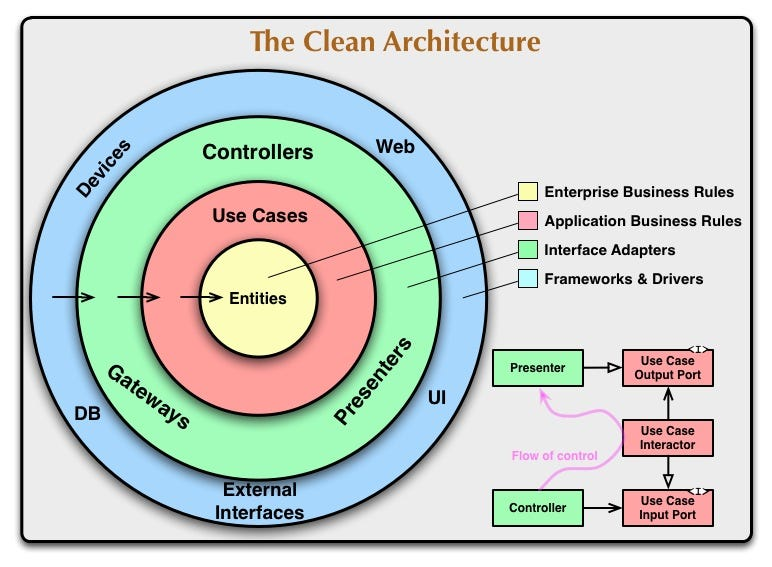
O sistema utiliza-se da arquitetura limpa que é focada na separação de preocupações e independência entre UI, regras de negócio e infraestrutura, exemplificado pela imagem abaixo.



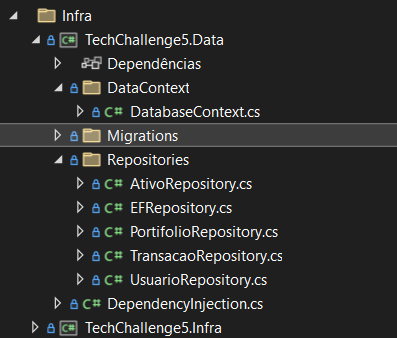
Na implementação do nosso sistema, a arquitetura se dá da seguinte forma:

**Frameworks & Drivers**

Nessa camada nós temos as classes responsáveis pela manipulação do banco de dados, como o DbContext do Entity Framework e as classes Repository.

Dessa forma a comunicação com o repositório de dados fica independente, fazendo com que as outras camadas não sejam dependentes de um local específico para o armazenamento de dados, assim como não se preocupem com a forma que é feita essa comunicação.

Além da comunicação com o banco de dados, o Swagger também entra nessa camada, sendo responsável pela interface que fará a comunicação do usuário com a API de fato.



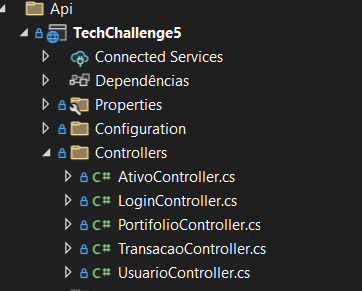
**Interface Adapters**

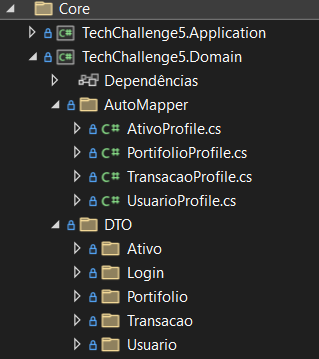
Nessa camada nós encontramos as Controllers, Mappers e DTOs da aplicação, sendo responsáveis pela ligação entre as classes de negócio com o mundo externo.

As Controllers recebem a requisição e definem qual regra de negócio deve ser utilizada para aquele caso, além de responder a informação ao usuário..

Os DTOs são classes que representam os dados fornecidos pelo usuário, e não como ele é usado pela lógica de negócios em si.

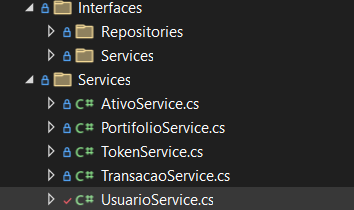
Os Mappers são responsáveis por transformar a classe DTO em uma classe de negócio para que ela seja utilizada em camadas mais internas, assim como transformar resultados internos do sistema para o formato necessário ao usuário.





**Use Cases**

Aqui nós temos nossas classes Service, responsáveis por manipular as entidades e implementar a lógica da aplicação.



**Entities**

Aqui são as entidades núcleo do negócio, com as regras mais fundamentais para a consistência do negócio.

