

Trabalho Integrado dos componentes de Banco de Dados II, Engenharia de Software I e Programação II: OrgFit

Alysson Antonietti
Emanuel Gomes Petry
Daniel Machado

Resumo

O uso de um software de gestão oferece facilidades e agilidade, sendo uma ferramenta essencial nos dias atuais, apesar disso, algumas empresas ainda hesitam em adotar essas soluções devido aos custos de implementação e à falta de conhecimento sobre suas vantagens. Essa resistência pode resultar em perda de tempo e dinheiro, pois muitas vezes as empresas acabam realizando tarefas desnecessárias que poderiam ser automatizadas.

O projeto em questão concentra-se em aprimorar a gestão de uma loja de roupas de pequeno porte, focalizando especialmente nas áreas de vendas e controle de estoque. Introduz soluções que permitem ao cliente organizar o comissionamento de seus funcionários, vinculando-os a cada venda realizada.

Apresentando o processo de desenvolvimento nas áreas de programação, engenharia de software e banco de dados para o sistema OrgFit.

Palavras-chave: Sistema, loja de roupas, gestão.

1. Introdução

A importância de um software que gerencie e administre qualquer tipo de negócio é muito grande, pois com ele a empresa pode contar com as facilidades e a agilidade que ele dispõe. Mas mesmo nos dias de hoje algumas empresas ainda não utilizam de tais meios para gerir seus negócios, por fatores que vão desde o valor da implantação do software e pela falta de conhecimento e procura do interessado, esses fatores fazem com que muitas vezes tais empresas possam estar perdendo dinheiro e tempo em trabalhos desnecessários.

O trabalho em questão tem como principal função melhorar o gerenciamento de uma loja de roupas de pequeno porte. Traz soluções principalmente na rotina de venda e controle de estoque. Possibilita o cliente organizar comissionamento de seus funcionários que são vinculados a cada venda realizada.

A implementação de um sistema em uma loja oferece várias vantagens que podem melhorar significativamente a eficiência operacional, a experiência do cliente e a gestão do negócio.

Os sistemas podem automatizar tarefas rotineiras, como gerenciamento de estoque, processamento de transações e controle de estoque. Isso reduz a carga de trabalho manual, minimiza erros e libera tempo para atividades mais estratégicas, sem contar que um sistema de gestão de estoque integrado permite um controle regado.

2. Materiais e métodos

Para encaminhar o desenvolvimento do projeto, de início realizamos a coleta dos requisitos para então poder formalizar o conceito do produto. Tendo como concluído essa etapa, partimos para a criação da modelagem do sistema, usando o Visual Paradigm para realizar o modelo relacional. Após discussão entre os envolvidos e aprovação da estrutura, foi gerado todo o script para gerar o banco de dados. Como requisito do sistema para o banco de dados foi feito uso da linguagem PostgreSQL, como ferramenta SGBD o DBeaver. Para o ambiente de programação, foi utilizado o Eclipse com o projeto gerado pelo Spring Boot, e uso do JPA para a comunicação com o banco de dados.

Realizamos os diagramas do sistema usando Visual Paradigm, conforme os requisitos atribuídos para esse sistema, usando o Trello para melhor organização e divisão das tarefas para melhor desempenho no desenvolvimento.

3. Desenvolvimento

3.1. Engenharia de Software

Para dar início, fizemos a junção dos requisitos e formalizamos já as etapas para o desenvolvimento para assim então organizar eles no Trello, definindo já as tarefas e seus respectivos responsáveis.

Requisitos funcionais do sistema:

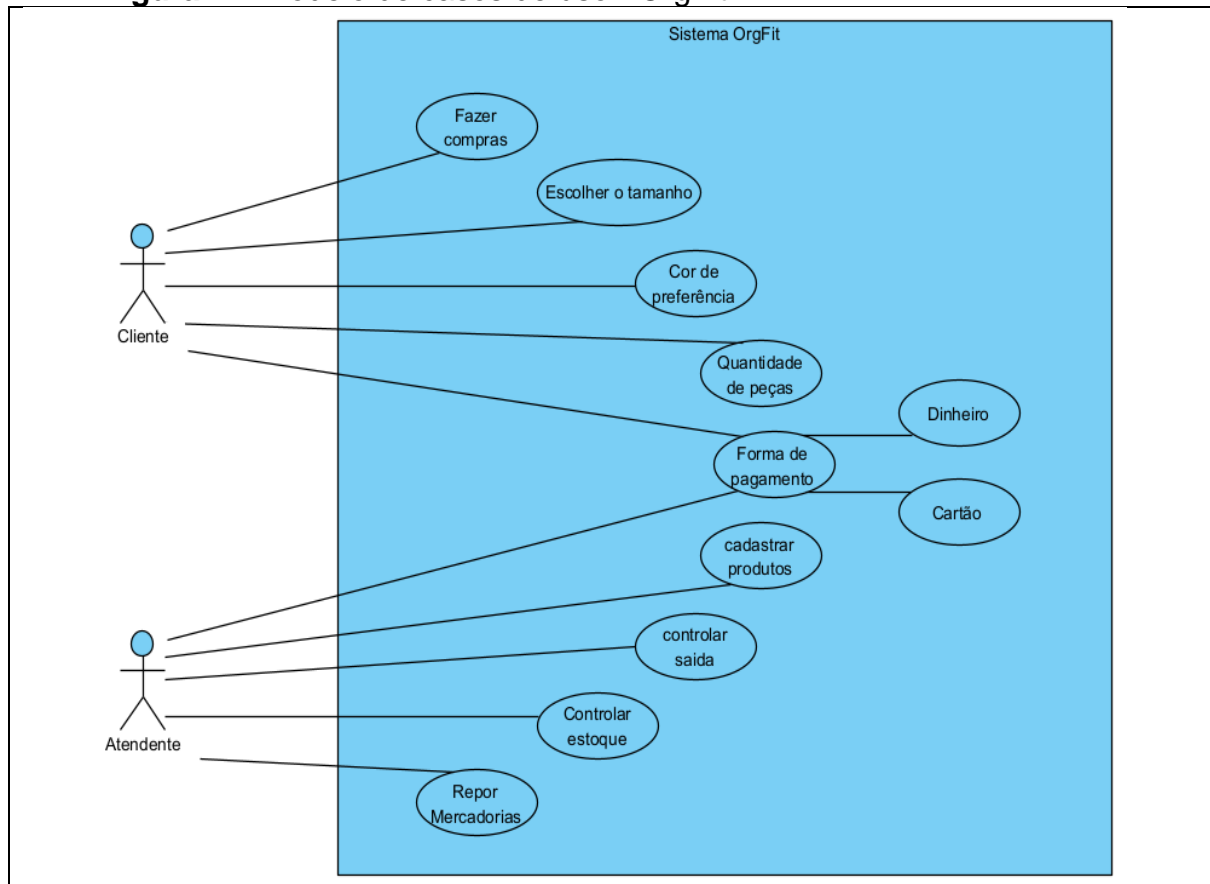
- Cadastro de Produtos.
- Gestão de Estoque.
- Sistema de Pagamento.
- Implementar um sistema de processamento de pagamentos, aceitando diferentes métodos, como cartões de crédito e débito.
- Gerenciamento de vendas.
- Relatórios de Vendas
- Cadastro de clientes.
- Cadastro de fornecedores.
- Cadastro de funcionários.

Requisitos não funcionais do sistema:

- Garantir que o sistema seja seguro.
- Banco de dados em PostgreSQL.
- O sistema deve ser capaz de lidar com picos de tráfego, durante promoções e períodos de alta demanda.
- A interface do usuário deve ser intuitiva.
- Compatibilidade com desktop e mobile.
- Backup e Recuperação.
- Legislação e Conformidade.
- Certificar de que a loja esteja conforme às regulamentações.

3.2. Diagrama de casos de uso

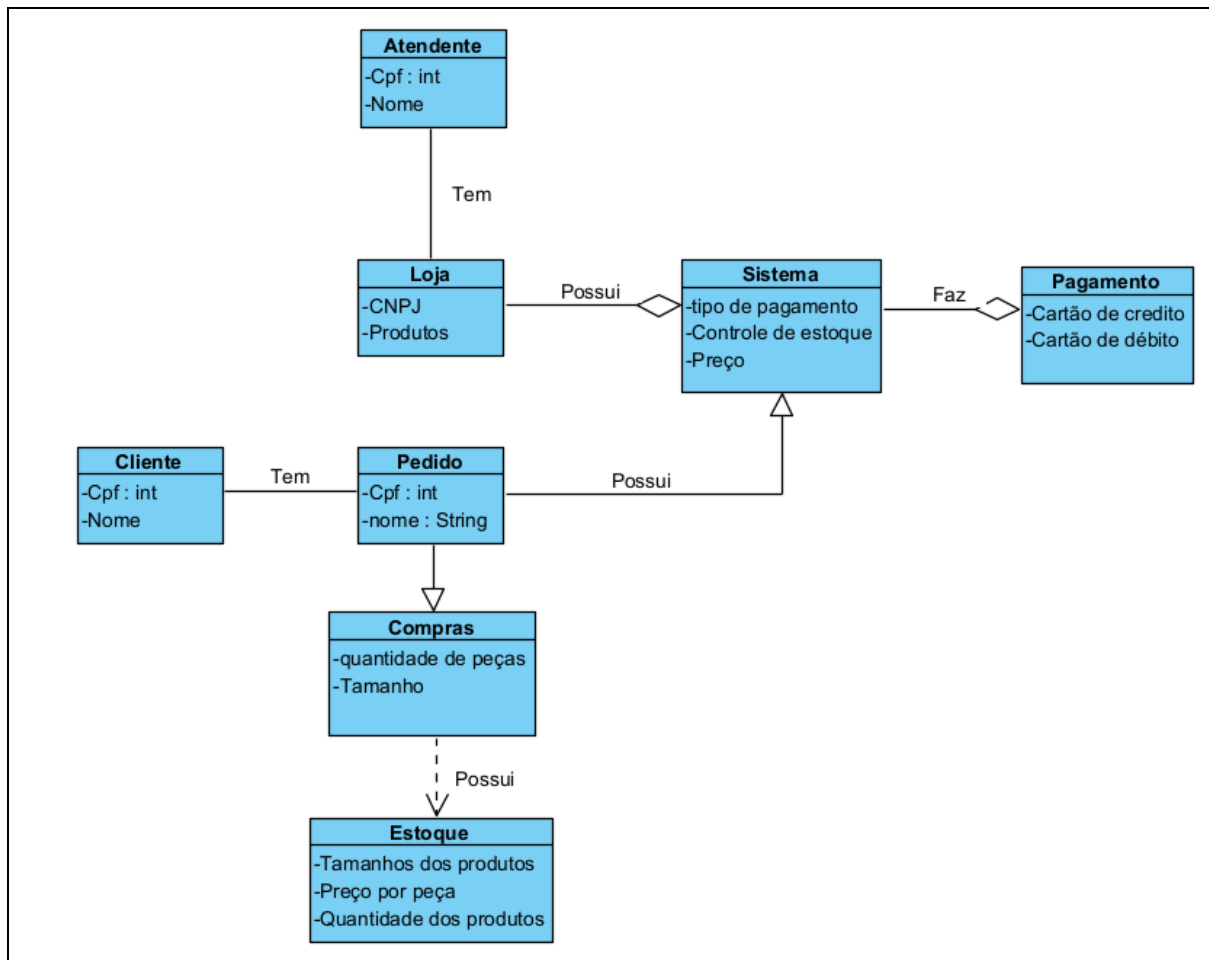
Figura 1 – Modelo de casos de uso - OrgFit



Fonte: Emanuel Petry(2023)

3.3. Diagrama de classes.

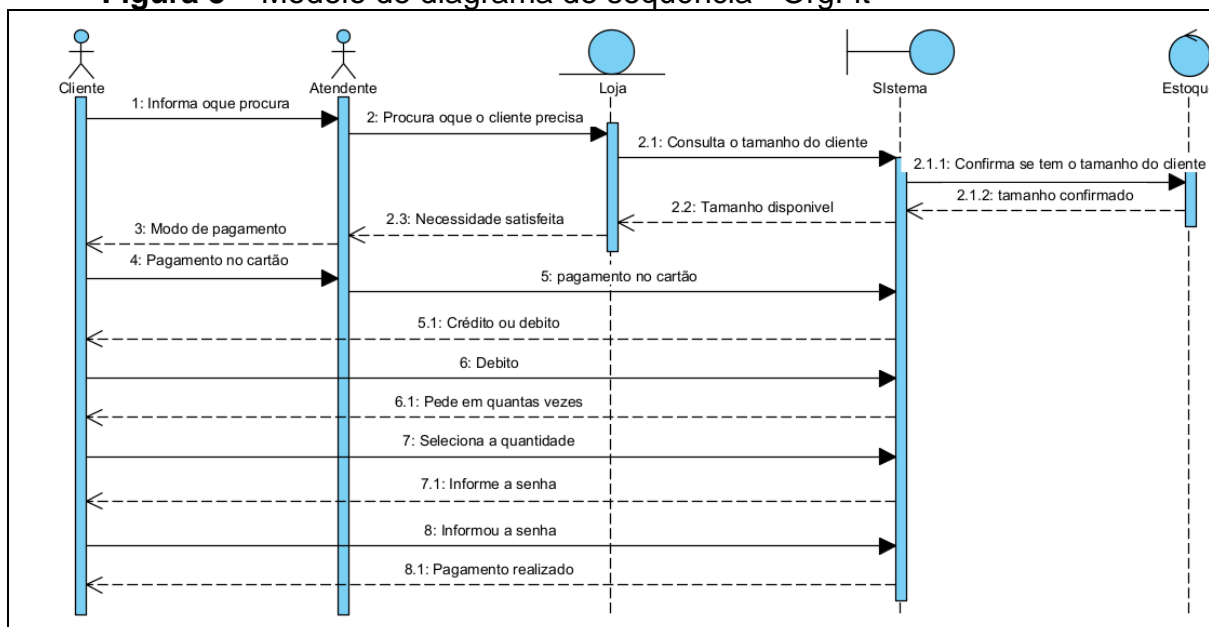
Figura 2 – Modelo de diagrama de classes - OrgFit



Fonte: Emanuel Petry(2023)

3.4. Diagrama de sequência.

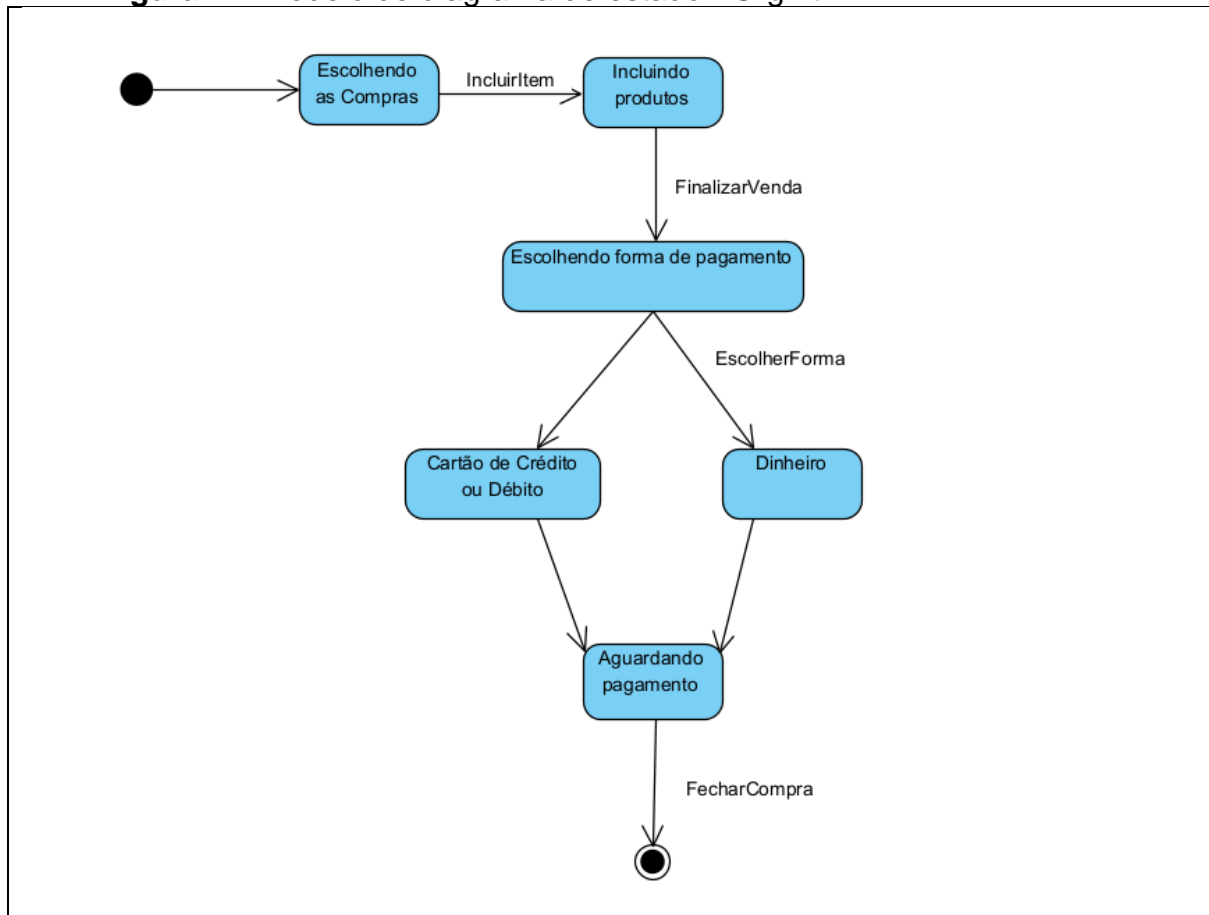
Figura 3 – Modelo de diagrama de sequência - OrgFit



Fonte: Emanuel Petry(2023)

3.5. Diagrama de estado.

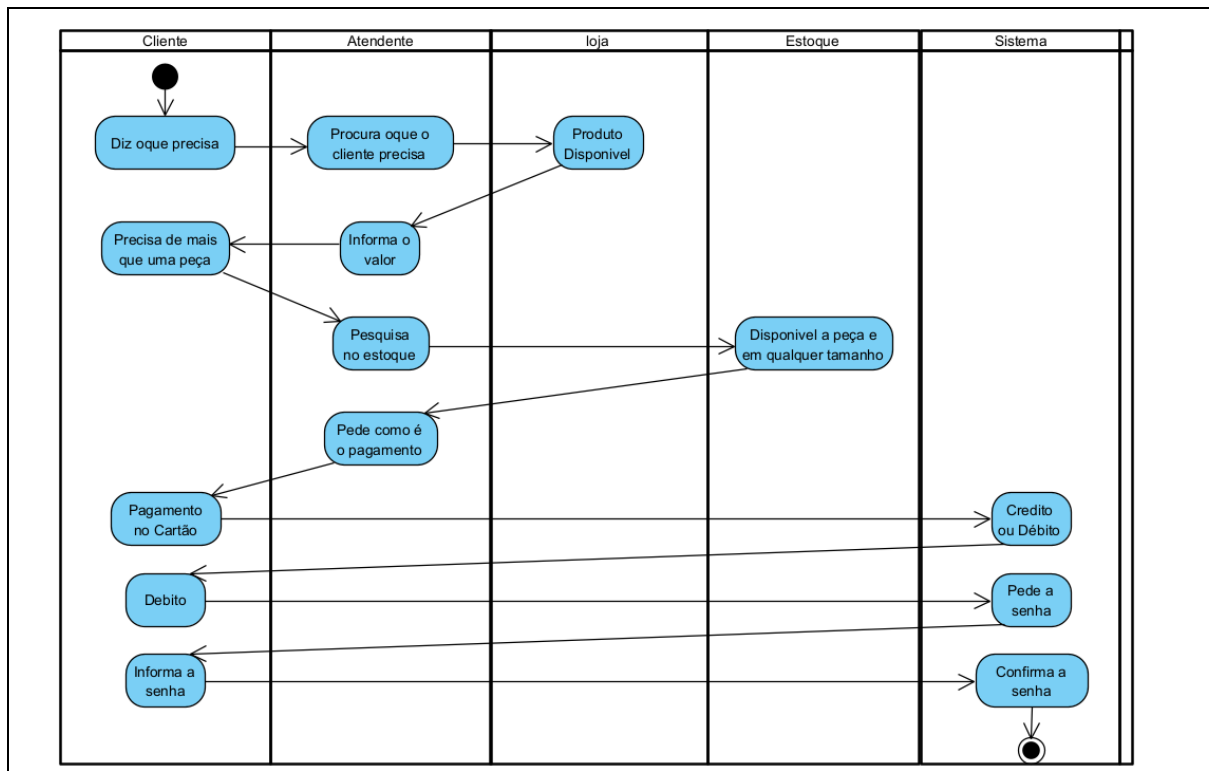
Figura 4 – Modelo de diagrama de estado - OrgFit



Fonte: Emanuel Petry(2023)

3.6. Diagrama de atividades.

Figura 7 - Modelo de diagrama de atividades - OrgFit

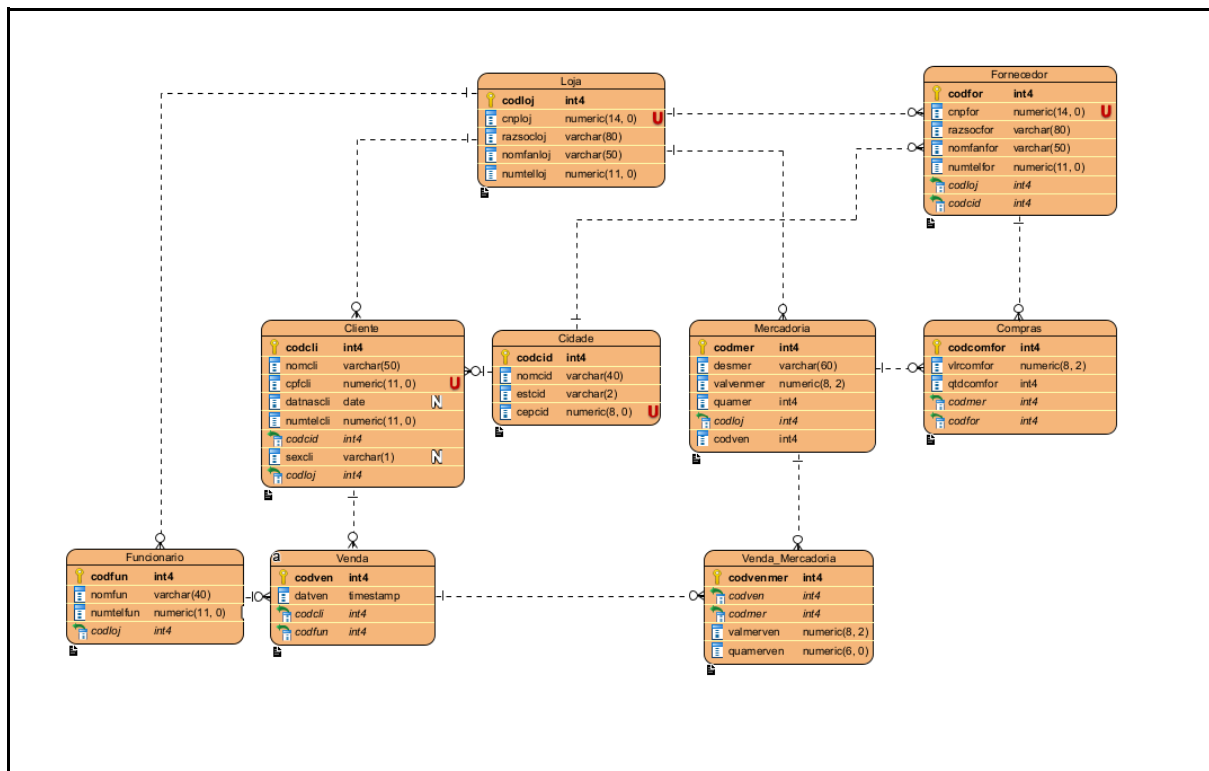


Fonte: Emanuel Petry(2023)

3.7. Banco de dados

Formalizado o Modelo Relacional, geramos os scripts para criar todas as tabelas no banco. Para assim, posteriormente realizar todos os scripts para controle e administração do banco de dados.

Figura 6 – Modelo Relacional



Fonte: Alysson Antonietti (2023)

3.8. Triggers

Implementação de dois gatilhos(triggers) para evitar mau uso do banco. A figura 7 apresenta um gatilho que não permite o cadastro de produtos com valor de venda igual a 0. Já na figura 8, apresenta um gatilho que bloqueia a venda de itens sem ter disponibilidade em estoque.

Figura 7 – Gatilho que controla valor de venda

```

Create or replace function controla_valor()
returns trigger
as
$body$
begin
    if (new.valvenmer <= 0) then
        raise exception 'Mercadoria % não pode ter preço de venda
        menor ou igual a %. Verifique!', new.desmer, new.valvenmer;
    end if;
    return new;
end
$body$
language plpgsql;

create or replace trigger mercadoria_bf_tg
before insert or update
on mercadoria
for each row

```

```
execute procedure controla_valor();
```

Fonte: Alysson Antonietti (2023)

Figura 8 – Gatilho que controla quantidade de estoque

```
create or replace function controla_estoque()
returns trigger
as
$body$
declare
    quantidade numeric;
begin
    select quamer into quantidade from mercadoria
    where codmer = new.codmer;
    if (new.quamerven >= quantidade) then
        raise exception 'Estoque insuficiente!';
    end if;
    return new;
end
$body$
language plpgsql;

create trigger venda_mercadoria_bf_tg
before insert
on venda_mercadoria
for each row
execute procedure controla_estoque();
```

Fonte: Alysson Antonietti (2023)

3.9. Stored Procedures

Implementação de procedimentos armazenados (stored procedures) para tarefas de calcular descontos de preço avista, demonstrado na figura 9. Também para calcular o preço de custo da mercadoria na figura 10.

Figura 9 – Calcula preço avista

```
create or replace function preco_avista(valor numeric)
returns numeric
as
$body$
declare
    valor_avista numeric := 0;
begin
    if (valor >= 100 and valor < 300) then
```



```

        valor_avista := valor * 0.97; -- 3%
    elsif (valor >= 300 and valor < 800) then
        valor_avista := valor * 0.95; -- 5%
    elsif (valor >= 800) then
        valor_avista := valor * 0.93; -- 7%
    else valor_avista := valor; -- sem desconto
    end if;
    return valor_avista;
end
$body$
language plpgsql;

```

Fonte: Alysson Antonietti (2023)

Figura 10 – Calcula preço de custo

```

create or replace function preco_custo(valor_total numeric, quantidade
integer)
returns numeric
as
$body$
begin
    return valor_total / quantidade;
end
$body$
language plpgsql;

```

Fonte: Alysson Antonietti (2023)

3.10. Políticas de acesso

Mapeamento de grupos de permissão para cada função no sistema. Criação de usuários e atribuição dos acessos apresentados na figura 11.

Figura 11 – Política de acesso e usuários

```

-- Grupos de permissão
create group gerencia;
create group vendedor;
create group administrativo;

-- Usuários
create user Joao with login createrole password 'gerencia';
create user Pedro with login password 'vendedor';
create user Isabela with login password 'administrativo';

-- Definição de permissões
grant select, insert, update, delete on Cliente, cidade, compras,

```

```

fornecedor, funcionario, loja, mercadoria, venda, venda_mercadoria to
gerencia;
grant select, insert, update, delete on cliente, fornecedor,
funcionario, mercadoria, cidade, compras to administrativo;
grant select on venda, venda_mercadoria to administrativo;
grant select, insert on venda, venda_mercadoria to vendedor;
grant select on cliente to vendedor;

-- Atribuição das permissões
grant gerencia to Joao;
grant administrativo to Isabela;
grant vendedor to Pedro;

```

Fonte: Alysson Antonietti (2023)

3.11. Índices

Criados scripts de índices para todas as chaves estrangeiras e para chaves candidatas conforme a figura 12, com o propósito de melhor desempenho em consultas no banco de dados.

Figura 12 – Criação de índices

```

create index for_nomloj_sk on fornecedor(codloj);
create index for_codcid_sk on fornecedor(codcid);
create index cli_codcid_sk on cliente(codcid);
create index cli_codloj_sk on cliente(codloj);
create index fun_codloj_sk on funcionario(codloj);
create index ven_codcli_sk on venda(codcli);
create index ven_codfun_sk on venda(codfun);
create index ven_mer_codven_sk on venda_mercadoria(codven);
create index ven_mer_codmer_sk on venda_mercadoria(codmer);
create index mer_codloj_sk on mercadoria(codloj);
create index com_codmer_sk on compras(codmer);
create index com_codfor_sk on compras(codfor);

create unique index cli_cpcli_uk on cliente(cpcli);
create unique index cid_cepcid_uk on cidade(cepcid);

```

Fonte: Alysson Antonietti (2023)

3.12. Restore e Backup

Realizado script de rotina de backup lógico e físico do banco de dados. Além também do restore para garantir a integridade do backup.

Formalizado para um banco hospedado em um sistema operacional Linux. Através da ferramenta contrab configuramos as rotinas de agendamento para a

execução automática dos backups. Sendo, o backup lógico todos os dias às 20h e o backup físico todas as quintas-feiras às 23h.

3.13. Programação

Gerado projeto Spring Boot para comunicação com o banco de dados. Foi criada a estrutura de Model, Controller, Repository para cada tabela no banco. Realizados testes com a ferramenta Thunder Cliente do Visual Studio. Na figura 13 é apresentado a interface web que permite o cadastro, edição, consulta e exclusão de mercadorias.

Figura 13 – Interface web



Cadastro de Mercadoria

CÓDIGO

DESCRIÇÃO DA MERCADORIA

Informe a descrição

PREÇO DE VENDA

ESTOQUE

CÓDIGO LOJA

Salvar Novo Pesquisar Excluir

Fonte: Alysson Antonietti (2023)

4. Conclusão

O artigo compreende o desenvolvimento de um sistema para uma loja de roupas de pequeno porte. Com base nos requisitos, modelamos o banco de dados e os scripts de controle e gestão do mesmo. Também foram construídos os diagramas de classe, sequência, de estado, caso de uso, e atividade.

Realizada a programação em java com uso do Spring Boot para fazer a criação de uma interface web que possibilita interagir com o banco de dados. Dessa forma, concluindo o desenvolvimento do sistema OrgFit.

Referências

DRKUSIC, Emil. **Learn SQL Triggers**. SQLShack, publicado em 20 mar. 2020. Disponível em: <https://www.sqlshack.com/learn-sql-sql-triggers/>. Acesso em: 16 nov. 2023.

LUIZ, Henrique. **Entendendo o crontab**. Totvs Developers, publicado em 9 abr. 2020. Disponível em: <https://medium.com/totvsdevelopers/entendendo-o-crontab-607bc9f00ed3>. Acesso em: 19 nov. 2023.

POSTGRESQL. **Backup and Restore**. PostgreSQL Documentation, 8.1, Disponível em: <https://www.postgresql.org/docs/8.1/backup.html>. Acesso em: 16 nov. 2023.

SPRING. **Building Java Projects with Maven**. Disponível em: <https://spring.io/guides/gs/maven/>. Acesso em: 22 nov. 2023.