Федеральное агентство связи

Сибирский Государственный Университет Телекоммуникаций и Информатики

СибГУТИ

Кафедра прикладной математики и кибернетики

РГР по дисциплине Визуальное программирование и человеко-машинное взаимодействие. Ч. 1

Выполнил: студент 2 курса, группы ИП-014

Альхимович Михаил Валерьевич  
Проверил: ст. преподаватель

Милешко Антон Владимирович

Новосибирск, 2022 г.

**РГР. Часть 1**

# Задание на РГР

Создать ПО для отображения и обработки статистических данных для определённого вида спорта. ПО должно включать 2 основных окна: окно отображающее таблицы БД со статистической информацией и результаты запросов к БД, переключение таблиц и результатов должно быть реализовано через вкладки; и окно для менеджера запросов к БД.

Первое окно должно давать возможность просматривать и изменять все таблицы БД, а также просматривать результаты запросов к БД. Должна иметься возможность удалить вкладки с результатами запросов, но не вкладки с таблицами. Также должна иметься возможность перейти к окну менеджера запросов.

Окно менеджера запросов должно предоставлять интерфейс для создания, сохранения, удаления, редактирования запросов. Созданные запросы должны отображаться в виде списка с названиями запросов, в который можно добавлять новые запросы, удалять, просматривать существующие. Для создания и редактирования запросов должен предоставляться визуальный интерфейс, а не язык запросов. Редактор запросов должен поддерживать операции выборки, соединения, группирования, подзапросы (в качестве подзапроса используются ранее сохранённые запросы).

Ход работы:

1. Исследование предметной области и создание ER диаграммы.
2. Перевод ER диаграммы в реляционную модель, создание и заполнение БД.
3. Проработка визуального интерфейса приложения
4. Создание диаграммы классов приложения
5. Реализация основного окна приложения
6. Реализация менеджера запросов
7. Тестирование и отладка

**Вариант задания - 1**

1. Футбол (клубы) ([https://www.whoscored.com/Statistic](https://www.whoscored.com/Statistics)s <https://www.soccerstats.com/> <https://fbref.com/en/>)

**Предметная область**

Тема: Футбол (клубы)

1. Футбольные клубы;

1.1 Чемпионская лига;

1.2 Дата проведения турнира;

1.3 Групповой этап (отборочные);

1.3.1 Группа ⍺, ⍺ - {A, B, …, Z};

1.3.2 Дата проведения игры;

1.3.3 Название команд;

1.3.4 Результат матча;

1.3.5 Сводка прошедших в лигу команд;

1.3.5.1 Общая статистика команд за группу(GP, W, D, L, etc.);

1.4 Final League Parts;

1.4.1 n-th Finals (где n - ;

1.4.2 Дата проведения игры;

1.4.3 Название команд;

1.4.4 Результат матча;

1.4.5 Итог турнира (финальный матч);

1.5 Общая статистика турнира;

1.5.1 Время гола ((0-15),(16-30),(31-45),(46-60),(61-75),(76-90)) + овертаймы;

1.5.2 Время гола по половинам ((0-45),(46-90));

1.5.3 Ср. время гола в матче;

1.5.4 Early Scoring analysis (крат. статистика);

1.5.5 Широкая статистика голов;

1.6 Игроки;

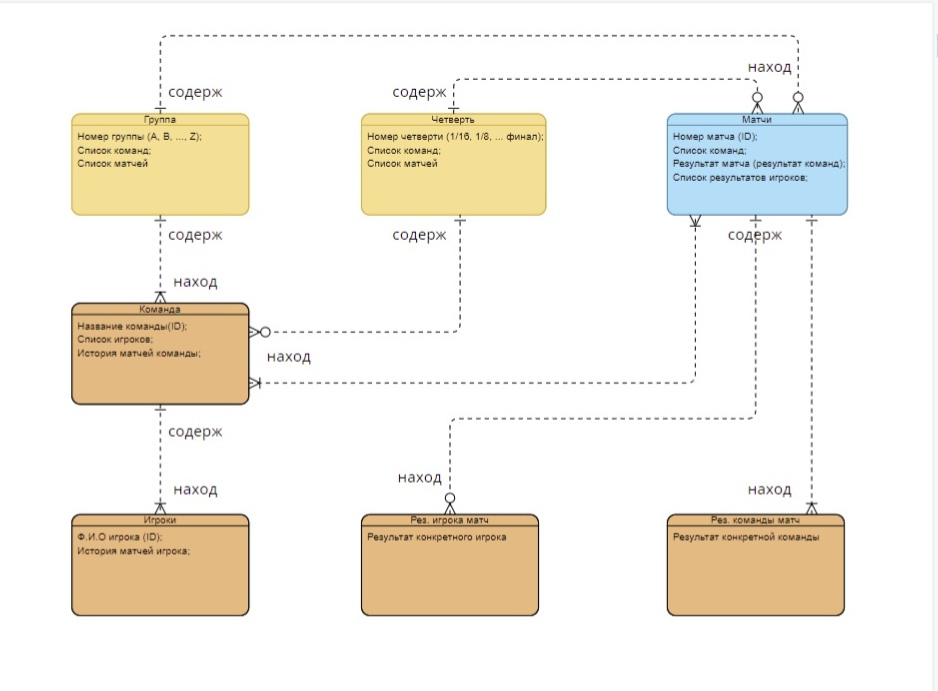
1.6.1 Ф.И.О игрока;

1.6.2 Команда игрока;

1.6.3 Статистика игрока;

**ER-диаграмма**

**Сущности**: четверти, группы, матчи, команды, игроки, результат мачта в команде, результат матча игрока.



**РГР. Часть 2**

# Задание на РГР

Создать ПО для отображения и обработки статистических данных для определённого вида спорта. ПО должно включать 2 основных окна: окно отображающее таблицы БД со статистической информацией и результаты запросов к БД, переключение таблиц и результатов должно быть реализовано через вкладки; и окно для менеджера запросов к БД.

Первое окно должно давать возможность просматривать и изменять все таблицы БД, а также просматривать результаты запросов к БД. Должна иметься возможность удалить вкладки с результатами запросов, но не вкладки с таблицами. Также должна иметься возможность перейти к окну менеджера запросов.

Окно менеджера запросов должно предоставлять интерфейс для создания, сохранения, удаления, редактирования запросов. Созданные запросы должны отображаться в виде списка с названиями запросов, в который можно добавлять новые запросы, удалять, просматривать существующие. Для создания и редактирования запросов должен предоставляться визуальный интерфейс, а не язык запросов. Редактор запросов должен поддерживать операции выборки, соединения, группирования, подзапросы (в качестве подзапроса используются ранее сохранённые запросы).

Ход работы:

1. Исследование предметной области и создание ER диаграммы.
2. Перевод ER диаграммы в реляционную модель, создание и заполнение БД.
3. Проработка визуального интерфейса приложения
4. Создание диаграммы классов приложения
5. Реализация основного окна приложения
6. Реализация менеджера запросов
7. Тестирование и отладка

**Вариант задания - 1**

1. Футбол (клубы) ([https://www.whoscored.com/Statistic](https://www.whoscored.com/Statistics)s <https://www.soccerstats.com/> <https://fbref.com/en/>)

**Перевод ER-диаграммы в реляционную модель**

Концептуальные модели позволяют более точно представить предметную область, чем реляционные и другие более ранние модели. Но в настоящее время существует немного систем управления базами данных, поддерживающих эти модели. На практике наиболее распространены системы, реализующие реляционную модель.

Поэтому необходим метод перевода концептуальной модели в реляционную. Такой метод основывается на формировании набора предварительных таблиц.

Каждой сущности создается таблица. Причем каждому атрибуту сущности соответствует столбец таблицы.

Правила генерации таблиц из ER-диаграмм опираются на два основных фактора – тип связи и класс принадлежности сущности. Изложим их:

**Правило 1.** Если связь типа 1:1 и класс принадлежности обеих сущностей является обязательным, то необходима только одна таблица. Первичным ключом этой таблицы может быть первичный ключ любой из двух сущностей.

**Правило 2.** Если связь типа 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности, для которой класс принадлежности является необязательным, добавляется как атрибут в таблицу для сущности с обязательным классом принадлежности.

**Правило 3.** Если связь типа 1:1 и класс принадлежности обеих сущностей является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

**Правило 4.** Если связь типа 1:М и класс принадлежности сущности на стороне М является обязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности на стороне 1 добавляется как атрибут в таблицу для сущности на стороне М.

**Правило 5.** Если связь типа 1:М и класс принадлежности сущности на стороне М является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

**Правило 6.** Если связь типа М, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

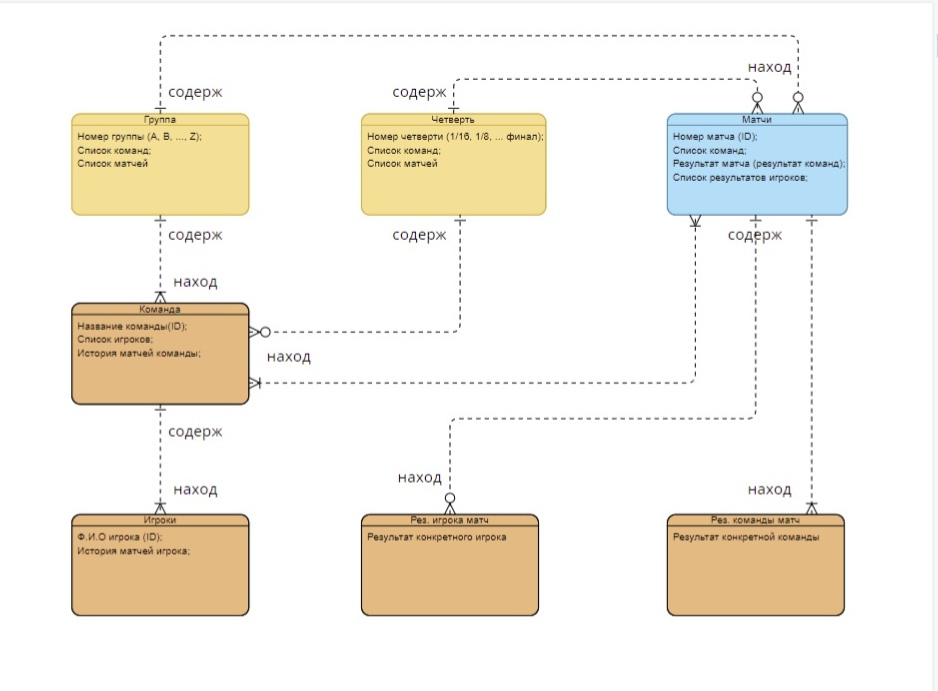
**Цели создания реляционной модели:**

* Обеспечение более высокой степени независимости от данных. Прикладные программы не должны зависеть от изменений внутреннего представления данных, в частности от изменений организации файлов, переупорядочивания записей и путей доступа.
* Создание прочного фундамента для решения семантических вопросов, а также проблем непротиворечивости и избыточности данных. В частности, в статье Кодда вводится понятие нормализованных отношений, т.е. отношений без повторяющихся групп.
* Расширение языков управления данными за счет включения операций над множествами.

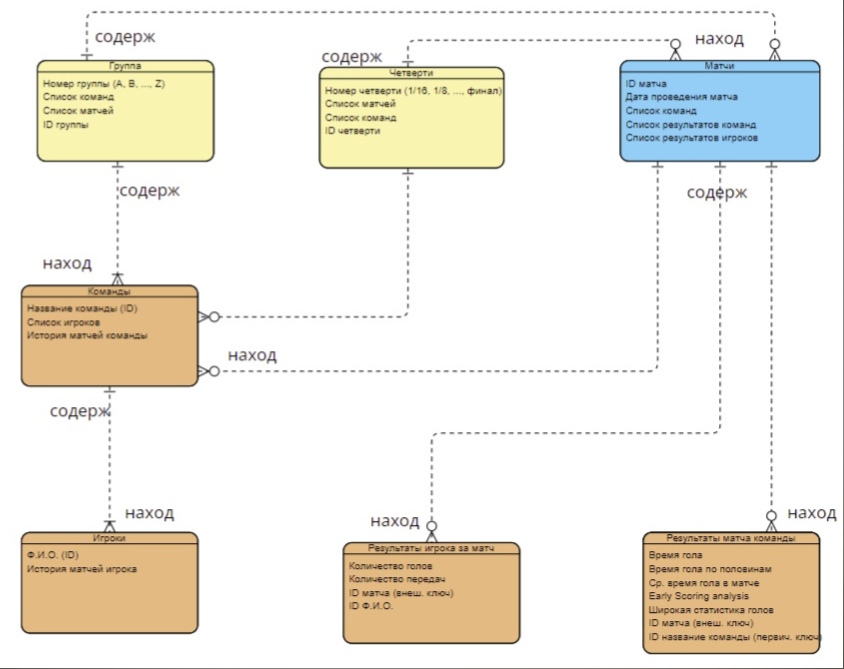
Реляционная модель основана на математическом понятии отношения, физическим представлением которого, является таблица.

**Попробуем преобразовать исходную ER-модель в реляционную модель:**

**Исходная ER-модель**



**Для начала уточним некоторые поля сущностей, чтобы более четко внести их в базу данных:**



**Распишем каждое отношение, его атрибуты и домены атрибутов:**

**1. Группа:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID группы** | **Номер группы** | **Список команд** | **Список матчей** |

**Домен ID группы (Суперключ):**

Имя домена: ID группы.

Содержимое домена: Множество всех допустимых ID команд.

Определение домена: Символьный, размер 4, диапазон от 0001 - FFFF.

**Домен номер группы:**

Имя домена: Номер группы.

Содержимое домена: Множество всех допустимых обозначений футбольных групп.

Определение домена: Символьный, размер 1, диапазон от A - Z.

**Домен список команд:**

Имя домена: Список команд.

Содержимое домена: список всех возможных команд Чемпионской лиги.

Определение домена: список отношений "Команды"**(?)**.

**Домен список матчей:**

Имя домена: Список матчей.

Содержимое домена: Список всех возможных матчей Чемпионской лиги.

Определение домена: Список отношений "Матчи"**(?)**.

**2. Четверти:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID четверти** | **Номер четверти** | **Список команд** | **Список матчей** |

**Домен ID четверти (Суперключ):**

Имя домена: ID четверти.

Содержимое домена: Множество всех допустимых ID четвертей.

Определение домена: Символьный, размер 4, диапазон от 0001 - FFFF.

**Домен номер группы:**

Имя домена: Номер четверти

Содержимое домена: Множество всех допустимых обозначений части финала.

Определение домена: Численный, размер 1, диапазон от - , n .

**Домен список команд:**

Имя домена: Список команд.

Содержимое домена: Список всех возможных команд Чемпионской лиги.

Определение домена: Список отношений "Команды"**(?)**.

**Домен список матчей:**

Имя домена: Список матчей.

Содержимое домена: Список всех возможных матчей Чемпионской лиги.

Определение домена: Список отношений "Матчи"**(?)**.

**3. Матчи:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID матча** | **Дата проведения мачта** | **Список команд** | **Список рез. команд** | **Список рез. игроков** |

**Домен ID матча (Первичный ключ):**

Имя домена: ID матча.

Содержимое домена: Множество всех допустимых ID матчей.

Определение домена: Символьный, размер 4, диапазон от 0001 - FFFF.

**Домен дата проведения матча:**

Имя домена: Дата

Содержимое домена: Все возможные даты.

Определение домена: Дата, диапазон от 2021/22, формат DD.MM.YY.

**Домен список команд:**

Имя домена: Список команд.

Содержимое домена: Список всех возможных команд Чемпионской лиги.

Определение домена: Список отношений "Команды"**(?)**.

**Домен список результатов команд:**

Имя домена: Результаты команд.

Содержимое домена: Список всех возможных результатов матча.

Определение домена: Список отношений "Результаты матча команды"**(?)**.

**Домен список результатов игроков:**

Имя домена: Результаты игроков.

Содержимое домена: Список всех возможных результатов игрока.

Определение домена: Список отношений "Результаты игрока за матч"**(?)**.

**4. Результаты матча команды:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название команды** | **Время гола** | **Время гола по половинам** | **Ср. время гола в матче** | **Early Scoring Analysis** | **Широкая статистика голов** | **ID матча** |

**Домен название команды (Первичный ключ):**

Имя домена: Название команды.

Содержимое домена: Список всех возможных команд Чемпионской лиги.

Определение домена: Буквенный, диапазон латинского и английского алфавитов.

**Домен время гола:**

Имя домена: Время гола (по 15 м.).

Содержимое домена: Все возможное время матча включая овертаймы.

Определение домена: Время, диапазон от 00:00 - 99:00, формат mm:ss.

**Время гола по половинам:**

Имя домена: Время гола по половинам.

Содержимое домена: Все возможное время матча включая овертаймы.

Определение домена: Время, диапазон от 00:00 - 99:00, формат mm:ss.

**Ср. время гола в матче:**

Имя домена: Ср. время гола.

Содержимое домена: Все возможное время матча включая овертаймы.

Определение домена: Время, диапазон от 00:00 - 99:00, формат mm:ss.

**Early Scoring Analysis:**

Имя домена: Краткая статистика.

Содержимое домена: Множество возможных статистик.

Определение домена: **(?)**.

**Домен широкая статистика гола:**

Имя домена: Широкая статистика.

Содержимое домена: Множество возможных статистик.

Определение домена: **(?)**.

**Домен ID матча (Внешний ключ):**

Имя домена: ID матча.

Содержимое домена: Множество всех допустимых ID матчей.

Определение домена: Символьный, размер 4, диапазон от 0001 - FFFF.

**(!?) Необходимо ли сделать название команды + ID матча - суперключом, для обозначения уникальности?**

**5. Результат игрока за матч:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ф.И.О.** | **Кол-во голов** | **Кол-во передач** | **ID матча** |

**Домен Ф.И.О. (Первичный ключ):**

Имя домена: Ф.И.О.

Содержимое домена: Множество всех возможных Ф.И.О. Чемпионской лиги.

Определение домена: Буквенный, диапазон латинского и английского алфавитов.

**Домен количество голов:**

Имя домена: Количество голов.

Содержимое домена: Множество положительных чисел;

Определение домена: Численный, диапазон от 0 - ∞.

**Домен количество передач:**

Имя домена: Количество передач.

Содержимое домена: Множество положительных чисел;

Определение домена: Численный, диапазон от 0 - ∞.

**Домен ID матча (Внешний ключ):**

Имя домена: ID матча.

Содержимое домена: Множество всех допустимых ID матчей.

Определение домена: Символьный, размер 4, диапазон от 0001 - FFFF.

**(!?) Необходимо ли сделать Ф.И.О. + ID матча - суперключом, для обозначения уникальности?**

**6. Команды:**

|  |  |  |
| --- | --- | --- |
| **Название команды** | **Список игроков** | **История матчей команды** |

**Домен название команды (Суперключ):**

Имя домена: Название.

Содержимое домена: Множество всех возможных результатов матча.

Определение домена: Буквенный, диапазон латинского и английского алфавитов.

**Домен список игроков:**

Имя домена: Список игроков.

Содержимое домена: Множество всех возможных игроков Чемпионской лиги.

Определение домена: Список отношений "Игроки"**(?)**.

**Домен история матчей команды:**

Имя домена: Список матчей.

Содержимое домена: Список всех возможных матчей Чемпионской лиги.

Определение домена: Список отношений "Матчи"**(?)**.

**7. Игроки:**

|  |  |
| --- | --- |
| **Ф.И.О.** | **История матчей игрока** |

**Домен Ф.И.О. (Первичный ключ):**

Имя домена: Ф.И.О.

Содержимое домена: Множество всех возможных Ф.И.О. Чемпионской лиги.

Определение домена: Буквенный, диапазон латинского и английского алфавитов.

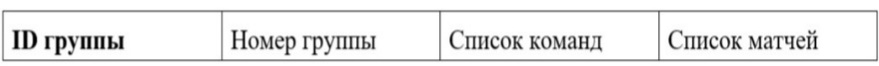
**Домен история матчей игрока:**

Имя домена: Список матчей.

Содержимое домена: Список всех возможных матчей Чемпионской лиги.

Определение домена: Список отношений "Матчи"**(?)**.

**Реляционная модель:**

****

1

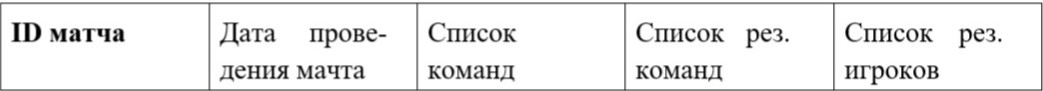
1

****

1

1

M

****

M

M

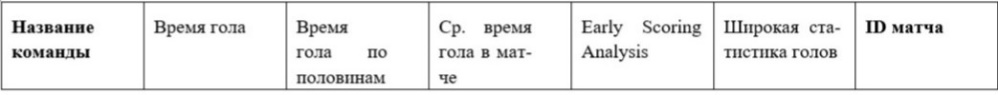
1

1

1

M

M

****

M

M

****

1

M

****

1

M

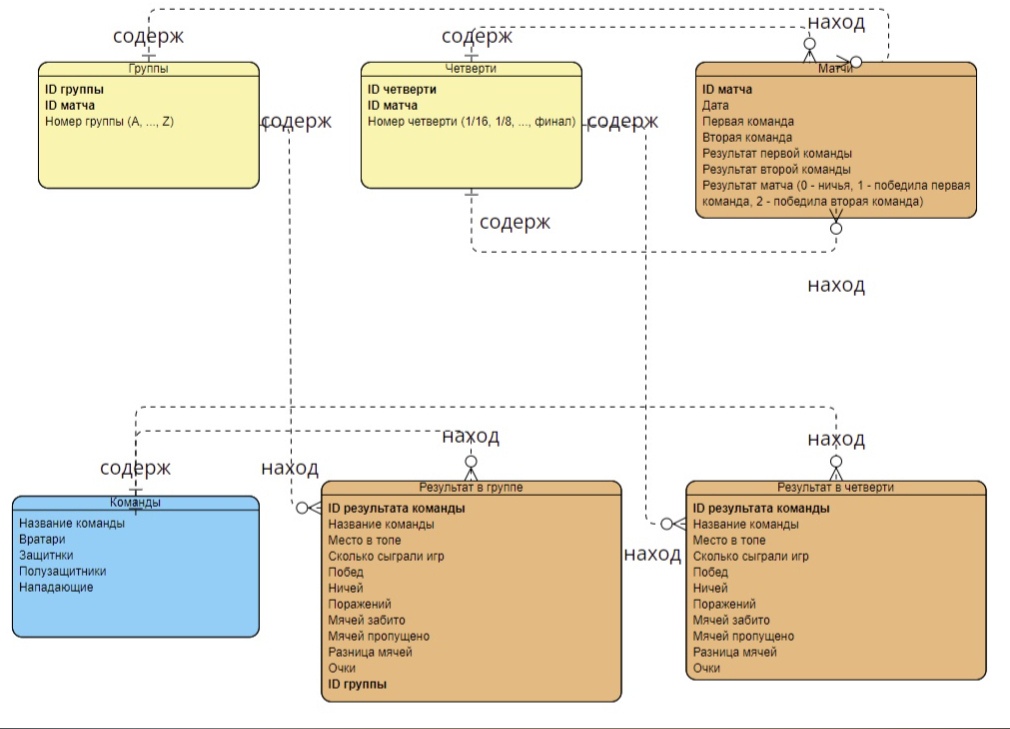
****

**Анализ состава атрибутов показывает, что некоторые отношения являются составными частями других отношений, поэтому я простраивал, в таких случаях, связь от атрибута к отношению, которое является составной частью другого отношения.** Не знаю, на сколько это правильно, а тем более как это делать, но пока так**.**

В SQLite тоже не получилось что-то подобное сделать…

Спустя 5 часов работы в базой данных, я ее закончил, но поменялась и реляционная модель и ER-диаграмма соответственно. То, что я сделал по изначальной ER-диаграмме больше похоже на структуру взаимодействия классов в программе, но никак не что-то для создания БД.

Новая ER-модель:



Сущности: Группы, Четверти, Матчи, Результаты в группе, Результаты в четверти, команды

**Распишем каждую сущность как отношение:**

**1. Группы:**

|  |  |  |
| --- | --- | --- |
| **ID ГРУППЫ** | **ID МАТЧА** | **НОМЕР ГРУППЫ** |

**2. Четверти:**

|  |  |  |
| --- | --- | --- |
| **ID ЧЕТВЕРТИ** | **ID МАТЧА** | **НОМЕР ЧЕТВЕРТИ** |

**3. Матчи:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID МАТЧА** | **ДАТА** | **ПЕРВАЯ**  **КОМАНДА** | **ВТОРАЯ**  **КОМАНДА** | **РЕЗ. ПЕРВОЙ КОМАНДЫ** | **РЕЗ. ВТОРОЙ КОМНАДЫ** | **РЕЗ. МАТЧА** |

**4. Результат в группе**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID РЕЗ. КОМАНДЫ В ГРУППЕ** | **НАЗВАНИЕ**  **КОМАНДЫ** | **МЕСТО ВТОПЕ** | **ОБЩЕЕ КОЛ-ВО ИГР** | **ПОБЕД** | **НИЧЕЙ** | **ПОРАЖЕНИЙ** | **МЯЧЕЙ ЗАБИТО** | **МЯЧЕЙ ПРОПУЩ-**  **ЕНО** | **РАЗНИЦА МЯЧЕЙ** | **ОЧКИ** | **ID ГРУППЫ** |

**5. Результат в четверти**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID РЕЗ. КОМАНДЫ В ЧЕТВЕРТИ** | **НАЗВАНИЕ**  **КОМАНДЫ** | **МЕСТО ВТОПЕ** | **ОБЩЕЕ КОЛ-ВО ИГР** | **ПОБЕД** | **НИЧЕЙ** | **ПОРАЖЕНИЙ** | **МЯЧЕЙ ЗАБИТО** | **МЯЧЕЙ ПРОПУЩ-**  **ЕНО** | **РАЗНИЦА МЯЧЕЙ** | **ОЧКИ** |

**6. Команды**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название команды** | **Вратари** | **Защитники** | **Полузащитники** | **Нападающие** |

**Новая реляционная модель:**

|  |  |  |
| --- | --- | --- |
| **ID ГРУППЫ** | **ID МАТЧА** | НОМЕР ГРУППЫ |

1

M

|  |  |  |
| --- | --- | --- |
| **ID ЧЕТВЕРТИ** | **ID МАТЧА** | НОМЕР ЧЕТВЕРТИ |

M

1

1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID МАТЧА** | ДАТА | ПЕРВАЯ  КОМАНДА | ВТОРАЯ  КОМАНДА | РЕЗ. ПЕРВОЙ КОМАНДЫ | РЕЗ. ВТОРОЙ КОМНАДЫ | РЕЗ. МАТЧА |

1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID РЕЗ. КОМАНДЫ В ГРУППЕ** | НАЗВАНИЕ  КОМАНДЫ | МЕСТО ВТОПЕ | ОБЩЕЕ КОЛ-ВО ИГР | ПОБЕД | НИЧЕЙ | ПОРАЖЕНИЙ | МЯЧЕЙ ЗАБИТО | МЯЧЕЙ ПРОПУЩ-  ЕНО | РАЗНИЦА МЯЧЕЙ | ОЧКИ | **ID ГРУППЫ** |

1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID РЕЗ. КОМАНДЫ В ЧЕТВЕРТИ** | НАЗВАНИЕ  КОМАНДЫ | МЕСТО ВТОПЕ | ОБЩЕЕ КОЛ-ВО ИГР | ПОБЕД | НИЧЕЙ | ПОРАЖЕНИЙ | МЯЧЕЙ ЗАБИТО | МЯЧЕЙ ПРОПУЩ-  ЕНО | РАЗНИЦА МЯЧЕЙ | ОЧКИ |

1

1

1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название команды** | Вратари | Защитники | Полузащитники | Нападающие |

**РГР. Часть 3**

# Задание на РГР

Создать ПО для отображения и обработки статистических данных для определённого вида спорта. ПО должно включать 2 основных окна: окно отображающее таблицы БД со статистической информацией и результаты запросов к БД, переключение таблиц и результатов должно быть реализовано через вкладки; и окно для менеджера запросов к БД.

Первое окно должно давать возможность просматривать и изменять все таблицы БД, а также просматривать результаты запросов к БД. Должна иметься возможность удалить вкладки с результатами запросов, но не вкладки с таблицами. Также должна иметься возможность перейти к окну менеджера запросов.

Окно менеджера запросов должно предоставлять интерфейс для создания, сохранения, удаления, редактирования запросов. Созданные запросы должны отображаться в виде списка с названиями запросов, в который можно добавлять новые запросы, удалять, просматривать существующие. Для создания и редактирования запросов должен предоставляться визуальный интерфейс, а не язык запросов. Редактор запросов должен поддерживать операции выборки, соединения, группирования, подзапросы (в качестве подзапроса используются ранее сохранённые запросы).

Ход работы:

1. Исследование предметной области и создание ER диаграммы.
2. Перевод ER диаграммы в реляционную модель, создание и заполнение БД.
3. Проработка визуального интерфейса приложения
4. Создание диаграммы классов приложения
5. Реализация основного окна приложения
6. Реализация менеджера запросов
7. Тестирование и отладка

**Вариант задания - 1**

1. Футбол (клубы) ([https://www.whoscored.com/Statistic](https://www.whoscored.com/Statistics)s <https://www.soccerstats.com/> <https://fbref.com/en/>)

**Проработка визуального интерфейса приложения**

**Главное окно приложения содержит:**

**Окно инструментов:**

**File** - кнопка, которая открывает вложенное окно, содержащее:

**Save** - кнопка, которая перезаписывает исходный файл базы данных.

**Load** - кнопка, которая загружает указанный файл в приложение и выводит все отношения, содержащиеся на данный момент в базе данных.

**Tables** - открытая по умолчанию вкладка, отображающая все отношения базы данных. Если пользователь находится во Write request, то при нажатии на кнопку Tables, пользователь будет возвращаться обратно ко всем отношениям базы данных.

**Write request** - кнопка, которая отвечает за переход к написанным нами SQL-запросам, в правой части приложения за место кнопок Table 1, 2, 3, …, n, появятся Request 1, 2, 3, …, n, За место атрибутов таблицы, будут выведены атрибуты запроса.

**About** - кнопка, которая открывает вложенное окно с карт. Описанием кнопок и информацией о создателе приложения.



**Панель отображения таблиц:**



В этой панели отображаются все отношения базы данных, при переходе в на панели инструментов во вкладку Request, все названия отношения поменяются на название запросов.

**Панель отображения атрибутов:**



Панель, в которой находятся все атрибуты каждого отношения. В каждом столбце атрибута, будет содержаться информация описанная в атрибуте. Столбцы можно будет скорллить как в лево, так и вправо с помощью скролл-полей.



**Окно написания запросов содержит:**

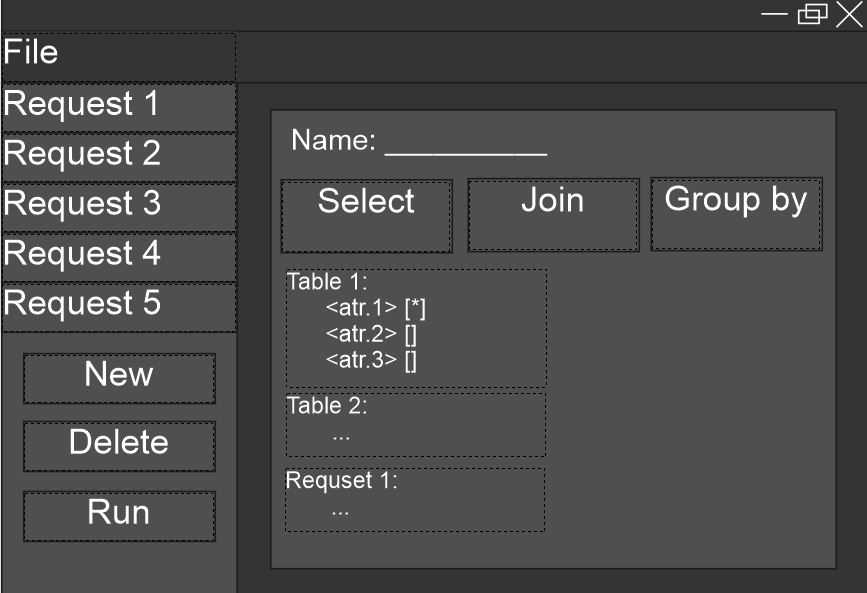
**Окно инструментов:**

**File -** некликабельное поле, отвечающее за панель взаимодействия с файлом.

**Список кнопок с запросами -** набор кнопок, который отвечает за переключение между запросами. Отображает название запроса и то, что в нем содержится.

**New -** кнопка, отвечающая за создание нового запроса.

**Delete -** кнопка, отвечающая за удаление уже существующего запроса.

**Run -** кнопка, отвечающая за запуск запроса и формирование отношения.

**Окно написание запроса:**

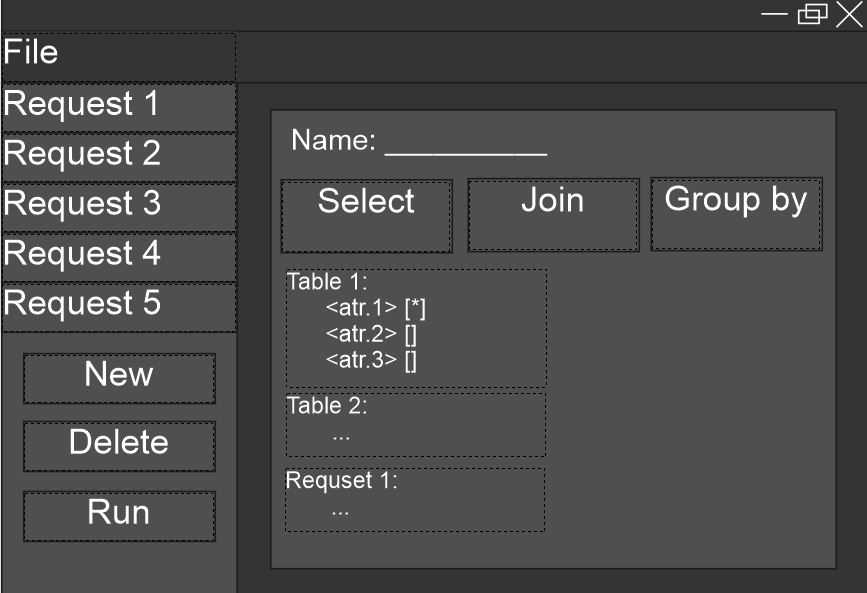
Поле Name отвечает за название запроса.

**Select -** кнопка, формирующаязапрос аналогичный запросу Select в SQL.

**Join -** кнопка, формирующаязапрос аналогичный запросу Join в SQL.

**Group by -** кнопка, формирующаязапрос аналогичный запросу Group by в SQL.

Представленные ниже поля будут отвечать за введения атрибутов для взаимодействия в запросе. А также из каких таблиц их брать. Поля с Request будут отвечать за подзапросы.



**РГР. Часть 4**

**Создание диаграммы классов приложения**

В первую очередь, стоит отметить, что архитектура приложения будет основана на архитектуре Avalonia MVVM (Model-View-ViewModel), с использованием кроссплатформенного XAML фреймворк для платформы .NET Avalonia.

Бизнес логика приложения, в паттерне первая буква “M” (**Model**). В данном слое описывается логика и основные задачи приложения. Взаимодействие с файловой системой, базой данных, API, описание сущностей системы и т.п. Часто общение с различными источниками данных, выделяют в отдельную под-часть (Services).

Интерфейс – в паттерне буква V (**View**) описывается с помощью языка разметки XAML.

ViewModel — связывающий слой между Model и View с помощью технологии привязки (Binding). Для понятия Binding, введем понятие свойства (Property) – изменяемое поле данных во ViewModel. Простыми словами, с помощью binding, все property, описанные в ViewModel доступны для View. Важным, также является **изменяемость** property – под этим следует, что любые изменения во View или Model о которых "узнает" ViewModel будут автоматически изменены в зависимости от того, откуда пришли изменение (ввод текстового поля, получение ответа от API и т.п)

**Описание слоев:**

**Model:**

Опишем здесь классы, которые являются объектами предметной области, которую мы моделируем.

**Champion\_League\_Football\_AlkhimovichContext : DbContext**  – контекстуальный класс, который позволяет работать с базой данных через модели. Каждый DbSet представляет собой сущности в БД

public virtual DbSet<Group> Groups - представление сущности отношения Groups;

public virtual DbSet<GroupsResault> GroupsResaults - представление сущности отношения Groups;

public virtual DbSet<Match> Matches - представление сущности отношения Groups;

public virtual DbSet<Quarter> Quarters - представление сущности отношения Groups;

public virtual DbSet<QuartersResault> QuartersResaults - представление сущности отношения Groups;

public virtual DbSet<Team> Team - представление сущности отношения Groups;

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) - метод считывания базы данных, из которой будет собираться информация.

protected override void OnModelCreating(ModelBuilder modelBuilder) - метод, который точно пересоздает каждое отношение нашей базы данных как объект, с которым можно взаимодействовать в коде.

**Классы представления нашей базы данных:**

**public partial class Group**

public string GroupId;

public string MatchId;

public string GroupNum;

public virtual Match Match;

**public partial class GroupsResault**

public string GroupsTeamResId;

public string? Team;

public long? Place;

public long? GamesPlayed;

public long? Wins;

public long? Draws;

public long? Loses;

public long? GfBallsScored;

public long? GaBallsConceded;

public long? GdAccountDifference;

public long? PtsPoints;

**public partial class Match**

public Match();

public string MatchId;

public string? Date;

public string FirstTeam;

public string SecondTeam;

public long FirstTeamResault;

public long SecondTeamResault;

public long WhoWon;

public virtual ICollection<Group> Groups;

public virtual ICollection<Quarter> Quarters;

**public partial class Quarter**

public string QuartersId;

public string MatchId;

public double QuartersNum;

public virtual Match Match;

**public partial class QuartersResault**

public string QuartersTeamResId;

public string? Teams;

public long? Place;

public long? GamesPlayed;

public long? Wins;

public long? Draws;

public long? Loses;

public long? GfBallsScored;

public long? GaBallsConceded;

public long? GdAccountDifference;

public long? PtsPoints;

public virtual Team? TeamsNavigation;

**public partial class Team**

public Team();

public string TeamName;

public string? Goalkeepers;

public string? Defenders;

public string? Midfielders;

public string? Fowards;

public virtual ICollection<QuartersResault> QuartersResaults;

**public class Request : INotifyPropertyChanged -** класс, отвечающий за написание запросов к базе данных.

**ViewModel:**

**public class MainWindowViewModel : ViewModelBase -** класс, в котором представлены все основные данные и методы для работы со слоем View, каждая ObservableCollection – наблюдаемая коллекция, содержимое которых будет отображаться в окне программы, также они непосредственно связаны с моделями из DatabaseContext.

public ObservableCollection<Group> Group;

public ObservableCollection<GroupsResault> GroupsResault;

public ObservableCollection<Match> Match;

public ObservableCollection<Quarter> Quarter;

public ObservableCollection<QuartersResault> QuartersResault;

public ObservableCollection<Team> Team;

public void CreateRequest();

public void DeleteRequest(Request e);

public void SQLRequestOpen();

public void SQLRequestRun();

public void DeleteGroup(Group entity);

public void DeleteGroupsResault(GroupsResault entity);

public void DeleteMatch(Match entity);

public void DeleteQuarter(Quarter entity);

public void DeleteQuartersResault(QuartersResault entity);

public void DeleteTeam(Team entity);

public void CreateGroup();

public void CreateGroupsResault();

public void CreateMatch();

public void CreateQuarter();

public void CreateQuartersResault();

public void CreateTeam();

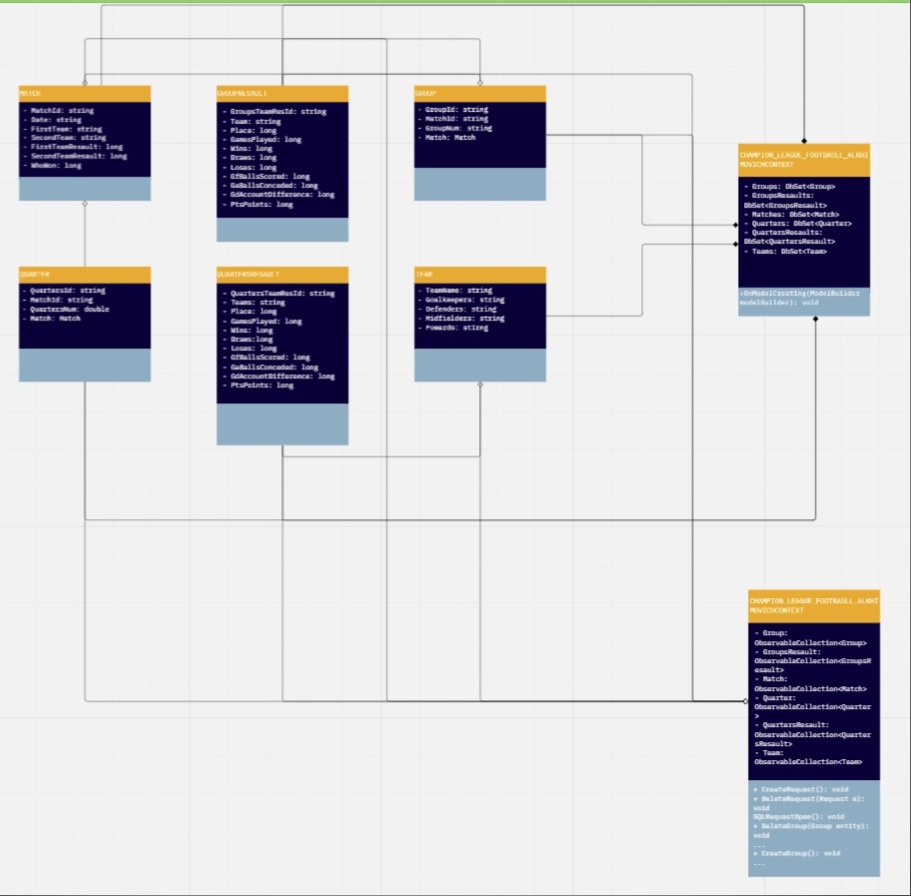
**View:**

**MainWindow -** отрисовка главного окна.

**DataBaseView -** отрисовка всех отношений в главном окне и методов взаимодействия с ними.

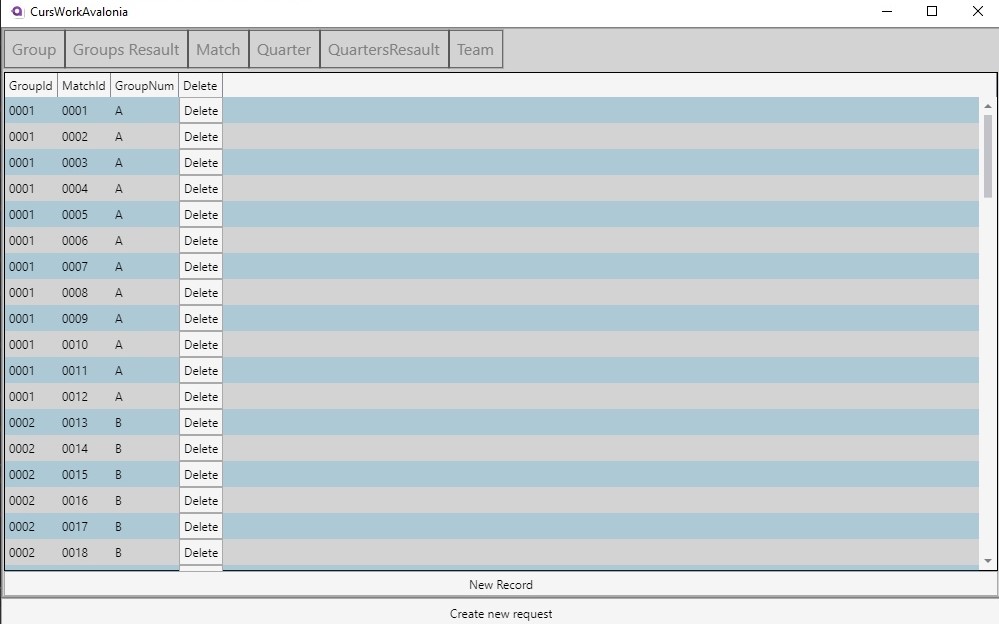
**SQLRequestView** - отрисовка запросов к бд.

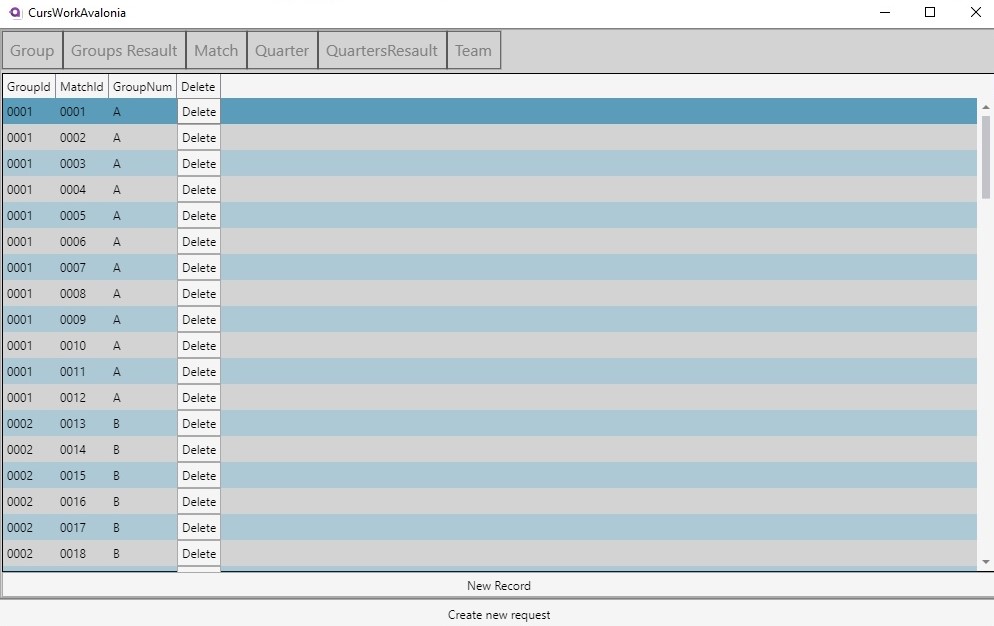
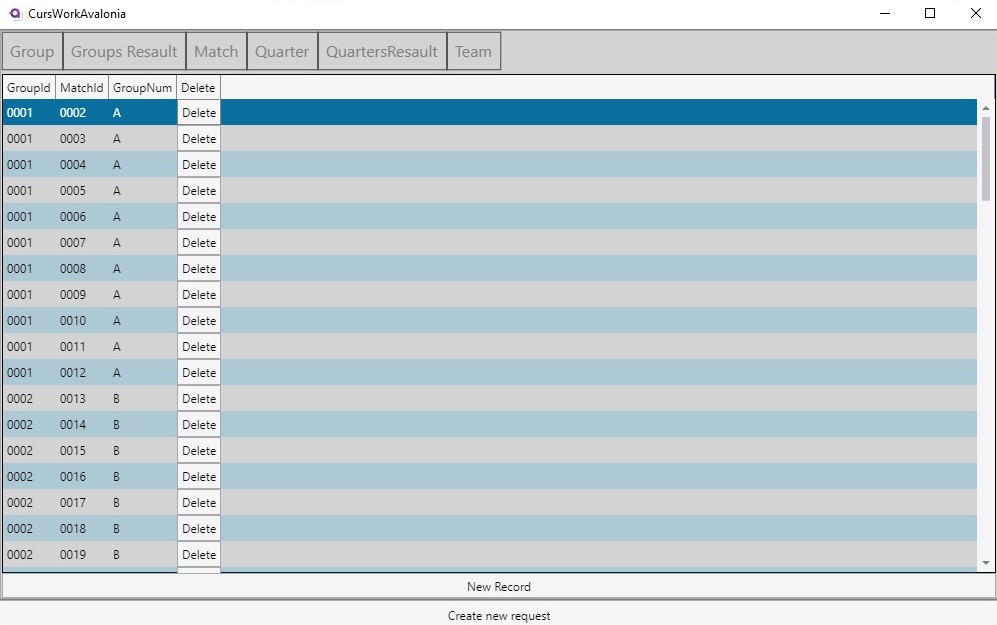
**UML-диаграмма:**

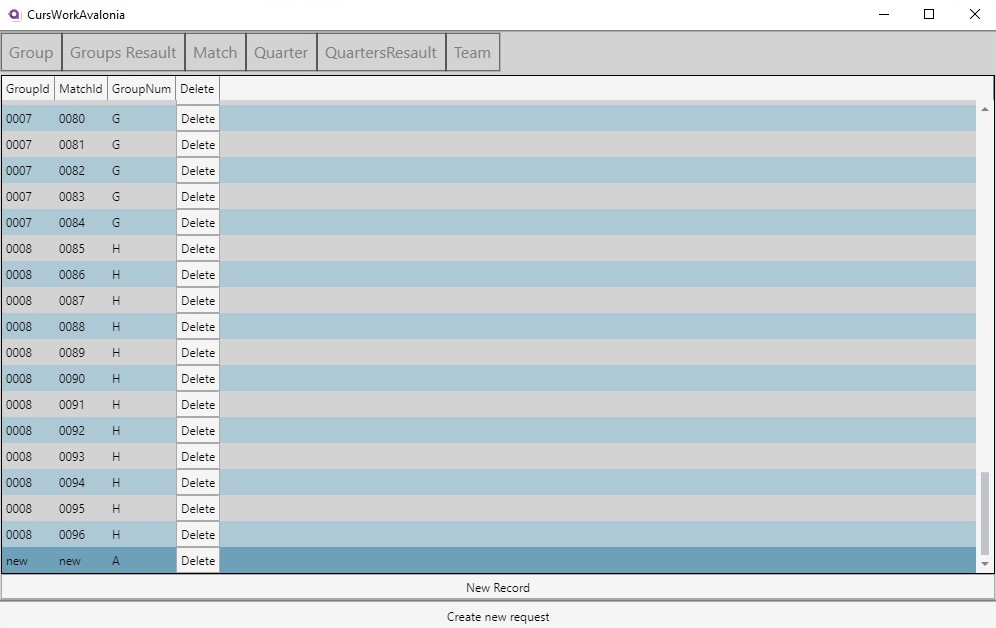
****

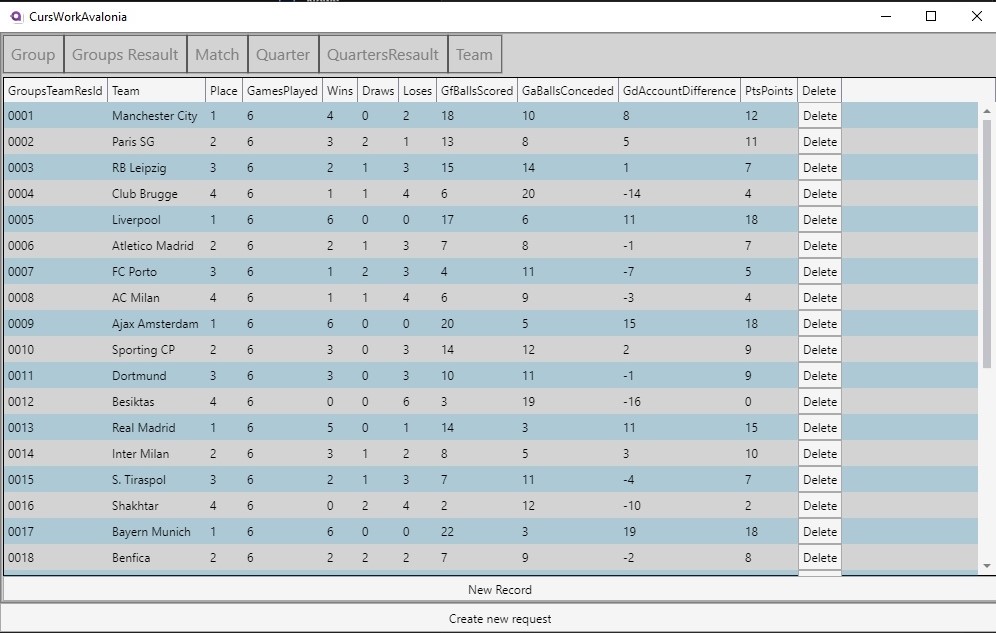
**РГР. Часть 5**

Создал основное окно приложения и реализовал методы взаимодействия с элементами базы данных (удаление и добавление).



****

****

****