

**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО**

Физико-механический институт

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

«Генерация диаграмм UML»
по дисциплине «Автоматизация научных исследований»

Выполнил: Смирнов Александр Дмитриевич
студент группы 5040102/50201

Преподаватель: Новиков Ф.А.

Санкт-Петербург
2025 г.

1. Цель работы

Разработать и проанализировать диаграммы UML (диаграмму вариантов использования и диаграмму классов) для выбранной информационной системы. Использовать PlantUML для генерации, предоставить исходный код, изображения диаграмм и пояснения.

2. Задание

Задание и требования приведены в методических указаниях к лабораторной работе №3 и включают построение двух диаграмм (Use Case и Class) с соблюдением стандартов UML и использованием PlantUML.

3. Выбор предметной области и описание системы

Предметная область: анализ и визуализация волейбольной статистики по матчам с привязкой к видеозаписям и разметкой розыгрышей.

Проектируемая система: «Система волейбольной статистики и видеоразметки», предназначенная для сбора данных матчей, расчёта KPI, визуализации статистики, формирования отчётов и хранения/разметки видео моментов.

3.1. Действующие лица (actors)

- Игрок — просматривает личную и командную статистику, а также видео моментов.
- Тренер/Аналитик — проводит глубокий анализ, выполняет разметку розыгрышей, формирует отчёты и экспортирует данные.
- Менеджер команды — загружает данные матчей и видео, формирует отчётность для команды/турнира.
- Администратор системы — управляет пользователями, ролями и интеграциями.
- Внешнее видеохранилище — источник/приёмник видео (интеграция при синхронизации).

3.2. Основные функциональные требования

- Аутентификация пользователей.
- Импорт данных матчей и загрузка видеозаписей.
- Просмотр статистики и расчёт KPI (с учётом предобработки данных).

- Просмотр видеомоментов и расширение функционала разметкой розыгрышей.
- Формирование отчётов и экспорт данных.
- Администрирование: управление пользователями и ролями, поддержка интеграции с видеохранилищем.

4. Исследование генерации диаграмм (2 модели ИИ)

В рамках исследования использовались две модели: GPT-5.2 Thinking и DeepSeek. Для каждой модели применялись 4 промпта для диаграммы вариантов использования и 4 промпта для диаграммы классов (от минимального к продвинутому).

4.1. Результаты: GPT-5.2 Thinking

4.1.1. Диаграмма вариантов использования (Use Case)

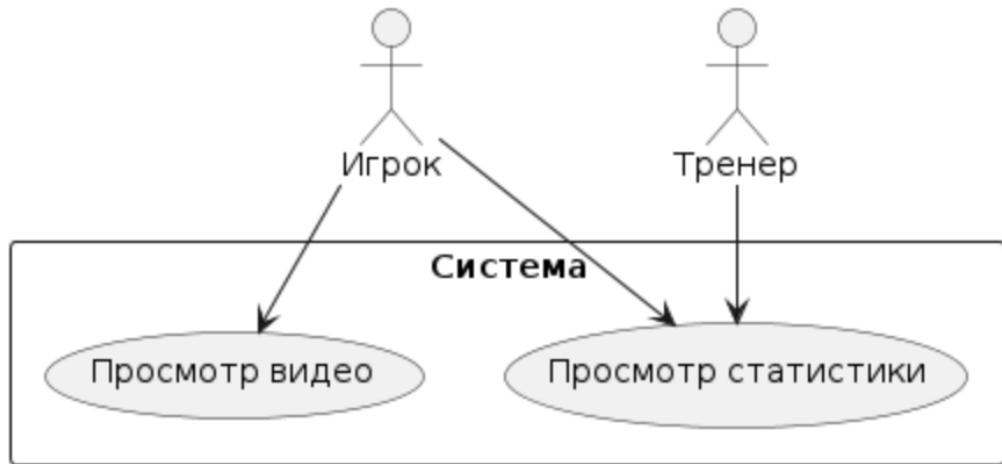
Промпт A1 (минимальный)

Промпт:

Сгенерируй PlantUML код для диаграммы вариантов использования системы волейбольной статистики и видеоразметки. Укажи действующие лица, перечисли основные случаи использования.

Ответ (PlantUML):

```
@startuml
actor Игрок
actor Тренер
rectangle "Система" {
    usecase "Просмотр статистики" as UC1
    usecase "Просмотр видео" as UC2
}
Игрок --> UC1
Игрок --> UC2
Тренер --> UC1
@enduml
```



Промпт А2 (базовый)

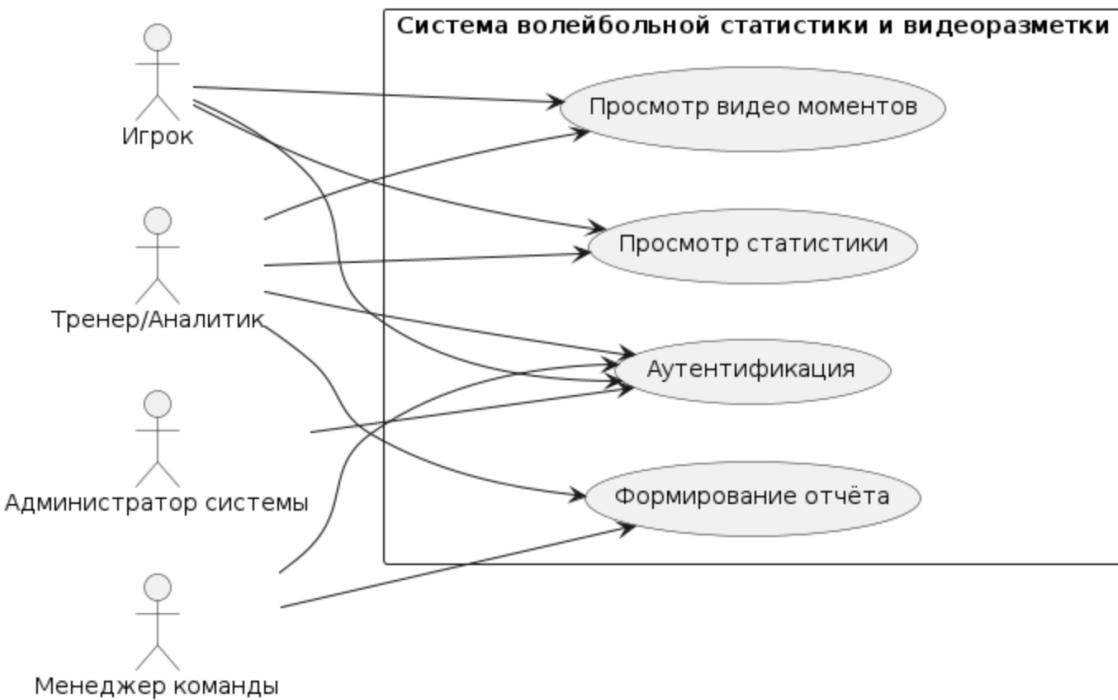
Промпт:

Сгенерируй PlantUML код для диаграммы вариантов использования системы волейбольной статистики и видеоразметки. Укажи акторов: Игрок, Тренер/Аналитик, Менеджер команды, Администратор системы. Перечисли основные случаи использования и добавь границу системы.

Ответ (PlantUML):

```

@startuml
left to right direction
skinparam packageStyle rectangle
actor "Игрок" as Player
actor "Тренер/Аналитик" as Coach
actor "Менеджер команды" as Manager
actor "Администратор системы" as Admin
rectangle "Система волейбольной статистики и видеоразметки" {
    usecase "Аутентификация" as UC_Login
    usecase "Просмотр статистики" as UC_ViewStats
    usecase "Просмотр видео моментов" as UC_ViewVideo
    usecase "Формирование отчёта" as UC_Report
}
Player --> UC_Login
Player --> UC_ViewStats
Player --> UC_ViewVideo
Coach --> UC_Login
Coach --> UC_ViewStats
Coach --> UC_ViewVideo
Coach --> UC_Report
Manager --> UC_Login
Manager --> UC_Report
Admin --> UC_Login
@enduml
    
```



Промпт А3 (расширенный)

Промпт:

Сгенерируй PlantUML код для диаграммы вариантов использования системы волейбольной статистики и видеоразметки. Перечисли use case по ролям и добавь include/extend там, где это логично. Добавь границу системы.

Ответ (PlantUML):

```

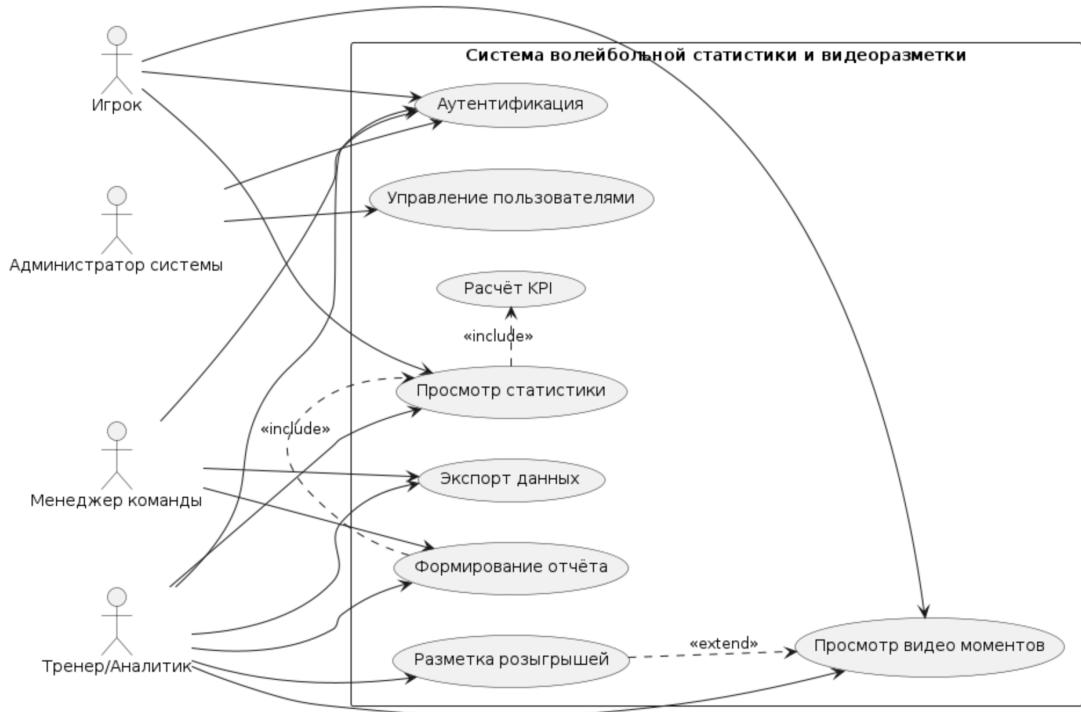
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Игрок" as Player
actor "Тренер/Аналитик" as Coach
actor "Менеджер команды" as Manager
actor "Администратор системы" as Admin
rectangle "Система волейбольной статистики и видеоразметки" {
    usecase "Аутентификация" as UC_Login
    usecase "Просмотр статистики" as UC_ViewStats
    usecase "Расчёт KPI" as UC_ComputeKPI
    usecase "Просмотр видео моментов" as UC_ViewVideo
    usecase "Разметка розыгрышей" as UC_TagRally
    usecase "Формирование отчёта" as UC_Report
    usecase "Экспорт данных" as UC_Export
    usecase "Управление пользователями" as UC_Users
}
UC_ViewStats .> UC_ComputeKPI : <<include>>
UC_TagRally ..> UC_ViewVideo : <<extend>>
UC_Report .> UC_ViewStats : <<include>>
Player --> UC_Login

```

```

Player --> UC_ViewStats
Player --> UC_ViewVideo
Coach --> UC_Login
Coach --> UC_ViewStats
Coach --> UC_ViewVideo
Coach --> UC_TagRally
Coach --> UC_Report
Coach --> UC_Export
Manager --> UC_Login
Manager --> UC_Report
Manager --> UC_Export
Admin --> UC_Login
Admin --> UC_Users
@enduml

```



Промпт А4 (продвинутый)

Промпт:

Сгенерируй PlantUML код для диаграммы вариантов использования системы волейбольной статистики и видеоразметки. Акторы: Игрок, Тренер/Аналитик, Менеджер команды, Администратор системы, Внешнее видеохранилище. Обязательно: граница системы, include/extend, логическая целостность.

Ответ (PlantUML):

```

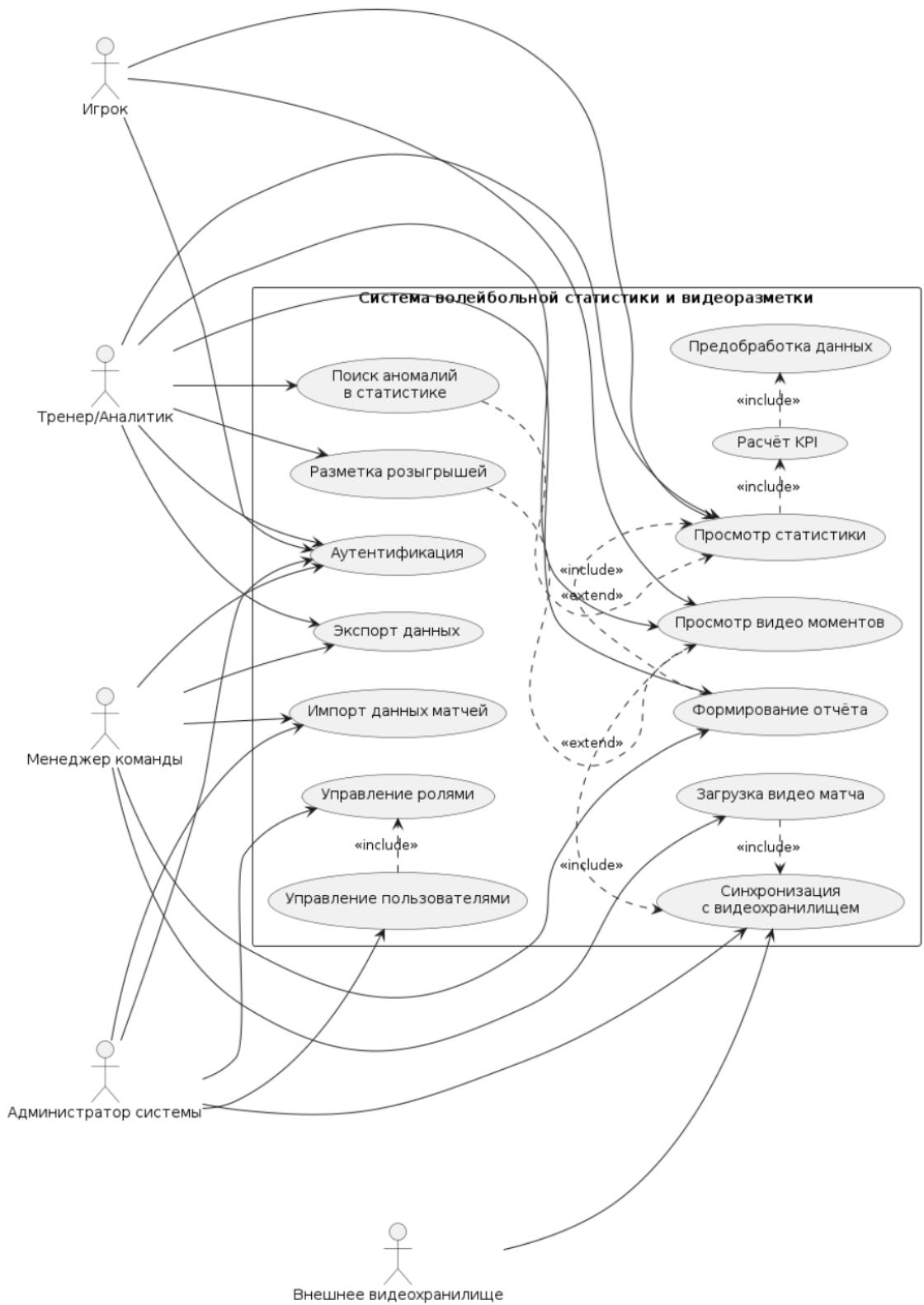
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Игрок" as Player
actor "Тренер/Аналитик" as Coach
actor "Менеджер команды" as Manager

```

```

actor "Администратор системы" as Admin
actor "Внешнее видеохранилище" as Storage
rectangle "Система волейбольной статистики и видеоразметки" {
    usecase "Аутентификация" as UC_Login
    usecase "Просмотр статистики" as UC_ViewStats
    usecase "Расчёт КПИ" as UC_ComputeKPI
    usecase "Предобработка данных" as UC_Preprocess
    usecase "Поиск аномалий\пв статистике" as UC_Anomaly
    usecase "Просмотр видео моментов" as UC_ViewVideo
    usecase "Синхронизация\пс видеохранилищем" as UC_SyncStorage
    usecase "Разметка розыгрышней" as UC_TagRally
    usecase "Загрузка видео матча" as UC_UploadVideo
    usecase "Импорт данных матчей" as UC_ImportData
    usecase "Формирование отчёта" as UC_Report
    usecase "Экспорт данных" as UC_Export
    usecase "Управление пользователями" as UC_Users
    usecase "Управление ролями" as UC_Roles
}
Player --> UC_Login
Player --> UC_ViewStats
Player --> UC_ViewVideo
Coach --> UC_Login
Coach --> UC_ViewStats
Coach --> UC_ViewVideo
Coach --> UC_TagRally
Coach --> UC_Report
Coach --> UC_Export
Coach --> UC_Anomaly
Manager --> UC_Login
Manager --> UC_ImportData
Manager --> UC_UploadVideo
Manager --> UC_Report
Manager --> UC_Export
Admin --> UC_Login
Admin --> UC_Users
Admin --> UC_Roles
Admin --> UC_ImportData
Admin --> UC_SyncStorage
Storage --> UC_SyncStorage
UC_ViewStats .> UC_ComputeKPI : <<include>>
UC_ComputeKPI .> UC_Preprocess : <<include>>
UC_Report .> UC_ViewStats : <<include>>
UC_ViewVideo .> UC_SyncStorage : <<include>>
UC_UploadVideo .> UC_SyncStorage : <<include>>
UC_Users .> UC_Roles : <<include>>
UC_Anomaly ..> UC_ViewStats : <<extend>>
UC_TagRally ..> UC_ViewVideo : <<extend>>
@enduml

```



4.1.2. Диаграмма классов (Class)

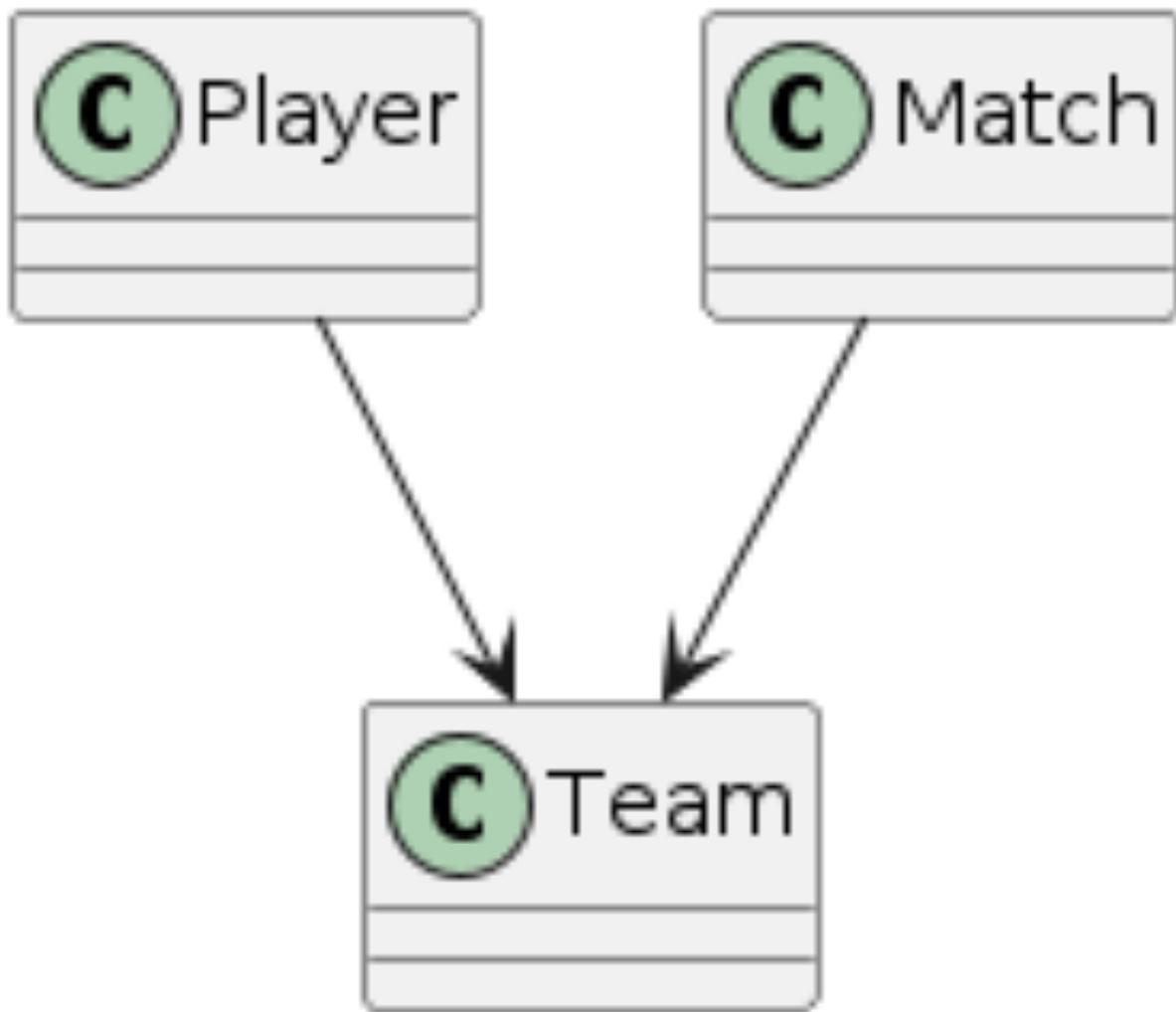
Промпт В1 (минимальный)

Промпт:

Сгенерируй PlantUML код для диаграммы классов системы волейбольной статистики и видеоразметки. Укажи основные классы и связи.

Ответ (PlantUML):

```
@startuml
class Player
class Match
class Team
Player --> Team
Match --> Team
@enduml
```



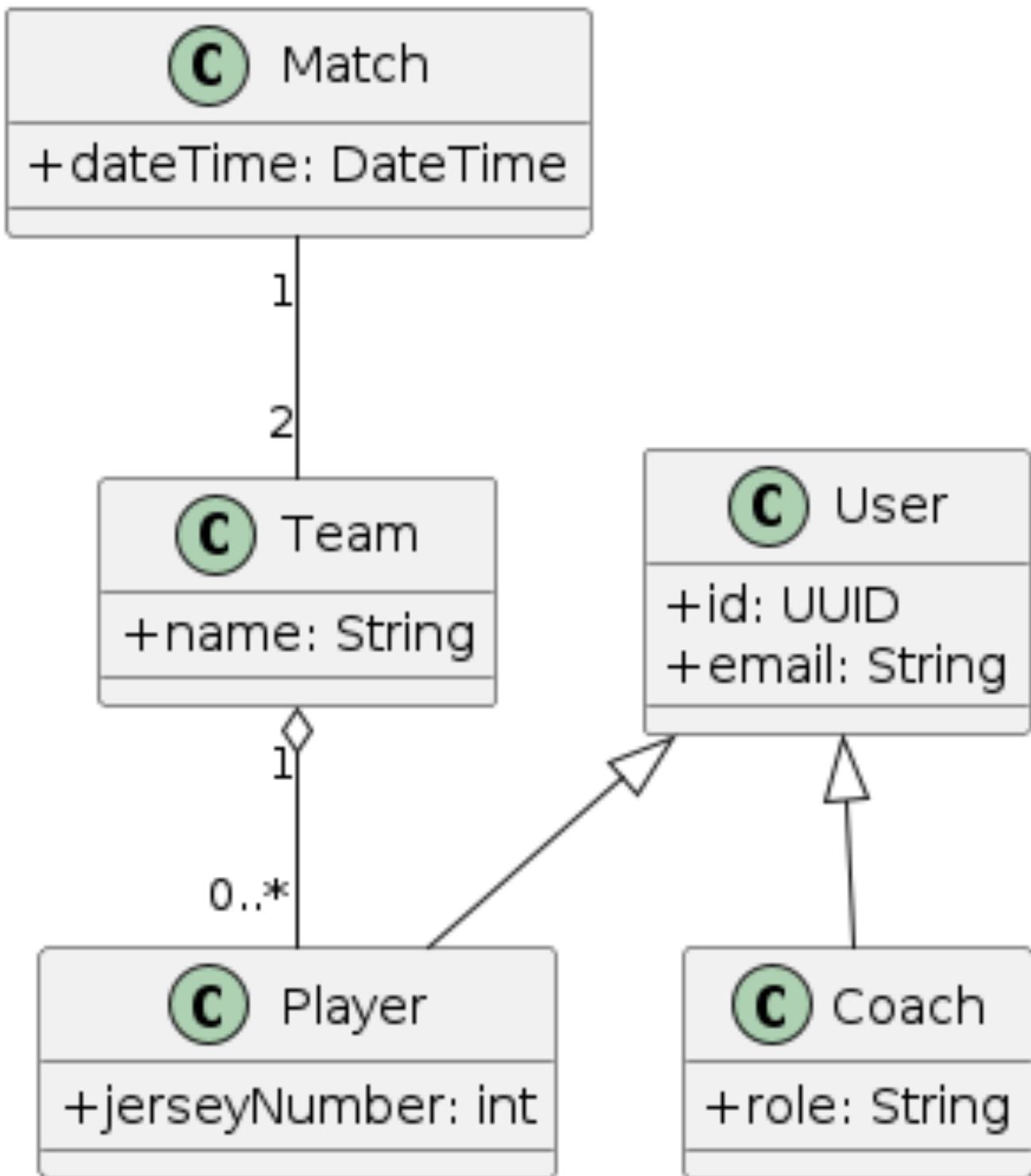
Промпт В2 (базовый)

Промпт:

Сгенерируй PlantUML код для диаграммы классов системы волейбольной статистики и видеоразметки. Классы: User, Player, Coach, Match, Team, Video, Report. Укажи атрибуты с типами и основные связи.

Ответ (PlantUML):

```
@startuml
skinparam classAttributeIconSize 0
class User {
    +id: UUID
    +email: String
}
class Player {
    +jerseyNumber: int
}
class Coach {
    +role: String
}
class Team {
    +name: String
}
class Match {
    +dateTime: DateTime
}
User <|-- Player
User <|-- Coach
Team "1" o-- "0..*" Player
Match "1" -- "2" Team
@enduml
```



Промпт В3 (расширенный)

Промпт:

Сгенерируй PlantUML код для диаграммы классов системы волейбольной статистики и видеоразметки. Добавь наследование User->Player/Coach/Manager/Admin, композиции

Match->Set->Rally->Action, кратности ассоциаций и enumeration для статусов/типов.

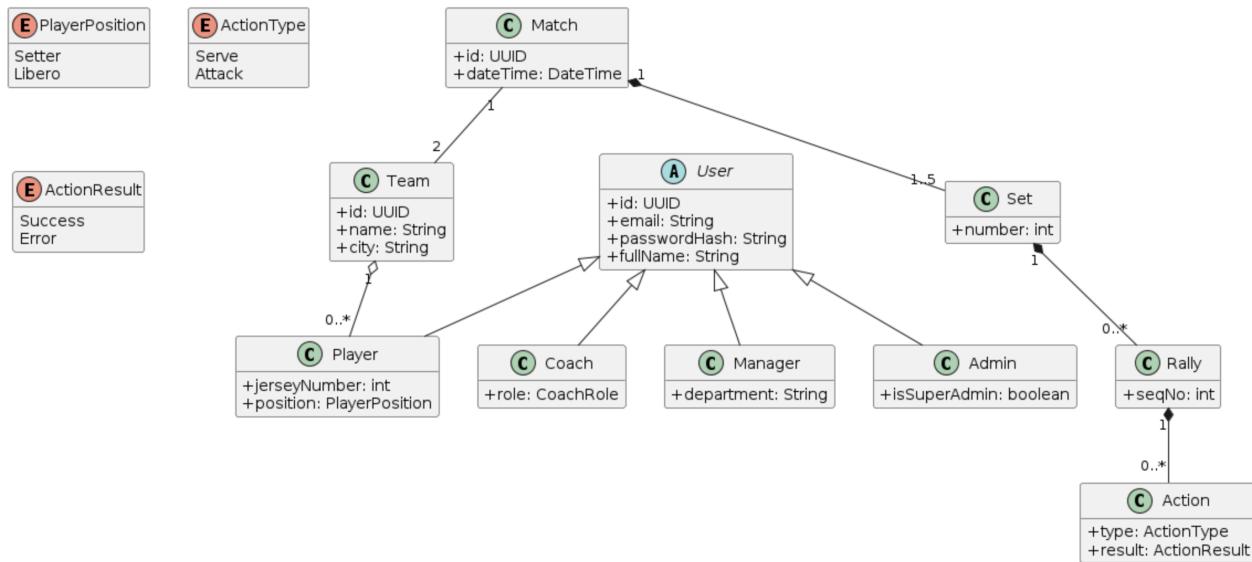
Ответ (PlantUML):

```
@startuml
skinparam classAttributeIconSize 0
hide empty members
abstract class User {
    +id: UUID
    +email: String
    +passwordHash: String
    +fullName: String
}
class Player {
    +jerseyNumber: int
    +position: PlayerPosition
}
class Coach {
    +role: CoachRole
}
class Manager {
    +department: String
}
class Admin {
    +isSuperAdmin: boolean
}
User <|-- Player
User <|-- Coach
User <|-- Manager
User <|-- Admin
class Team {
    +id: UUID
    +name: String
    +city: String
}
class Match {
    +id: UUID
    +dateTime: DateTime
}
class Set {
    +number: int
}
class Rally {
    +seqNo: int
}
class Action {
    +type: ActionType
    +result: ActionResult
}
enum PlayerPosition {
    Setter
    Libero
}
enum ActionType {
    Serve
    Attack
}
enum ActionResult {
```

```

Success
Error
}
Match "1" *-- "1..5" Set
Set "1" *-- "0..*" Rally
Rally "1" *-- "0..*" Action
Team "1" o-- "0..*" Player
Match "1" -- "2" Team
@enduml

```



Промпт:

Сгенерируй PlantUML код для диаграммы классов системы волейбольной статистики и видеоразметки. Укажи: наследование, композиция/агрегация/ассоциация, кратности, атрибуты с типами данных, enum (PlayerPosition, ActionType, ActionResult, ReportType, Side, CoachRole).

Ответ (PlantUML):

```

@startuml
skinparam classAttributeIconSize 0
hide empty members
abstract class User {
    +id: UUID
    +email: String
    +passwordHash: String
    +fullName: String
}
class Player {
    +jerseyNumber: int
    +position: PlayerPosition
    +dateOfBirth: Date
}
class Coach {
    +role: CoachRole
}
class Manager {

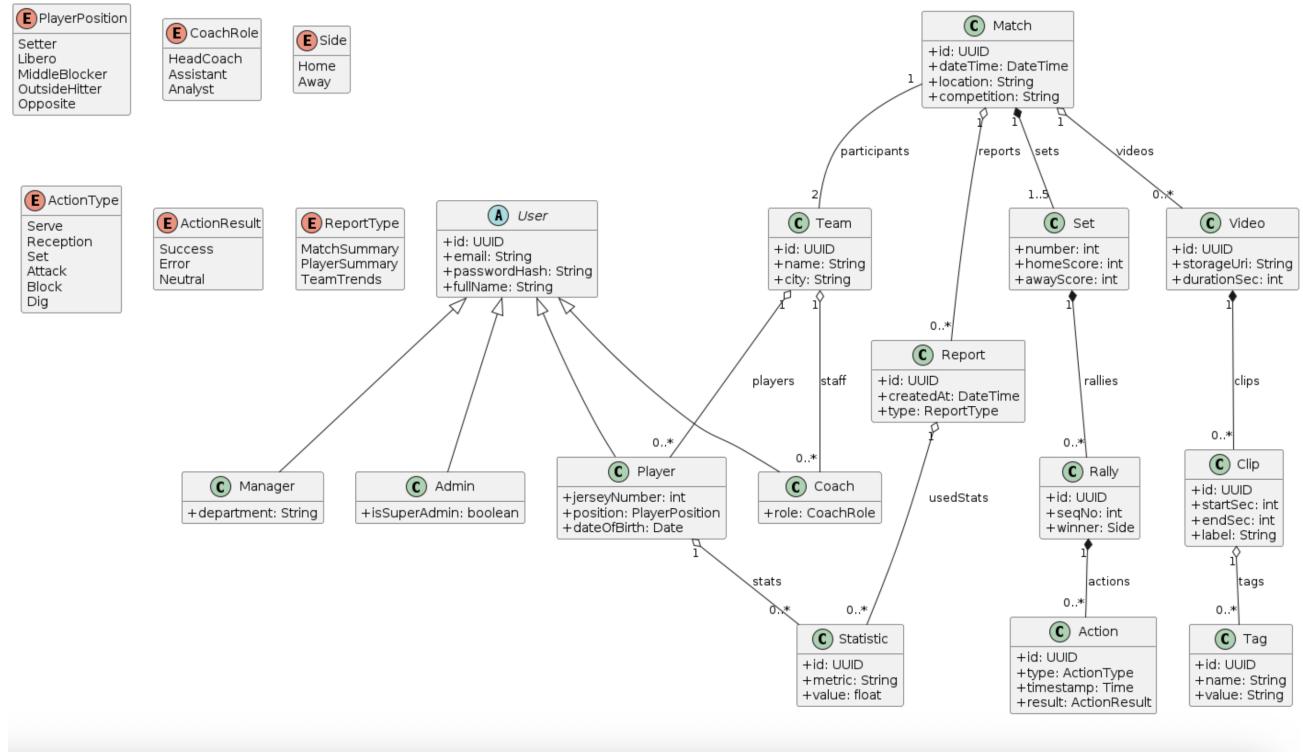
```

```
    +department: String
}
class Admin {
    +isSuperAdmin: boolean
}
User <|-- Player
User <|-- Coach
User <|-- Manager
User <|-- Admin
class Team {
    +id: UUID
    +name: String
    +city: String
}
class Match {
    +id: UUID
    +dateTime: DateTime
    +location: String
    +competition: String
}
class Set {
    +number: int
    +homeScore: int
    +awayScore: int
}
class Rally {
    +id: UUID
    +seqNo: int
    +winner: Side
}
class Action {
    +id: UUID
    +type: ActionType
    +timestamp: Time
    +result: ActionResult
}
class Statistic {
    +id: UUID
    +metric: String
    +value: float
}
class Video {
    +id: UUID
    +storageUri: String
    +durationSec: int
}
class Clip {
    +id: UUID
    +startSec: int
    +endSec: int
    +label: String
}
class Tag {
    +id: UUID
    +name: String
    +value: String
}
class Report {
    +id: UUID
```

```

+createdAt: DateTime
+type: ReportType
}
enum PlayerPosition {
    Setter
    Libero
    MiddleBlocker
    OutsideHitter
    Opposite
}
enum CoachRole {
    HeadCoach
    Assistant
    Analyst
}
enum Side {
    Home
    Away
}
enum ActionType {
    Serve
    Reception
    Set
    Attack
    Block
    Dig
}
enum ActionResult {
    Success
    Error
    Neutral
}
enum ReportType {
    MatchSummary
    PlayerSummary
    TeamTrends
}
Team "1" o-- "0..*" Player : players
Team "1" o-- "0..*" Coach : staff
Match "1" -- "2" Team : participants
Match "1" *-- "1..5" Set : sets
Set "1" *-- "0..*" Rally : rallies
Rally "1" *-- "0..*" Action : actions
Player "1" o-- "0..*" Statistic : stats
Match "1" o-- "0..*" Video : videos
Video "1" *-- "0..*" Clip : clips
Clip "1" o-- "0..*" Tag : tags
Match "1" o-- "0..*" Report : reports
Report "1" o-- "0..*" Statistic : usedStats
@enduml

```



4.2. Результаты: DeepSeek

4.2.1. Диаграмма вариантов использования (Use Case)

Промпт А1 (минимальный)

Промпт:

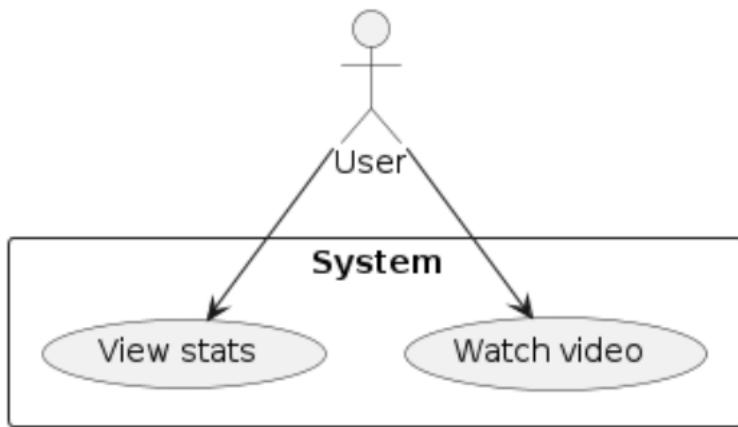
Сгенерируй PlantUML код для диаграммы вариантов использования системы волейбольной статистики и видеоразметки. Укажи действующие лица, перечисли основные случаи использования.

Ответ (PlantUML):

```

@startuml
actor User
rectangle "System" {
    usecase "View stats" as U1
    usecase "Watch video" as U2
}
User --> U1
User --> U2
@enduml

```



Промпт А2 (базовый)

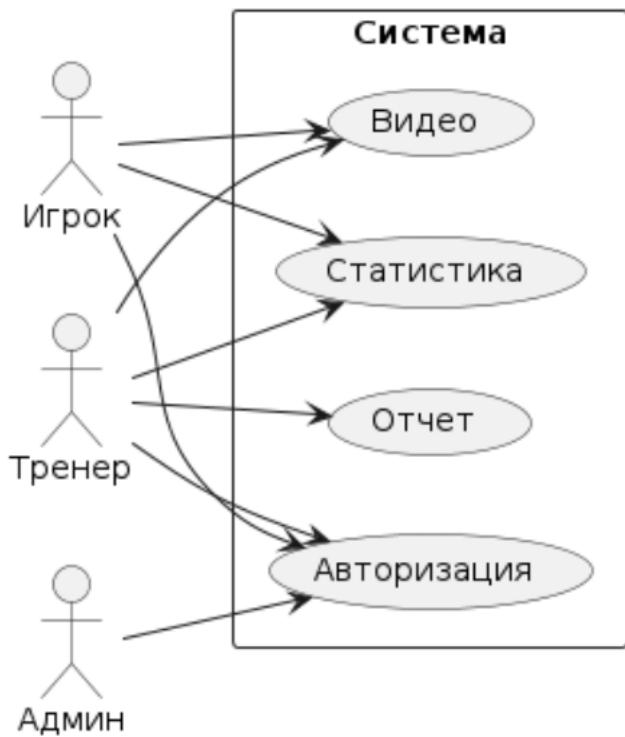
Промпт:

Сгенерируй PlantUML код для диаграммы вариантов использования системы волейбольной статистики и видеоразметки. Укажи акторов: Игрок, Тренер/Аналитик, Менеджер команды, Администратор системы. Перечисли основные случаи использования и добавь границу системы.

Ответ (PlantUML):

```

@startuml
left to right direction
actor "Игрок" as P
actor "Тренер" as C
actor "Админ" as A
rectangle "Система" {
    (Авторизация) as L
    (Статистика) as S
    (Видео) as V
    (Отчет) as R
}
P --> L
P --> S
P --> V
C --> L
C --> S
C --> V
C --> R
A --> L
@enduml
    
```



Промпт А3 (расширенный)

Промпт:

Сгенерируй PlantUML код для диаграммы вариантов использования системы волейбольной статистики и видеоразметки. Перечисли use case по ролям и добавь include/extend там, где это логично. Добавь границу системы.

Ответ (PlantUML):

```

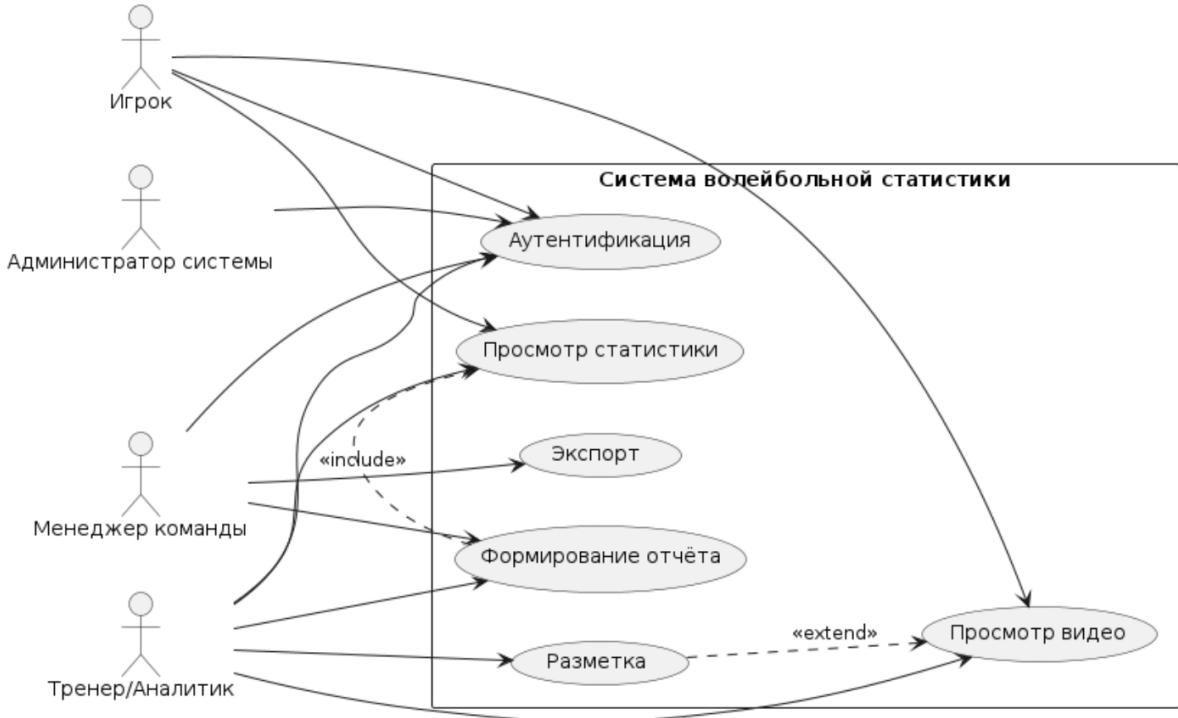
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Игрок" as Player
actor "Тренер/Аналитик" as Coach
actor "Менеджер команды" as Manager
actor "Администратор системы" as Admin
rectangle "Система волейбольной статистики" {
    usecase "Аутентификация" as UC_Login
    usecase "Просмотр статистики" as UC_ViewStats
    usecase "Формирование отчёта" as UC_Report
    usecase "Экспорт" as UC_Export
    usecase "Просмотр видео" as UC_ViewVideo
    usecase "Разметка" as UC_Tag
}
UC_Report .> UC_ViewStats : <<include>>
UC_Tag ..> UC_ViewVideo : <<extend>>
Player --> UC_Login
Player --> UC_ViewStats
Player --> UC_ViewVideo
Coach --> UC_Login

```

```

Coach --> UC_ViewStats
Coach --> UC_ViewVideo
Coach --> UC_Tag
Coach --> UC_Report
Manager --> UC_Login
Manager --> UC_Report
Manager --> UC_Export
Admin --> UC_Login
@enduml

```



Промпт А4 (продвинутый)

Промпт:

Сгенерируй PlantUML код для диаграммы вариантов использования системы волейбольной статистики и видеоразметки. Акторы: Игрок, Тренер/Аналитик, Менеджер команды, Администратор системы, Внешнее видеохранилище. Обязательно: граница системы, include/extend, логическая целостность.

Ответ (PlantUML):

```

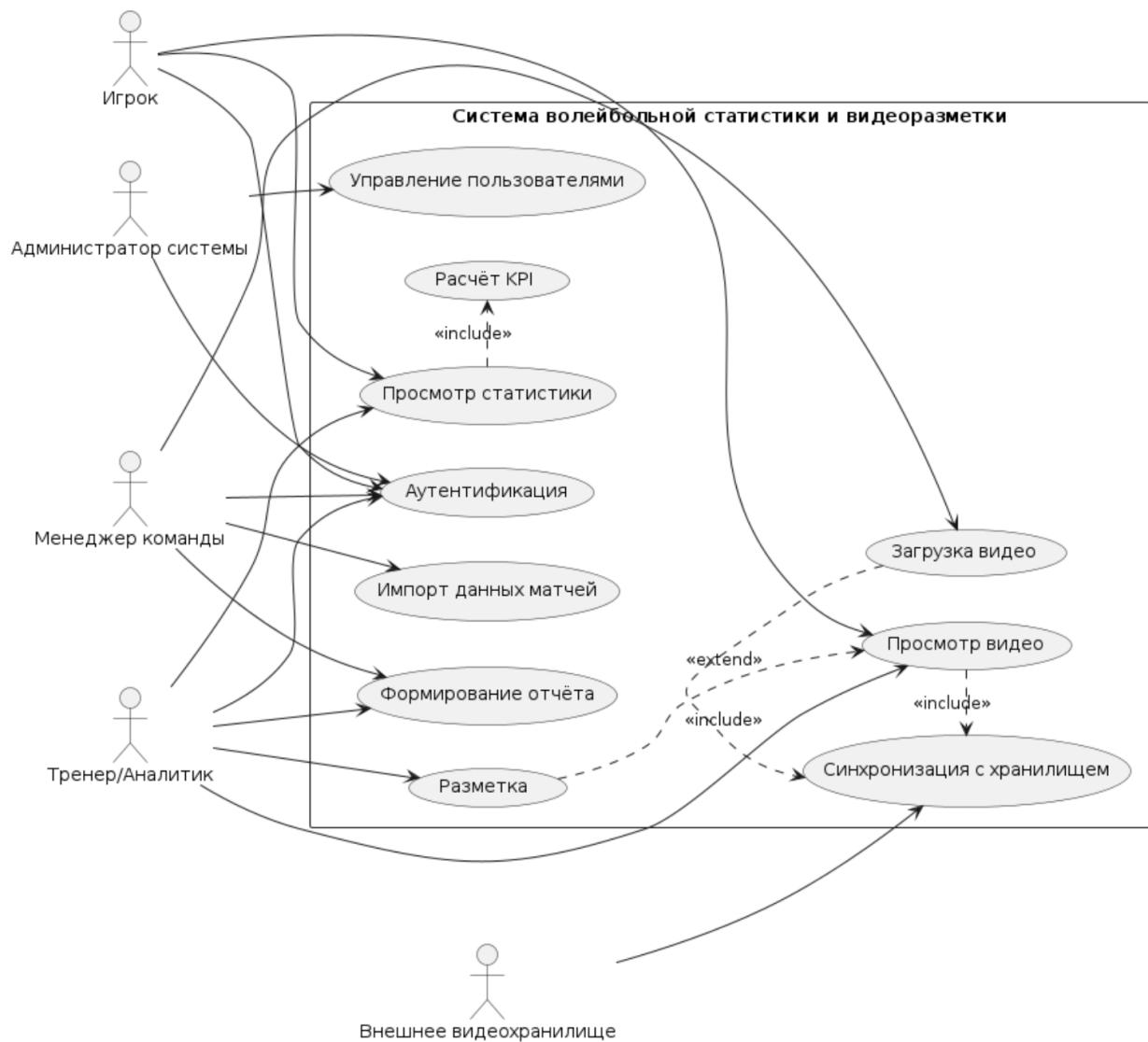
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Игрок" as Player
actor "Тренер/Аналитик" as Coach
actor "Менеджер команды" as Manager
actor "Администратор системы" as Admin
actor "Внешнее видеохранилище" as Storage
rectangle "Система волейбольной статистики и видеоразметки" {
    usecase "Аутентификация" as UC_Login
    usecase "Просмотр статистики" as UC_ViewStats
    usecase "Расчёт KPI" as UC_KPI
}

```

```

usecase "Просмотр видео" as UC_Video
usecase "Синхронизация с хранилищем" as UC_Sync
usecase "Разметка" as UC_Tag
usecase "Импорт данных матчей" as UC_Import
usecase "Загрузка видео" as UC_Upload
usecase "Формирование отчёта" as UC_Report
usecase "Управление пользователями" as UC_Users
}
UC_ViewStats .> UC_KPI : <<include>>
UC_Video .> UC_Sync : <<include>>
UC_Upload .> UC_Sync : <<include>>
UC_Tag ..> UC_Video : <<extend>>
Player --> UC_Login
Player --> UC_ViewStats
Player --> UC_Video
Coach --> UC_Login
Coach --> UC_ViewStats
Coach --> UC_Video
Coach --> UC_Tag
Coach --> UC_Report
Manager --> UC_Login
Manager --> UC_Import
Manager --> UC_Upload
Manager --> UC_Report
Admin --> UC_Login
Admin --> UC_Users
Storage --> UC_Sync
@enduml

```



4.2.2. Диаграмма классов (Class)

Промпт B1 (минимальный)

Промпт:

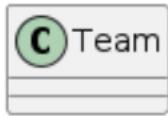
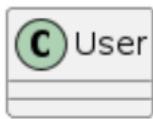
Сгенерируй PlantUML код для диаграммы классов системы волейбольной статистики и видеоразметки. Укажи основные классы и связи.

Ответ (PlantUML):

```

@startuml
class User
class Match
class Team
@enduml

```



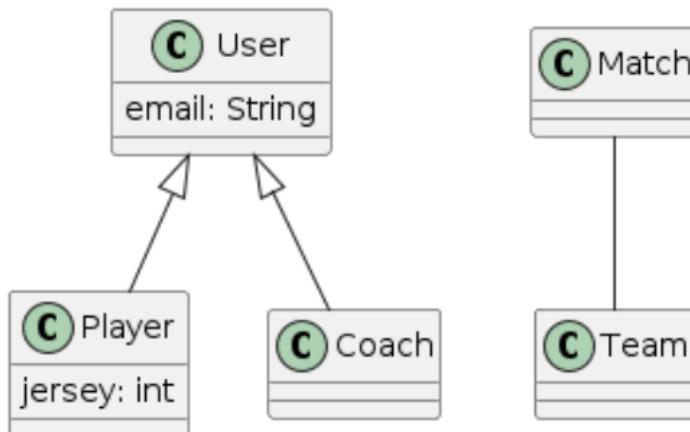
Промпт B2 (базовый)

Промпт:

Сгенерируй PlantUML код для диаграммы классов системы волейбольной статистики и видеоразметки. Классы: User, Player, Coach, Match, Team, Video, Report. Укажи атрибуты с типами и основные связи.

Ответ (PlantUML):

```
@startuml
class User {
    email: String
}
class Player {
    jersey: int
}
class Coach {
}
User <|-- Player
User <|-- Coach
class Match
class Team
Match -- Team
@enduml
```



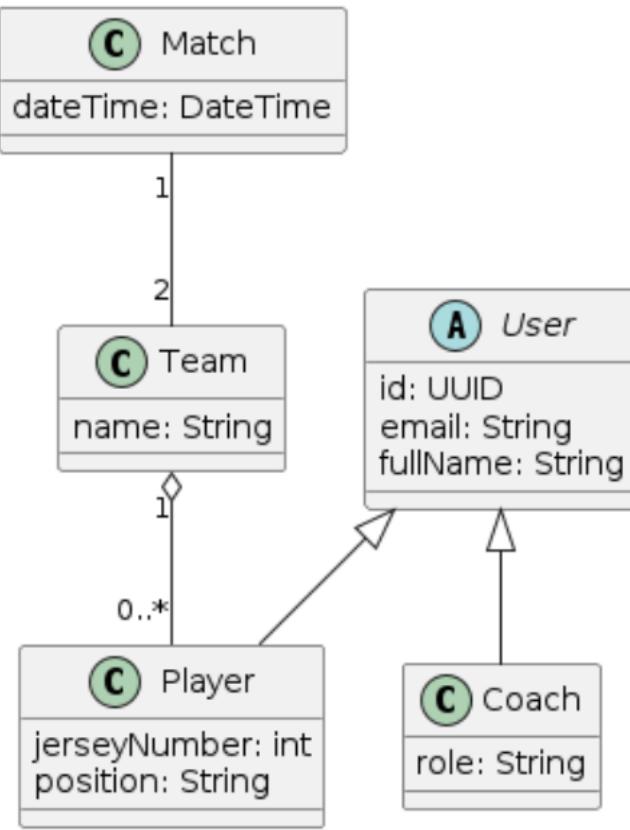
Промпт В3 (расширенный)

Промпт:

Сгенерируй PlantUML код для диаграммы классов системы волейбольной статистики и видеоразметки. Добавь наследование User->Player/Coach/Manager/Admin, композиции Match->Set->Rally->Action, кратности ассоциаций и enumeration для статусов/типов.

Ответ (PlantUML):

```
@startuml
skinparam classAttributeIconSize 0
abstract class User {
    id: UUID
    email: String
    fullName: String
}
class Player {
    jerseyNumber: int
    position: String
}
class Coach {
    role: String
}
class Team {
    name: String
}
class Match {
    dateDateTime: DateTime
}
User <|-- Player
User <|-- Coach
Team "1" o-- "0..*" Player
Match "1" -- "2" Team
@enduml
```



Промпт В4 (продвинутый)

Промпт:

Сгенерируй PlantUML код для диаграммы классов системы волейбольной статистики и видеоразметки. Укажи: наследование, композиция/агрегация/ассоциация, кратности, атрибуты с типами данных, enum (PlayerPosition, ActionType, ActionResult, ReportType, Side, CoachRole).

Ответ (PlantUML):

```

@startuml
skinparam classAttributeIconSize 0
hide empty members
abstract class User {
    +id: UUID
    +email: String
    +passwordHash: String
    +fullName: String
}
class Player {
    +jerseyNumber: int
    +position: PlayerPosition
    +dateOfBirth: Date
}
class Coach {
    +role: CoachRole
}
class Manager {
    +department: String
}
class Admin {
    +isSuperAdmin: boolean
}
User <|-- Player
User <|-- Coach
User <|-- Manager
User <|-- Admin
class Team {
    +id: UUID
    +name: String
    +city: String
}
class Match {
    +id: UUID
    +dateTime: DateTime
    +location: String
}
class Set {
    +number: int
}
class Rally {
    +seqNo: int
}
class Action {
    +type: ActionType
    +result: ActionResult
}
class Video {
    +storageUri: String
}
class Report {
    +createdAt: DateTime
}
enum PlayerPosition {
    Setter
    Libero
}
enum ActionType {
    Serve
}

```

```

        Attack
    }
enum ActionResult {
    Success
    Error
}
Team "1" o-- "0..*" Player
Match "1" -- "2" Team
Match "1" *-- "1..5" Set
Set "1" *-- "0..*" Rally
Rally "1" *-- "0..*" Action
Match "1" o-- "0..*" Video
Match "1" o-- "0..*" Report
@enduml

```

5. Итоговые диаграммы и комментарии

В качестве итогового варианта выбраны результаты продвинутых промптов (A4 и B4) для модели GPT-5.2 Thinking, так как они наиболее полно отражают требования задания (граница системы, include/extend, наследование, композиций, кратности).

5.1. Итоговая диаграмма вариантов использования

Исходный код PlantUML (выбранный вариант):

```

@startuml
left to right direction
skinparam packageStyle rectangle
actor "Игрок" as Player
actor "Тренер/Аналитик" as Coach
actor "Менеджер команды" as Manager
actor "Администратор системы" as Admin
actor "Внешнее видеохранилище" as Storage
rectangle "Система волейбольной статистики и видеоразметки" {
    usecase "Аутентификация" as UC_Login
    usecase "Просмотр статистики" as UC_ViewStats
    usecase "Расчёт KPI" as UC_ComputeKPI
    usecase "Предобработка данных" as UC_Preprocess
    usecase "Поиск аномалий\пв статистике" as UC_Anomaly
    usecase "Просмотр видео моментов" as UC_ViewVideo
    usecase "Синхронизация\пс видеохранилищем" as UC_SyncStorage
    usecase "Разметка розыгрышей" as UC_TagRally
    usecase "Загрузка видео матча" as UC_UploadVideo
    usecase "Импорт данных матчей" as UC_ImportData
    usecase "Формирование отчёта" as UC_Report
    usecase "Экспорт данных" as UC_Export
    usecase "Управление пользователями" as UC_Users
    usecase "Управление ролями" as UC_Roles
}
Player --> UC_Login
Player --> UC_ViewStats
Player --> UC_ViewVideo
Coach --> UC_Login

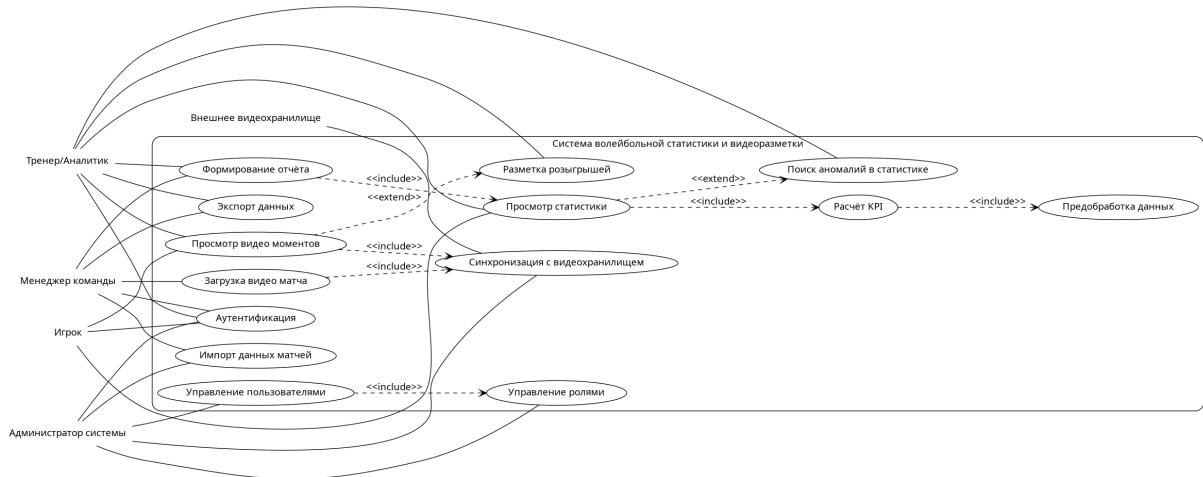
```

```

Coach --> UC_ViewStats
Coach --> UC_ViewVideo
Coach --> UC_TagRally
Coach --> UC_Report
Coach --> UC_Export
Coach --> UC_Anomaly
Manager --> UC_Login
Manager --> UC_ImportData
Manager --> UC_UploadVideo
Manager --> UC_Report
Manager --> UC_Export
Admin --> UC_Login
Admin --> UC_Users
Admin --> UC_Roles
Admin --> UC_ImportData
Admin --> UC_SyncStorage
Storage --> UC_SyncStorage
UC_ViewStats .> UC_ComputeKPI : <<include>>
UC_ComputeKPI .> UC_Preprocess : <<include>>
UC_Report .> UC_ViewStats : <<include>>
UC_ViewVideo .> UC_SyncStorage : <<include>>
UC_UploadVideo .> UC_SyncStorage : <<include>>
UC_Users .> UC_Roles : <<include>>
UC_Anomaly ..> UC_ViewStats : <<extend>>
UC_TagRally ..> UC_ViewVideo : <<extend>>
@enduml

```

Сгенерированная диаграмма:



Комментарий к ключевым элементам (Use Case)

- UC_ViewStats включает UC_ComputeKPI, а UC_ComputeKPI включает UC_Preprocess — расчёты KPI предполагают обязательную предобработку данных.

- UC_ViewVideo и UC_UploadVideo включают UC_SyncStorage — просмотр/загрузка работают через обязательную синхронизацию с хранилищем.
- UC_Anomaly расширяет UC_ViewStats (extend) — поиск аномалий выполняется как дополнительный (опциональный) анализ статистики.
- UC_TagRally расширяет UC_ViewVideo (extend) — разметка является опциональным действием в рамках просмотра видео.

5.2. Итоговая диаграмма классов

Исходный код PlantUML (выбранный вариант):

```

@startuml
skinparam classAttributeIconSize 0
hide empty members
abstract class User {
    +id: UUID
    +email: String
    +passwordHash: String
    +fullName: String
}
class Player {
    +jerseyNumber: int
    +position: PlayerPosition
    +dateOfBirth: Date
}
class Coach { +role: CoachRole }
class Manager { +department: String }
class Admin { +isSuperAdmin: boolean }
User <|-- Player
User <|-- Coach
User <|-- Manager
User <|-- Admin
class Team { +id: UUID +name: String +city: String }
class Match { +id: UUID +dateTime: DateTime +location: String +competition: String }
class Set { +number: int +homeScore: int +awayScore: int }
class Rally { +id: UUID +seqNo: int +winner: Side }
class Action { +id: UUID +type: ActionType +timestamp: Time +result: ActionResult }
class Statistic { +id: UUID +metric: String +value: float }
class Video { +id: UUID +storageUri: String +durationSec: int }
class Clip { +id: UUID +startSec: int +endSec: int +label: String }
class Tag { +id: UUID +name: String +value: String }
class Report { +id: UUID +createdAt: DateTime +type: ReportType }
enum PlayerPosition { Setter Libero MiddleBlocker OutsideHitter Opposite }
enum CoachRole { HeadCoach Assistant Analyst }
enum Side { Home Away }
enum ActionType { Serve Reception Set Attack Block Dig }

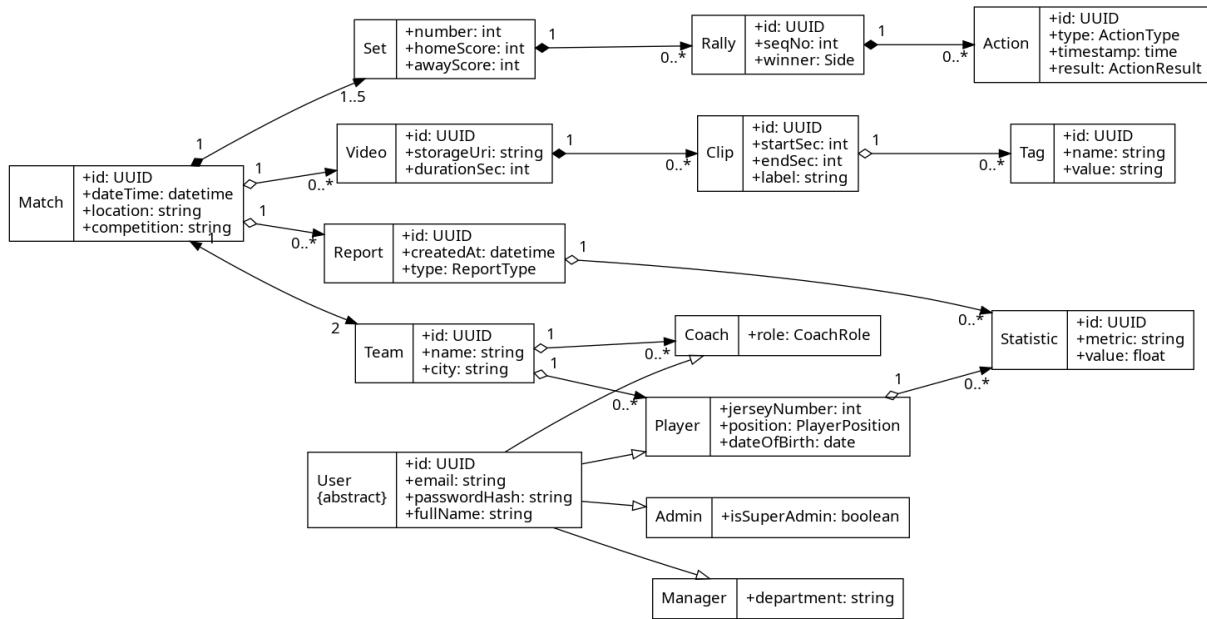
```

```

enum ActionResult { Success Error Neutral }
enum ReportType { MatchSummary PlayerSummary TeamTrends }
Team "1" o-- "0..*" Player : players
Team "1" o-- "0..*" Coach : staff
Match "1" -- "2" Team : participants
Match "1" *-- "1..5" Set : sets
Set "1" *-- "0..*" Rally : rallies
Rally "1" *-- "0..*" Action : actions
Player "1" o-- "0..*" Statistic : stats
Match "1" o-- "0..*" Video : videos
Video "1" *-- "0..*" Clip : clips
Clip "1" o-- "0..*" Tag : tags
Match "1" o-- "0..*" Report : reports
Report "1" o-- "0..*" Statistic : usedStats
@enduml

```

Сгенерированная диаграмма:



Комментарий к ключевым элементам (Class)

- Наследование User → Player/Coach/Manager/Admin позволяет избежать дублирования общих атрибутов (id, email и т.п.).
- Композиция Match → Set → Rally → Action отражает «жёсткую» структуру матча: без матча нет сетов, без сетов нет розыгрышей, и т.д.
- Агрегации (например Team o-- Player) показывают принадлежность объектов без жёсткого владения жизненным циклом.
- Перечисления (enum) фиксируют допустимые значения для ролей, позиций и типов действий, что повышает строгость модели.

6. Сравнительный анализ

Сравнение выполнялось по критериям: полнота требований, соответствие UML, читаемость, структурность PlantUML-кода и устойчивость к неоднозначностям (насколько хорошо модель добавляет `include/extend` и кратности без явного указания).

Критерий	GPT-5.2 Thinking	DeepSeek	Комментарий	Итог
Полнота Use Case	Высокая (A3–A4)	Средняя (A3–A4)	DeepSeek чаще упрощает состав акторов и use case на ранних промптах.	Лучше GPT
Корректность <code>include/extend</code>	Корректно и последовательно	Корректно, но менее полно	У DeepSeek меньше вспомогательных сценариев.	Лучше GPT
Полнота Class	Высокая (B3–B4)	Средняя/высокая (B4)	DeepSeek добавляет меньше атрибутов и связей по умолчанию.	Лучше GPT
Наследование и кратности	Есть и читаемо	Есть, но менее детально	Обе модели справляются, GPT точнее структурирует.	Лучше GPT
Качество итоговой визуализации	Высокое	Среднее	Итоговый вариант выбран из GPT-5.2.	Лучше GPT

7. Выводы

- Выбранная предметная область позволяет естественно продемонстрировать include/extend (обязательные этапы расчётов и optionalные действия разметки/поиска аномалий) и различные типы связей в диаграмме классов.
- Наиболее качественные результаты получены при использовании продвинутых промптов (A4 и B4), где явно заданы акторы, требования к include/extend, наследование и кратности.
- В качестве рекомендуемого решения выбран вариант от GPT-5.2 Thinking (A4, B4), так как он обеспечивает максимальную логическую целостность и полноту модели.