D S T Q Q S S

4/1/22

Interfaces: Muitos dos métodos dos carros são comuns em vários automóveis. Tanto um carro quanto uma motocicleta são classes cujos objetos podem acelerar, parar, acender o farol etc, pois são ações comuns a automóveis. Podemos dizer então que ambas as classes carro e moto são automóveis.

Quando duas (ou mais) classes possuem comportamentos comuns que podem ser separados em uma outra classe, dizemos que a "classe comum" é a interface, que pode ser "herdada" pelas outras classes, entre aspas, porque uma interface não é exatamente uma classe, mas sim um conj. de métodos que todas as classes que herdarem dela devem possuir (implementar).

Portanto, uma interface não é "herdada" por uma classe, e sim implementada.

Arquiteturas de software

MVC → Model, View, Controller

• Model Camada que possui a lógica da aplicação, responsável pelas regras de negócios, persistência com o BD e as classes de entidades. O model recebe requisições vindas do controller e gera respostas a partir dessas requisições.

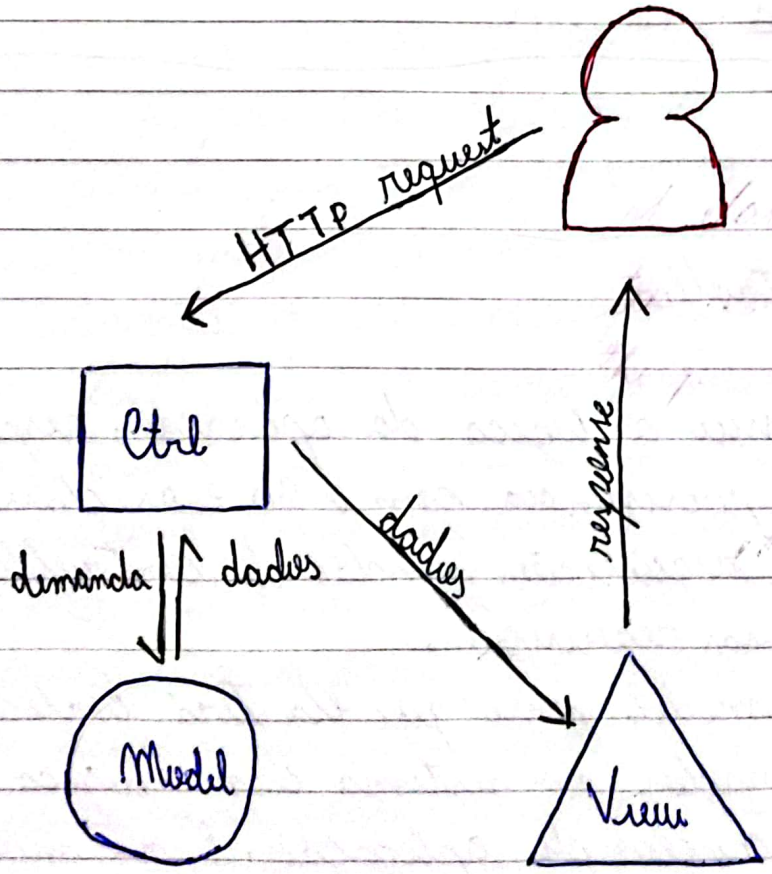
Sempre que pensar na model, pense que ela terá conhecimento apenas dos dados armazenados no sistema e na lógica sobre eles, por se tratar do núcleo da aplicação. É na model também que o crud deve ser realizado.

4/J/22

DSTO

• View é a camada de visualização e representa a parte do sistema que interage com o usuário, e é por ela que haverá a entrada dos dados inseridos pelos usuários e também a saída de informações que serão exibidas para eles. A view não contém lógica de negócios, portanto todo o processamento é feito pela camada model e então repassado para a view pelo controller.

• Controller é a camada que faz a comunicação entre a view e a model, serve como um intermediário que organiza os eventos da interface e os direciona para a camada de modelo, e vice-versa.



S T Q Q S S

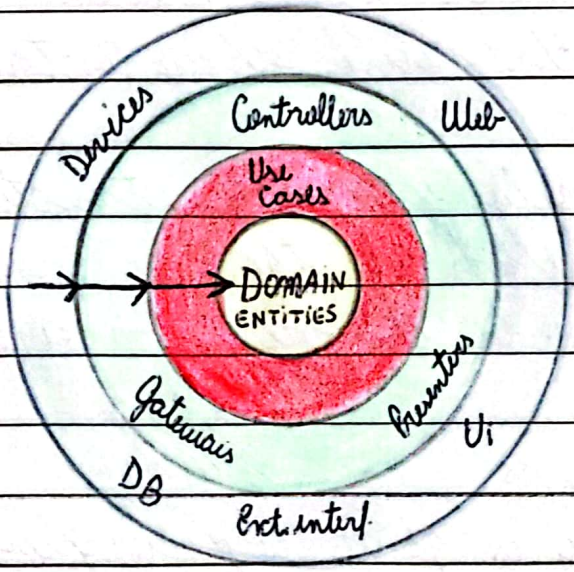
5/1/22

Clean Architecture.

Padrão de arquitetura de software em camadas, tendo como obj:

- Independencia de framework.
- Independencia de U.I
- Independencia de banco de dados
- Independencia de qualquer agente externo.

A clean Architecture divide um software em 4 camadas, que respeitam a regra de "de fora pra dentro", que diz que as camadas mais internas jamais devem depender de camadas mais externas, apenas o contrario.



- Enterprise Bussiness Rules
- Application Bussiness Rules
- Interface Adapters
- Frameworks e drivers

- Domain

Entities encapsulate enterprise wide bussiness rules. An entity can be an object with methods, or it can be a set of data structures and functions. It doesn't matter so long as the entities could be used by many different applications in the enterprise.

5/1/22

D S T Q Q S S

• Use cases

The software in this layer contains application specific business rules. It encapsulates and implements all of the use cases of the system. These use cases orchestrate the flow of data to and from the entities, and direct these entities to use their enterprise wide business rules to achieve the goals of the use case.

• Interface adapters

The software in this layer is a set of adapters that convert data from the format most convenient for the use cases and entities, to the format most convenient for some external agency such as the database or the web.

Is it this the layer that will wholy contain the MVC architecture of a gui.

The models are likely just data structures that are passed from the controllers to the use cases, and then back from the use cases to the presenters and views

• Frameworks and drivers

The outermost layer is generally composed of frameworks and tools such as the database, the web framework etc.

Generally you dont write so much code on this layer.