Отчёт по курсовому проекту по дисциплине «Базы данных» Вариант 54. «Эстафеты»

Выполнил: Миннахметов Э.Ю.
Проверил: Петренко А.А.

Цель: Разработать информационную систему «Эстафеты».

Задачи:

1) провести анализ предметной области;

2) разработать структуру базы данных;

3) разработать приложение для работы с базой данных.

Текст варианта 54

Разработайте реляционную базу данных Эстафета.

В базе данных должны храниться сведения о:

- спортсменах;
- показанных ими результатах;
- принятых участиях;
- результатах предыдущих эстафет;
- результатах соперников.

Необходимо решать задачи формирования эстафетной команды, способной победить в новых эстафетах.

Среды реализации моделей

Тип приложения	Достоинства	Недостатки
Консольное приложение	- нет ГПИ, достаточно написать диалоговый интерфейс - быстрота выполнения - не требуется многопоточность	- некрасивый внешний вид
Десктопное приложение	- красивый внешний вид - быстрота выполнения	- необходима многопоточность - сложнее в написании
Мобильное приложение	- красивый внешний вид - быстрота выполнения	- необходима многопоточность - сложнее в написании
Веб-приложение	- браузер – готовый клиент - богатство инструментария - не требуется многопоточность	- медленно в исполнении

Языки программирования

Язык программирования	Достоинства	Недостатки
PHP	- лёгок для начинающих веб-программистов	- отсутствие полноценного ООП - динамическая типизация
Python	- мощные средства для веб-разработки	- динамическая типизация
C#	- мощные средства для веб-разработки	- требует сервер на Windows
Java	- мощные средства для веб-разработки	- скромнее, по сравнению с C#

Среды разработки

Язык программирования	Достоинства	Недостатки
Visual Studio Code	- легковесная - богатый центр расширений	- требует настройки для каждого языка программирования
Eclipse	- подсветка кода - подсказки	- не удобный интерфейс
Intellij IDEA Community	- подсветка кода - подсказки - удобный интерфейс	- требовательна к аппаратному обеспечению

Вспомогательные инструменты

Фреймворк Spring Boot позволит обрабатывать запросы в методах классов-контроллеров. Альтернатива Java EE.

Шаблонизатор ThymeLeaf позволит генерировать HTML-текст. Идет в комплекте со Spring Boot.

Почему Spring Boot?

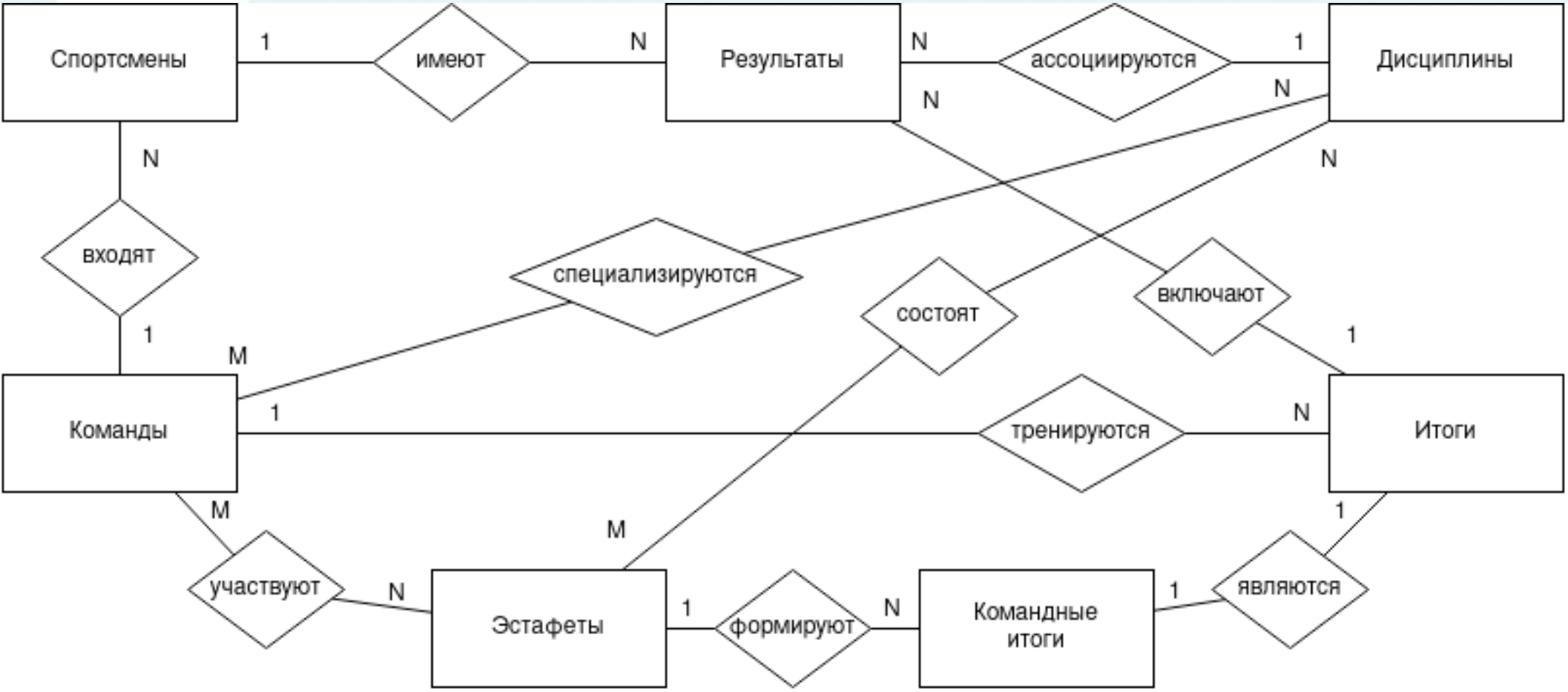
Java EE

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    }
```

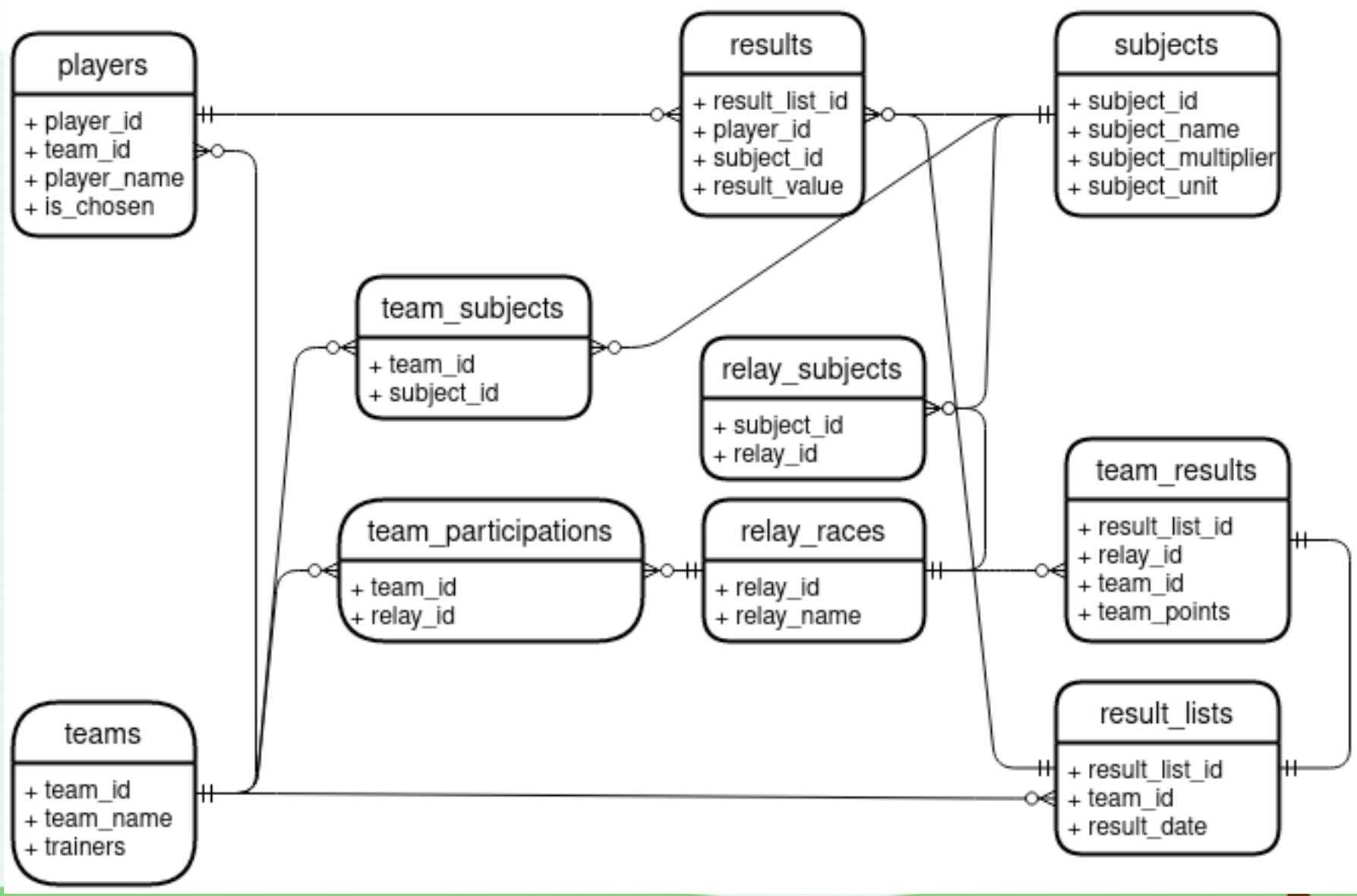
Spring Boot

```
@Controller
public class MainController extends AController {

    @GetMapping("/home")
    public String home() {
        return "general/home";
    }

    @GetMapping("/about")
    public String about() {
        return "general/about";
    }
```

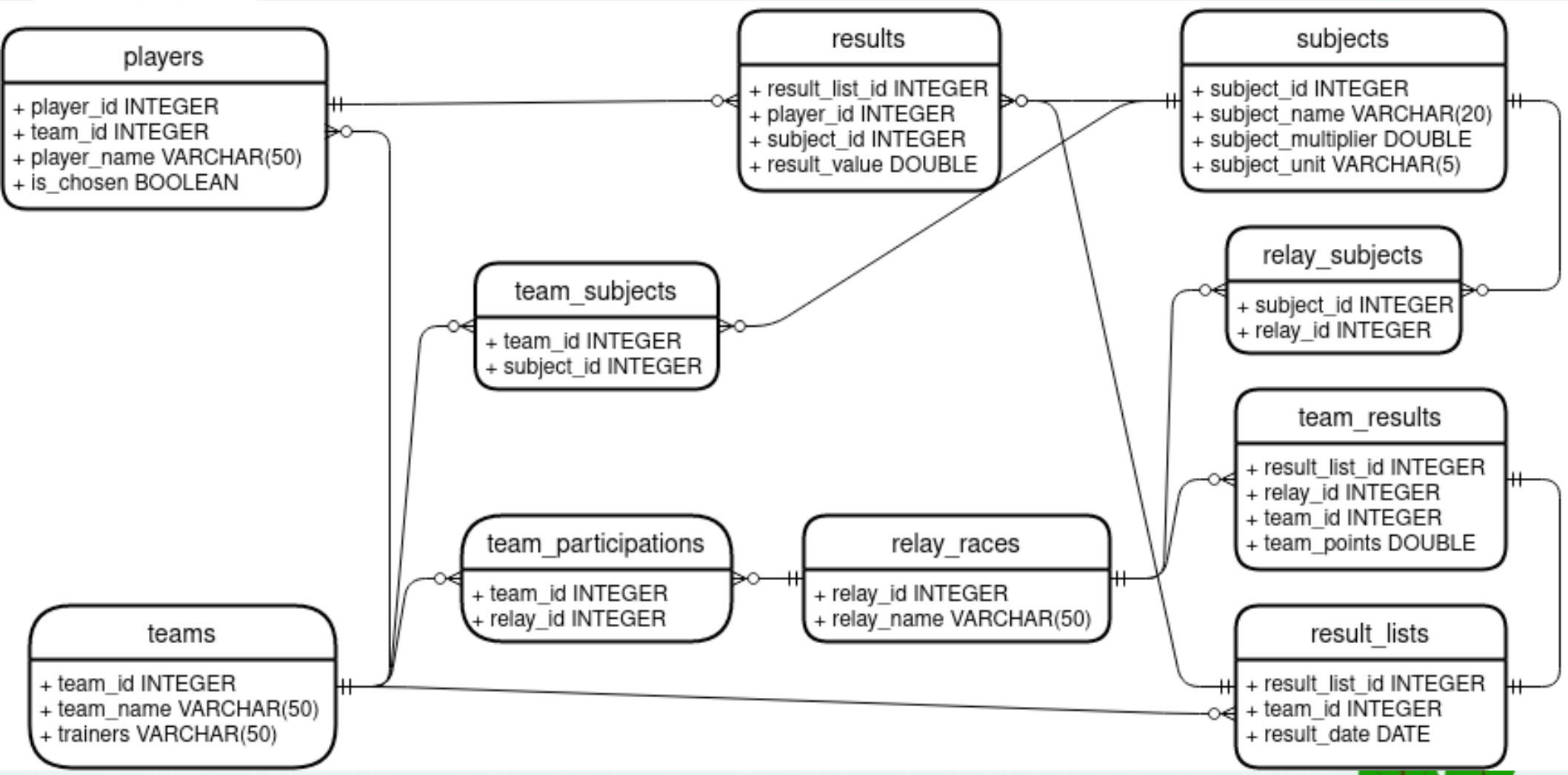
Концептуальная модель



Логическая модель



Физическая модель



Системы управления базами данных (СУБД)

СУБД	Достоинства	Недостатки	Возможность для реализации поставленной задачи
Oracle	- богатый функционал	платная	+
PostgreSQL	- богатый функционал - бесплатная	- не удобный клиент pgAdmin	+
MySQL	- богатый функционал - бесплатная - удобный клиент MySQL Workbench		+

Создание базы данных

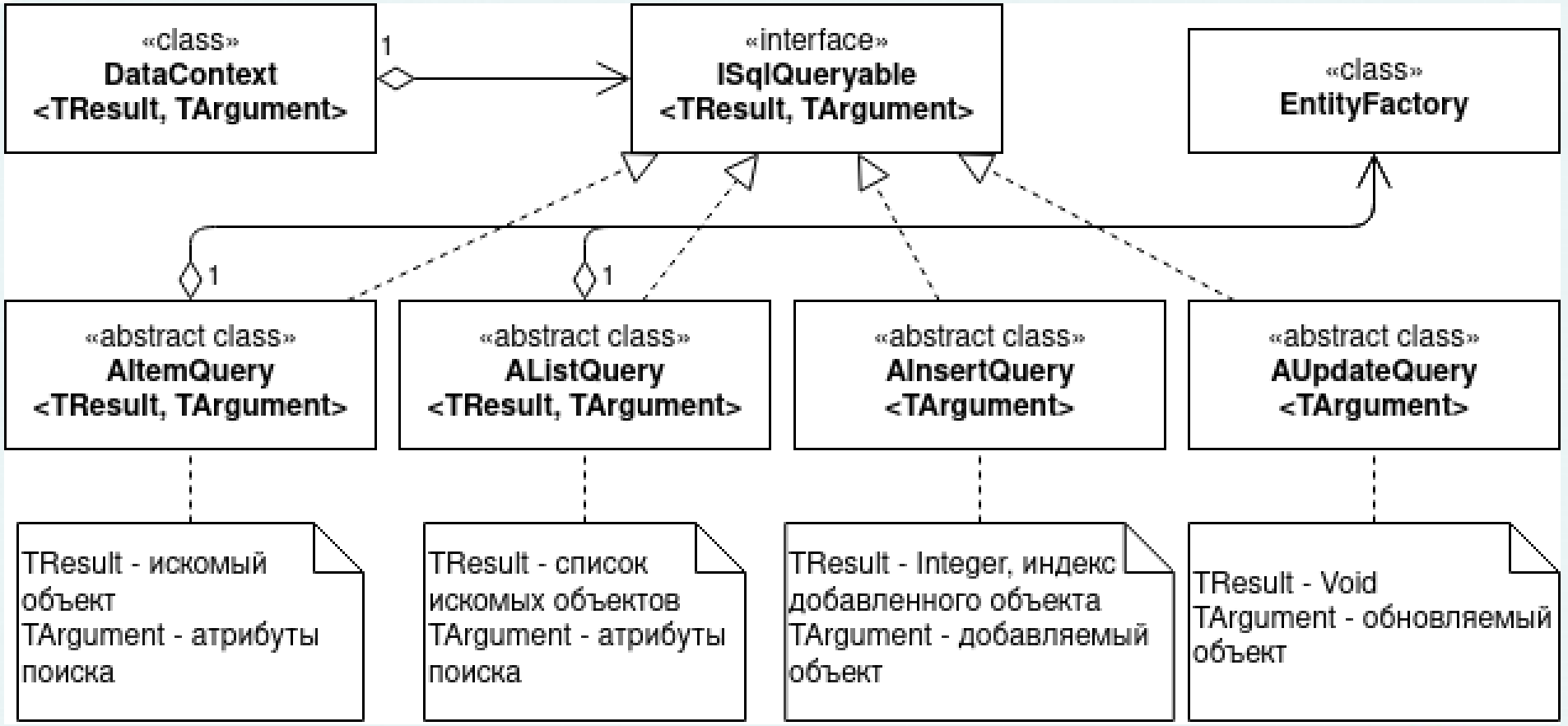
Представление player_views

```
CREATE VIEW player_views
(player_id, player_name, team_id, team_name)
AS SELECT players.player_id, players.player_name, players.team_id, teams.team_name
FROM players JOIN teams ON players.team_id = teams.team_id
UNION SELECT players.player_id, players.player_name, players.team_id, NULL AS team_name
FROM players WHERE players.team_id IS NULL;
```

Процедуры выборки списка спортсменов и поиска конкретного

```
DELIMITER //
CREATE PROCEDURE get_player_list()
    BEGIN
        SELECT * FROM player_views;
    END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE find_player(IN arg_id INT)
    BEGIN
        SELECT * FROM player_views WHERE player_id = arg_id;
    END //
DELIMITER ;
```

Построение архитектуры приложения



EntityFactory

```
public class EntityFactory {

    public Player player() throws SQLException {
        Player item = new Player();
        item.setPlayerId(_set.getInt( columnLabel: "player_id"));
        item.setTeamId(_set.getInt( columnLabel: "team_id"));
        item.setPlayerName(_set.getString( columnLabel: "player_name"));
        item.setTeamName(_set.getString( columnLabel: "team_name"));
        return item;
    }

    public Subject subject() throws SQLException {
        Subject item = new Subject();
        item.setSubjectId(_set.getInt( columnLabel: "subject_id"));
        item.setSubjectName(_set.getString( columnLabel: "subject_name"));
        item.setSubjectUnit(_set.getString( columnLabel: "subject_unit"));
        item.setSubjectMultiplier(_set.getDouble( columnLabel: "subject_multiplier"));
        return item;
    }
```

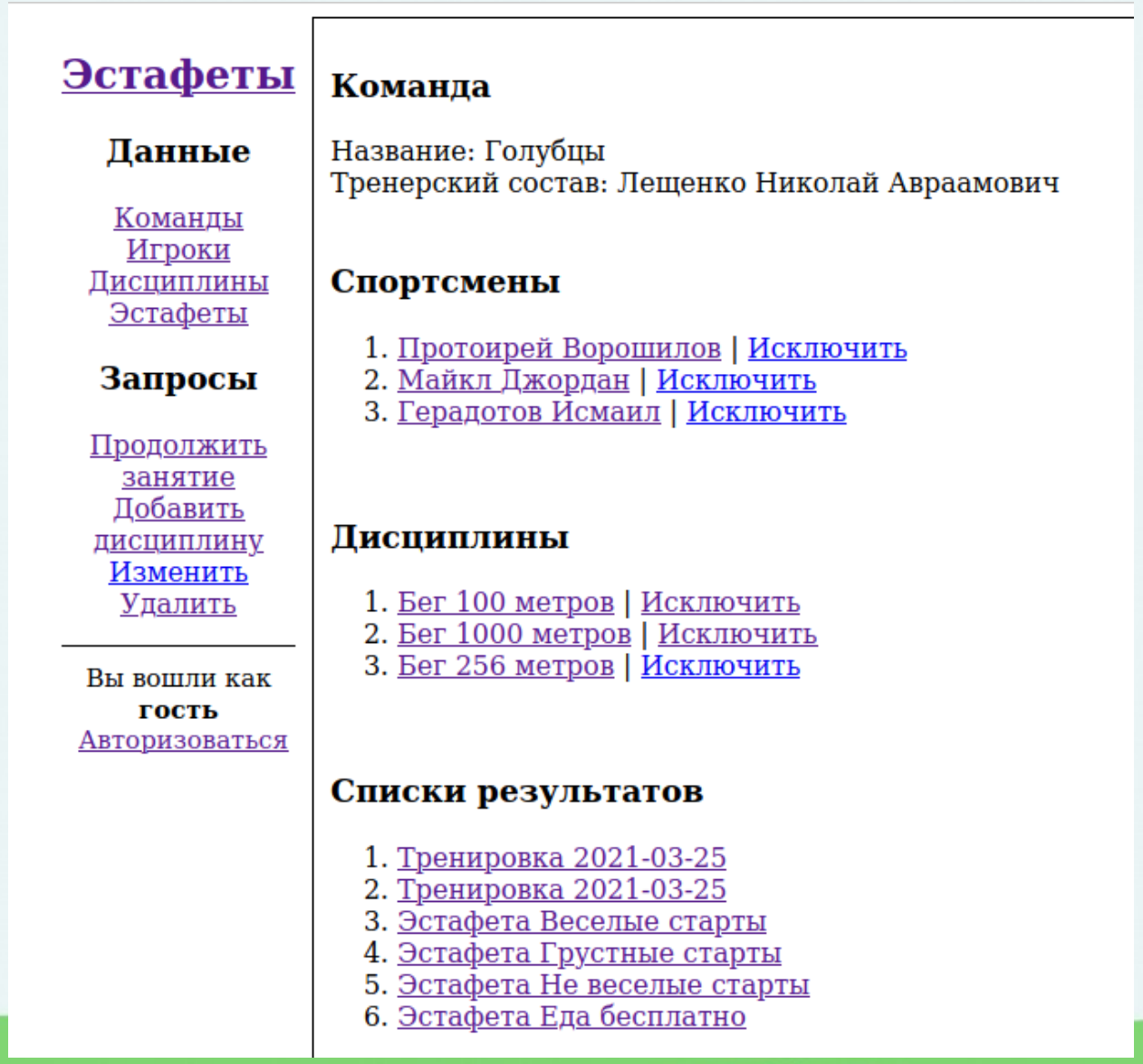
Типичный класс запроса

```
public class PlayerQuery extends AItemQuery<Player, String> {
    @Override
    protected String query(String arg) {
        return String.format("CALL find_player(%s);", arg);
    }

    @Override
    protected Player item(EntityFactory builder) throws SQLException {
        return builder.player();
    }
}
```

Типичный класс-контроллер

```
@Controller
public class MainController extends AController {

    @GetMapping("/player")
    public String player(@RequestParam(name = "id") String id, Model model) {
        Player player = (Player)(new DataContext(new PlayerQuery()).provide(id));
        Collection<Result> results = (Collection<Result>)(new DataContext(new PlayerResultListQuery()).provide(id));
        model.addAttribute( attributeName: "player", player);
        model.addAttribute( attributeName: "results", results);
        return "general/player";
    }

    @GetMapping("/players")
    public String players(Model model) {
        Collection<Player> players = (Collection<Player>)(new DataContext(new PlayerListQuery()).provide( argument: null));
        model.addAttribute( attributeName: "players", players);
        return "general/players";
    }
```

Графический пользовательский интерфейс



Заключение

Перспективы дальнейшей разработки:

- возможность установки ограничений на число команд, участвующих в эстафете;
- позволять/запрещать внутри команды выставлять разных игроков для каждой дисциплины;
- усовершенствовать средствами CSS и языка написания сценариев JavaScript.

Данная информационная система может использоваться на легкоатлетических соревнованиях, в качестве администратора информационной системы будет выступать организатор соревнования, редакторов – тренера команд и представители жюри, гостя – спортсмены и наблюдатели соревнований.