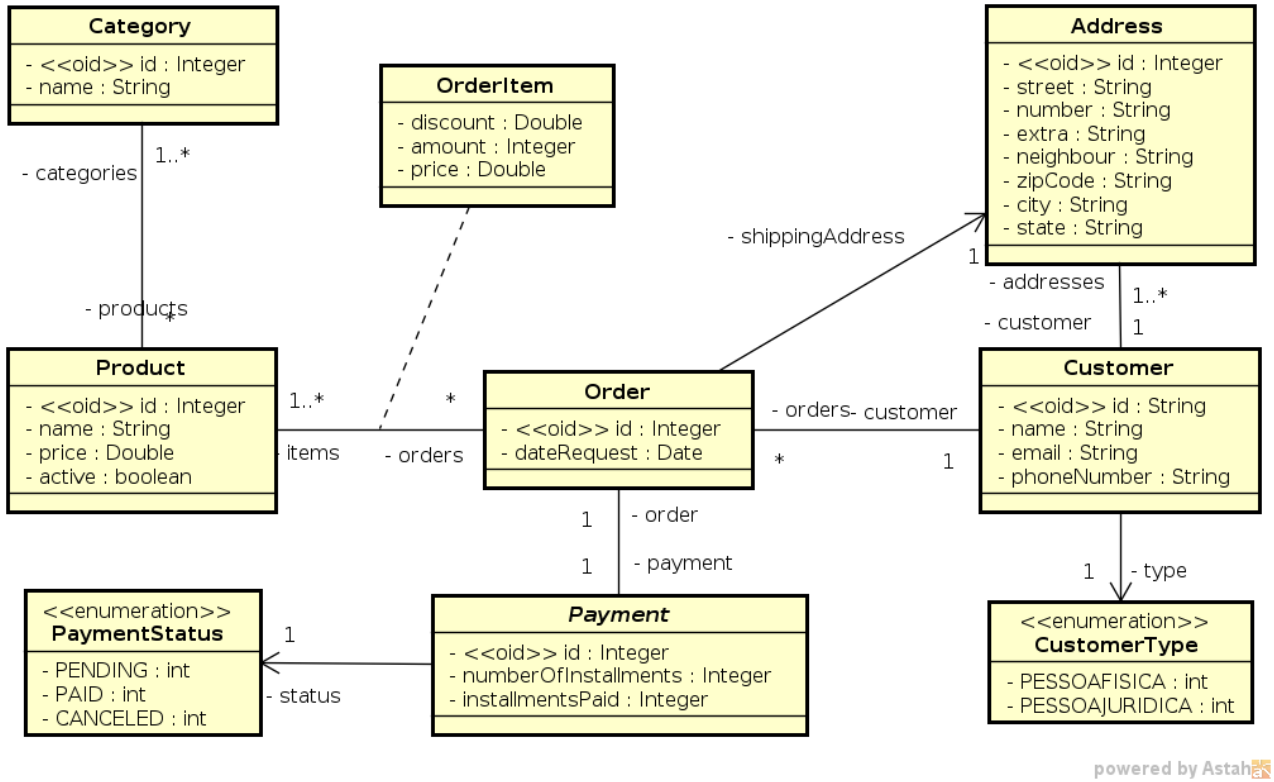
Aula 11 - Exercícios

Analisar o modelo de domínio do projeto



Analisar o diagrama de classes
Criar as classes Entity de acordo com o modelo.

Aula 12 - Exercícios

1) Implementando endpoins de Category

Criar: Entity, DTO, Repository, Service e Controller.
Especificações do endpoint:
Resource URL: /categories

Class method	Request Body	Path Variable	HTTP Method	Path mapping	Response Status
findAll	-	-	GET	-	HttpStatus.OK
findById	-	Integer	GET	/{id}	HttpStatus.OK
createCategory	CategoryDTO		POST	-	HttpStatus.CREATED
updateCategory	CategoryDTO	Integer	PUT	/{id}	HttpStatus.NO_CONTENT
deleteCategory	-	Integer	DELETE	/{id}	HttpStatus.NO_CONTENT

Incluir validação.

Executar o projeto e testar usando Postman.

Aula 13 - Exercícios

2) Implementando endpoins de Customer

Criar: Entity, DTO, Repository, Service e Controller.
Especificações do endpoint:
Resource URL: /customers

Class method	Request Body	Path Variable	HTTP Method	Path mapping	Response Status
findAll	-	-	GET	-	HttpStatus.OK
findById	-	Integer	GET	/{id}	HttpStatus.OK
createCustomer	CustomerDTO		POST	-	HttpStatus.CREATED
updateCustomer	CustomerDTO	Integer	PUT	/{id}	HttpStatus.NO_CONTENT
deleteCustomer	-	Integer	DELETE	/{id}	HttpStatus.NO_CONTENT

Não esqueça do AddressRepository para persistir e buscar os endereços junto com o cliente.
Incluir validação.
Executar o projeto e testar usando Postman.

Analisar a necessidade de criar uma classe CustomerNewDTO apenas para receber os dados na hora de criar um cliente novo, devido a possíveis dados diferentes entre ele e o CustomerDTO

Aula 14 - Exercícios

3) Implementando endpoins de Product

Criar: Entity, DTO, Repository, Service e Controller.
Especificações do endpoint:
Resource URL: /products

Class method	Request Body	Path Variable	HTTP Method	Path mapping	Response Status
findAll	-	-	GET	-	HttpStatus.OK
findById	-	Integer	GET	/{id}	HttpStatus.OK
createProduct	ProductDTO		POST	-	HttpStatus.CREATED
updateProduct	ProductDTO	Integer	PUT	/{id}	HttpStatus.NO_CONTENT

Observações:

- o ProductDTO deve ter apenas o nome da categoria do produto.
- o createProduct deve criar uma categoria nova caso ela ainda não existir.

- não tem DELETE porque deve utilizar o atributo “active”.

Incluir validação.
Executar o projeto e testar usando Postman.

Aula 15 - Exercícios

4) Implementando endpoins de Order

Criar: Entity, DTO, Repository, Service e Controller.
Especificações do endpoint:
Resource URL: /orders

Class method	Request Body	Path Variable	HTTP Method	Path mapping	Response Status
findAll	-	-	GET	-	HttpStatus.OK
findById	-	Integer	GET	/{id}	HttpStatus.OK
findByCustomer	OrderDTO	Integer	GET	/{customerId}	HttpStatus.CREATED
createOrder	ProductDTO	Integer	POST	/{id}	HttpStatus.NO_CONTENT

Observações:

- o createOrder não deve criar um pedido caso haja alguma informação inconsistente/ faltante.
- Colocar o @Transactional no createOrder do controller.
- As alterações no pedido devem atualizar o Payment também.

Incluir validação.
Executar o projeto e testar usando Postman.

Aula 16 - Exercícios

Fazer ajustes e correções.

Extra:
Editar o arquivo application.properties e colocar as seguintes informações:
server.contextPath=/topicos
server.port=5555
Executar a classe com o método main.
Acessar a URL: localhost:12345/topicos