**UNIVERSIDADE DO OESTE DE SANTA CATARINA – UNOESC**

ADRIEL MATHEUS GRAUPP

ALAN BENDER

GUSTAVO BERWIAN

LUCAS BONAMIGO

**SIM – SLIPPER INDUSTRY MANAGEMENT**

São Miguel do Oeste

2022

ADRIEL MATHEUS GRAUPP

ALAN BENDER

GUSTAVO BERWIAN

LUCAS BONAMIGO

**SIM – SLIPPER INDUSTRY MANAGEMENT**

Trabalho apresentado ao curso de Ciência da Computação como requisito para obtenção de nota para o componente curricular de Banco de Dados II, Engenharia de Software I e Programação II.

Orientadores: Roberson Junior Fernandes Alves, Franciele Carla Petry, Otilia Donato Barbosa

São Miguel do Oeste

2022

**SUMÁRIO:**

Introdução3

Slipper Industry Managemnt4

Oque nos foi requisitado4

Funções do Sistema4

Diagramas5

Codificação10

Dicionario de Dados...............................................................................................................15

Modelo ER .............................................................................................................................16

Criando o Banco....................................................................................................................17

Relatórios................................................................................................................................21

Políticas de Acesso.................................................................................................................22

Triggers...................................................................................................................................24

Procedures..............................................................................................................................25

Backup e Restore....................................... ............................................................................26

**Conclusão..............................................................................................................................28**

**Referências............................................................................................................................29**

**INTRODUÇÃO**

O presente trabalho é sobre o desenvolvimento do Slipper Industry Management – SIM, trata-se de um sistema gerenciador para indústria de calçados, mais especificadamente chinelos, o sistema foi baseado diante da demanda de um sistema requisitado pela empresa Cia Beach da cidade de Guarujá do Sul em Santa Catarina. O sistema tem como objetivo auxiliar a fábrica na gestão de vendas, no controle de produção, estoque e também fazer a gestão dos funcionários.

O texto apresenta informações de suma importância para o desenvolvimento de um sistema gerenciador de fábrica de chinelo, tal conhecimentos poderão ser aplicados para o desenvolvimento de qualquer sistema. Apresentaremos ao leitor principais objetivos e funções presentes no sistema, o artigo traz imagens, código e diagramas que podem auxiliar o leitor no entendimento do conteúdo, mostraremos na prática a criação e planejamento de projeto, tanto banco de dados utilizando Postgres e também na parte de programação onde usamos Java e suas várias funções junto com HTML.

**SIM – SLIPPER INDUSTRY MANAGEMENT**

O sistema foi desenvolvido buscando facilitar o dia-a-dia de funcionários e gestores de uma indústria, ele conta com funções que auxiliam desde o processo produtivo como a gestão e estoque da produção e da matéria prima, até a parte administrativa provendo controle de funcionários, cadastramento de clientes e mecanismos para facilitar a venda para o atacado e varejo.

**O SISTEMA QUE NOS FOI REQUISITADO**

O sistema foi desenvolvido com base no que nos foi requisitado pela empresa Cia Beach e nos conteúdos trabalhados em sala de aula nas matérias de Banco de Dados II, Engenharia de Software I e Programação II.

* Precisamos de um sistema que auxilie no: PCP (Planejamento e controle de produção), Estoque, Vendas, Compras, Fiscal, Financeiro, Comercial, Cadastro de clientes, RH e Fornecedores.
* o sistema não pode gerar o fiscal caso haja alguma restrição no CGC (Cadastro Geral de Contribuintes), também o cadastro pode ser bloqueado caso o mesmo tenha inadimplência com a empresa. Também a regras no PCP / estoque que emitem alertas de estoque baixo ou caso o fluxo de Vendas passe o limite de produção e matéria prima para a sua produção.
* sistema deverá ter uma comunicação fluente entre os terminais e Sub terminais de vendas mantendo sempre os mesmos atualizados.
* Cada colaborador deve ter acesso com login e senha tendo acesso somente ao que lhe cabe, Ex: setor financeiro, somente financeiro.
* Os produtos (chinelos) são registrados pelo modelo (referencia (Wk02)), e também pelo código.

**FUNÇÕES DO SISTEMA:**

Gestão Vendas: Uma das principais funções do sistema é fazer a gestão das vendas, para realizar as vendas o sistema leva em consideração a disponibilidade do produto no estoque. O sistema conta com duas formas de venda, para pessoa física e pessoa jurídica, já que o sistema é focado para uma indústria no formato de distribuidora.

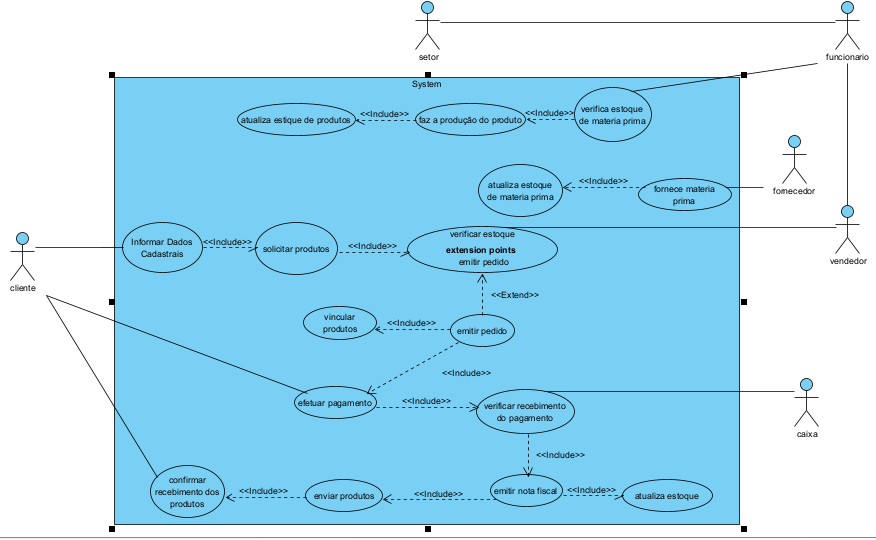
Gestão de Produção/ Estoque: o sistema faz o controle de produção, levando em consideração o estoque, onde são registrados todos os produtos com suas características e quantidades.

Gestão de Funcionários: O sistema auxilia no controle e registro de funcionários, onde são separados pelo setor ou função.

**DIAGRAMAS**

**Casos de Uso:**

Na Linguagem de modelagem unificada (UML), o diagrama de caso de uso resume os detalhes dos usuários do seu sistema (também conhecidos como atores) e as interações deles com o sistema. Na criação de um, use-se um conjunto de símbolos e conectores especializados. O diagrama de caso de uso tem como principais finalidade ajudar sua equipe a representar e discutir cenários e interações entre usuário e sistema. Logo abaixo você poderá conferir a interação entre usuário e sistema no projeto SIM:



**Fluxos de casos de uso:**

Os fluxos de caso de uso definem como será o fluxo do sistema, em nosso trabalho apresentaremos dois tipos de fluxos, o fluxo principal, que define o caminho básico, ou caminho ótimo, onde mostra o que o sistema fará passo a passo no fluxo mais comum, como logo abaixo:

1- Informar dados cadastrais

2- Solicitar produtos

3- Verificar estoque

4- Emitir pedido

5- Vincular produtos

6- Efetuar pagamento

7- Verificar recebimento do pagamento

8- Emitir nota fiscal

9- Atualizar estoque

10- Enviar produtos

11- Cliente confirmar recebimento dos produtos

Mas também há o fluxo alternativo, onde mostra os possíveis caminhos alternativos que poderá ocorrer com o sistema, como o que está descrito logo abaixo:

Fluxo Alternativo: irregularidade nos dados cadastrais

1a- sistema informa irregularidade nos dados

1b - cliente corrige os dados

retoma passo 1.

Fluxo Alternativo: Produtos sem estoque

3a - Sistema informa a falta de produtos

3b - cliente muda a quantidade de produtos do pedido ou seleciona outros produtos

retoma ao passo 2.

Fluxo Alternativo: Pagamento não recebido

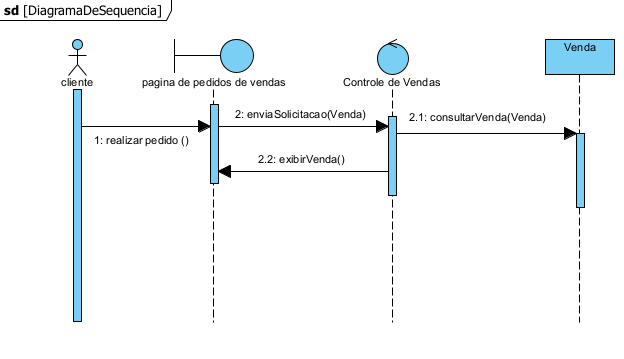
7a - informar ao cliente que o pagamento não foi efetuado

7b - cliente efetua pagamento

retoma ao passo 6.

**Diagrama de Sequência:**

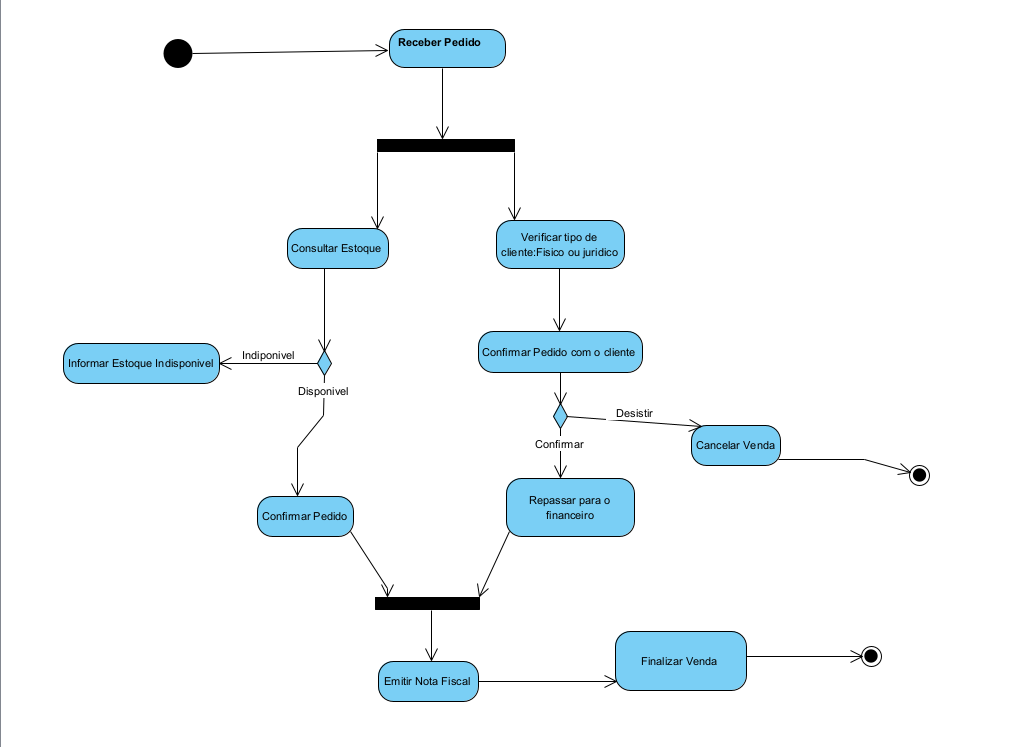
O diagrama de sequência descreve como, e em qual ordem, um grupo de objetos trabalha em conjunto. Estes diagramas são usados para entender as necessidades de um novo sistema, seus cenários e eventos ou para documentar um processo existente. O seguinte diagrama de sequencia descreve o trecho da parte de pedidos de venda do sistema.



**Diagrama de Atividades:**

Diagramas de atividade ajudam a entender o processo e comportamento do sistema. Para criar um diagrama de atividade, é necessário um conjunto de símbolos especiais, incluindo aqueles para dar partida, encerrar, fundir ou receber etapas no fluxo.

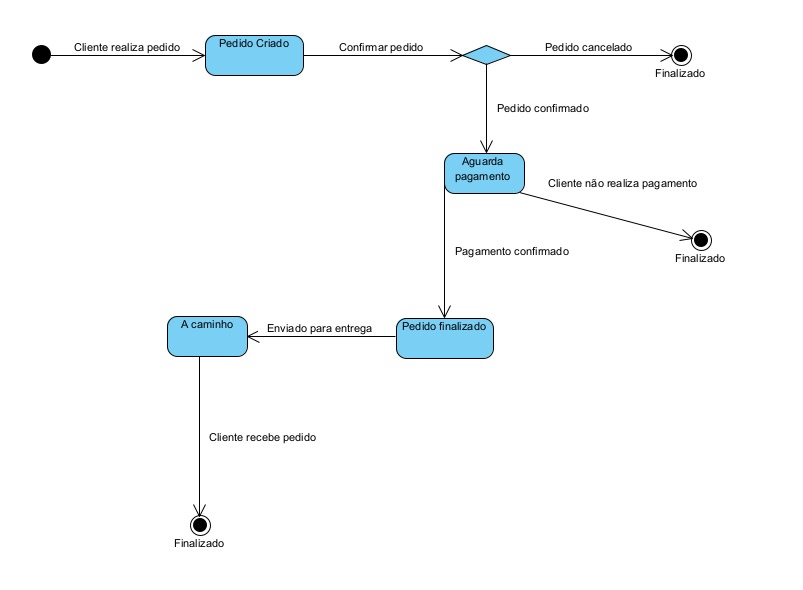
No seguinte diagrama pode ser identificado como funciona o sistema, seu fluxo em geral, suas funções e seus possíveis caminho, desde o recebimento do pedido, emissão da nota fiscal, até finalizar a venda.



**Diagrama de Estados**

Em um diagrama de estado, um objeto possui um comportamento e um estado, o estado de um objeto depende da atividade na qual ele está processando. Um diagrama de estado mostra os possíveis estados de um objeto e as transações responsáveis pelas suas mudanças de estado.

O seguinte diagrama foi baseado nos possíveis estados quando o cliente realiza um pedido.



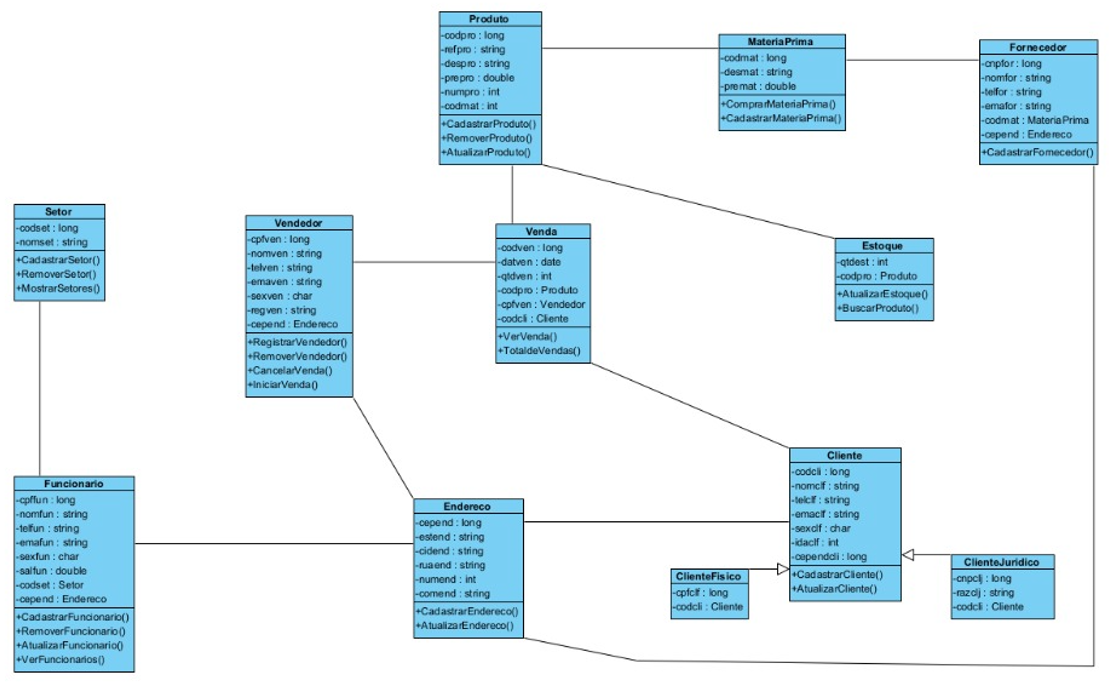
**Diagrama de Classes**

Diagramas de classes servem para mapear de forma clara a estrutura de um determinado sistema ao modelar suas classes, seus atributos, operações e relações entre objetos.

Os diagramas de classes são um tipo de diagrama da estrutura porque descrevem o que deve estar presente no sistema a ser modelado. Podemos afirmar de maneira mais simples que seria um conjunto de objetos com as mesmas características, assim saberemos identificar objetos e agrupá-los, de forma a encontrar suas respectivas classes.

Na imagem logo abaixo pode ser identificada todas as classes, métodos e atributos que foram necessários para o desenvolvimento do projeto.

Classes: Setor, Funcionário, Matéria Prima, Fornecedor, Endereço, Vendedor, Cliente (Físico e Jurídico), Produto, Estoque e Venda.



**CODIFICAÇÃO**

Para a codificação do sistema usamos as linguagens de programação de Java e HTML, o sistema foi desenvolvido na ideia de SpringBoot.

O Spring Boot é um framework que torna fácil a criação de aplicações Spring autossuficientes e robustas, possibilitando a execução imediata. Contudo isso só é possível por conta da abordagem opinativa sobre a plataforma Spring e bibliotecas de terceiros, que permite ao desenvolvedor gastar o mínimo de tempo possível configurando o projeto, e sim codificando suas regras de negócio.

Spring Boot utiliza o conceito de convenção sobre configuração, ou seja, o Spring Boot já possui uma série de configurações padrões para as bibliotecas Spring ou de terceiros que você deseja utilizar.

Para começarmos criamos a classe Aplication no Package br.edu.unoesc.springboot.sim, essa será nossa classe principal, que chamará nossas outras classes do Package Model.

package br.edu.unoesc.springboot.sim;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

/\*\*

\*

\* Spring Boot application starter class

\*/

@EntityScan(basePackages = "br.edu.unoesc.springboot.sim.model")

@SpringBootApplication

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

Após isso foi construída a classe Getings Controllers que é responsável por armazenar os end points, e todas as requisições de salvar, deletar, pesquisar...

package br.edu.unoesc.springboot.sim.controllers;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import br.edu.unoesc.springboot.sim.model.cliente;

import br.edu.unoesc.springboot.sim.model.produto;

import br.edu.unoesc.springboot.sim.model.setor;

import br.edu.unoesc.springboot.sim.repository.ClienteRepository;

import br.edu.unoesc.springboot.sim.repository.ProdutoRepository;

import br.edu.unoesc.springboot.sim.repository.SetorRepository;

import br.edu.unoesc.springboot.sim.model.setor;

import br.edu.unoesc.springboot.sim.repository.SetorRepository;

/\*\*

\*

\* A sample greetings controller to return greeting text

\*/

@RestController

public class GreetingsController {

/\*\*

\*

\* @param name the name to greet

\* @return greeting text

\*/

@Autowired // injeção de dependência

private ClienteRepository clienteRepository;

@PostMapping(value = "salvarcliente")

@ResponseBody

public ResponseEntity<cliente> salvarcliente(@RequestBody cliente cliente){

cliente client = clienteRepository.save(cliente);

return new ResponseEntity<cliente>(client, HttpStatus.CREATED);

}

@DeleteMapping(value = "deletecliente")

@ResponseBody

public ResponseEntity<String> deletecliente(@RequestParam Long idclient) {

clienteRepository.deleteById(idclient);

return new ResponseEntity<String>("Cliente excluido com sucesso", HttpStatus.OK);

}

@GetMapping(value = "buscaruseridcliente")

@ResponseBody

public ResponseEntity<cliente> buscaruseridcliente(@RequestParam(name = "codigocliente") Long codigocliente){

cliente cliente = clienteRepository.findById(codigocliente).get();

return new ResponseEntity<cliente>(cliente, HttpStatus.OK);

}

@PutMapping(value = "atualizarcliente")

@ResponseBody

public ResponseEntity<?> atualizarcliente(@RequestBody cliente cliente){

if(cliente.getCodigocliente()==null) {

return new ResponseEntity<String>("Id não informado para atualizar", HttpStatus.OK);

}

cliente client = clienteRepository.saveAndFlush(cliente);

return new ResponseEntity<cliente>(client, HttpStatus.OK);

}

@GetMapping(value = "buscarpornomecliente")

@ResponseBody

public ResponseEntity<List<cliente>>bucarPorNomeCliente(@RequestParam(name = "nome") String nome){

List<cliente> client = clienteRepository.buscarPorNomeCliente(nome.trim().toUpperCase());

return new ResponseEntity<List<cliente>>(client, HttpStatus.OK);

}

}

Para a criação do projeto foi feita algumas classes que armazenam os atributos e dados, assim dentro de um Package chamado br.edu.unoesc.springboot.sim.model criamos as seguintes classes: Setor, Funcionário, Endereço, Fornecedor, Matéria Prima, Produto, vendedor, Cliente (Cliente físico e jurídico) Estoque e Venda.

Segue de exemplo a classe cliente:

package br.edu.unoesc.springboot.sim.model;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.SequenceGenerator;

@Entity

@SequenceGenerator(name = “seq\_cliente”, sequenceName = “seq\_cliente”, allocationSize = 1, initialValue = 1)

public class cliente implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = “seq\_cliente”)

@Column(name = “codcli”)

private Long codigocliente;

@Column(name = “nomclf”)

private String nomecliente;

@Column(name = “telclf”)

private String telefonecliente;

@Column(name = “emaclf”)

private String emailcliente;

@Column(name = “sexclf”)

private char sexocliente;

@Column(name = “idaclf”)

private int idadecliente;

@Column(name = “cependcli”)

private Long cependerecocliente;

//Getters e Setters – Gerados automaticamente pelo Eclipse

}

Todas as outras classes do sistema foram construídas seguindo o mesmo modelo, definindo os atributos com seu respectivo nome que terá no banco de dados, e no final os Getters e Setters que podem ser gerados de maneira automática pelo programa Eclipse que está sendo utilizado para o desenvolvimento deste projeto.

Foram criadas as interfaces Repository dentro do Package br.edu.unoesc.springboot.sim.repository, foi desenvolvida para todas as classes que foi, essas interfaces são de suma importância, pois são responsáveis por armazenar os dados no banco.

Segue o exemplo da codificação da interface clienteRepository

package br.edu.unoesc.springboot.sim.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Repository;

import br.edu.unoesc.springboot.sim.model.cliente;

@Repository

public interface ClienteRepository extends JpaRepository<cliente, Long> {

@Query(value = "select c from cliente c where upper(trim(c.nomecliente)) like %?1%")

List<cliente> buscarPorNomeCliente(String nome);

}

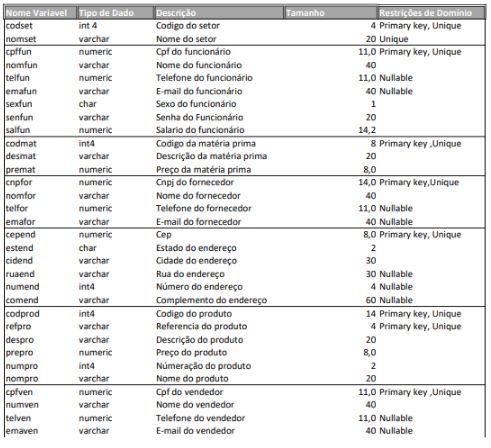
Por último foi criado o arquivo index.html, que é onde foi criado as telas para o usuário utilizar o programa, criamos as telas de cadastro de clientes e produtos. Fizemos os campos para serem informados os dados do cliente e configurado os botões de salvar, deletar e pesquisar, o mesmo foi feito para o cadastro de produtos.

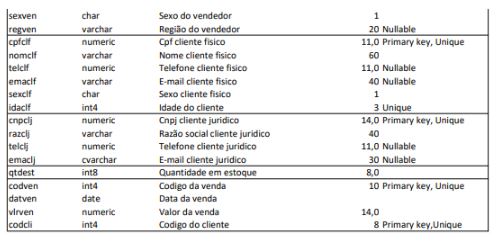
Criamos scripts usando Ajax para realizar as operações de pesquisar, salvar e deletar.

O script completo está disponível no repositório SIM lá no GitHub.

**DICIONARIO DE DADOS:**

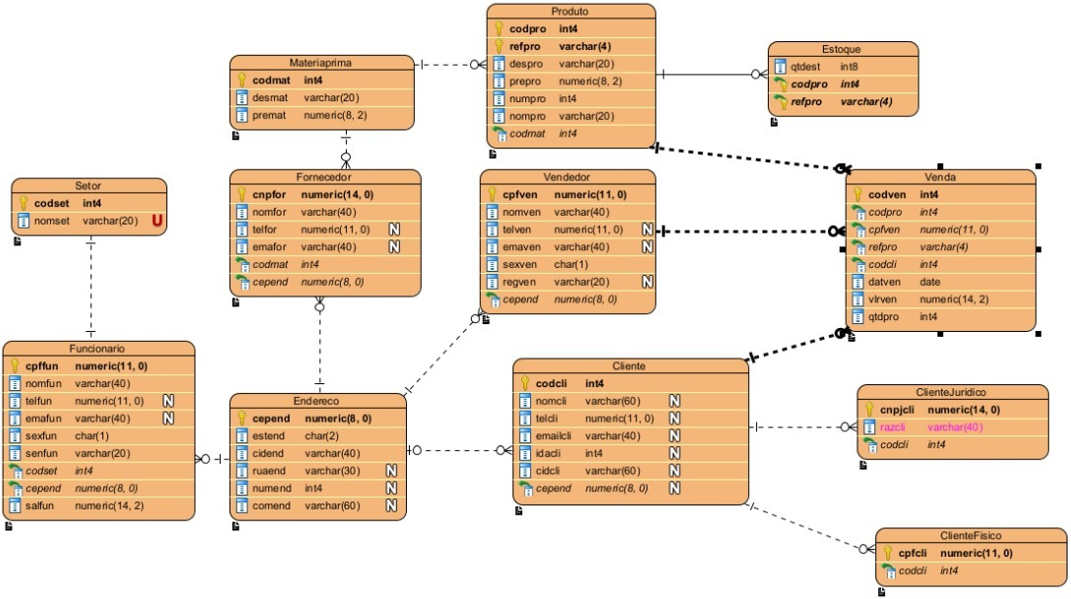
O dicionário de dados consiste numa lista organizada de todos os elementos de dados que são pertinentes para o sistema. O dicionário de dados o modelo descreve entradas, saídas, composição de depósitos de dados e alguns cálculos intermédios, consiste num ponto de referência de todos os elementos envolvidos na medida em que permite associar um significado a cada termo utilizado.

****



**Modelo ER (Entidade Relacionamento)**

O Modelo Entidade Relacionamento, descreve todas entidades (tabelas) necessárias para o desenvolvimento do projeto, no modelo ER são descritos também todos os atributos de cada tabela, mostra as relações entre as entidades, chaves primárias, estrangeiras e o tipo do atributo



**CRIANDO O BANCO**

Para criar o banco de dados escrevemos o seguinte script, no script foi feito primeiramente a criação das tabelas, e no fim as declarações de chaves estrangeiras (Foreign Key – FK):

**Script de criação das tabelas do banco:**

CREATE TABLE Cliente (

codcli int,

nomcli varchar(60),

telcli numeric(11,0),

emailcli varchar(40),

idacli int,

cidcli varchar (60),

cepend numeric(8,0),

PRIMARY KEY (codcli)

);

COMMENT ON TABLE Cliente IS 'registro do cliente';

COMMENT ON COLUMN Cliente.codcli IS 'codigo do cliente';

COMMENT ON COLUMN Cliente.nomcli IS 'nome do cliente';

COMMENT ON COLUMN Cliente.telcli IS 'telefone do cliente';

COMMENT ON COLUMN Cliente.emailcli IS 'e-mail do cliente';

COMMENT ON COLUMN Cliente.idacli IS 'idade do cliente';

CREATE TABLE ClienteFisico (

cpfcli numeric(11, 0),

codcli int,

PRIMARY KEY (cpfcli)

);

COMMENT ON TABLE ClienteFisico IS 'Tabela de cadastro de clientes (pessoa fisica)';

COMMENT ON COLUMN ClienteFisico.cpfcli IS 'cpf do cliente';

CREATE TABLE ClienteJuridico (

cnpjcli numeric(14, 0) NOT NULL,

razcli varchar(40) NOT NULL,

codcli int,

PRIMARY KEY (cnpjcli)

);

COMMENT ON TABLE ClienteJuridico IS 'cadastro de cliente (pessoa juridica)';

COMMENT ON COLUMN ClienteJuridico.cnpjcli IS 'cnpj do cliente';

COMMENT ON COLUMN ClienteJuridico.razcli IS 'razao social';

CREATE TABLE Endereco (

cepend numeric(8, 0) NOT NULL,

estend char(2),

cidend varchar(40),

ruaend varchar(30),

numend int4,

comend varchar(60),

PRIMARY KEY (cepend));

COMMENT ON TABLE Endereco IS 'tabela de registro de endereço';

COMMENT ON COLUMN Endereco.cepend IS 'cep';

COMMENT ON COLUMN Endereco.estend IS 'estado endereço';

COMMENT ON COLUMN Endereco.cidend IS 'nome da cidade';

COMMENT ON COLUMN Endereco.ruaend IS 'rua';

COMMENT ON COLUMN Endereco.numend IS 'numero do endereço';

COMMENT ON COLUMN Endereco.comend IS 'complemento do endereço';

CREATE TABLE Estoque (

qtdest int8 NOT NULL,

codpro int4 NOT NULL,

PRIMARY KEY (codpro)

);

COMMENT ON TABLE Estoque IS 'tabela de estoque';

COMMENT ON COLUMN Estoque.qtdest IS 'quantidade do produto';

CREATE TABLE Fornecedor (

cnpfor numeric(14, 0) NOT NULL,

nomfor varchar(40) NOT NULL,

telfor numeric(11, 0),

emafor varchar(40),

codmat int4 not null,

cepend numeric(8, 0),

PRIMARY KEY (cnpfor));

COMMENT ON TABLE Fornecedor IS 'Cadastro de fornecedores';

COMMENT ON COLUMN Fornecedor.cnpfor IS 'cnpj do fornecedor';

COMMENT ON COLUMN Fornecedor.nomfor IS 'nome do fornecedor';

COMMENT ON COLUMN Fornecedor.telfor IS 'telefone do fornecedor';

COMMENT ON COLUMN Fornecedor.emafor IS 'e-mail do fornecedor';

CREATE TABLE Materiaprima (

codmat SERIAL NOT NULL,

desmat varchar(20),

premat numeric(8, 2),

PRIMARY KEY (codmat));

COMMENT ON TABLE Materiaprima IS 'cadastro da materia prima';

COMMENT ON COLUMN Materiaprima.codmat IS 'codigo da materia prima';

COMMENT ON COLUMN Materiaprima.desmat IS 'descrição da materia prima';

COMMENT ON COLUMN Materiaprima.premat IS 'preço da materia prima';

CREATE TABLE Setor (

codset SERIAL NOT NULL,

nomset varchar(20) NOT NULL UNIQUE,

PRIMARY KEY (codset));

COMMENT ON TABLE Setor IS 'Cadastro do setor';

COMMENT ON COLUMN Setor.codset IS 'codigo do setor';

COMMENT ON COLUMN Setor.nomset IS 'nome do setor';

CREATE TABLE Venda (

codven SERIAL NOT NULL,

codpro int4 NOT NULL,

cpfven numeric(11, 0) NOT NULL,

codcli int4 NOT NULL,

datven date NOT NULL,

vlrven numeric(14, 2) NOT NULL,

qtdpro int ,

PRIMARY KEY (codven)

);

COMMENT ON TABLE Venda IS 'Cadastro de vendas';

COMMENT ON COLUMN Venda.codven IS 'codigo de vendas';

COMMENT ON COLUMN Venda.datven IS 'data da venda';

COMMENT ON COLUMN Venda.vlrven IS 'valor da venda';

CREATE TABLE Vendedor (

cpfven numeric(11, 0) NOT NULL,

nomven varchar(40) NOT NULL,

telven numeric(11, 0),

emaven varchar(40),

sexven char(1),

regven varchar(20),

cepend numeric(8, 0),

PRIMARY KEY (cpfven));

COMMENT ON TABLE Vendedor IS 'cadastro dos vendedores';

COMMENT ON COLUMN Vendedor.cpfven IS 'cpf do vendedor';

COMMENT ON COLUMN Vendedor.nomven IS 'nome do vendedor';

COMMENT ON COLUMN Vendedor.telven IS 'telefone do vendedor';

COMMENT ON COLUMN Vendedor.emaven IS 'e-mail do vendedor';

COMMENT ON COLUMN Vendedor.sexven IS 'sexo do vendedor';

COMMENT ON COLUMN Vendedor.regven IS 'região em que o vendedor vende';

CREATE TABLE Produto (

codpro int4 NOT NULL,

refpro varchar(4) NOT NULL,

despro varchar(20) NOT NULL,

prepro numeric(8, 2) NOT NULL,

numpro int4 NOT NULL,

nompro varchar(20) NOT NULL,

codmat int4 NOT NULL,

PRIMARY KEY (codpro)

);

COMMENT ON TABLE Produto IS 'Cadastro de produtos';

COMMENT ON COLUMN Produto.codpro IS 'codigo de barras do produto';

COMMENT ON COLUMN Produto.refpro IS 'referencia do produto';

COMMENT ON COLUMN Produto.despro IS 'descrição do produto';

COMMENT ON COLUMN Produto.prepro IS 'preço do produto';

COMMENT ON COLUMN Produto.numpro IS 'numeração do produto';

COMMENT ON COLUMN Produto.nompro IS 'nome do produto';

CREATE TABLE Funcionario (

cpffun numeric(11, 0) NOT NULL,

nomfun varchar(40) NOT NULL,

telfun numeric(11, 0),

emafun varchar(40),

sexfun char(1),

senfun varchar(20),

codset int4 not null,

cepend numeric(8, 0),

salfun numeric(14, 2) NOT NULL,

PRIMARY KEY (cpffun));

COMMENT ON TABLE Funcionario IS 'Cadastro de funcionarios';

COMMENT ON COLUMN Funcionario.cpffun IS 'cpf do funcionario';

COMMENT ON COLUMN Funcionario.nomfun IS 'nome do funcionario';

COMMENT ON COLUMN Funcionario.telfun IS 'telefone do funcionario';

COMMENT ON COLUMN Funcionario.emafun IS 'e-mail do funcionario';

COMMENT ON COLUMN Funcionario.sexfun IS 'sexo do funcionario';

COMMENT ON COLUMN Funcionario.senfun IS 'senha do funcionario';

COMMENT ON COLUMN Funcionario.salfun IS 'salário do funcionario';

ALTER TABLE Produto ADD CONSTRAINT FKProduto272416 FOREIGN KEY (codmat) REFERENCES Materiaprima (codmat);

ALTER TABLE Funcionario ADD CONSTRAINT FKFuncionari869107 FOREIGN KEY (codset) REFERENCES Setor (codset);

ALTER TABLE Venda ADD CONSTRAINT FKVenda583852 FOREIGN KEY (cpfven) REFERENCES Vendedor (cpfven);

ALTER TABLE Venda ADD CONSTRAINT FKVenda891507 FOREIGN KEY (codpro) REFERENCES Produto (codpro);

ALTER TABLE Vendedor ADD CONSTRAINT FKVendedor796523 FOREIGN KEY (cepend) REFERENCES Endereco (cepend);

ALTER TABLE Funcionario ADD CONSTRAINT FKFuncionari985887 FOREIGN KEY (cepend) REFERENCES Endereco (cepend);

ALTER TABLE Fornecedor ADD CONSTRAINT FKFornecedor138171 FOREIGN KEY (cepend) REFERENCES Endereco (cepend);

ALTER TABLE Fornecedor ADD CONSTRAINT FKFornecedor781853 FOREIGN KEY (codmat) REFERENCES Materiaprima (codmat);

ALTER TABLE Estoque ADD CONSTRAINT FKEstoque495509 FOREIGN KEY (codpro) REFERENCES Produto (codpro);

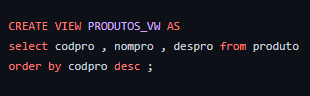
alter table Cliente add constraint cli\_cepend\_fk FOREIGN KEY (cepend) references endereco(cepend);

alter table ClienteFisico add constraint clif\_codcli\_fk foreign key (codcli) references Cliente (codcli);

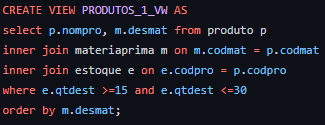
alter table ClienteJuridico add constraint clij\_codcli\_fk foreign key (codcli) references cliente(codcli);

**Relatórios:**

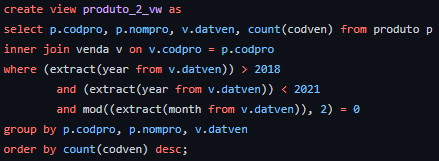
- Relacionar o código, nome e tipo de todos os produtos. Ordene o relatório de forma descendente pelo código;



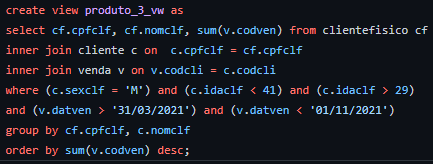
- Relacionar o nome do produto e o nome do fornecedor para os produtos. Filtre somente produtos com estoque entre 15 e 30. Ordene o relatório de forma ascendente pelo nome do fornecedor;



- Relacionar o código do produto, nome do produto, mês da venda, quantidade total de vendas no mês para meses pares entre anos 2019 e 2021. Ordene o relatório do 3 produto com mais vendas(em termos de quantidade) para o produto com menos vendas;



- Relacionar o cpf do cliente, nome do cliente e o total gasto em produtos. Filtrar somente clientes do sexo masculino, com idades entre 30 e 40 anos e que realizaram compras entre os meses de março e outubro de 2021. Ordene o relatório do cliente com mais gastos para o cliente com menos gastos.



**Políticas de Acesso**

Para que usuários não acessem dados indesejáveis, são configuradas politicas de acesso, onde cada colaborador tem acesso apenas ao seu setor.

create group vendedor;

create group gerente;

create group estoquista;

create group financeiro;

create user Gustavo

password 'gustavo';

create user Adriel

password 'adriel';

create user Lucas

password 'lucas';

create user Alan

password 'alan';

grant select, insert,update,delete

on table cliente,clientefisico,clientejuridico,estoque,fornecedor,funcionario,materiaprima,produto,

setor,venda,vendedor

to gerente;

grant select,insert,update

on table cliente, clientefisico,endereco,produto,venda

to vendedor;

grant select ,insert,update,delete

on table estoque , fornecedor, materiaprima,produto

to estoquista;

grant select, insert ,update,delete

on table cliente,clientefisico,clientejuridico,endereco,estoque,produto,venda,vendedor,funcionario

to financeiro;

grant vendedor to Gustavo;

grant gerente to Adriel;

grant estoquista to Lucas;

grant financeiro to Alan;

**Triggers**

São Gatilhos que são disparados em um evento determinado no sistema.

Como por exemplo:

Trigger para controle de estoque

create or replace function controla\_estoque() returns trigger

as

$body$

declare

quantidade numeric;

begin

select qtdest into quantidade from estoque

where codpro = new.codpro;

if(new.qtdest >= quantidade) then

raise exception 'Estoque insuficiente!';

end if;

return new;

end

$body$

language plpgsql;

create trigger controla\_estoque\_tg

before insert

on estoque

for each row

execute procedure controla\_estoque();

Trigger que aceita apenas registro de chinelos que tenham a numeração entre 14 e 48

create or replace function controla\_tamanho() returns trigger

as

$body$

begin

if(new.numpro < 13) or (new.numpro > 48) then

raise exception 'Tamanho Inválido!';

end if;

return new;

end

$body$

language plpgsql;

create trigger controla\_tamanho\_tg

before insert

on produto

for each row

execute procedure controla\_tamanho();

**Procedures**

Procedure para atualizar salário dos funcionários:

create procedure reajuste\_sal (salfun numeric )

return void

$$

begin

if salfun <= 2000 then

alter table funcionario = salfun \*1.3;

elsif salfun <= 4000 then

alter table funcionario = salfun\*1.2;

else

alter table funcionario = salfun\*1.1;

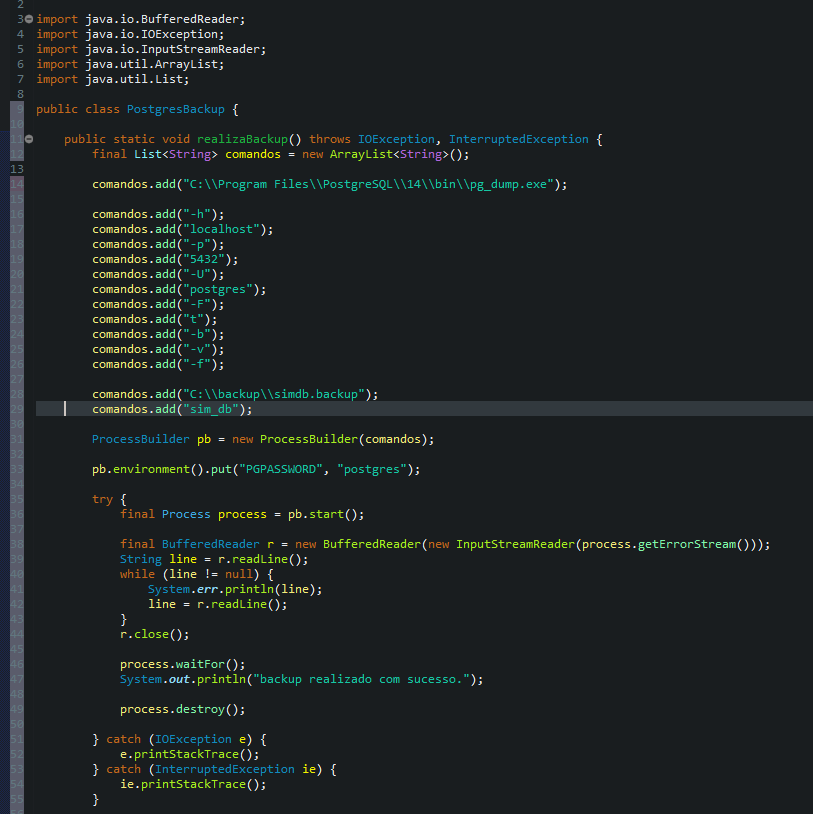
end if;

end;

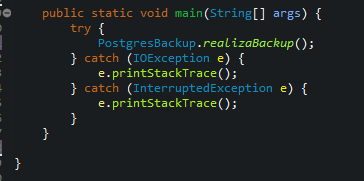
$$

**Backup e Restore**

O backup do banco de dados foi realizado utilizando o pg\_dump, manualmente através de uma classe java criada no eclipse, armazenando o arquivo de backup no diretório “C:\\backup\\simdb.backup””.

Método utilizado: 

Main com o chamado do método:



O restore do backup foi feito de forma manual através do pg\_restore utilizando o seguinte comando:

pg\_restore.exe -h localhost -p 5432 -U postgres -d sim\_db C:\backup\backup\_simdb.backup

**CONCLUSÃO**

Neste trabalho abordamos o desenvolvimento de um sistema SIM – Slipper Industry Management, e concluímos que a uma grande importância na organização, a criação dos diagramas que nos auxiliam muito na hora do desenvolvimento, na criação de um banco de dados, e na parte de codificação utilizando Spring Boot no Java e HTML.

Os objetivos foram cumpridos com sucesso, uma vez que o sistema teve um bom resultado, um sistema funcional, banco de dados consistente. E que conseguimos passar um conhecimento fundamental para a criação de um sistema completo apresentando na pratica o funcionamento do sistema.

**REFERÊNCIAS:**

Banco de Dados. Wikipedia, 2022. Disponível em: https://pt.wikipedia.org/wiki/Banco\_de\_dados. Acesso em: 07 mar 2022.

Erp para confecção. Sistema para Confecção 2022. Disponível em: https://www.sistemaparaconfeccao.com.br/erp-para-confeccao-ad/?gclid=Cj0KCQjwntCVBhDdARIsAMEwAClrkrcKTpoA\_RF0ejprNjADVQJHGLrfIkV0BDM7PEn7rVCgd-rubjsaApbyEALw\_wcB. Acesso em: 23 mai. 2022.

Principais comandos SQL. DevMedia, 2007. Disponível em: https://www.devmedia.com.br/principais-comandos-sql/5748. Acesso em: 28 mai. 2022

Dicionario de Dados. Disponível em: http://www.estgv.ipv.pt/paginaspessoais/ajas/AS/Apontamentos%20Te%C3%B3ricos/as\_3\_4.pdf. Acesso em: 1 dez. 2022.

Diagrama de caso de uso UML: O que é, como fazer e exemplos. LucidChart. Disponível em: https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml. Acesso em: 23 mai. 2022.

Orientações básicas na elaboração de um diagrama de classes. DevMedia. Disponível em: https://www.devmedia.com.br/orientacoes-basicas-na-elaboracao-de-um-diagrama-de-classes/37224. Acesso em: 1 dez 2022.

O que é diagrama de atividades UML?. LucidChart. Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-atividades-uml. Acesso em: 1 dez 2022.