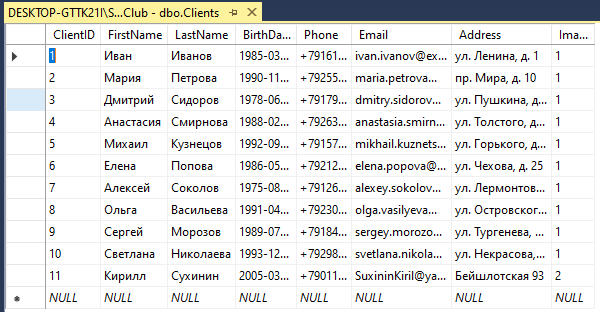


Рис. 20 Таблица «Clients»

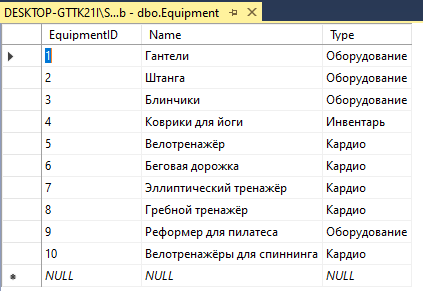


Рис. 21 Таблица «Equipment»

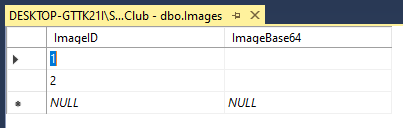


Рис. 22 Таблица «Images» (Примечание: base64 не отображается в SSMS)

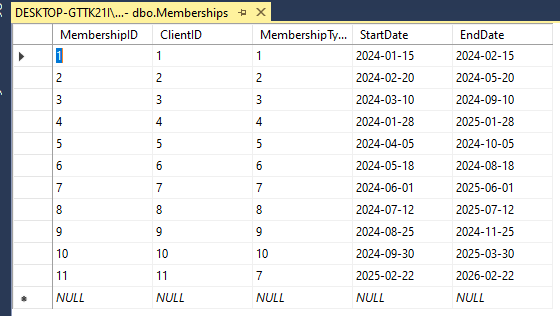


Рис. 23 Таблица «Memberships»

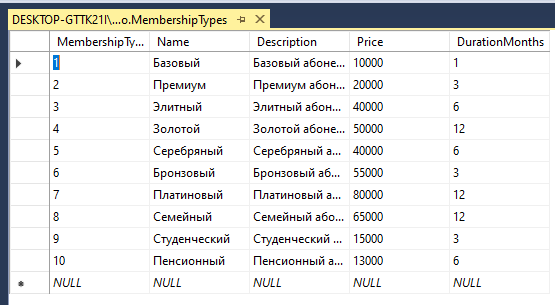


Рис. 24 Таблица «MembershipTypes»

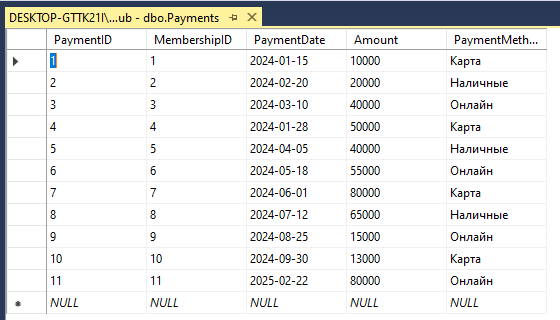


Рис. 25 Таблица «Payments»

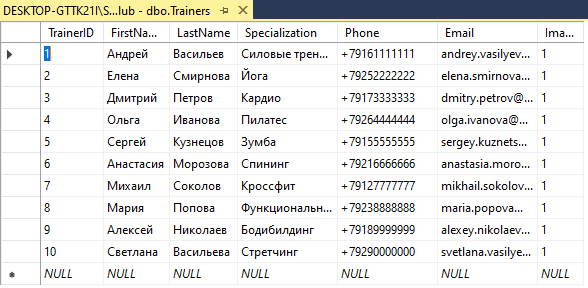


Рис. 26 Таблица «Trainers»

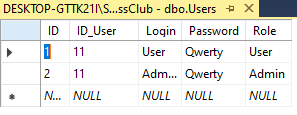


Рис. 27 Таблица «Users»

**2.8 Создание запросов**

1. Запрос на выборку

SELECT FirstName, LastName

FROM Clients;

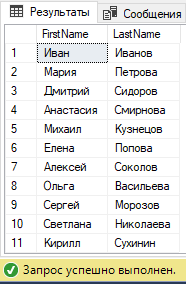


Рис. 28 Запрос №1 Вывод имён клиентов

SELECT c.FirstName, c.LastName

FROM Clients c

JOIN ClientClassRegistrations r ON c.ClientID = r.ClientID

JOIN ClassSchedules s ON r.ClassScheduleID = s.ClassScheduleID

WHERE s.ScheduledDate = '2024-03-07'

ORDER BY c.LastName;

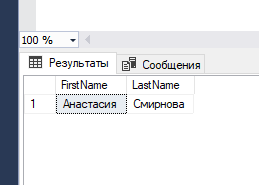


Рис. 29 Запрос №2 Вывод клиента по дате

SELECT mt.Name, COUNT(m.ClientID) AS NumberOfClients

FROM Memberships m

JOIN MembershipTypes mt ON m.MembershipTypeID = mt.MembershipTypeID

GROUP BY mt.Name

HAVING COUNT(m.ClientID) >= 2;

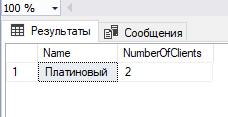


Рис. 30 Запрос №3 Абонемент приобретен 2 раза.

2. Запрос на добавление

INSERT INTO Clients (ClientID, FirstName, LastName, ImageID)

VALUES (12, 'Иван', 'Петров', 1);

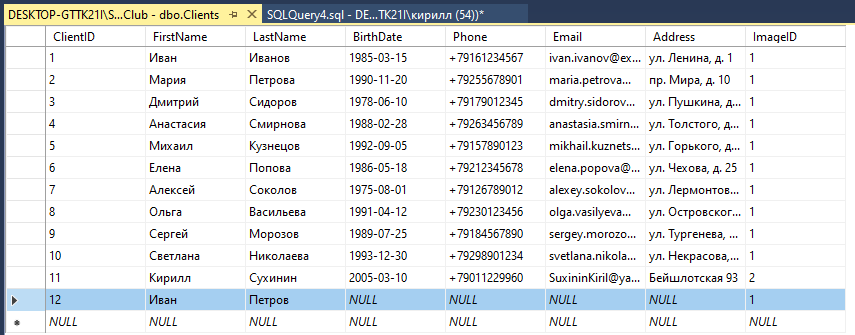


Рис. 31 Запрос №4 Добавление клиента

3. Запрос на изменения

UPDATE Trainers

SET Phone = '+79991234567'

WHERE TrainerID = 1;

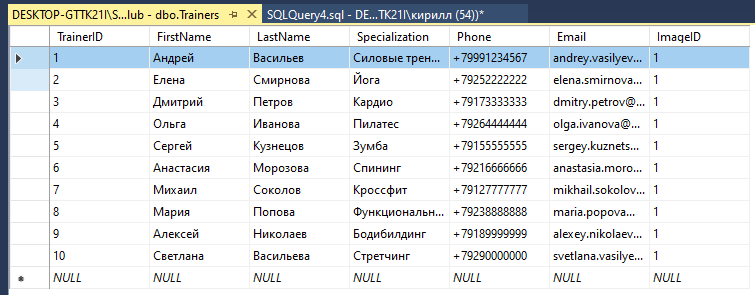


Рис. 32 Запрос №5 Изменение номера у тренера

4. Запрос на удаление

DELETE FROM Clients

WHERE ClientID = 12;

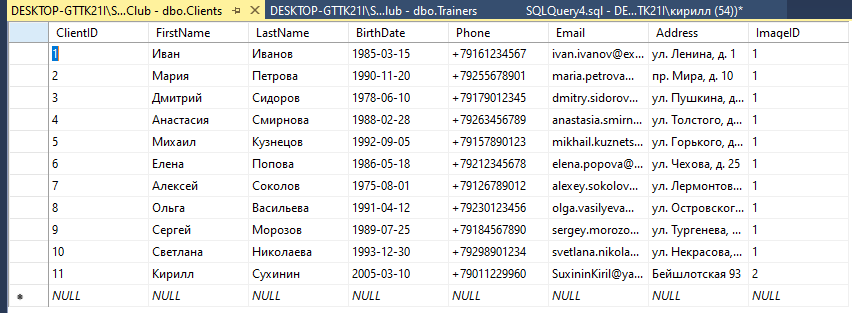


Рис. 33 Запрос №6 Удаление клиента

5. Запрос на вычисление

SELECT AVG(Price) AS AverageMembershipPrice

FROM MembershipTypes;

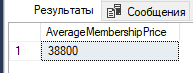


Рис. 34 Запрос №7 Вычисление среднюю стоимость абонемента

**ГЛАВА 3. РАЗРАБОТКА ПРИЛОЖЕНИЯ**

**3.1 Разработка пользовательского интерфейса**

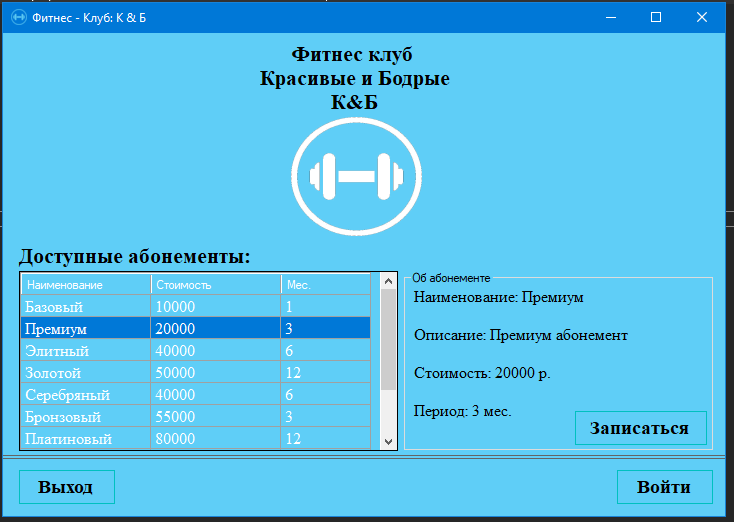


Рис. 35 Титульная форма

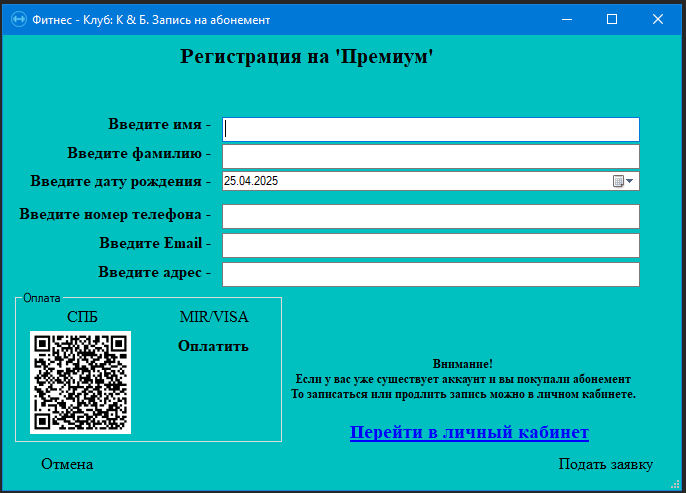


Рис. 36 Запись на абонемент

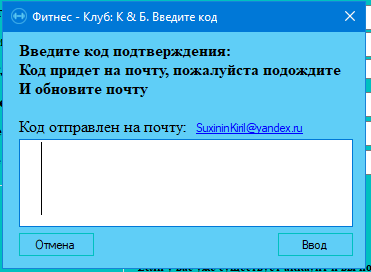


Рис. 37 Форма кода с почты при регистрации

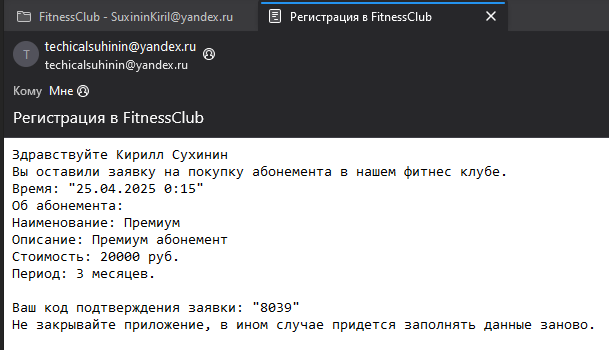


Рис. 38 Письмо для подтверждения почты

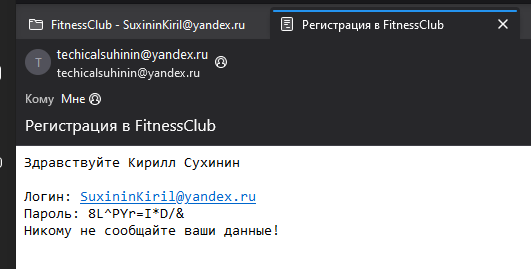


Рис. 39 Письмо после подтверждение (Данные для входа)

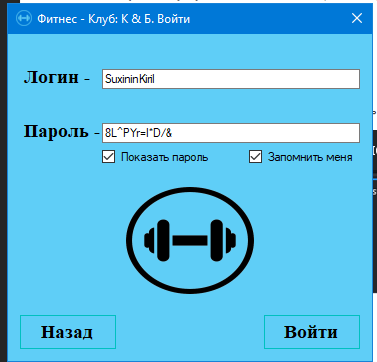


Рис. 40 Форма авторизации

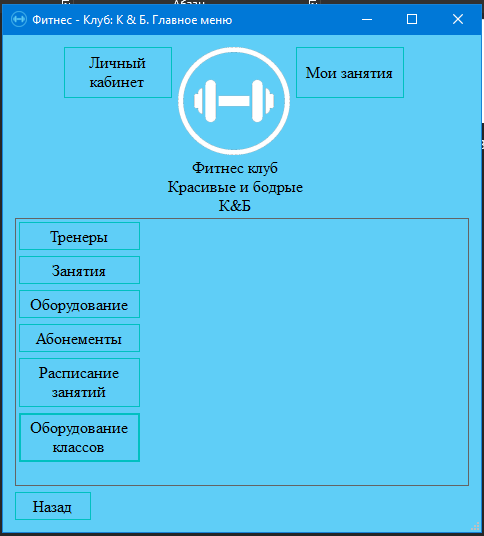


Рис. 41 Главная форма

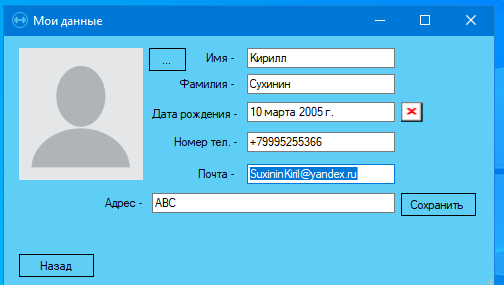


Рис. 42 Форма аккаунта

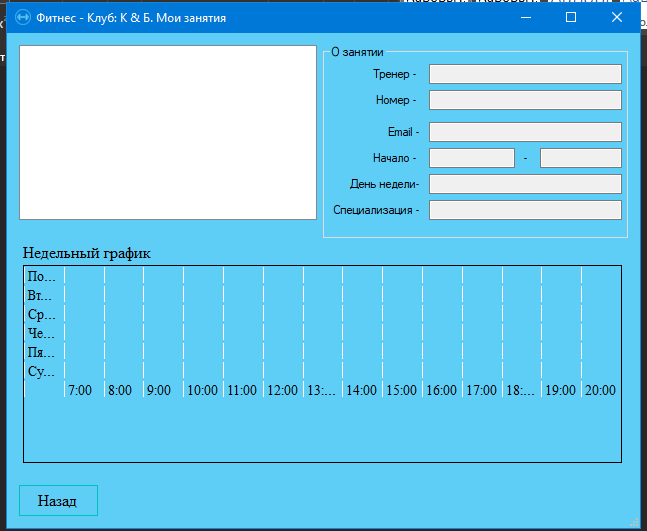


Рис. 43 Форма «Мои занятия»

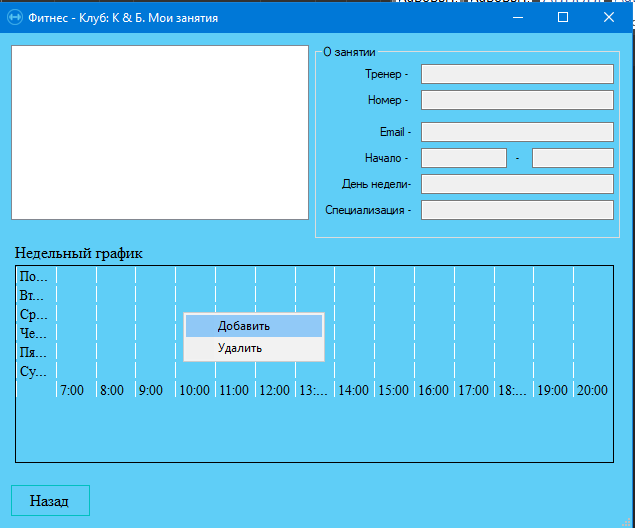


Рис. 44 Контекстное меню на «Недельный график» так и в ListBox

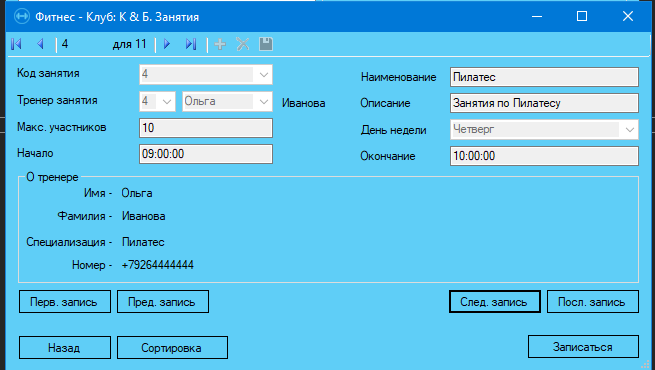


Рис. 45 Запись на занятия

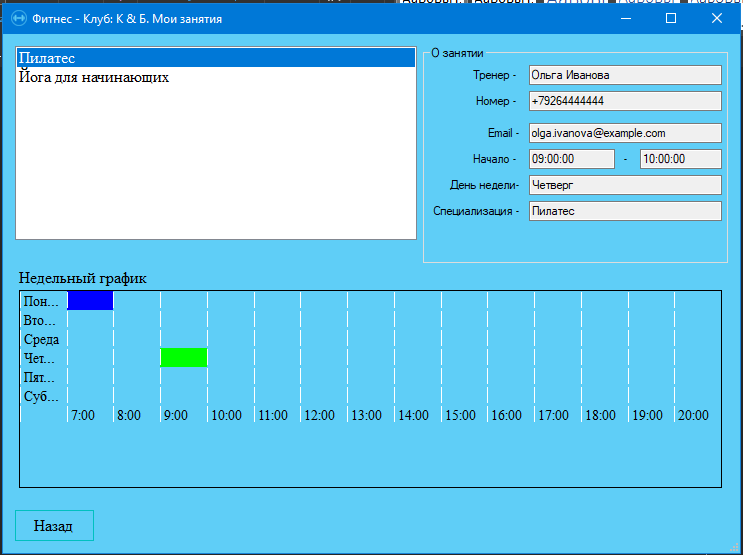


Рис. 46 После записи занятия отображаются на форме «Мои занятия»

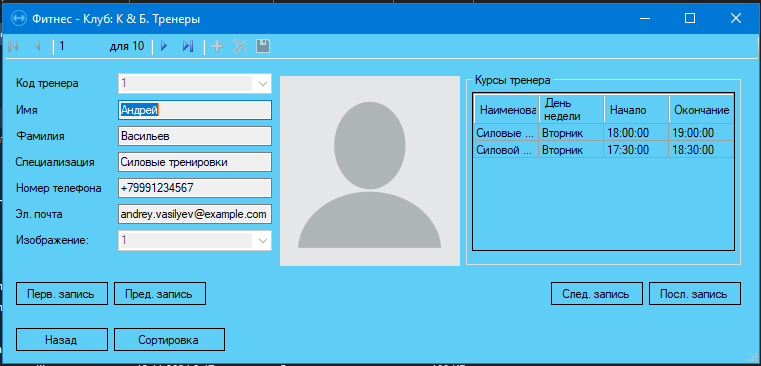


Рис. 47 Форма «Тренеры» (добавлены курсы тренера)

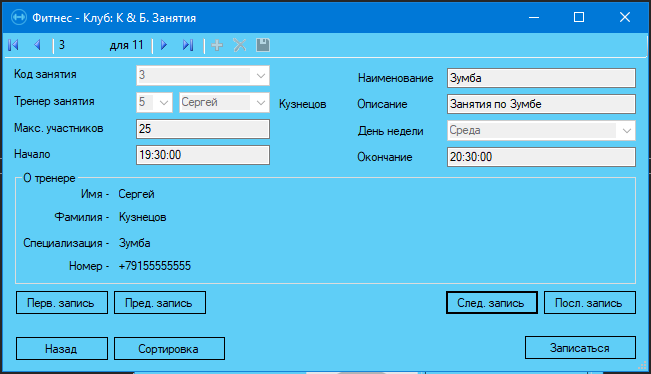


Рис. 48 Форма «Занятия»

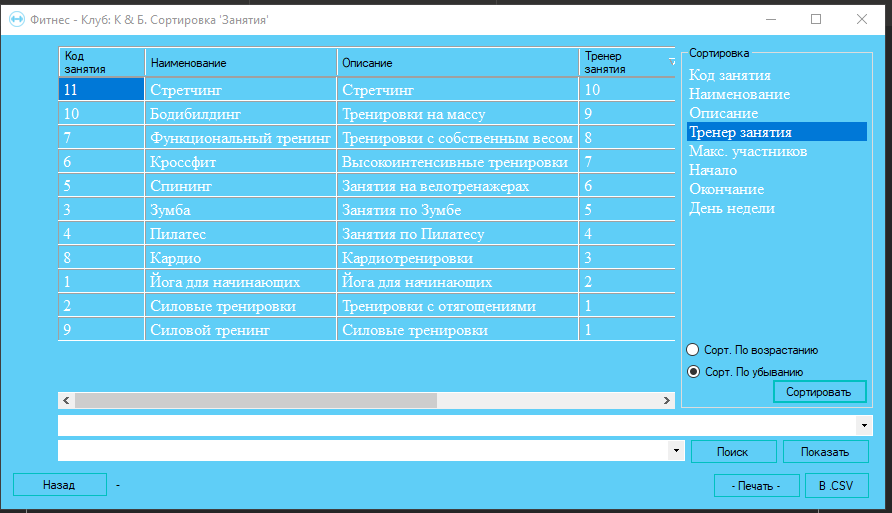


Рис. 49 Форма «Сортировки и поиска» Сортировка

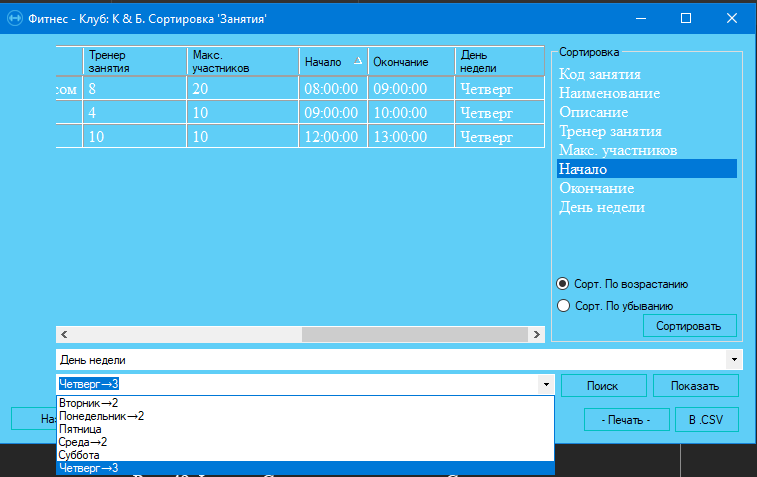


Рис. 50 Форма «Сортировки и поиска» Сортировка и поиск

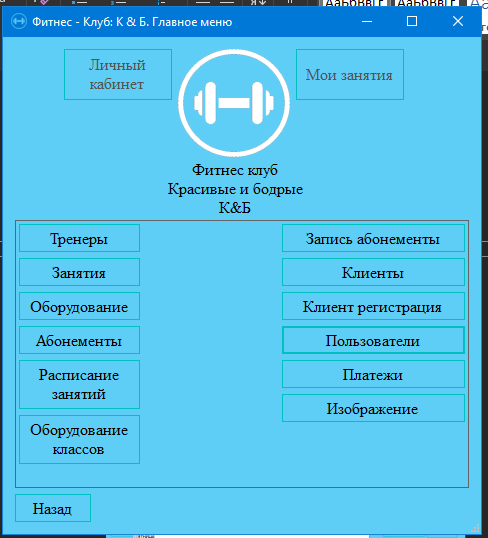


Рис. 51 Главная форма от администратора («Личный кабинет» и «Мои занятия» не доступны)

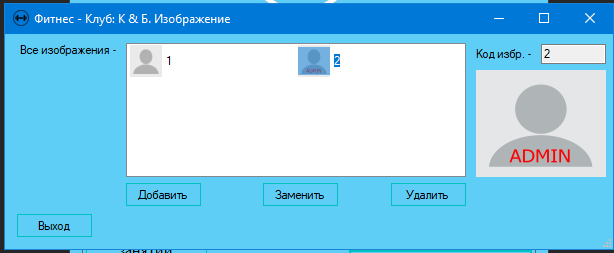


Рис. 52 Дополнительная форма для администратора «Редактор изображений БД»

**3.2 Разработка библиотек класса**

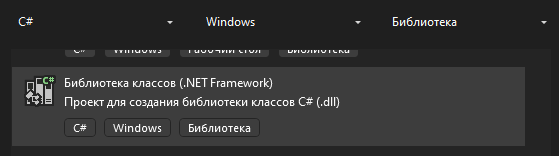


Рис. 53 Создание проекта библиотеки класса

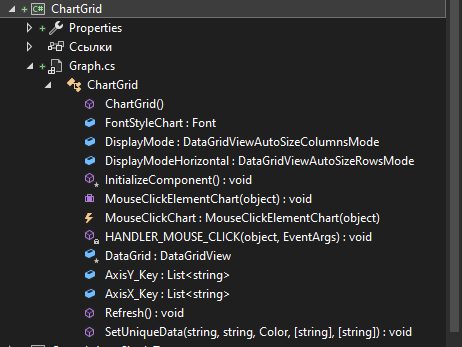


Рис. 54 Создание функционала библиотеки

В моём случае класс ChartGrid (дополнительный элемент для отображения «недельного расписания» для формы «Мои занятия»)

Листинг библиотеки ChartGrid : Panel

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace ChartGrid

{

/// <summary>

/// Элемент предоставляет возможность отображить граффик с цветовыми точками

/// </summary>

public class ChartGrid : Panel

{

public ChartGrid()

{

this.MinimumSize = new System.Drawing.Size(50,50);

}

[Category("Appearance")]

public Font FontStyleChart = new Font("Times New Roman", 10f);

[Category("Appearance")]

public DataGridViewAutoSizeColumnsMode DisplayMode = DataGridViewAutoSizeColumnsMode.Fill;

[Category("Appearance")]

public DataGridViewAutoSizeRowsMode DisplayModeHorizontal = DataGridViewAutoSizeRowsMode.AllCells;

/// <summary>

/// Инициализация элементов

/// </summary>

protected void InitializeComponent()

{

//График по сути таблица где первый столбец и последняя строка представляет собой набор данных пользователя

this.Controls.Clear();

DataGrid = new DataGridView();

DataGrid.Anchor = this.Anchor;

DataGrid.Location = new System.Drawing.Point(0, 0);

DataGrid.Size = new System.Drawing.Size(this.Size.Width, this.Size.Height);

foreach(var x in AxisX\_Key)

{

DataGrid.Columns.Add(x,x);

}

DataGrid.Columns.Add("", "");

DataGrid.Rows.Add(AxisY\_Key.Count);

int index\_y\_row = 0;

foreach(var y in AxisY\_Key)

{

DataGrid.Rows[index\_y\_row].SetValues(y);

index\_y\_row++;

}

List<string> add\_x\_axis = new List<string>();

add\_x\_axis.Add("");//Empty data for first collumn where we have Y-axis label

foreach(var x in AxisX\_Key)

{

add\_x\_axis.Add(x);

}

int index = DataGrid.Rows.Add(add\_x\_axis.Select(x => x).ToArray<string>());

for (int i = 1; i < DataGrid.Rows[index].Cells.Count; i++)

{

DataGrid.Rows[index].Cells[i].Style.Font = FontStyleChart;

}

for(int i = 0; i < DataGrid.Rows.Count; i++)

{

DataGrid.Rows[i].Cells[0].Style.Font = FontStyleChart;

}

//STYLE

DataGrid.EnableHeadersVisualStyles = false;

DataGrid.RowHeadersVisible = false;

DataGrid.AllowUserToAddRows = false;

DataGrid.ReadOnly = true;

DataGrid.AutoSize = false;

DataGrid.SelectionMode = DataGridViewSelectionMode.RowHeaderSelect;

DataGrid.MultiSelect = false;

DataGrid.AutoSizeColumnsMode = DisplayMode;

DataGrid.AutoSizeRowsMode = DisplayModeHorizontal;

DataGrid.CellBorderStyle = DataGridViewCellBorderStyle.None;

DataGrid.AdvancedCellBorderStyle.Left = DataGridViewAdvancedCellBorderStyle.OutsetPartial;

DataGrid.AllowUserToDeleteRows = false;

DataGrid.AllowUserToOrderColumns = false;

DataGrid.AllowUserToResizeColumns = false;

DataGrid.AllowUserToResizeRows = false;

DataGrid.GridColor = this.BackColor;

DataGrid.RowsDefaultCellStyle.BackColor = this.BackColor;

DataGrid.RowsDefaultCellStyle.SelectionForeColor = this.ForeColor;

DataGrid.RowsDefaultCellStyle.SelectionBackColor = Color.Transparent;

DataGrid.SelectionChanged -= HANDLER\_MOUSE\_CLICK;

DataGrid.SelectionChanged += HANDLER\_MOUSE\_CLICK;

DataGrid.BackgroundColor = this.BackColor;

DataGrid.ColumnHeadersVisible = false;

this.Controls.Add(DataGrid);

DataGrid.ClearSelection();

}

/// <summary>

/// Делегат который бэкапит действие обратно на главный уровень при выборе элемента графика

/// </summary>

/// <param name="NameAxisX"></param>

/// <param name="NameAxisY"></param>

public delegate void MouseClickElementChart(object data);

public event MouseClickElementChart MouseClickChart;

private void HANDLER\_MOUSE\_CLICK(object sender,EventArgs e)

{

if(DataGrid.SelectedCells != null && DataGrid.SelectedCells.Count > 0)

{

if (DataGrid.SelectedCells[0].ColumnIndex <= AxisX\_Key.Count && DataGrid.SelectedCells[0].RowIndex < AxisY\_Key.Count && DataGrid.SelectedCells[0].ColumnIndex > 0)

{

// Debugger.Log(0, "Debug", $"HANDLER\_MOUSE\_CLICK: X:{DataGrid.SelectedCells[0].ColumnIndex} Y:{DataGrid.SelectedCells[0].RowIndex}\n");

// Debugger.Log(0, "Debug", $"HANDLER\_MOUSE\_CLICK: {AxisX\_Key[DataGrid.SelectedCells[0].ColumnIndex]} {AxisY\_Key[DataGrid.SelectedCells[0].RowIndex]}\n");

int x = DataGrid.SelectedCells[0].ColumnIndex;

int y = DataGrid.SelectedCells[0].RowIndex;

if (DataGrid.SelectedCells[0].Tag != null)

MouseClickChart?.Invoke(DataGrid.SelectedCells[0].Tag);

}

DataGrid.ClearSelection();

}

}

protected DataGridView DataGrid;

/// <summary>

/// Ключ - тех название

/// Значение - отображение

/// </summary>

[Category("Data")]

[Description("Описание элементов по Y оси данных")]

public List<string> AxisY\_Key = new List<string>();

/// <summary>

/// Ключ - тех название

/// Значение - отображение

/// </summary>

[Category("Data")]

[Description("Описание элементов по Y оси данных")]

public List<string> AxisX\_Key = new List<string>();

public override void Refresh()

{

//HERE WE HANDLED ME ELEMENTS OR RE-DRAWING

InitializeComponent();

base.Refresh();

}

public void SetUniqueData(string NameAxisX,string NameAxisY,Color ColorData,string tagforbackup = null,string ToolTip = null)

{

int indexX = -1;

int indexY = -1;

for (int i = 0; i < DataGrid.Rows.Count; i++)

{

if (DataGrid.Rows[i].Cells[0].Value.ToString().ToUpper() == NameAxisY.ToUpper())

{

indexY = i;

}

}

for(int i = 0; i < DataGrid.Columns.Count; i++)

{

if (DataGrid.Columns[i].Name.ToUpper() == NameAxisX.ToUpper())

{

indexX = i;

}

}

if (indexY != -1 && indexX != -1)

{

DataGrid.Rows[indexY].Cells[indexX+1].Style.BackColor = ColorData;

DataGrid.Rows[indexY].Cells[indexX + 1].ToolTipText = ToolTip;

DataGrid.Rows[indexY].Cells[indexX + 1].Tag = tagforbackup;

}

}

}

}

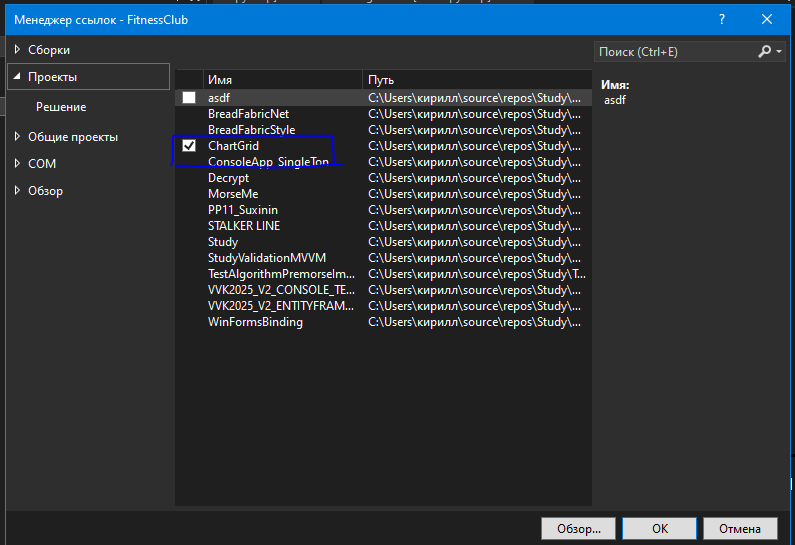


Рис. 55 Добавляем ссылку на библиотеку в проект

**3.3 Листинг программы**

Главный код приложения (Program.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Data.Sql;

using System.Data;

using System.Diagnostics;

using FitnessClub.JSON;

using Newtonsoft.Json;

using System.Security.AccessControl;

using System.Reflection.Emit;

using System.Drawing;

using System.IO;

using System.Threading;

namespace FitnessClub

{

public static class Debugger

{

/// <summary>

/// Дебаггер для удобства записи и вывода ошибок (немного поглащения)

/// </summary>

/// <param name="i"></param>

/// <param name="level"></param>

/// <param name="message"></param>

public static void Log(int i,string level,string message)

{

try

{

//Если уровень стоит на запись в файл (по умолчанию Debug)

if (level == "Debug/File")

{

System.IO.File.AppendAllText("DebugFile.log", $"{Environment.NewLine}{System.DateTime.Now.ToShortDateString()} {DateTime.Now.ToShortTimeString()} -> {i} {level}{Environment.NewLine}{message}");

}

//Определяем уровень если ошибка вывод в консоли КРАСНЫМ, если предупреждение ЖЁЛТЫМ в ином случае белый

if (message.ToUpper().Contains("WARNING") || message.ToUpper().Contains("ПРЕДУПРЕЖДЕНИЕ"))

Console.ForegroundColor = ConsoleColor.DarkYellow;

if (message.ToUpper().Contains("ERROR") || message.ToUpper().Contains("ОШИБКА"))

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"i: {i}\nLevel: {level}\nMessage:\n{message}");

System.Diagnostics.Debugger.Log(i, level.Split('/')[0], message);

Console.ForegroundColor = ConsoleColor.White;

}

catch(Exception g)

{

//Обрабатываем исключение т.к. одновременный вызов функции на уровне записи может привести к ошибки доступа

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("\tMultiThread Debugger ERROR");

Console.ForegroundColor = ConsoleColor.White;

}

}

}

/// <summary>

/// Роли

/// </summary>

public enum Role

{

User = 0,

Admin = 1

}

public static class Program

{

//P.S> ДА я знаю что Program уже божественный класс

/// <summary>

/// Относительно истины/лож вернём роль как перечисление

/// </summary>

/// <param name="r">Роль: Истина/Лож</param>

/// <returns></returns>

public static Role GetRole(bool r)

{

if (r)

return Role.Admin;

else

return Role.User;

}

/// <summary>

/// Перегрузка прошлого метода (только наоборот)

/// </summary>

/// <param name="role">Перечисление</param>

/// <returns></returns>

public static bool GetRole(Role role)

{

if (role == Role.Admin)

return true;

else

return false;

}

/// <summary>

/// Конвертор изображение в base64 строку

/// </summary>

/// <param name="image">Изображение</param>

/// <returns>Строка base64</returns>

public static string BitmapToBase64(Bitmap image)

{

using (var ms = new MemoryStream())

{

using (var bitmap = new Bitmap(image))

{

bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

var SigBase64 = Convert.ToBase64String(ms.GetBuffer()); //Get Base64

return SigBase64;

}

}

}

/// <summary>

/// Конвертор строки base64 в изображение

/// </summary>

/// <param name="base64String">Строка base64</param>

/// <returns>Bitmap изображение</returns>

public static Bitmap Base64ToBitmap(string base64String)

{

// Convert Base64 String to byte[]

byte[] imageBytes = Convert.FromBase64String(base64String);

MemoryStream ms = new MemoryStream(imageBytes, 0,

imageBytes.Length);

// Convert byte[] to Image

ms.Write(imageBytes, 0, imageBytes.Length);

Image image = Image.FromStream(ms, true);

return (Bitmap)image;

}

/// <summary>

/// МЕТОД ДЛЯ ЗАПИСИ В БАЗУ

/// </summary>

/// <param name="sql">Запрос</param>

/// <returns>Код результата</returns>

public static int Insert(string sql)

{

try

{

//Блокаем выполнение исходного потока ради подпотока (решает баг когда у нас одновременно обновляются изрб. на форме и в базе)

Program.ThreadLoadImage.Join();

var command = SERVER\_SQL.CreateCommand();

command.CommandText = sql;

Debugger.Log(0, "Debug", $"\n\tINSERT: {sql}\n");

int res = command.ExecuteNonQuery();

Debugger.Log(0, "Debug", $"\n\t\t=======> {res}\n");

return res;

}

catch(Exception g)

{

Debugger.Log(0, "Debug/File", g.ToString());

}

return 0;

}

/// <summary>

/// МЕТОД ДЛЯ ЗАГРУЗКИ ИЗ БАЗЫ

/// </summary>

/// <param name="sql">Запрос</param>

/// <returns>DataTable без манипуляций</returns>

public static DataTable Select(string sql,bool ignoreDebug = false)

{

DataTable table = new DataTable();

try

{

// Незаметная рекурсия т.к. этот метод выполняется в самом подпотоке то подпоток начинает ждать самого себя

// Program.ThreadLoadImage.Join();

var com = SERVER\_SQL.CreateCommand();

com.CommandText = sql;

if (!ignoreDebug)

Debugger.Log(0, "Debug", $"\n\tSELECT: {sql}\n");

var ireader = com.ExecuteReader();

table.Load(ireader);

if (!ignoreDebug)

Debugger.Log(0, "Debug", $"\n\t\t=======> {table.Rows.Count}\n");

var tmp\_sereliz = JsonConvert.SerializeObject(table, Formatting.Indented);

if (!ignoreDebug)

Debugger.Log(0, "Debug", $"{tmp\_sereliz.ToString()}");

ireader.Close();

return table;

}

catch(Exception g)

{

Debugger.Log(0, "Debug/File", JsonConvert.SerializeObject(g,Formatting.Indented));

MessageBox.Show(g.Message, "SQL ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

//Если отработал catch(Exception) возращаем пустую таблицу (формы будут думать что данные отсутствуют)

return table;

}

/// <summary>

/// Метод для получения максимального ID из таблицы

/// </summary>

/// <param name="table"></param>

/// <returns></returns>

public static int GetMaxID(string table,string namerow)

{

int id = -1;

table = table.Trim('[', ']');

namerow = namerow.Trim('[', ']');

var data = Select($"SELECT TOP(1) {namerow} FROM [{table}] ORDER BY [{namerow}] DESC");

if (data.Rows.Count <= 0 || data.Rows.Count > 1)

{

return 0;

}

else

id = int.Parse(data.Rows[0].ItemArray[0].ToString());

return id;

}

/// <summary>

/// Метод для получения и последующего преобразования данных из запроса в динамический тип JArray

/// </summary>

/// <param name="sql">Select - запрос</param>

/// <returns>dynamic объект представления JArray из DataTable</returns>

public static dynamic Select\_Dynamic(string sql)

{

var data = Select(sql);

var tmp\_sereliz = JsonConvert.SerializeObject(data, Formatting.Indented);

var tmp\_dereliz = JsonConvert.DeserializeObject(tmp\_sereliz);

dynamic dyn\_obj = tmp\_dereliz;

return dyn\_obj;

}

/// <summary>

/// Метод подпотока для загрузки изображения

/// </summary>

static internal void UpdateImage()

{

while (SERVER\_SQL.State != ConnectionState.Open)

{

Debugger.Log(0, "Debug", "ThreadLoadImage wait 1000ms for connect server!");

Thread.Sleep(1000);

}

var command = Program.Select("SELECT \* FROM [Images]",true);

Images.Clear();

foreach (DataRow row in command.Rows)

{

Images.Add(int.Parse(row.ItemArray[0].ToString()), Program.Base64ToBitmap(row.ItemArray[1].ToString()));

Debugger.Log(0, "Debug/File", "Image was load: " + row.ItemArray[0].ToString() + " | SIZE: " + row.ItemArray[1].ToString().Length + " .");

}

}

/// <summary>

/// Экземпляр подключения для доп. обращений (берём строку подключения из ресурсов, чтобы не было конфликтов при переносе БД и ПО)

/// </summary>

static internal SqlConnection SERVER\_SQL = new SqlConnection(Properties.Settings.Default.FitnessClubConnectionString);

/// <summary>

/// Словарик для изображений

/// </summary>

static internal Dictionary<int, Bitmap> Images = new Dictionary<int, Bitmap>();

/// <summary>

/// Подпоток для обновления изображений базы

/// </summary>

static internal Thread ThreadLoadImage;

/// <summary>

/// Объект серелиза для перевода

/// </summary>

public static Rootobject\_Translater Translater;

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

//Устанавливаем специальный вывод исключений (Чтобы ловить исключений привязки от MS)

Application.SetUnhandledExceptionMode(UnhandledExceptionMode.ThrowException);

//Подпоток для реализации и проверки подключения к БД

System.Threading.Thread taskConnect = new System.Threading.Thread(() =>

{

try

{

SERVER\_SQL.Open();

}

catch (Exception g)

{

MessageBox.Show("Критическая ошибка подключение к серверу!\n" + g.Message, "Ошибка сервера!", MessageBoxButtons.OK, MessageBoxIcon.Error);

Application.Exit();

}

})

{

IsBackground = true

};

taskConnect.Start();

//Подпоток для фоновой загрузки избр. из быза

ThreadLoadImage = new Thread(UpdateImage)

{

IsBackground = true

};

ThreadLoadImage.Start();

try

{

//Проверка существования файла переводчика

if (System.IO.File.Exists($"{Application.StartupPath}\\rus.json"))

{

//Десериализация java script object notation (JSON)

Translater = JsonConvert.DeserializeObject<JSON.Rootobject\_Translater>(System.IO.File.ReadAllText($"{Application.StartupPath}\\rus.json", System.Text.Encoding.UTF8));

//Проверка списка таблиц

if (Translater == null || Translater.Rus == null || Translater.Rus.Count <= 0)

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

foreach(var ele in Translater.Rus)

{

//Тут мы определяем чего не хватает в сереализаторе

if(ele == null)

{

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

throw new Exception("Ошибка синтаксиса перевода!\nГлавный элемент не найден!");

}

if(ele.NameTable == null)

{

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

throw new Exception("Ошибка синтаксиса перевода!\nНазвание таблицы перевода не найдено!");

}

if(ele.HeaderTable == null)

{

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

throw new Exception("Ошибка синтаксиса перевода!\nПеревод элемента не найден!");

}

foreach(var proper in ele.Property)

{

if(proper == null)

{

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

throw new Exception("Ошибка синтаксиса перевода!\nГлавный элемент свойства не найден! (Proper)");

}

if(proper.HeaderProperty == null)

{

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

throw new Exception("Ошибка синтаксиса перевода!\nПеревод элемента таблицы не найден! (HeaderProperty)");

}

if(proper.NameProperty == null)

{

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

throw new Exception("Ошибка синтаксиса перевода!\nПеревод элемента свойства не найден! (NameProperty)");

}

}

}

}

else

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

}

catch (Exception g)

{

MessageBox.Show("Program.cs [Translater initializator]\n" + g.Message, "PROGRAMM ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

MessageBox.Show("В инициализаторе перевода возникла ошибка!\nДальнейшей перевод не возможен!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

//При полном отсутствие или неполноценности данных инициализуем пустой экземпляр (в ином случае будет NullRefrences везде где используется этот экземпляр, по другому говоря это заглушка)

Translater = new JSON.Rootobject\_Translater() { Rus = new List<JSON.Tables>() };

}

Properties.Settings.Default.Role = GetRole(Role.User);

Properties.Settings.Default.Save();

//Базовый код WinForms для запуска

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

Титульная форма

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net.NetworkInformation;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.VisualStyles;

using static FitnessClub.FitnessClubDataSet;

namespace FitnessClub

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

dynamic\_style.Enable(this);

}

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

{

this.Close();

}

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

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "fitnessClubDataSet.MembershipTypes". При необходимости она может быть перемещена или удалена.

this.membershipTypesTableAdapter.Fill(this.fitnessClubDataSet.MembershipTypes);

}

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

{

AuthMainForm a = new AuthMainForm();

this.Hide();

a.ShowDialog(this);

this.Show();

this.MainForm\_Load(sender, e);

}

private void dataGridMembershipsType\_CurrentCellChanged(object sender, EventArgs e)

{

var rowDATA = GetRowByGrid();

labtextinfo.Text = $"Наименование: {rowDATA[1]}{Environment.NewLine}{Environment.NewLine}Описание: {rowDATA[2]}{Environment.NewLine}{Environment.NewLine}Стоимость: {rowDATA[3]} р.{Environment.NewLine}{Environment.NewLine}Период: {rowDATA[4]} мес.";

}

private object[] GetRowByGrid()

{

object[] a = new object[5];

a[0] = int.Parse((membershipTypesBindingSource.Current as DataRowView)[0].ToString());

a[1] = (membershipTypesBindingSource.Current as DataRowView)[1].ToString();

a[2] = (membershipTypesBindingSource.Current as DataRowView)[2].ToString();

a[3] = int.Parse((membershipTypesBindingSource.Current as DataRowView)[3].ToString());

a[4] = int.Parse((membershipTypesBindingSource.Current as DataRowView)[4].ToString());

return a;

}

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

{

var data = GetRowByGrid();

MainRegisterForm regF = new MainRegisterForm((int)data[0], data[1].ToString(), data[2].ToString(), (int)data[3], data[4].ToString());

this.Hide();

var result = regF.ShowDialog(this);

this.Show();

if (result == DialogResult.Retry)

{

btnAuth?.PerformClick();

}

}

}

}

Форма регистрации

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Net.Mail;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FitnessClub

{

public partial class MainRegisterForm : Form

{

public MainRegisterForm(int id,string name,string description,int price,string duration)

{

InitializeComponent();

dynamic\_style.Enable(this);

this.lab\_reg\_on.Text = $"Регистрация на \'{name}\'";

this.INDEX = id;

this.NameA = name;

this.DescriptionA = description;

this.PriceA = price;

this.DurationA = duration;

}

private int INDEX;

private string NameA;

private string DescriptionA;

private int PriceA;

private string DurationA;

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

{

Process.Start(@"https://online.sberbank.ru/");

}

public bool IsValid(string emailaddress)

{

try

{

MailAddress m = new MailAddress(emailaddress);

return true;

}

catch

{

return false;

}

}

private static Random rand = new Random();

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

{

Control[] validation\_control = new Control[]

{

txt\_name,

txt\_surname,

txt\_date,

txtNumber,

txtAddress,

txtEmail,

};

bool validationOnNull = true;

DateTime dateBirth = new DateTime();

if (string.IsNullOrWhiteSpace(txt\_name.Text) || string.IsNullOrWhiteSpace(txt\_surname.Text) || !DateTime.TryParse(txt\_date.Text, out dateBirth) || string.IsNullOrWhiteSpace(txtNumber.Text) || string.IsNullOrWhiteSpace(txtAddress.Text))

validationOnNull = false;

if (!IsValid(txtEmail.Text))

validationOnNull = false;

if (validationOnNull)

{

int code = rand.Next(1111, 9999);

Thread th = new Thread(() =>

{

try

{

string from = "techicalsuhinin@yandex.ru";

string pass = "gezmmycdvpihuuay";//Раньше тут был настоящий пароль от аккаунта

SmtpClient client = new SmtpClient();

client.Host = "smtp.yandex.ru";

//Порт - незащищённый

client.Port = 587;

//Но при этом используем SSL защиту

client.EnableSsl = true;

client.Credentials = new System.Net.NetworkCredential(from, pass);

Debugger.Log(0, "Debug", $"\n\tНачало отправки: {code}\n");

client.Send(from, txtEmail.Text, "Регистрация в FitnessClub", $"Здравствуйте {txt\_name.Text} {txt\_surname.Text}{Environment.NewLine}Вы оставили заявку на покупку абонемента в нашем фитнес клубе.{Environment.NewLine}Время: \"{System.DateTime.Now.ToShortDateString()} {System.DateTime.Now.ToShortTimeString()}\"\nОб абонемента:{Environment.NewLine}Наименование: {NameA}{Environment.NewLine}Описание: {DescriptionA}{Environment.NewLine}Стоимость: {PriceA} руб. {Environment.NewLine}Период: {DurationA} месяцев. {Environment.NewLine}{Environment.NewLine}Ваш код подтверждения заявки: \"{code}\"{Environment.NewLine}Не закрывайте приложение, в ином случае придется заполнять данные заново.");

Debugger.Log(0, "Debug", $"\n\tОкончание отправки!\n");

}

catch(Exception g)

{

MessageBox.Show(this, g.Message, "Ошибка сети SMTP протокол", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

});

th.Start();

GetRegisterCodeForm codeform = new GetRegisterCodeForm(txtEmail.Text);

var result = codeform.ShowDialog(this);

if (result == DialogResult.OK && codeform.GetCode == code)

{

//ПОСЛЕ ТОГО КАК ПОЛЬЗОВАТЕЛЬ ВВЕЛ СВОИ ДАННЫЕ И ПОСЛЕ ВВОДКА КОДА С ПОЧТЫ

//ДЕЛАЕМ ЗАПИСЬ НА СОЗДАНИЯ КЛИЕНТА (КОТОРЫЙ В БУДУЩЕМ АДМИН БУДЕТ РЕШАТЬ ОДОБРИТЬИЛИ НЕТ)

var command = Program.SERVER\_SQL.CreateCommand();

int id\_new\_member = -1;

var tmp\_command = Program.SERVER\_SQL.CreateCommand();

tmp\_command.CommandText = "SELECT TOP(1) [ClientID] FROM [Clients] ORDER BY [ClientID] DESC";

var tmp\_reader = tmp\_command.ExecuteReader();

DataTable tmp\_table = new DataTable();

tmp\_table.Load(tmp\_reader);

tmp\_reader.Close();

if (int.TryParse(tmp\_table.Rows[0].ItemArray[0].ToString(), out id\_new\_member))

{

//ЗАПИСЬ В ТАБЛИЦУ КЛИЕНТЫ

id\_new\_member++;

command.CommandText = $"INSERT INTO [Clients] VALUES({id\_new\_member},\'{txt\_name.Text}\',\'{txt\_surname.Text}\',\'{txt\_date.Text}\',\'{txtNumber.Text}\',\'{txtEmail.Text}\',\'{txtAddress.Text}\',NULL)";

command.ExecuteNonQuery();

//ЗАПИСЬ В ТАБЛИЦУ Memberships (КОД INDEX - ИД ДЛЯ АБОНЕМЕНТА, КОД ПЗ)

//STARTDATE - Высчитывается от момента заявки до +МЕСЯЦ АБОНЕМЕНТА

int id\_new\_memberships = Program.GetMaxID("Memberships","MembershipID")+1;

string startDate = System.DateTime.Now.ToShortDateString();

string endDate = (System.DateTime.Now.AddMonths(int.Parse(DurationA))).ToShortDateString();

Program.Insert($"INSERT INTO [Memberships] VALUES({id\_new\_memberships},{id\_new\_member},{INDEX},\'{startDate}\',\'{endDate}\')");

//ДОБАВЛЕНИЕ КОДА ЗАПИСИ Memberships в ТАБЛИЦУ Payments ДЛЯ ПОНИМАНИЯ ОБ ОПЛАТЕ

int id\_new\_payment = Program.GetMaxID("[Payments]", "[PaymentID]") + 1;

Program.Insert($"INSERT INTO [Payments] VALUES({id\_new\_member},{id\_new\_memberships},\'{System.DateTime.Now.ToShortDateString()}\',{PriceA},\'Онлайн\')");

//ОТКАЗ ОТ ЛОГИКИ ОДОБРЕНИЯ РЕГИСТРАЦИИ ПОЛЬЗОВАТЕЛЯ

//ДОП.ФОРМА: ОКОНЧАТЕЛЬНАЯ РЕГИСТРАЦИИ ИНФ.АККАУНТА В ТАБЛИЦЕ USERS

string passN = System.Web.Security.Membership.GeneratePassword(12, 0);

int id\_new\_users = Program.GetMaxID("Users", "ID")+1;

Program.Insert($"INSERT INTO [Users] VALUES({id\_new\_users},{id\_new\_member},\'{txtEmail.Text.Split('@')[0]}\',\'{passN}\',\'User\')");

try

{

string from = "techicalsuhinin@yandex.ru";

string pass = "gezmmycdvpihuuay";

SmtpClient client = new SmtpClient();

client.Host = "smtp.yandex.ru";

client.Port = 587;

client.EnableSsl = true;

client.Credentials = new System.Net.NetworkCredential(from, pass);

Debugger.Log(0, "Debug", $"\n\tНачало отправки: {passN}\n");

client.Send(from, txtEmail.Text, "Регистрация в FitnessClub", $"Здравствуйте {txt\_name.Text} {txt\_surname.Text}{Environment.NewLine}{Environment.NewLine}Логин: {txtEmail.Text}{Environment.NewLine}Пароль: {passN}{Environment.NewLine}Никому не сообщайте ваши данные!");

Debugger.Log(0, "Debug", $"\n\tОкончание отправки!\n");

MessageBox.Show(this, "Вы успешно зарегистрировались!\nЛогин и пароль для входа отправлен вам на почту!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.DialogResult = DialogResult.Retry;

this.Close();

}

catch(Exception g)

{

MessageBox.Show(this, g.Message, "Ошибка сети SMTP протокол", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

MessageBox.Show(this, "Не удалось зарегистрировать пользователя с такими ключём!.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if(result == DialogResult.OK && codeform.GetCode != code)

{

if(MessageBox.Show(this,"Введен не правильный код!\nПовторить попытку?","Ошибка",MessageBoxButtons.YesNo,MessageBoxIcon.Warning) == DialogResult.Yes)

{

btnRegister.PerformClick();

}

}

else if(result == DialogResult.Abort)

{

//ЕСЛИ КОД НЕ ПРИШЁЛ ДЕЛАЕМ ЛОКАЛЬНЫЙ КОД

MessageBox.Show("Ошибка авторизации","Не удалось",MessageBoxButtons.OK,MessageBoxIcon.Hand);

}

}

else

{

MessageBox.Show(this, "Не все поля заполнены!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.DialogResult = DialogResult.Cancel;

this.Close();

}

private void btnGO\_IN\_A\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

this.DialogResult = DialogResult.Retry;

this.Close();

}

}

}

Главная форма авторизации

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 FitnessClub

{

public partial class AuthMainForm : Form

{

public AuthMainForm()

{

InitializeComponent();

dynamic\_style.Enable(this);

if (Properties.Settings.Default.SaveAuth)

{

ckc\_save.Checked = true;

if (!string.IsNullOrWhiteSpace(Properties.Settings.Default.Login) || !string.IsNullOrWhiteSpace(Properties.Settings.Default.Password))

{

txtLogin.Text = Properties.Settings.Default.Login;

txtPass.Text = Properties.Settings.Default.Password;

}

}

}

private void chc\_pass\_CheckedChanged(object sender, EventArgs e)

{

txtPass.UseSystemPasswordChar = !chc\_pass.Checked;

}

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

{

this.DialogResult = DialogResult.Cancel;

this.Close();

}

private void ckc\_save\_CheckedChanged(object sender, EventArgs e)

{

Properties.Settings.Default.SaveAuth = ckc\_save.Checked;

}

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

{

Properties.Settings.Default.Login = txtLogin.Text;

}

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

{

Properties.Settings.Default.Password = txtPass.Text;

}

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

{

Properties.Settings.Default.Save();

}

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

{

if (!string.IsNullOrWhiteSpace(txtLogin.Text) && !string.IsNullOrWhiteSpace(txtPass.Text))

{

var command = Program.SERVER\_SQL.CreateCommand();

command.CommandText = $"SELECT \* FROM [Users] WHERE [Login] = \'{txtLogin.Text}\' and [Password] = \'{txtPass.Text}\'";

var reader = command.ExecuteReader();

DataTable tmp\_table = new DataTable();

tmp\_table.Load(reader);

reader.Close();

//Проверка если нету элементов аккаунтов или аккаунтов больше чем нужно

if (tmp\_table.Rows.Count <= 0 || tmp\_table.Rows.Count > 1)

{

MessageBox.Show(this, "Введен неверный логин или пароль!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

string roly = tmp\_table.Rows[0].ItemArray[4].ToString();

if (roly.ToUpper() == "ADMIN")

Properties.Settings.Default.Role = Program.GetRole(Role.Admin);

else

Properties.Settings.Default.Role = Program.GetRole(Role.User);

string ids = tmp\_table.Rows[0].ItemArray[1].ToString();

int id = -1;

if (int.TryParse(ids, out id))

{

Properties.Settings.Default.ID = id;

//Сохраняем выше указанные параметры только в том случае если авторизация успешна и пользователь существует

Properties.Settings.Default.Save();

MainHandlerForm handler = new MainHandlerForm();

this.Hide();

handler.ShowDialog(this);

this.Show();

}

else if(Properties.Settings.Default.Role == Program.GetRole(Role.User))

{

MessageBox.Show(this, "Не удалось идентифицировать пользователя!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if(Properties.Settings.Default.Role == Program.GetRole(Role.Admin))

{

Properties.Settings.Default.ID = -1;

Properties.Settings.Default.Save();

MainHandlerForm handler = new MainHandlerForm();

this.Hide();

handler.ShowDialog(this);

this.Show();

}

}

}

else

{

MessageBox.Show(this, "Не все поля заполнены!", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void txtLogin\_KeyUp(object sender, KeyEventArgs e)

{

if(e.KeyCode == Keys.Enter)

{

btnAuth.PerformClick();

}

}

}

}

Главная форма для выбора таблиц

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FitnessClub

{

public partial class MainHandlerForm : Form

{

public MainHandlerForm()

{

InitializeComponent();

dynamic\_style.Enable(this);

}

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

{

this.DialogResult = DialogResult.Cancel;

this.Close();

}

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

{

try

{

Lenta.Trainers\_Form trainers = new Lenta.Trainers\_Form();

this.Hide();

trainers.ShowDialog(this);

}

catch(Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.ClassEquipment\_Form equi = new Lenta.ClassEquipment\_Form();

this.Hide();

equi.ShowDialog(this);

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.Classes\_Form classes = new Lenta.Classes\_Form();

this.Hide();

classes.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

private void btnMe\_Classes\_Click(object sender, EventArgs e)

{

//ДАННОЕ ОКНО ПРЕДСТАВЛЯЕТ СОБОЙ ВОЗМОЖНОСТЬ ПРОСМОТРЕТЬ ЗАНЯТИЯ НА КОТОРЫЕ ПОЛЬЗОВАТЕЛЬ

//БЫЛ ЗАПИСАН

//ОТСЛЕЖИВАНИЕ ИДЁТ ПО ЛОГИКИ -> ТАБЛИЦА ClientClassRegistration И ЕСЛИ ЕСТЬ ЗАПИСЬ КОНКРЕТНОГО\_ПОЛЬЗОВАТЕЛЯ и КОНКРЕТНОГО\_ЗАНЯТИЯ зНАЧИТ ОН БЫЛ НА ЗАНЯТИЕ

try

{

Other.MeClasses\_Form me = new Other.MeClasses\_Form();

this.Hide();

me.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

Other.MeAccount\_Form account = new Other.MeAccount\_Form();

this.Hide();

account.ShowDialog(this);

this.Show();

}

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

{

try

{

Lenta.Equipment\_Form equipment = new Lenta.Equipment\_Form();

this.Hide();

equipment.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.Membershis\_Form memberships = new Lenta.Membershis\_Form();

this.Hide();

memberships.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.MembershipTypes\_Form membershipsT = new Lenta.MembershipTypes\_Form();

this.Hide();

membershipsT.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.Clients\_Form clients = new Lenta.Clients\_Form();

this.Hide();

clients.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.ClassSchedules\_Form classS = new Lenta.ClassSchedules\_Form();

this.Hide();

classS.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.ClientClassRegistrations\_Form reg = new Lenta.ClientClassRegistrations\_Form();

this.Hide();

reg.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.Users\_Form reg = new Lenta.Users\_Form();

this.Hide();

reg.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.Payments\_Form pay = new Lenta.Payments\_Form();

this.Hide();

pay.ShowDialog(this);

this.Show();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

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

{

try

{

Lenta.Images\_Form img = new Lenta.Images\_Form();

img.ShowDialog(this);

}

catch(Exception g)

{

MessageBox.Show(this, g.Message, "Критическая ошибка! Действие приостановлено", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

this.Show();

}

}

}

}

Форма сортировки и поиска (Обобщённая)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

//////////////////////////

// П-46 //

// Автор: Сухинин К.Д. //

// 10.2024 //

//P.S> Использую //

//обобщённый //

//подход к данным //

//////////////////////////

namespace FitnessClub.Tables

{

public partial class Table\_Common\_Handler : Form

{

public Table\_Common\_Handler(string NameTable)

{

InitializeComponent();

dynamic\_style.Enable(this);

this.NameTable = NameTable;

}

public void REFRESH\_TEST(string newTable)

{

this.NameTable = newTable;

RefreshTable();

}

public Table\_Common\_Handler(IDbCommand table)

{

InitializeComponent();

dynamic\_style.Enable(this);

this.NameTable = null;

this.\_TABLE = table;

}

/// <summary>

/// Поле для хранения названия таблицы (тех.название)

/// </summary>

private string NameTable;

private IDbCommand \_TABLE;

/// <summary>

/// Загружаем и отображаем актуальные данные таблицы

/// </summary>

private void RefreshTable()

{

var translater = Program.Translater.Rus.Where(x => x.NameTable == NameTable);

JSON.Tables translater\_prop = null;

if (NameTable != null)

{

string title = NameTable;

if (translater != null)

{

if (translater.Count() > 0)

{

title = translater.FirstOrDefault().HeaderTable;

translater\_prop = translater.FirstOrDefault();

}

}

this.Text = "Сортировка \'" + title + "\'";

}

else

{

// this.Text = $"Сортировка \'НЕ ОПРЕДЕЛЕННО\'";

}

var dbo\_tableView = Program.SERVER\_SQL.CreateCommand();

if (NameTable != null)

dbo\_tableView.CommandText = $"SELECT \* FROM [{NameTable}]";

else

dbo\_tableView = (SqlCommand)\_TABLE;

var reader = dbo\_tableView.ExecuteReader();

DataTable table = new DataTable();

table.Load(reader);

list\_columns.Items.Clear();

cb\_collumns.Items.Clear();

foreach (DataColumn co in table.Columns)

{

string co\_columnName = co.ColumnName;

if (translater\_prop != null && translater\_prop.Property.Where(x => x.NameProperty == co.ColumnName).Count() > 0)

{

co\_columnName = translater\_prop.Property.Where(x => x.NameProperty == co.ColumnName).FirstOrDefault().HeaderProperty;

}

else if (translater\_prop == null)

{

foreach (var v in Program.Translater.Rus)

{

var vs = v.Property.Where(x => x.NameProperty == co\_columnName);

if (vs != null && vs.Count() > 0)

co\_columnName = vs.FirstOrDefault().HeaderProperty;

}

}

list\_columns.Items.Add(co\_columnName);

cb\_collumns.Items.Add(co\_columnName);

}

dataGridView1.DataSource = table;

foreach (DataGridViewColumn cl in dataGridView1.Columns)

{

if (translater\_prop != null && translater\_prop.Property.Where(x => x.NameProperty == cl.Name).Count() > 0)

cl.HeaderText = translater\_prop.Property.Where(x => x.NameProperty == cl.Name).FirstOrDefault().HeaderProperty;

else if (translater\_prop == null)

{

foreach (var v in Program.Translater.Rus)

{

var vs = v.Property.Where(x => x.NameProperty == cl.Name);

if (vs != null && vs.Count() > 0)

cl.HeaderText = vs.FirstOrDefault().HeaderProperty;

}

}

}

reader.Close();

dynamic\_style.Enable(this);

}

private void Table\_Common\_Handler\_Load(object sender, EventArgs e)

{

RefreshTable();

}

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

{

this.DialogResult = DialogResult.Cancel;

this.Close();

}

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

{

//Проверка на выбор колонки

if (cb\_collumns.SelectedIndex == -1)

{

MessageBox.Show(this, "Выберите колонку из выпадающего списка.", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);

//Выход из функции если не был выбран индекс в вышестоящем элементе (В ином случае: System.ArgumentOutOfRangeException обеспечен!)

return;

}

//Убираем выбранную ячейку пользователя (В ином случае мы не сможем её скрыть если он не содержит нужные данные поиска)

dataGridView1.CurrentCell = null;

//Скрываем все данные

foreach (DataGridViewRow vc in dataGridView1.Rows)

{

vc.Visible = false;

}

//Определяем значение поиска (Возможно использование только Text или Items[INDEX]

string teFind = "NONE";

if (cbFind.SelectedIndex != -1)

{

teFind = cbFind.Items[cbFind.SelectedIndex].ToString();

}

else

teFind = cbFind.Text;

//Удаляем указательные данные для пользователя на кол.во совпадений

if (teFind.Contains("→"))

teFind = teFind.Split('→')[0];

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

//Проверяем содержит ли указаная ячейка нужные данные

if (dataGridView1.Rows[i].Cells[cb\_collumns.SelectedIndex].Value.ToString().Contains(teFind))

{

//Отображаем всё строку

dataGridView1.Rows[i].Visible = true;

//Подсвечиваем (нужно если большое кол.во совпадений)

// dataGridView1.Rows[i].Cells[cb\_collumns.SelectedIndex].Style.BackColor = Color.GreenYellow;

}

}

}

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

{

RefreshTable();

}

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

{

//Тернарный оператор для перечисления на основе Радио кнопок

ListSortDirection sortAlgorithm = radioButton1.Checked ? ListSortDirection.Ascending : ListSortDirection.Descending;

if (list\_columns.SelectedIndex != -1)

{

//Примитивная сортировка

var column = dataGridView1.Columns[list\_columns.SelectedIndex];

dataGridView1.Sort(column, sortAlgorithm);

dynamic\_style.Enable(this);

}

}

private void cb\_collumns\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cb\_collumns.SelectedIndex == -1)

return;

cbFind.Items.Clear();

//Лист на проверку данных (Чтобы одни и те же данные не добавлялись)

List<string> checkList = new List<string>();

//Лист для подсчёта повторяющихся данных

Dictionary<string, int> checkerFunc = new Dictionary<string, int>();

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

//Проверка на чек-листе

if (checkList.Where(x => x == dataGridView1.Rows[i].Cells[cb\_collumns.SelectedIndex].Value.ToString()).Count() > 0)

{

//Ветления для ведение подсчёта кол.во повторяющихся данных имеет один и тот же вид данных. (И Соответственно примитивная логика проверки для Dictionary<T>)

//Если таймер/счётчик для вида данных существует то увеличиваем его

//В ином случае мы добавляем этот вид данных и инициализуем ему счётчик с числом 2 т.к. первое вхождение уже было зафиксировано при первичном добавлении данных в рабочий элемент

if (checkerFunc.Where(x => x.Key == dataGridView1.Rows[i].Cells[cb\_collumns.SelectedIndex].Value.ToString()).Count() > 0)

checkerFunc[dataGridView1.Rows[i].Cells[cb\_collumns.SelectedIndex].Value.ToString()]++;

else

checkerFunc.Add(dataGridView1.Rows[i].Cells[cb\_collumns.SelectedIndex].Value.ToString(), 2);

//Пропускаем добавление одних и тех же данных

continue;

}

checkList.Add(dataGridView1.Rows[i].Cells[cb\_collumns.SelectedIndex].Value.ToString());

cbFind.Items.Add(dataGridView1.Rows[i].Cells[cb\_collumns.SelectedIndex].Value);

}

//Добавление итогов кол.во повторений имеет один вид данных

for (int i = 0; i < cbFind.Items.Count; i++)

{

//Добавляем счётчики к контрольному элементу

if (checkerFunc.Where(x => x.Key == cbFind.Items[i].ToString()).Count() > 0)

{

if (checkerFunc[cbFind.Items[i].ToString()] > 0)

cbFind.Items[i] = $"{cbFind.Items[i]}→{checkerFunc.Where(x => x.Key == cbFind.Items[i].ToString()).FirstOrDefault().Value}";

}

}

//Сортируем отображения текстовых подсказок в один порядок

//В ИНОМ СЛУЧАЕ: При внешней сортировки (пользователем) данные отображаются в зависимости от сортировки

//ИЗМ: Пхаха переводим в список объектов, потом в лист, лист итерационно в строку и IEnumerable<string> переводим обратно в List<Object> в ином случае сортировка типов decimal,int,double,float,DateTime и д.р. невозможна!

var copyObj = cbFind.Items.Cast<object>().ToList().Select(x => x.ToString()).ToList<object>();

//Удаляем старые несортированные данные

cbFind.Items.Clear();

//Сортируем список

copyObj.Sort();

//Добавляем на контрольный элемент

cbFind.Items.AddRange(copyObj.ToArray());

}

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

{

//P.S> Как по мне лучше использовать средства Excel чем обращатся к устаревшим интерфейсам печати

//Алгоритм сохранение данных в CSV таблицу для дальнейшей печати

//Диалоговое окно создания/выборки файла для сохранения

SaveFileDialog save = new SaveFileDialog();

save.Title = "Сохранить .csv файл";

//Установка ограничение на создание ТОЛЬКО .CSV ФАЙЛОВ

//Выбрал .CSV файла т.к. они легко интепретируются (создаются/изменяются) в любом формате

//И после создания таково файла возможна печать с помощью средств Office

save.Filter = "\*.csv|\*.csv";

if (save.ShowDialog(this) == DialogResult.OK)

{

try

{

//Строки для создания в .CSV

var lines = new List<string>();

//Выборка столбцов (для реализации изменения)

List<string> columns = dataGridView1.Columns.Cast<DataGridViewColumn>().Select(x => x.Name).ToList();

if (MessageBox.Show(this, "Изменить отображение столбцов?", "?", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Other.ChangeStructureOutputTableForm handlerColumns = new Other.ChangeStructureOutputTableForm(columns, dataGridView1.Rows[0].Cells.Cast<DataGridViewCell>().Select(x => x.Value.ToString()).ToList());

if (handlerColumns.ShowDialog(this) == DialogResult.OK)

columns = handlerColumns.Result;

}

//Наименование столбцов в виде IEnumerable<string>

string[] columnNames = dataGridView1.Columns.Cast<DataGridViewColumn>().Where(x => columns.Where(z => z == x.Name).FirstOrDefault() != null).Cast<DataGridViewColumn>().Select(x => x.HeaderText).ToArray();

//Подставляем ориг. именование столбцов если нужны тех. значение в заголовках .csv

if (MessageBox.Show(this, $"Оставить оригинальные наименование столбцов?\nПример: \'{dataGridView1.Columns[0].HeaderText}\' -> \'{dataGridView1.Columns[0].Name}\'", "?", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

columnNames = dataGridView1.Columns.Cast<DataGridViewColumn>().Where(x => columns.Where(z => z == x.Name).FirstOrDefault() != null).Cast<DataGridViewColumn>().Select(x => x.Name).ToArray();

}

//Объединяем IEnumerable<string> в string через ';' т.к. это основной формат .csv

var header = string.Join(";", columnNames.Select(name => $"\"{name}\""));

//Добавляем заголовки/столбцы для прим. отображения

lines.Add(header);

//IEnumerable<string> который в себя содержит все представленные rows

List<string> valueLines = new List<string>();

//Алгоритм добавления ROW[N] в итоговую таблицу с учётом форматирование

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

//Только доступные строки

if (dataGridView1.Rows[i].Visible)

{

//IEnumerable<string> который будет содержать обрезанный row относительно тех столбцов которые оставил пользователь на моменте "Изменение структуры столбцов"

List<string> add = new List<string>();

for (int c = 0; c < dataGridView1.Rows[i].Cells.Count; c++)

{

//Проверка является ли столбец отображаемым или нет

if (columns.Where(x => x == dataGridView1.Columns[c].Name).Count() > 0)

{

add.Add(dataGridView1.Rows[i].Cells[c].Value.ToString());

}

else

continue;

}

//Если в ROW есть хоть одно значение то добавляем его с форматирование через ';' формата .csv

if (add.Count > 0)

valueLines.Add(string.Join(";", add));

}

}

//Добавляем IEnumerable<string> представления N экземпляра данных (Всех rows) к HEADER

lines.AddRange(valueLines);

//Итог: мы получим .csv таблицу которая будет содержать только те столбцы которые оставил пользователь

//НЕ БАГ А ФИЧА: т.к. данные берутся напрямую из DataGridView, а не из Источника то в таблице будут данные которые видит пользователь на форме отображения в момент нажатия данной кнопки, со всеми сортировками и со всеми фильтрами (кроме цвета и выделения)

//P.S.> Не знаю как менять стили ячейкам в .csv когда сам .csv файл состоит из примитивного текста с разделителями (Стили реализуемы, но факт того что при добавлении html и иных тэгов для стиля может слететь кириллица в данных существует)

System.IO.File.WriteAllLines(save.FileName, lines, System.Text.Encoding.UTF8);

MessageBox.Show(this, $"Таблица успешна записана в \'{new FileInfo(save.FileName).Name}\' и готова для печати.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

try

{

//без понятия зачем вообще эта кнопка нужна)

printDialog1.Document = printDocument1;

printDialog1.ShowDialog(this);

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void printDocument1\_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)

{

try

{

Bitmap bt = new Bitmap(dataGridView1.Width + 10, dataGridView1.Height + 10);

dataGridView1.DrawToBitmap(bt, dataGridView1.Bounds);

e.Graphics.DrawImage(bt, new PointF(0, 0));

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView1\_CurrentCellChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedCells.Count > 0)

{

int indexRow = dataGridView1.SelectedCells[0].RowIndex;

lab\_counter.Text = $"{indexRow + 1} для {dataGridView1.Rows.Count}";

}

else

{

lab\_counter.Text = $"-";

}

}

}

}

Листинг форма классов

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 FitnessClub.Lenta

{

public partial class Classes\_Form : Form

{

public Classes\_Form()

{

InitializeComponent();

dynamic\_style.Enable(this);

}

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

{

try

{

this.Validate();

this.classesBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.fitnessClubDataSet);

MessageBox.Show(this, "Сохранение успешно!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch(Exception g)

{

MessageBox.Show(this, "Не удалось применить сохранения!\n"+g.Message+"\nИзменения будут отменены!", "Критическая ошибка данных!", MessageBoxButtons.OK, MessageBoxIcon.Error);

classesBindingSource.CancelEdit();

}

}

private void Classes\_Form\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "fitnessClubDataSet.Trainers". При необходимости она может быть перемещена или удалена.

this.trainersTableAdapter.Fill(this.fitnessClubDataSet.Trainers);

// TODO: данная строка кода позволяет загрузить данные в таблицу "fitnessClubDataSet.Classes". При необходимости она может быть перемещена или удалена.

this.classesTableAdapter.Fill(this.fitnessClubDataSet.Classes);

}

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

{

classesBindingSource.MoveFirst();

}

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

{

classesBindingSource.MovePrevious();

}

private void btn\_Next\_Click(object sender, EventArgs e)

{

classesBindingSource.MoveNext();

}

private void btn\_last\_Click(object sender, EventArgs e)

{

classesBindingSource.MoveLast();

}

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

{

classesBindingSource.CancelEdit();

}

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

{

this.DialogResult = DialogResult.Cancel;

this.Close();

}

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

{

Tables.Table\_Common\_Handler classes = new Tables.Table\_Common\_Handler("Classes");

classes.Show(this);

}

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

{

try

{

if (MessageBox.Show(this, "Вы уверены что хотите удалить запись?", "?", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

bindingNavigatorDeleteItem.PerformClick();

if (MessageBox.Show(this, "Применить удаление?\nЭто действие нельзя будет вернуть", "Применить?", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

classesBindingNavigatorSaveItem?.PerformClick();

// MessageBox.Show(this, "Сохранение успешно!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Ошибка удаления!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

bindingNavigatorAddNewItem?.PerformClick();

int newUID = int.Parse(bindingNavigatorPositionItem.Text);

classIDComboBox.Text = newUID.ToString();

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Ошибка добавления!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

classesBindingNavigatorSaveItem?.PerformClick();

//MessageBox.Show(this, "Данные сохранены успешно!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception g)

{

MessageBox.Show(this, g.Message, "Ошибка сохранения!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btn\_registrationOnClasses\_Click(object sender, EventArgs e)

{

int ClientID = Properties.Settings.Default.ID;

if(ClientID == -1)

{

MessageBox.Show(this, "Пользователь для записи не найден!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int ClassesID = int.Parse((classesBindingSource.Current as DataRowView).Row.ItemArray[0].ToString());

var dataSchedule = Program.Select\_Dynamic($"SELECT \* FROM [ClassSchedules] WHERE [ClassID] = \'{ClassesID}\'");

bool wasRegistration = false;

if(dataSchedule != null)

{

foreach(var schedule in dataSchedule)

{

var clientRegistration = Program.Select\_Dynamic($"SELECT \* FROM [ClientClassRegistrations] WHERE [ClientID] = \'{ClientID}\' and [ClassScheduleID] = \'{schedule.ClassScheduleID.ToString()}\'");

if((int)(clientRegistration.Count) > 0)

{

wasRegistration = true;

MessageBox.Show(this, $"Данный пользователь был записан в \'{clientRegistration[0].RegisteredAt.ToString()}\'", "Ошибка записи!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

if (!wasRegistration)

{

if(MessageBox.Show(this,$"Вы уверены что хотите записаться на \'{(classesBindingSource.Current as DataRowView).Row.ItemArray[1]}\'","Подтвердите",MessageBoxButtons.YesNo,MessageBoxIcon.Question) == DialogResult.Yes)

{

int schedules\_id\_new = Program.GetMaxID("ClassSchedules", "ClassScheduleID")+1;

int schedule\_result = Program.Insert($"INSERT INTO [ClassSchedules] VALUES({schedules\_id\_new},{ClassesID},\'{System.DateTime.Now.ToShortDateString()}\')");

if(schedule\_result == 1)

{

int registration\_id\_new = Program.GetMaxID("ClientClassRegistrations","RegistrationID")+1;

int registration\_result = Program.Insert($"INSERT INTO [ClientClassRegistrations] VALUES({registration\_id\_new},{ClientID},{schedules\_id\_new},\'{System.DateTime.Now.ToShortDateString()} {DateTime.Now.ToShortTimeString()}\')");

if(registration\_result == 1)

{

MessageBox.Show(this, "Вы успешно записаны на занятие!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show(this, "Запись [ClientClassRegistrations] вернула: " + registration\_result.ToString() + "\nОбратитесь к администратору системы!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show(this, "Запись [ClassSchedules] вернула: " + schedule\_result.ToString() + "\nОбратитесь к администратору системы!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

}

Листинг форма тренеры

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FitnessClub.Lenta

{

public partial class Trainers\_Form : Form

{

public Trainers\_Form()

{

InitializeComponent();

dynamic\_style.Enable(this);

}

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

{

try

{

this.Validate();

this.trainersBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.fitnessClubDataSet);

MessageBox.Show(this, "Сохранение успешно!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch(Exception g)

{

MessageBox.Show(this, "Не удалось применить сохранения!\n" + g.Message + "\nИзменения будут отменены!", "Критическая ошибка данных!", MessageBoxButtons.OK, MessageBoxIcon.Error);

trainersBindingSource.CancelEdit();

}

}

private void Trainers\_Form\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "fitnessClubDataSet.Images". При необходимости она может быть перемещена или удалена.

this.imagesTableAdapter.Fill(this.fitnessClubDataSet.Images);

// TODO: данная строка кода позволяет загрузить данные в таблицу "fitnessClubDataSet.ClassEquipment". При необходимости она может быть перемещена или удалена.

this.classEquipmentTableAdapter.Fill(this.fitnessClubDataSet.ClassEquipment);

// TODO: данная строка кода позволяет загрузить данные в таблицу "fitnessClubDataSet.Classes". При необходимости она может быть перемещена или удалена.

this.classesTableAdapter.Fill(this.fitnessClubDataSet.Classes);

// TODO: данная строка кода позволяет загрузить данные в таблицу "fitnessClubDataSet.Trainers". При необходимости она может быть перемещена или удалена.

this.trainersTableAdapter.Fill(this.fitnessClubDataSet.Trainers);

}

private void Initialize\_ListClassesTrainers()

{

// lst\_classes\_by\_trainers.Items.Clear();

// lst\_classes\_by\_trainers.ForeColor = Color.Black;

var data = Program.Select\_Dynamic($"SELECT \* FROM [Classes] WHERE [TrainerID] = \'{(trainersBindingSource.Current as DataRowView).Row.ItemArray[0]}\'");

if(((int)(data.Count) > 0))

{

dataGrid\_classe\_by\_trainer.Columns.Clear();

dataGrid\_classe\_by\_trainer.Rows.Clear();

dataGrid\_classe\_by\_trainer.DataSource = null;

dataGrid\_classe\_by\_trainer.Columns.Add("Наименование", "Наименование");

dataGrid\_classe\_by\_trainer.Columns.Add("День недели","День недели");

dataGrid\_classe\_by\_trainer.Columns.Add("Начало", "Начало");

dataGrid\_classe\_by\_trainer.Columns.Add("Окончание", "Окончание");

dataGrid\_classe\_by\_trainer.RowsDefaultCellStyle.SelectionBackColor = this.BackColor;

foreach (var din in data)

{

dataGrid\_classe\_by\_trainer.Rows.Add(din.Name.ToString(),din.DayOfWeek.ToString(),din.StartTime.ToString(),din.EndTime.ToString());

}

dataGrid\_classe\_by\_trainer.ClearSelection();

}

else

{

dataGrid\_classe\_by\_trainer.Columns.Clear();

dataGrid\_classe\_by\_trainer.Rows.Clear();

dataGrid\_classe\_by\_trainer.Columns.Add("", "");

int index = dataGrid\_classe\_by\_trainer.Rows.Add("Занятия отсутствуют");

dataGrid\_classe\_by\_trainer.RowsDefaultCellStyle.SelectionBackColor = Color.Red;

dataGrid\_classe\_by\_trainer.Rows[index].Cells[0].Style.BackColor = Color.Red;

dataGrid\_classe\_by\_trainer.ClearSelection();

}

}

private void trainersBindingSource\_CurrentItemChanged(object sender, EventArgs e)

{

Initialize\_ListClassesTrainers();

}

private void dataGrid\_classe\_by\_trainer\_SizeChanged(object sender, EventArgs e)

{

Initialize\_ListClassesTrainers();

}

private void trainersBindingSource\_CurrentChanged(object sender, EventArgs e)

{

Initialize\_ListClassesTrainers();

}

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

{

this.DialogResult = DialogResult.Cancel;

this.Close();

}

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

{

Tables.Table\_Common\_Handler train = new Tables.Table\_Common\_Handler("Trainers");

train.Show(this);

}

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

{

trainersBindingSource.MoveFirst();

}

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

{

trainersBindingSource.MovePrevious();

}

private void btn\_Next\_Click(object sender, EventArgs e)

{

trainersBindingSource.MoveNext();

}

private void btn\_last\_Click(object sender, EventArgs e)

{

trainersBindingSource.MoveLast();

}

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

{

trainersBindingSource.CancelEdit();

}

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

{

try

{

if (MessageBox.Show(this, "Вы уверены что хотите удалить запись?", "?", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

bindingNavigatorDeleteItem.PerformClick();

if(MessageBox.Show(this,"Применить удаление?\nЭто действие нельзя будет вернуть","Применить?",MessageBoxButtons.YesNo,MessageBoxIcon.Question) == DialogResult.Yes)

{

trainersBindingNavigatorSaveItem?.PerformClick();

}

}

}catch(Exception g)

{

MessageBox.Show(this, g.Message, "Ошибка удаления!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

bindingNavigatorAddNewItem?.PerformClick();

int newUID = int.Parse(bindingNavigatorPositionItem.Text);

trainerIDComboBox.Text = newUID.ToString();

}

catch(Exception g)

{

MessageBox.Show(this, g.Message, "Ошибка добавления!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

trainersBindingNavigatorSaveItem?.PerformClick();

}

catch(Exception g)

{

MessageBox.Show(this, g.Message, "Ошибка сохранения!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

}

private void trainersBindingSource\_PositionChanged(object sender, EventArgs e)

{

pictureBox1.Image = null;

if (!string.IsNullOrWhiteSpace(imgCombo.Text))

{

if (Program.Images.Where(x => x.Key.ToString() == imgCombo.Text).Count() > 0)

pictureBox1.Image = Program.Images[int.Parse(imgCombo.Text)];

}

}

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

{

pictureBox1.Image = null;

if (!string.IsNullOrWhiteSpace(imgCombo.Text))

{

if (Program.Images.Where(x => x.Key.ToString() == imgCombo.Text).Count() > 0)

pictureBox1.Image = Program.Images[int.Parse(imgCombo.Text)];

}

}

}

}

Листинг формы «Мои занятия»

using System;

using System.Drawing;

using System.Windows.Forms;

namespace FitnessClub.Other

{

public partial class MeClasses\_Form : Form

{

public MeClasses\_Form()

{

InitializeComponent();

dynamic\_style.Enable(this);

}

dynamic ME\_DATA;

private void MeClasses\_Form\_Load(object sender, EventArgs e)

{

int ID = Properties.Settings.Default.ID;

ME\_DATA = Program.Select\_Dynamic($"SELECT \* FROM [ClientClassRegistrations] JOIN ClassSchedules ON (ClientClassRegistrations.ClassScheduleID = ClassSchedules.ClassScheduleID) JOIN [Classes] ON (ClassSchedules.ClassID = Classes.ClassID) JOIN [Trainers] ON (Classes.TrainerID = Trainers.TrainerID) WHERE ClientClassRegistrations.ClientID = \'{ID}\'");

int count\_classes = ME\_DATA.Count;

lst\_classes.Items.Clear();

chartGrid1.AxisX\_Key.Clear();

chartGrid1.AxisY\_Key.Clear();

for (int i = 0; i < count\_classes; i++)

{

lst\_classes.Items.Add($"{ME\_DATA[i].Name}");

}

//РАБОТА С ГРАФИКОМ

//График отображает дни недели, часы занятия

for(int i = 0; i < 14; i++)

{//10 - 17

chartGrid1.AxisX\_Key.Add($"{7 + i}:00");

}

chartGrid1.AxisY\_Key.Add("Понедельник");

chartGrid1.AxisY\_Key.Add("Вторник");

chartGrid1.AxisY\_Key.Add("Среда");

chartGrid1.AxisY\_Key.Add("Четверг");

chartGrid1.AxisY\_Key.Add("Пятница");

chartGrid1.AxisY\_Key.Add("Суббота");

chartGrid1.Refresh();

foreach(var me in ME\_DATA)

{

TimeSpan startTime = TimeSpan.Parse(me.StartTime.ToString());

TimeSpan endTime = TimeSpan.Parse(me.EndTime.ToString())-TimeSpan.FromHours(1);

for(TimeSpan inter = startTime; inter <= endTime; inter += TimeSpan.FromHours(1))

{

chartGrid1.SetUniqueData($"{inter.Hours}:00", me.DayOfWeek.ToString(), Color.Blue,me.ClassID.ToString(), me.Name.ToString());

}

}

chartGrid1.MouseClickChart += (Id\_Class) =>

{

int ClassID = int.Parse(Id\_Class.ToString());

foreach(var v in ME\_DATA)

{

if(ClassID == int.Parse(v.ClassID.ToString()))

{

for(int j = 0; j < lst\_classes.Items.Count; j++)

{

if (lst\_classes.Items[j].ToString().ToUpper() == v.Name.ToString().ToUpper())

{

lst\_classes.SelectedIndex = j;

break;

}

}

}

}

};

ContextMenu change = new ContextMenu();

MenuItem add\_menu = new MenuItem("Добавить");

add\_menu.Click += (\_sender, \_e) =>

{

Lenta.Classes\_Form fs = new Lenta.Classes\_Form();

fs.ShowDialog(this);

MeClasses\_Form\_Load(sender, e);

};

MenuItem del\_menu = new MenuItem("Удалить");

del\_menu.Click += (\_sender, \_e) =>

{

if(lst\_classes.SelectedIndex != -1)

{

string name = lst\_classes.SelectedItem.ToString();

if(MessageBox.Show(this,$"Вы уверены что хотите удалить \'{name}\' из своего графика?","Подтвердите",MessageBoxButtons.YesNo,MessageBoxIcon.Question) == DialogResult.Yes)

{

int idClass = -1;

foreach(var V in ME\_DATA)

{

if(V.Name.ToString().ToUpper() == name.ToUpper())

{

idClass = (int)V.ClassID;

}

}

if(idClass != -1)

{

int idClient = Properties.Settings.Default.ID;

if(idClient != -1)

{

var ClassSchedule = Program.Select\_Dynamic($"SELECT \* FROM [ClassSchedules] WHERE [ClassID] = \'{idClass}\'");

foreach(var schedule in ClassSchedule)

{

var ClientClassRegistration = Program.Select\_Dynamic($"SELECT \* FROM [ClientClassRegistrations] WHERE [ClientID] = \'{idClient}\' and [ClassScheduleID] = \'{schedule.ClassScheduleID}\'");

if((int)(ClientClassRegistration.Count) > 0)

{

// MessageBox.Show($"{ClientClassRegistration[0].RegistrationID} {ClientClassRegistration[0].ClassScheduleID}");

int del\_reg = Program.Insert($"DELETE FROM [ClientClassRegistrations] WHERE [RegistrationID] = \'{ClientClassRegistration[0].RegistrationID}\'");

int del\_schedule = Program.Insert($"DELETE FROM [ClassSchedules] WHERE [ClassScheduleID] = \'{ClientClassRegistration[0].ClassScheduleID}\'");

if(del\_reg == 1 && del\_schedule == 1)

{

MessageBox.Show(this, "Изменения сохранены!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

MeClasses\_Form\_Load(sender,e);

}

else

{

MessageBox.Show(this, $"Ошибка транзакции!\ndel\_reg: {del\_reg}\ndel\_schedule: {del\_schedule}", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

MeClasses\_Form\_Load(sender, e);

}

break;

}

}

}

}

}

}

};

change.MenuItems.Add(add\_menu);

change.MenuItems.Add(del\_menu);

lst\_classes.ContextMenu = change;

chartGrid1.ContextMenu = change;

}

private void lst\_classes\_SelectedIndexChanged(object sender, EventArgs e)

{

if(lst\_classes.SelectedIndex != -1)

{

var that = ME\_DATA[lst\_classes.SelectedIndex];

txtNameSurnameTrainer.Text = $"{that.FirstName} {that.LastName}";

txtNumberTrainer.Text = $"{that.Phone}";

txtEmailTrainer.Text = $"{that.Email}";

txtStart.Text = $"{that.StartTime}";

txtEnd.Text = $"{that.EndTime}";

txtDayOfWeeks.Text = $"{that.DayOfWeek}";

txtSpecialization.Text = $"{that.Specialization}";

foreach (var me in ME\_DATA)

{

TimeSpan startTime = TimeSpan.Parse(me.StartTime.ToString());

TimeSpan endTime = TimeSpan.Parse(me.EndTime.ToString()) - TimeSpan.FromHours(1);

for (TimeSpan inter = startTime; inter <= endTime; inter += TimeSpan.FromHours(1))

{

if (me == that)

{

chartGrid1.SetUniqueData($"{inter.Hours}:00", me.DayOfWeek.ToString(), Color.Lime, me.ClassID.ToString(), me.Name.ToString());

}

else

chartGrid1.SetUniqueData($"{inter.Hours}:00", me.DayOfWeek.ToString(), Color.Blue, me.ClassID.ToString(), me.Name.ToString());

}

}

}

}

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

{

this.DialogResult = DialogResult.Cancel;

this.Close();

}

}

}

Листинг дополнительного класса для определения стилей и мини паттерна (ограничителя по тэгам)

Примечание: каждая форма и элемент имеет ряд тэгов по которым определяется их логика (ONLY\_USER,ONLY\_ADMIN,SAVE\_SIZE и т.д.)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

//====================

//АВТОР СУХИНИН К.Д. |

//====================

namespace FitnessClub

{

/// <summary>

/// Класс для применений цвета проекта и доп. обработка

/// </summary>

public static class dynamic\_style

{

private static Color FromHEX(string hex)

{

int argb = Int32.Parse(hex.Replace("#", ""), NumberStyles.HexNumber);

Color clr = Color.FromArgb(argb);

return clr;

}

/// <summary>

/// Метод применения цветовой палитры

/// </summary>

/// <param name="form">Форма на которой нужен стиль</param>

public static void Enable(Form form)

{

//HERE WE STANDART FORM STYLE FOR DOTHER FORM

form.KeyUp -= Handler\_KeyUp;

form.KeyUp += Handler\_KeyUp;

if (!form.Text.ToUpper().Contains("Фитнес - Клуб: К & Б".ToUpper()))

{

form.Text = $"Фитнес - Клуб: К & Б. {form.Text}";

}

//ДАННОЕ СОБЫТИЕ ОБРАБАТЫВАЕТ ФОРМЫ С ТЭГОМ SAVE\_SIZE И ОТКРЫВАЕТ ИХ С СОСТОЯНИЕМ ИХ РОДИТЕЛЕЙ. (РАЗВЕРНУТ НА ВЕСЬ ЭКРАН ИЛИ ОБЫЧНЫЙ. СКРЫТЫЙ РЕЖИМ НЕ ОБРАБАТЫВАЕТ)

form.Load -= LOAD;

form.Load += LOAD;

//Рекурсивно добавляем обработчик для горячих клавиш

foreach (var vsCW\_form in form.Controls)

{

try

{

(vsCW\_form as Control).KeyUp -= Handler\_KeyUp;

(vsCW\_form as Control).KeyUp += Handler\_KeyUp;

if (vsCW\_form is GroupBox || vsCW\_form is Panel)

{

Recursive\_Enable(vsCW\_form as Control);

}

else

InvokeStyle(vsCW\_form as Control);

}

catch (Exception g)

{

Debugger.Log(0, "Debug", $"\n{g}");

}

}

}

/// <summary>

/// Событие загрузки формы для обработки состояния WindowState относительно родителей.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private static void LOAD(object sender, EventArgs e)

{

var @\_form = (sender as Form);

if (@\_form.Owner != null && @\_form.Tag != null && @\_form.Tag.ToString().ToUpper() == "SAVE\_SIZE")

{

@\_form.WindowState = @\_form.Owner.WindowState;

}

}

/// <summary>

/// Событие горячих клавиш

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private static void Handler\_KeyUp(object sender, KeyEventArgs e)

{

if (((sender as Control).FindForm() is Tables.Table\_Common\_Handler))

{

if (e.KeyCode == Keys.F2)

{

Other.ChangeTable\_TEST change = new Other.ChangeTable\_TEST();

if (change.ShowDialog((sender as Control).FindForm() as IWin32Window) == DialogResult.OK)

{

((sender as Control).FindForm() as Tables.Table\_Common\_Handler).REFRESH\_TEST(change.ReturnText);

}

}

}

if (e.KeyCode == Keys.F1)

{

}

if(e.KeyCode == Keys.F3)

{

if (((sender as Control).FindForm() is MainHandlerForm))

{

Lenta.Class\_Trainers\_Equipment\_Form cl = new Lenta.Class\_Trainers\_Equipment\_Form();

cl.ShowDialog((sender as Control).FindForm());

}

}

if(e.KeyCode == Keys.F4)

{

Environment.Exit(0);

}

}

/// <summary>

/// Рекурсивный метод добавления горячих клавиш и стилей

/// </summary>

/// <param name="vControl"></param>

private static void Recursive\_Enable(Control vControl)

{

foreach (var element in vControl.Controls)

{

(element as Control).KeyUp -= Handler\_KeyUp;

(element as Control).KeyUp += Handler\_KeyUp;

if (element is GroupBox || element is Panel)

Recursive\_Enable(element as Control);

else

InvokeStyle(element as Control);

}

}

/// <summary>

/// Метод обработки элемента

/// </summary>

/// <param name="vs">Элемент</param>

/// <exception cref="Exception">Если элемент не существует или был уничтожен</exception>

private static void InvokeStyle(Control vs)

{

if (vs == null)

throw new Exception("Ошибка стилей!\nНе найден элемент применения стилей.");

//Нужно если в будущем будут подпотоки

if (vs.IsDisposed)

throw new Exception("Ошибка стилей\nЭлемент был уничтожен");

//УСТАНОВКА ЛОГИКИ РАБОТЫ ЭЛЕМЕНТОВ ОТНОСИТЕЛЬНО ПРАВ

if (vs is Button)

{

if (vs.AccessibleDescription != null)

{

if (vs.AccessibleDescription.ToString().ToUpper() == "ONLY\_ADMIN".ToUpper())

{

if (Program.GetRole(Properties.Settings.Default.Role) == Role.User)

{

vs.Visible = false;

}

}

if (vs.AccessibleDescription.ToString().ToUpper() == "ONLY\_USER".ToUpper())

{

if (Program.GetRole(Properties.Settings.Default.Role) == Role.Admin)

{

vs.Enabled = false;

}

}

}

}

//Тэг обработки для исключения элемента из обработки

if (vs.AccessibleName != null)

if (vs.AccessibleName == "NONESTYLE")

return;

if(vs is DateTimePicker)

{

if(Program.GetRole(Properties.Settings.Default.Role) == Role.User)

{

(vs as DateTimePicker).Enabled = false;

}

}

if (vs is Button)

{

vs.BackColor = Button\_Style.button\_backcolor;

vs.ForeColor = Button\_Style.button\_forecolor;

(vs as Button).FlatAppearance.BorderColor = Button\_Style.button\_bordercolor;

(vs as Button).FlatAppearance.BorderSize = Button\_Style.button\_bordersize;

}

if (vs is TextBox)

{

if (Program.GetRole(Properties.Settings.Default.Role) == Role.User)

{

(vs as TextBox).ReadOnly = true;

}

}

if (vs is Label)

{

vs.ForeColor = Label\_Style.label\_forecolor;

vs.BackColor = Label\_Style.label\_backcolor;

}

//Для данного элемента не предусмотрен стиль, поэтому добавляем только ограничение для пользователя

if (vs is ComboBox)

{

if (Program.GetRole(Properties.Settings.Default.Role) == Role.User)

{

if ((vs as ComboBox).Enabled)

vs.Enabled = false;

}

}

if (vs is BindingNavigator)

{

var navigator = (vs as BindingNavigator);

navigator.BackColor = (vs.FindForm()).BackColor;

navigator.PositionItem.BackColor = (vs.FindForm()).BackColor;

navigator.GripStyle = ToolStripGripStyle.Hidden;

if (navigator.Tag != null && navigator.Tag.ToString().ToUpper() == "ONLY\_ADMIN".ToUpper() && Program.GetRole(Properties.Settings.Default.Role) == Role.User)

{

//УБИРАЕМ ДОСТУП К РЕД ФУНКЦИОНАЛУ ДЛЯ ПОЛЬЗОВАТЕЛЯ

navigator.AddNewItem.Enabled = false;

navigator.DeleteItem.Enabled = false;

foreach (Component vsControl in navigator.Items)

{

if (vsControl is ToolStripButton)

{

if ((vsControl as ToolStripButton).Name.ToUpper().Contains("BindingNavigatorSaveItem".ToUpper()))

(vsControl as ToolStripButton).Enabled = false;

}

}

}

}

if(vs is DataGridView && false)

{

DataGridView DataGrid = (vs as DataGridView);

DataGrid.EnableHeadersVisualStyles = DataGridView\_NoneStyle.EnableHeadersVisualStyles;

DataGrid.RowHeadersVisible = DataGridView\_NoneStyle.RowHeadersVisible;

DataGrid.AllowUserToAddRows = DataGridView\_NoneStyle.AllowUserToAddRows;

DataGrid.ReadOnly = DataGridView\_NoneStyle.ReadOnly;

DataGrid.AutoSize = DataGridView\_NoneStyle.AutoSize;

DataGrid.SelectionMode = DataGridView\_NoneStyle.SelectionMode;

DataGrid.MultiSelect = DataGridView\_NoneStyle.MultiSelect;

DataGrid.AutoSizeColumnsMode = DataGridView\_NoneStyle.AutoSizeColumnsMode;

DataGrid.AutoSizeRowsMode = DataGridView\_NoneStyle.AutoSizeRowsMode;

DataGrid.CellBorderStyle = DataGridView\_NoneStyle.CellBorderStyle;

DataGrid.AdvancedCellBorderStyle.Left = DataGridView\_NoneStyle.AdvancedCellBorderStyleLeft;

DataGrid.AllowUserToDeleteRows = DataGridView\_NoneStyle.AllowUserToDeleteRows;

DataGrid.AllowUserToOrderColumns = DataGridView\_NoneStyle.AllowUserToOrderColumns;

DataGrid.AllowUserToResizeColumns = DataGridView\_NoneStyle.AllowUserToResizeColumns;

DataGrid.AllowUserToResizeRows = DataGridView\_NoneStyle.AllowUserToResizeRows;

DataGrid.GridColor = (vs.FindForm()).BackColor;

DataGrid.RowsDefaultCellStyle.BackColor = DataGridView\_NoneStyle.BackgroundColor;

DataGrid.RowsDefaultCellStyle.SelectionForeColor = DataGridView\_NoneStyle.SelectionForeColor;

DataGrid.RowsDefaultCellStyle.SelectionBackColor = DataGridView\_NoneStyle.SelectionBackColor;

}

//ЛОГИКА TRANSLATE (перевода) ДЛЯ ЛЕНТОЧНЫХ ФОРМ

if (vs is Label)

{

if ((vs.FindForm() != null) && vs.FindForm().AccessibleName != null && (vs.FindForm()).AccessibleName.ToUpper() == "FORM\_TRANSLATE")

{

try

{

if (Program.Translater == null || Program.Translater.Rus == null)

throw new Exception("Не удалось найти данные для перевода.");

var table = Program.Translater.Rus.Where(x => x.NameTable == vs.FindForm().AccessibleDescription).FirstOrDefault();

if (vs.AccessibleDescription != null && !string.IsNullOrWhiteSpace(vs.AccessibleDescription.ToString()))

{

table = Program.Translater.Rus.Where(x => x.NameTable == vs.AccessibleDescription.ToString()).FirstOrDefault();

}

if (table == null)

throw new Exception($"Warning: не удалось перевести данные для таблица \'{vs.FindForm().AccessibleDescription}\' относительно \'{vs.Name}\'");

var property = table.Property.Where(x => x.NameProperty.ToUpper() == vs.Name.ToUpper().Replace("Label".ToUpper(), "")).FirstOrDefault();

Debugger.Log(0, "Debug", $"{vs.Name.ToUpper().Replace("Label".ToUpper(), "")}{Environment.NewLine}");

if (property != null)

vs.Text = property.HeaderProperty;

}

catch (Exception g)

{

throw g;

}

}

}

}

/// <summary>

/// Стат класс для определения стилей DataGridView

/// </summary>

private static class DataGridView\_NoneStyle

{

public static bool EnableHeadersVisualStyles = false;

public static bool RowHeadersVisible = false;

public static bool AllowUserToAddRows = false;

public static bool ReadOnly = false;

public static bool AutoSize = false;

public static DataGridViewSelectionMode SelectionMode = DataGridViewSelectionMode.RowHeaderSelect;

public static bool MultiSelect = false;

public static DataGridViewAutoSizeColumnsMode AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

public static DataGridViewAutoSizeRowsMode AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

public static DataGridViewCellBorderStyle CellBorderStyle = DataGridViewCellBorderStyle.None;

public static DataGridViewAdvancedCellBorderStyle AdvancedCellBorderStyleLeft = DataGridViewAdvancedCellBorderStyle.OutsetPartial;

public static bool AllowUserToDeleteRows = false;

public static bool AllowUserToOrderColumns = false;

public static bool AllowUserToResizeColumns = false;

public static bool AllowUserToResizeRows = false;

public static Color SelectionForeColor = Color.Transparent;

public static Color SelectionBackColor = Color.Transparent;

public static Color BackgroundColor = Color.Transparent;

public static bool ColumnHeadersVisible = false;

}

/// <summary>

/// Стат класс для определения стилей кнопки

/// </summary>

private static class Button\_Style

{

public static Color button\_forecolor = Color.Black;

public static Color button\_backcolor = Color.Transparent;

public static int button\_bordersize = 1;

public static Color button\_bordercolor = Color.FromArgb(0, 192, 192);

}

/// <summary>

/// Стат класс для определения стилей текста Label (Если иные не описаны то стили в Designer.cs)

/// </summary>

private static class Label\_Style

{

public static Color label\_forecolor = Color.Black;

public static Color label\_backcolor = Color.Transparent;

}

}

}

Листинг вспомогательный класс для сереализации перевода

using Newtonsoft.Json;

using System.Collections.Generic;

namespace FitnessClub.JSON

{

public class Rootobject\_Translater

{

public List<Tables> Rus { get; set; }

}

public class Tables

{

[JsonProperty(DefaultValueHandling = DefaultValueHandling.Include)]

public string NameTable { get; set; }

[JsonProperty(DefaultValueHandling = DefaultValueHandling.Include)]

public string HeaderTable { get; set; }

[JsonProperty(DefaultValueHandling = DefaultValueHandling.Include)]

public List<Property1> Property { get; set; }

}

public class Property1

{

[JsonProperty(DefaultValueHandling = DefaultValueHandling.Include)]

public string NameProperty { get; set; }

[JsonProperty(DefaultValueHandling = DefaultValueHandling.Include)]

public string HeaderProperty { get; set; }

}

}

**ЗАКЛЮЧЕНИЕ**

Основная информация

Название приложения: Фитнес-клуб: К&Б.

Цель приложения: Помочь работникам клуба предоставить учёт, хранения, обработку заказов, информации о занятиях и т.д.

Функции: Для пользователей есть возможность просматривать записи. Для администраторов кроме возможности просмотра записей, так есть могут редактировать информацию (добавлять, удалять записи).

Результаты

Разработка приложения была выполнена успешно;

Цели и задачи были достигнуты и выполнены;

Разработка базы данных и приложения для фитнес клуба - это был интересный и познавательный опыт. Я узнал много нового о структурировании данных, оптимизации запросов и разработке удобного пользовательского интерфейса.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Jacob VanWagoner, Modern SQL: A Practical Guide for Working with Relational Databases (2021, 400 стр.)
2. Джозеф Л. Форджоне, SQL Server 2022: A Beginner's Guide (2022, 592 стр.)
3. Джонатан Кларк, C# in Depth (2021, 1008 стр.)
4. Дэвид Хэмилтон, Visual Studio 2022: A Beginner's Guide (2022, 576 стр.)
5. Иэн Гриффитс, Mastering Visual Studio 2022 (2022, 960 стр.)
6. Крис Феа, SQL for Data Analysis (2020, 432 стр.)
7. Кэтти Сиэрс и Билл МакКоун, Head First C# (2018, 752 стр.)
8. Линн Бейли, Билл МакКоун и Кэтти Сиэрс, Head First SQL (2018, 768 стр.)
9. Роберт МакГрат и Боб Тавелла, Professional SQL Server 2019 (2019, 1280 стр.)
10. Роберт Уайлднер, Pro Visual Studio 2019: The Expert's Guide to Microsoft's Integrated Development Environment (2019, 1344 стр.)
11. Мартин Фаулер, Refactoring: Improving the Design of Existing Code (2018, 464 стр.)
12. Стивен К. МакКоннелл, Code Complete: A Practical Handbook of Software Construction (2004, 960 стр.)
13. Брэдфорд Т. Форд, SQL Cookbook (2020, 864 стр.)
14. Бен Гриффитс и Дэн Рат, Programming Microsoft SQL Server 2019 (2019, 1056 стр.)
15. Скотт Хансельман, The ASP.NET Core 6.0 MVC Cheat Sheet (2021, 120 стр.)
16. Мик С. Доэрти и Шон Б. Шеннон, Applied Microsoft SQL Server 2019 Analysis Services (2019, 672 стр.)
17. Дон Джонс, Learn SQL Quickly (2019, 160 стр.)