UDESC

IBIRAMA

UNIVERSIDADE ESTADUAL DE SANTA CATARINA
CEAVI - CENTRO DE EDUCAÇÃO SUPERIOR DO ALTO VALE DO ITAJAÍ
DEPARTAMENTO DE ENGENHARIA DE SOFTWARE

Augusto Rustick e Denis Zickuhr

Entrega 1 - Projeto Integrador III

Ibirama

1. TEMA

O tema escolhido foi fazer um sistema de uma loja, onde são registrados a quantidade de vendas de um funcionário e a entrega de produtos que um transportador leva para cada departamento.

Cada departamento vende apenas um tipo de produto, e pode possuir quaisquer quantidades de funcionários ou de transportadores.

2. DIAGRAMAS

Segue neste tópico os diagramas que o sistema possui, sendo eles: o diagrama de classes e o diagrama da aplicação do servidor.

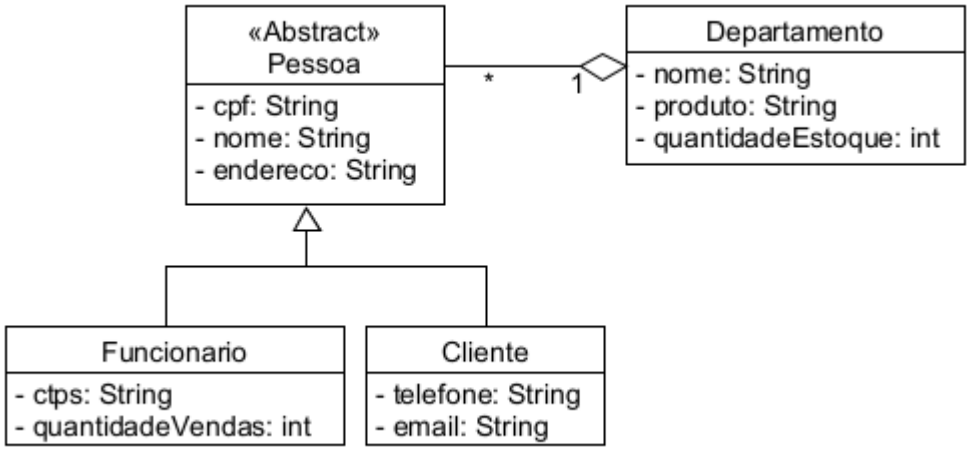
2.1. Diagrama de classes

As nossas entidades do sistema sofreram uma pequena remodelagem, devido a percepção da necessidade de alteração da modelagem antiga. Essa mudança ocorreu na entidade Cliente, que passou a ser a entidade Transportador, e o motivo dessa mudança foi por causa da falta de consistência em dizer que um cliente estava atrelado a um departamento.

Além do nome da entidade, também foi feita uma mudança na sua propriedade de "email", que passou a se chamar "carregamento"

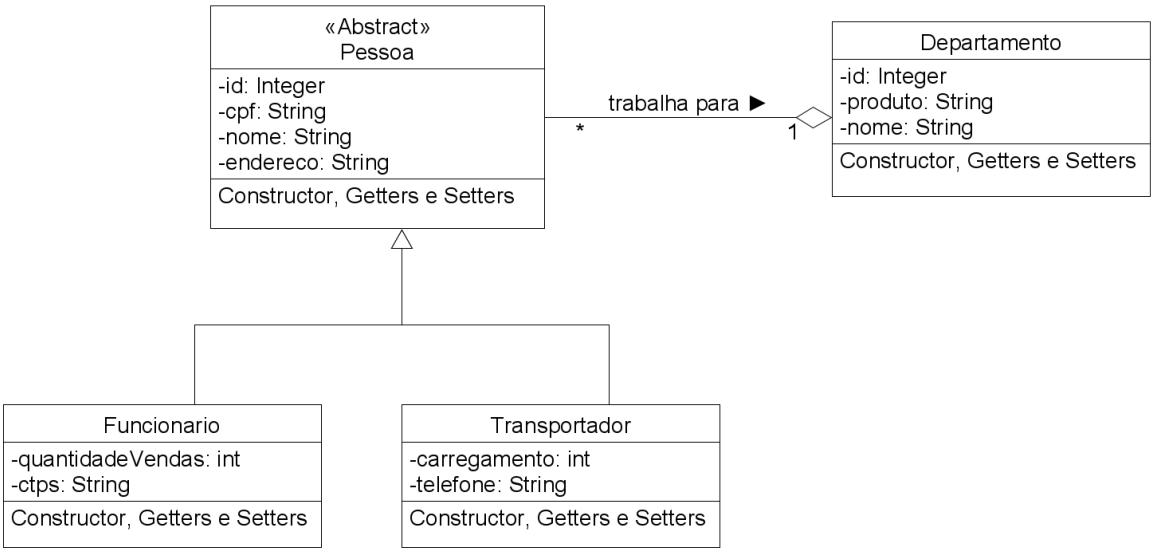
2.1.1. DIAGRAMA ANTIGO

O diagrama antigo possuía a entidade cliente, que fora substituída ao decorrer do projeto



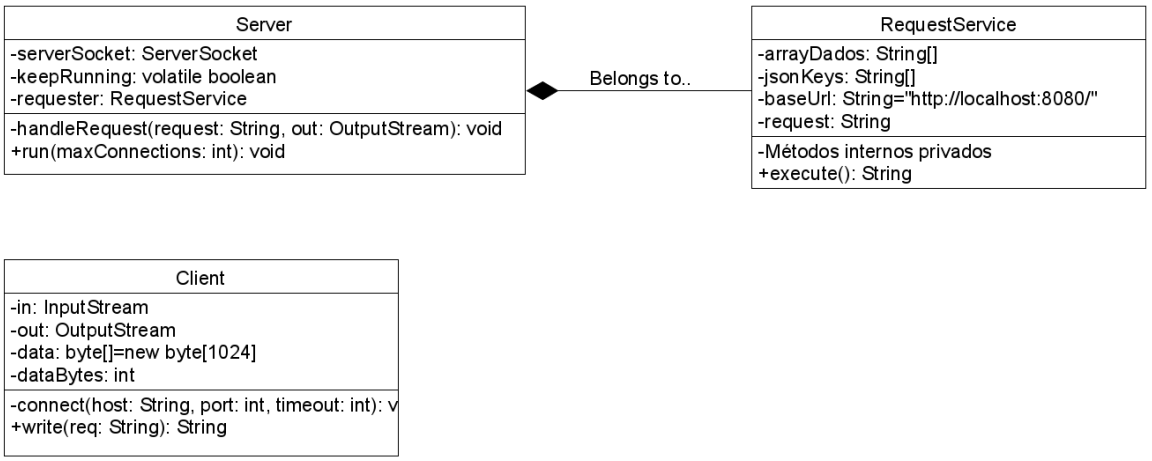
2.1.2. DIAGRAMA REMODELADO

A nova modelagem consta com o transportador no lugar de Cliente. Basicamente essa modelagem segue a abordagem de que uma pessoa trabalha para um departamento, e, por polimorfismo, essa pessoa pode ser tanto um funcionário, que vende as mercadorias desse departamento, quanto um transportador, que abastece o departamento em questão.



2.2. Diagrama da aplicação Servidor

O diagrama a seguir inclui as classes de cliente e servidor, que trocam bytes entre si, além do da classe de RequestService, que faz o consumo das APIs desenvolvidas em Spring.



3. FORMATAÇÃO DAS OPERAÇÕES

3.1. Formatação base de Requests

A formatação dos requests é muito específica. Para fazer um pedido ao banco de dados, feito em Spring, é necessária a chamada de uma classe do sistema chamada RequestService, que faz o consumo da API desenvolvida no Spring. Essa classe de request é instanciada pelo servidor, e o cliente apenas envia a requisição que ele deseja ao servidor, que faz a chamada do service, consumindo a API e retornando ao cliente a resposta da requisição.

Para cada uma das classes, é necessário enviar a request de uma maneira formatada em particular. Todos os parâmetros necessários devem ser passados em uma ordem certa:

Entidade	Alguma entre: departamento; funcionario; transportador;
Método	Algum dos seguintes: INSERT; UPDATE; GET; LIST; REMOVE
Campos	Todos os que vão ser passados adiante
Formatação	"entidade"; "tipoRequisicao"; campo1; campo2; campo3…

3.2. Formatação de Departamento

3.2.1. FORMATAÇÃO DE INSERT

Entidade	Departamento
Campos	Entidade: departamento Método: INSERT Nome: Loja Produto: Coisa Quantidade em estoque: x; $x \in Z$ id: nulo
Chaves do JSON	"entity", "requestType", "nome", "produto", "quantidadeEstoque"
Formatação	“departamento”; “INSERT”;”Loja”;”Coisa”; “x”
Sucesso	Requisição de tipo INSERT foi um sucesso
Erro	Erro ao cadastrar, verifique os campos e tente novamente

3.2.2. FORMATAÇÃO DE UPDATE

Entidade	Departamento
Campos	Entidade: departamento Método: INSERT Nome: Lojinha Produto: Coisa Quantidade em estoque: x; $x \in Z$ id: y; $y \in Z$
Chaves do JSON	"entity", "requestType", "nome", "produto", "quantidadeEstoque", "id"
Formatação	“departamento”; “UPDATE”;”Lojinha”;”Coisa”; “x”, “y”
Sucesso	Requisição de tipo UPDATE foi um sucesso
Erro	Erro ao alterar , verifique os campos e tente novamente

3.2.3. FORMATAÇÃO DE GET

Entidade	Departamento
Campos	Entidade: departamento Método: GET Nome: loja
Formatação	“departamento”; “GET”;”Loja”
Sucesso	GET: Loja Coisa X

	Pessoas: (Quantidade de pessoas) Pessoa 1: cpf1, Pessoa 2: cpf2, …
Erro	Nenhum registro encontrado com esse nome

3.2.4. FORMATAÇÃO DE LIST

Entidade	Departamento
Campos	Entidade: departamento Método: LIST
Formatação	“departamento”; “LIST”
Sucesso com itens	LIST: Quantidade: (número de departamentos) Loja 1: nome1, produto1, quantidadeEstoque1, Pessoas: (Quantidade de pessoas) Pessoa 1: cpf1, Pessoa 2: cpf2, …
Sucesso sem itens	List: 0

3.2.5. FORMATAÇÃO DE DELETE

Entidade	Departamento
Campos	Entidade: departamento Método: REMOVE Nome: loja
Formatação	“departamento”; “REMOVE”;”Loja”
Sucesso	A operação DELETE foi um sucesso
Erro	Não foi possível encontrar um departamento com esse nome para deletar

3.3. Formatação de Funcionário

3.3.1. FORMATAÇÃO DE INSERT

Entidade	Transportador
Campos	Entidade: transportador Método: INSERT Cpf: 11122233344 Nome: Fulano Endereço: Rua xy CTPS: 12345678 Departamento: x; $x \in Z$
Chaves do JSON	"entity", "requestType", "cpf", "nome", "endereco", "ctps", "quantidadeVendas", "departamento"
Formatação	“transportador”; “UPDATE”;”11122233344”;”Fulano”; “Rua xy”, “12345678”; “x”
Sucesso	Requisição de tipo UPDATE foi um sucesso
Erro	Erro ao alterar , verifique os campos e tente novamente

3.3.2. FORMATAÇÃO DE UPDATE

Entidade	Transportador
Campos	Entidade: transportador Método: INSERT Cpf: 11122233344 Nome: Fulano Endereço: Rua xy CTPS: 12345678 Quantidade de Vendas: z; **y** z Z Departamento: x; **x** ∈ Z id: y; **y** ∈ Z
Chaves do JSON	"entity", "requestType", "cpf", "nome", "endereco", "ctps", "quantidadeVendas", "departamento", "id"
Formatação	“transportador”; “UPDATE”;”11122233344”;”Fulano”; “Rua xy”, “12345678”; “z”; “x”; “y”
Sucesso	Requisição de tipo UPDATE foi um sucesso
Erro	Erro ao alterar , verifique os campos e tente novamente

3.3.3. FORMATAÇÃO DE GET

Entidade	Funcionário
Campos	Entidade: funcionario Método: GET Nome: loja
Formatação	“funcionario”; “GET”;”cpf”
Sucesso	GET: cpf nome rua

Erro	Nenhum registro encontrado com esse cpf

3.3.4. FORMATAÇÃO DE LIST

Entidade	Funcionário
Campos	Entidade: funcionario Método: LIST
Formatação	“funcionario”; “LIST”
Sucesso com itens	LIST: Quantidade: (número de departamentos) Funcionario 1: cpf1, nome1, endereco1, …
Sucesso sem itens	List: 0

3.3.5. FORMATAÇÃO DE DELETE

Entidade	Funcionário
Campos	Entidade: funcionario Método: REMOVE cpf: 11122233344
Formatação	“departamento”; “REMOVE”;”11122233344”
Sucesso	A operação DELETE foi um sucesso

Erro	Não foi possível encontrar um funcionário com esse cpf para deletar

3.4. Formatação de Transportador

3.4.1. FORMATAÇÃO DE INSERT

Entidade	Transportador
Campos	Entidade: transportador Método: INSERT Cpf: 11122233344 Nome: Lojinha Endereço: Rua xy Telefone: 123456789 Carregamento: z; **y** z Z Departamento: x; **x** ∈ Z
Chaves do JSON	"entity"; "requestType"; "cpf"; "nome"; "endereco"; "telefone"; "carregamento"; "departamento"
Formatação	“transportador”; “INSERT”;”11122233344”;”Fulano”; “Rua xy”, “123456789”; “z”; “x”
Sucesso	Requisição de tipo UPDATE foi um sucesso
Erro	Erro ao alterar , verifique os campos e tente novamente

3.4.2. FORMATAÇÃO DE UPDATE

Entidade	Transportador
Campos	Entidade: transportador Método: INSERT Cpf: 11122233344 Nome: Fulano Endereço: Rua xy Telefone: 123456789 Carregamento: z; **y** z Z Departamento: x; **x** ∈ Z

	id: y; $y \in Z$
Chaves do JSON	"entity"; "requestType"; "cpf"; "nome"; "endereco"; "telefone"; "carregamento"; "departamento"; "id"
Formatação	“transportador”; “UPDATE”;”11122233344”;”Fulano”; “Rua xy”, “123456789”; “z”; “x”; “y”
Sucesso	Requisição de tipo UPDATE foi um sucesso
Erro	Erro ao alterar , verifique os campos e tente novamente

3.4.3. FORMATAÇÃO DE GET

Entidade	Transportador
Campos	Entidade: transportador Método: GET Nome: loja
Formatação	“transportador”; “GET”;”cpf”
Sucesso	GET: cpf nome rua
Erro	Nenhum registro encontrado com esse cpf

3.4.4. FORMATAÇÃO DE LIST

Entidade	Transportador

Campos	Entidade: transportador Método: LIST
Formatação	“transportador”; “LIST”
Sucesso com itens	LIST: Quantidade: (número de departamentos) Transportador 1: cpf1, nome1, endereco1, …
Sucesso sem itens	List: 0

3.4.5. FORMATAÇÃO DE DELETE

Entidade	Transportador
Campos	Entidade: transportador Método: REMOVE cpf: 11122233344
Formatação	“transportador”; “REMOVE”;”11122233344”
Sucesso	A operação DELETE foi um sucesso
Erro	Não foi possível encontrar um transportador com esse cpf para deletar

4. TECNOLOGIAS USADAS

As tecnologias usadas no projeto foram as seguintes:

- IDE: foi usada, principalmente o InteliJ para fazer o projeto, mas o começo das API’s foram feitas em Eclipse;
- Banco de dados: foi usado o PostgreSQL com PgAdmin para fazer a persistência, e o DataGrip para monitorar os registros;

- Testes: apenas o Postman foi usado, para verificar o funcionamento das APIs e se foi executado corretamente o consumo;
- Versionamento: todo feito em GitHub

5. COMO EXECUTAR O PROJETO

Para executar o projeto, segue em anexo um print do arquivo ReadMe, contido no GitHub do projeto