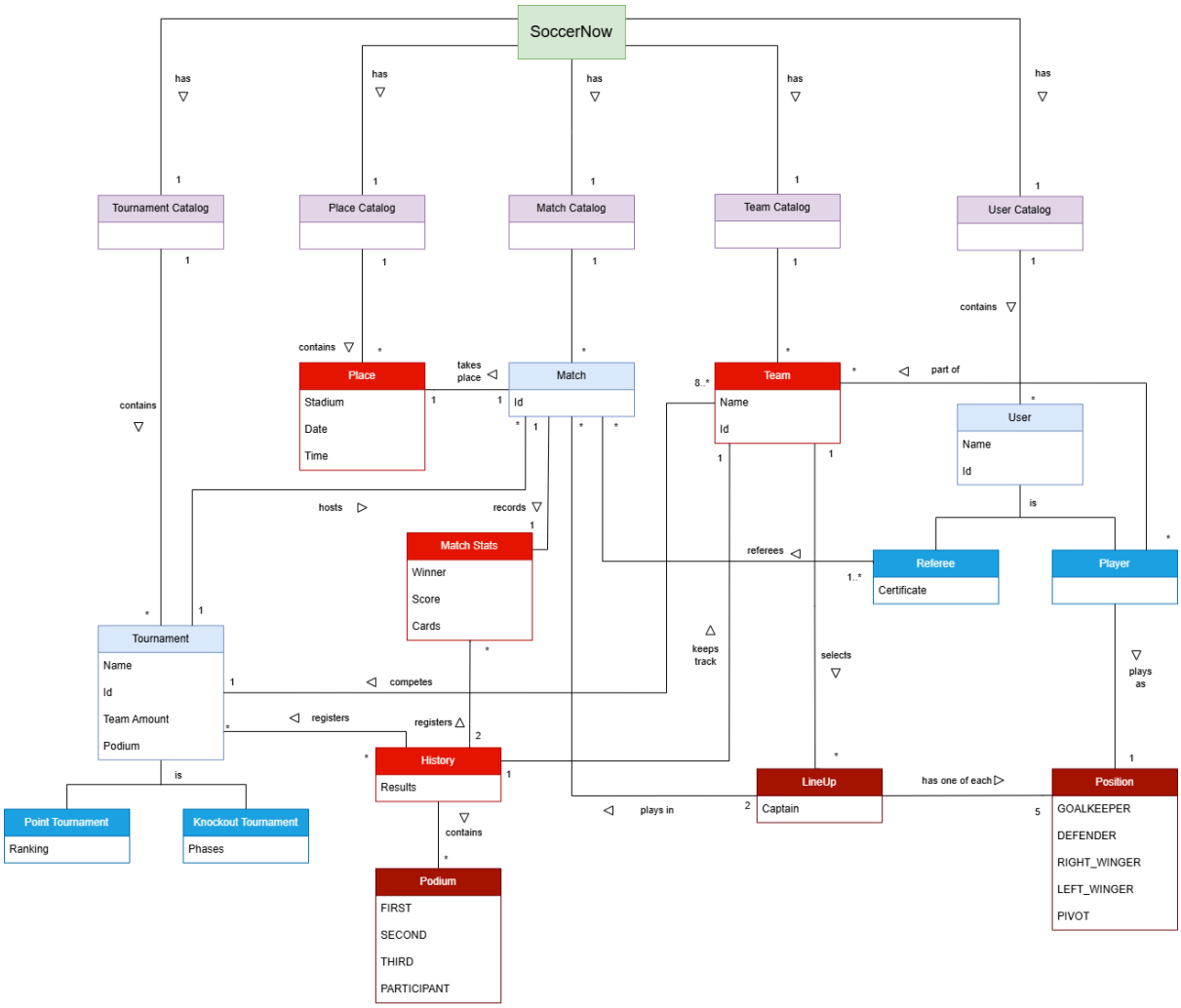
Relatório CSS

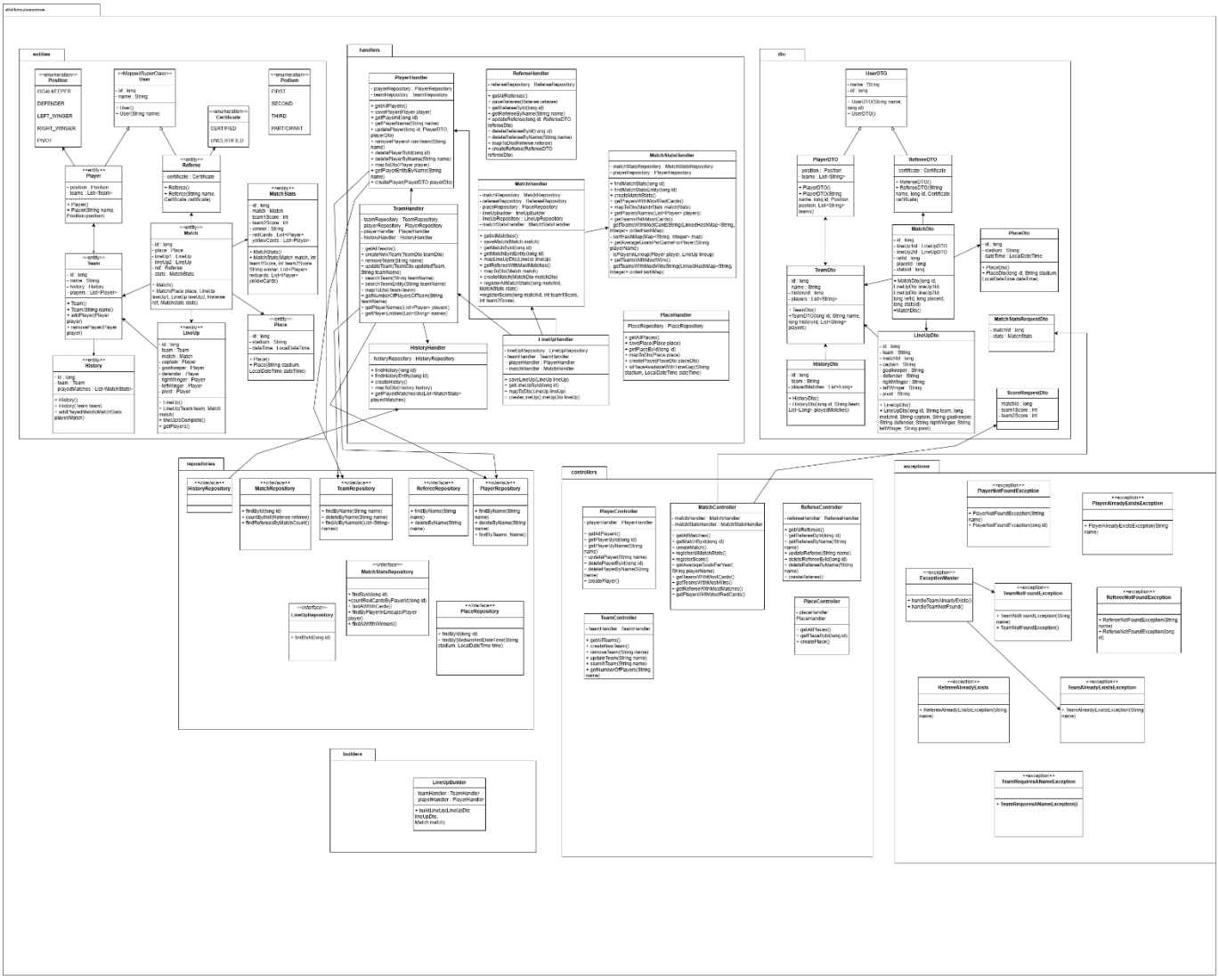
Arquitetura em camadas:

A camada mais abaixo é a camada de dados. Esta está contida numa base de dados PostgresSQL. A camada acima é a camada de acesso aos dados, que é feita através dos repositórios. A próxima camada é a camada de negócio. A gestão desta é feita através dos *handlers*. A camada a seguir é a camada de apresentação. Aqui residem os *controllers* que realizam chamadas aos *handlers*. A última camada é a camada da interface gráfica, que no nosso sistema é o Swagger.

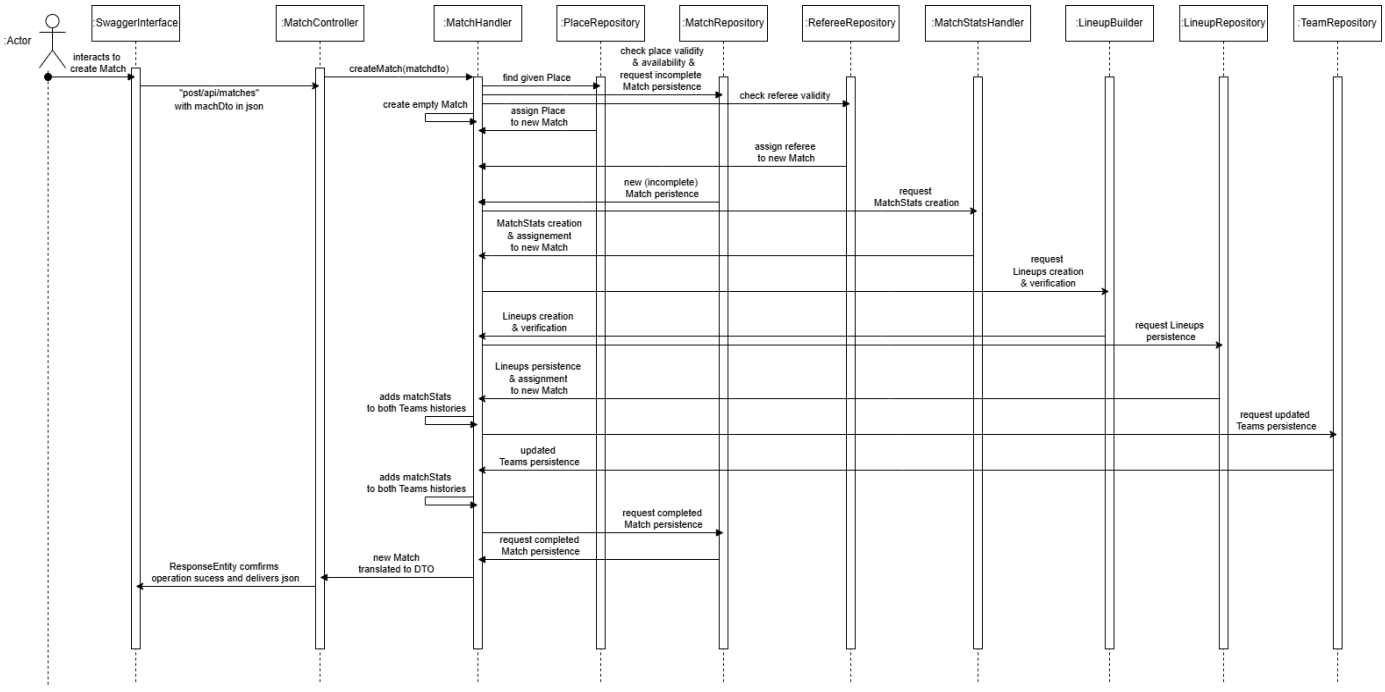
Modelo de domínio



## Diagrama de classes



## SSD do Caso de Uso H



As entidades que implementamos:

- *Team* (representa a equipa)
- *User* (mapped super class, classe mãe dos jogadores e dos árbitros)
- *Referee* (representa um árbitro)
- *Player* (representa um jogador)
- *Match* (representa um jogo)
- *Place* (representa a marcação de um jogo, contendo o local, data e hora)
- *History* (representa o histórico de uma equipa, com as partidas e os torneios disputados)
- *LineUp* (representa o plantel que joga num jogo)
- *MatchStats* (representa as estatísticas de um jogo)
- *Certificate* (enumerado que representa o certificado de árbitro)
- *Position* (enumerado que representa a posição de um jogador)
- *Podium* (enumerado que representa a classificação de uma equipa num torneio)

Mapeamento entre as entidades

- Relação *Player* → *Team*
  - Como uma equipa contém vários jogadores e um jogador pode estar associado a várias equipas, decidimos mapear esta relação como uma @*ManyToMany*. É bidirecional.
- Relação Team → *History*
  - Como uma equipa contém apenas um histórico e um histórico pertence a apenas uma equipa, mapeamos esta relação como uma @*OneToOne* bidirecional.
- Relação Match → *Place*
  - O *Place* representa o local, a data e a hora de um jogo. Sendo assim, só pode haver um único jogo num *Place*. Assim, esta relação é uma @*OneToOne* unidirecional.
- Relação Match → *LineUp*
  - O mesmo *lineUp* pode ser usado em diversos jogos, enquanto que um jogo tem apenas um *lineUp* de cada equipa. Logo esta relação é uma @*ManyToOne* bidirecional.
- Relação *Match* → *Referee*
  - O mesmo árbitro participa em vários jogos, mas cada jogo tem apenas um árbitro. Assim a relação é @*ManyToOne* e é unidirecional.
- Relação *Match* → *MatchStats*
  - Só faz sentido um jogo ter uma entidade de estatísticas, e o *MatchStats* pertence a apenas um jogo. Logo, @*OneToOne* bidirecional.
- Relação *MatchStats* → *Player*
  - Um *MatchStats* guarda as estatísticas de cada jogador participante, como cartões e número de golos, e, como cada jogador joga em vários jogos e pode aparecer em inúmeros *MatchStats*, esta relação é @*ManyToMany* unidirecional.
- Relação *LineUp* → *Player*
  - Um *LineUp* tem apenas um único jogador de cada posição, enquanto que cada jogador pode fazer parte de vários *LineUps*. Assim, esta relação é uma @*ManyToOne* unidirecional.
- Relação *LineUp* → *Team*

  - Dentro de uma equipa podem existir vários *lineUps*, mas estes pertencem sempre à mesma equipa. Assim, a relação é @*ManyToOne* unidirecional.
- Relação *History* → *MatchStats*
  - Um histórico de uma equipa guarda várias estatísticas de diversos jogos, sendo que estas estarão contidas nos históricos de duas equipas. Assim, a relação é @*ManyToMany* unidirecional.
- Relação *Player* → *User*
  - Um jogador é uma especificação de utilizador, ou seja, um jogador herda os atributos e métodos de um utilizador. A notação @*MappedSuperClass* presente no utilizador indica que *User* não irá ser persistido numa tabela, sendo apenas uma classe com o propósito de ser herdada por outras entidades.
- Relação *Referee* → *User*
  - Relação semelhante à de *Player* → *User*.

Garantias da lógica de negócio

- Ao eliminarmos uma equipa, o sistema garante que os jogadores associados a esta não serão igualmente eliminados, apenas desassociados desta.
- O sistema não permite que se altere o histórico de uma equipa. O mesmo histórico é sempre associado à mesma equipa, sendo este automaticamente criado com a equipa.
- Quando a equipa é eliminada, o seu histórico continua a existir e poderá ser consultado.
- Uma equipa não poderá ser eliminada se ainda tiver jogos por disputar.
- Não são permitidas equipas com o mesmo nome, ou equipas sem nome.
- Não podem existir jogadores e árbitros com o mesmo nome.
- Não são permitidos dois *Places* cujos tempos têm uma diferença baixo de 2 horas e 30 minutos.
- Um Match não poderá ser concretizado se algum dos *LineUps* estiver incompleto, ou se não existir um *Place* ou *Referee* associado.
- O resultado de um jogo não pode ser alterado enquanto este não se iniciar. O resultado também não pode ser negativo.
- Quando se cria um *Match*, o seu respetivo *MatchStats* é criado automaticamente. Quando o Match é eliminado, o *MatchStats* ainda existe e poderá ser consultado no futuro.

Decisões de lógica de negócio

- Os nomes de jogadores são únicos.
- Os nomes dos árbitros são únicos.
- Os nomes das equipas são únicos.
- Podem existir várias instâncias do mesmo sítio em *Place*, desde que sejam horários diferentes.
- *LineUp* contém cada posição possível, porém um jogador pode ser um *GOALKEEPER*, mas ir a jogo como, por exemplo, *PIVOT*.

- Jogadores guardam as equipas a que pertencem.
- *History* apenas existe quando há uma equipa correspondente.
- *MatchStats* apenas existe quando há uma partida correspondente.
- Quando um *MatchStats* é criado, este não contém nenhuma informação.
- As estatísticas de cada jogo são registadas através de *endpoints* bem definidos.
- O resultado de um jogo é determinado automaticamente através dos golos registados.
- Criámos exceções personalizadas (*TeamNotFoundException*, *PlayerNotFoundException, RefereeAlreadyExists,* etc.).

Limitações do Trabalho

- Diagrama de classes não contém todas as relações entre as classes para efeitos de clareza. Estão representadas relações entre *packages* e algumas relações entre *packages* para efeitos demonstrativos.
- Alguns métodos não estão representados no diagrama de classes.
- Algumas exceções em alguns *endpoints* não são completamente tratadas.

Notas:

- Acrescentámos uma pasta adicional designada por '*components*' com o modelo de domínio, diagrama de classes e SSD do caso de uso 'H', tanto em formato *PDF* como *PNG*.