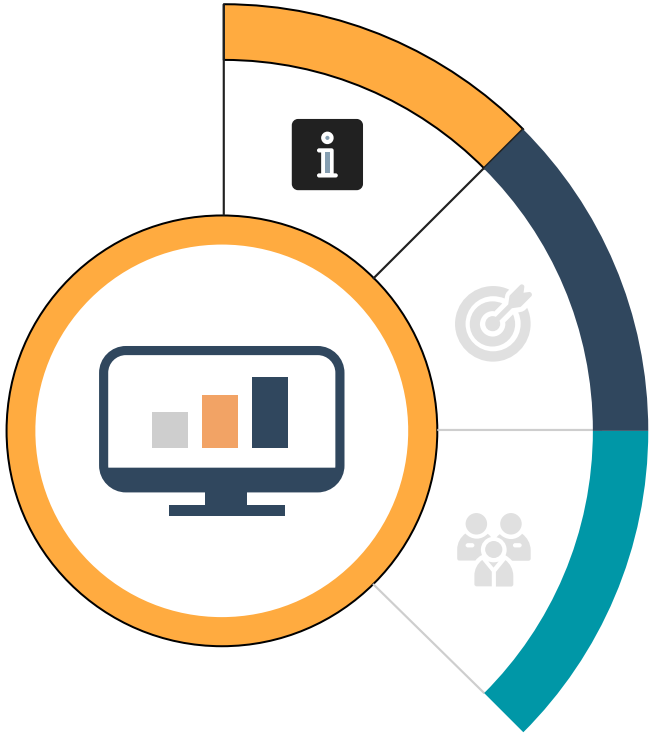
Market Manager

Ana Clara
Aniel Melo
Hugo Diniz

Market Manager



01

Requisitos.

02

Público alvo.

03

Arquitetura.

REQUISITOS

Controle de usuários

Controle de vendas

Controle de produtos

Controle de desconto

REQUISITOS - Controle de produtos

DOC - Visão

Necessidade 2	**Benefício**
Controle de Produtos	**Crítico**

Id Func.	**Descrição das Funcionalidades/atores envolvidos**
F2.1	Inclusão de Novos Produtos
	Estoquista– Fornece os dados do produto solicitados pelo sistema (nome, preço, quantidade, marca, etc.).
F2.2	Pesquisa / Listagem de Produtos
	Estoquista – Fornece o nome do produto (ou outros dados) e o sistema retorna uma lista de possíveis elementos encontrados.
F2.3	Alteração de dados do Produto
	Estoquista – Pesquisa um produto existente e fornece dados atualizados para este.
F2.4	Exclusão de Produto (Desativação)
	Estoquista – Pesquisa um produto existente e solicita ao sistema a exclusão deste. O produto não é apagado do banco de dados, apenas é marcado como INATIVO.

Market Manager

01

Requisitos.

02

Público alvo.

03

Arquitetura

Market Manager

01

Requisitos.

02

Público alvo.

03

Arquitetura.

Tecnologias utilizadas

Spring Boot
Framework web

Java 21
Linguagem de programação

Keycloak
Sistema de autenticação e autorização

Docker
Ferramenta para criar containers

Docker-compose
Ferramenta para orquestrar containers

Postgresql
Banco de dados

Tecnologias utilizadas

Spring Boot
Framework web

Java 21
Linguagem de programação

Keycloak
Sistema de autenticação e autorização

Docker
Ferramenta para criar containers

Docker-compose
Ferramenta para orquestrar containers

Postgresql
Banco de dados

Tecnologias utilizadas

Spring Boot
Framework web

Java 21
Linguagem de programação

Keycloak
Sistema de autenticação e autorização

Docker
Ferramenta para criar containers

Docker-compose
Ferramenta para orquestrar containers

Postgresql
Banco de dados

Tecnologias utilizadas

Spring Boot
Framework web

Java 21
Linguagem de programação

Keycloak
Sistema de autenticação e autorização

Docker
Ferramenta para criar containers

Docker-compose
Ferramenta para orquestrar containers

Postgresql
Banco de dados

Tecnologias utilizadas

Spring Boot
Framework web

Java 21
Linguagem de programação

Keycloak
Sistema de autenticação e autorização

Docker
Ferramenta para criar containers

Docker-compose
Ferramenta para orquestrar containers

Postgresql
Banco de dados

Tecnologias utilizadas

Spring Boot
Framework web

Java 21
Linguagem de programação

Keycloak
Sistema de autenticação e autorização

Docker
Ferramenta para criar containers

Docker-compose
Ferramenta para orquestrar containers

Postgresql
Banco de dados

Diagramas

Componentes
Representa módulos e suas dependências no sistema.

Sequencial
Exibe a interação entre objetos ao longo do tempo.

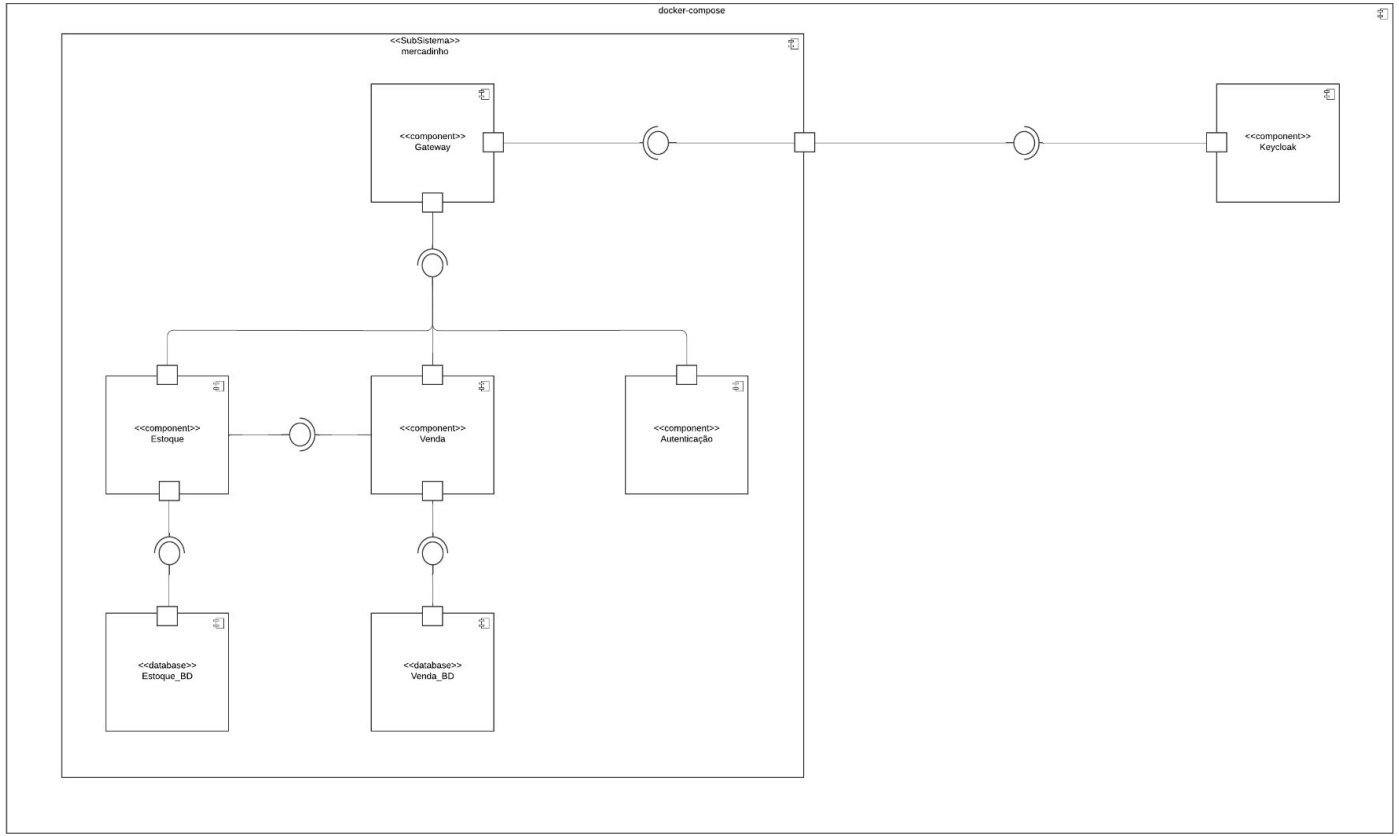
Atividade
Mostra o fluxo de atividades e decisões.

Conceitual
Planeja a estrutura de dados abstrata.

Classe
Define a estrutura e relacionamento de classes.

Caso de uso
Ilustra a interação do usuário com o sistema.

COMPONENTES



Diagramas

Componentes
Representa módulos e suas dependências no sistema.

Sequencial
Exibe a interação entre objetos ao longo do tempo.

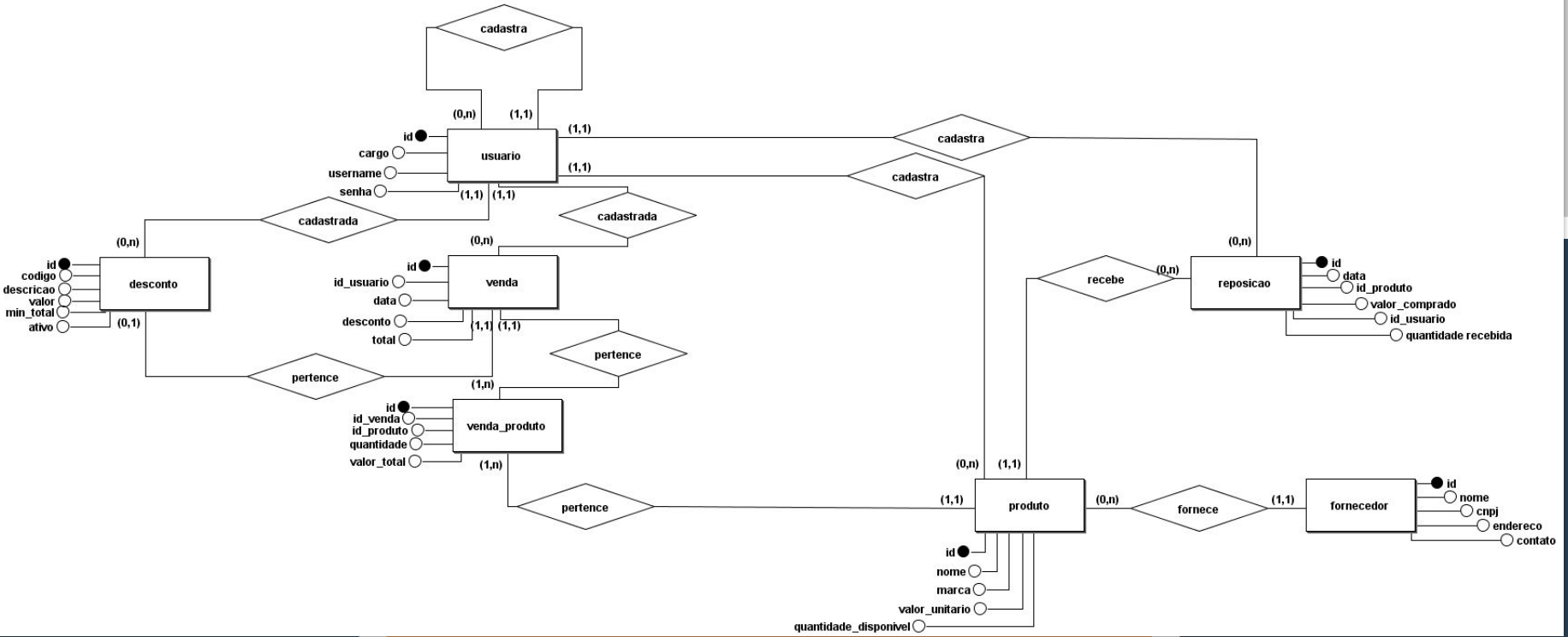
Atividade
Mostra o fluxo de atividades e decisões.

Conceitual
Planeja a estrutura de dados abstrata.

Classe
Define a estrutura e relacionamento de classes.

Caso de uso
Ilustra a interação do usuário com o sistema.

CONCEITUAL


cadastra
(0,n)
(1,1)
id
cargo
username
senha
usuario
(1,1)
(1,1)
(1,1)
cadastra
cadastra
(1,1)
(1,1)
cadastrada
cadastrada
(0,n)
(0,n)
id
codigo
descricao
valor
min_total
ativo
desconto
(0,1)
id
id_usuario
data
desconto
total
venda
(1,1)
(1,1)
pertence
pertence
(1,n)
id
id_venda
id_produto
quantidade
valor_total
venda_produto
(1,n)
pertence
(0,n)
(1,1)
recebe
(0,n)
reposicao
id
data
id_produto
valor_comprado
id_usuario
quantidade recebida
(1,1)
(0,n)
produto
(0,n)
fornece
(1,1)
fornecedor
id
nome
cnpj
endereco
contato
id
nome
marca
valor_unitario
quantidade_disponivel

Diagramas

Componentes
Representa módulos e suas dependências no sistema.

Sequencial
Exibe a interação entre objetos ao longo do tempo.

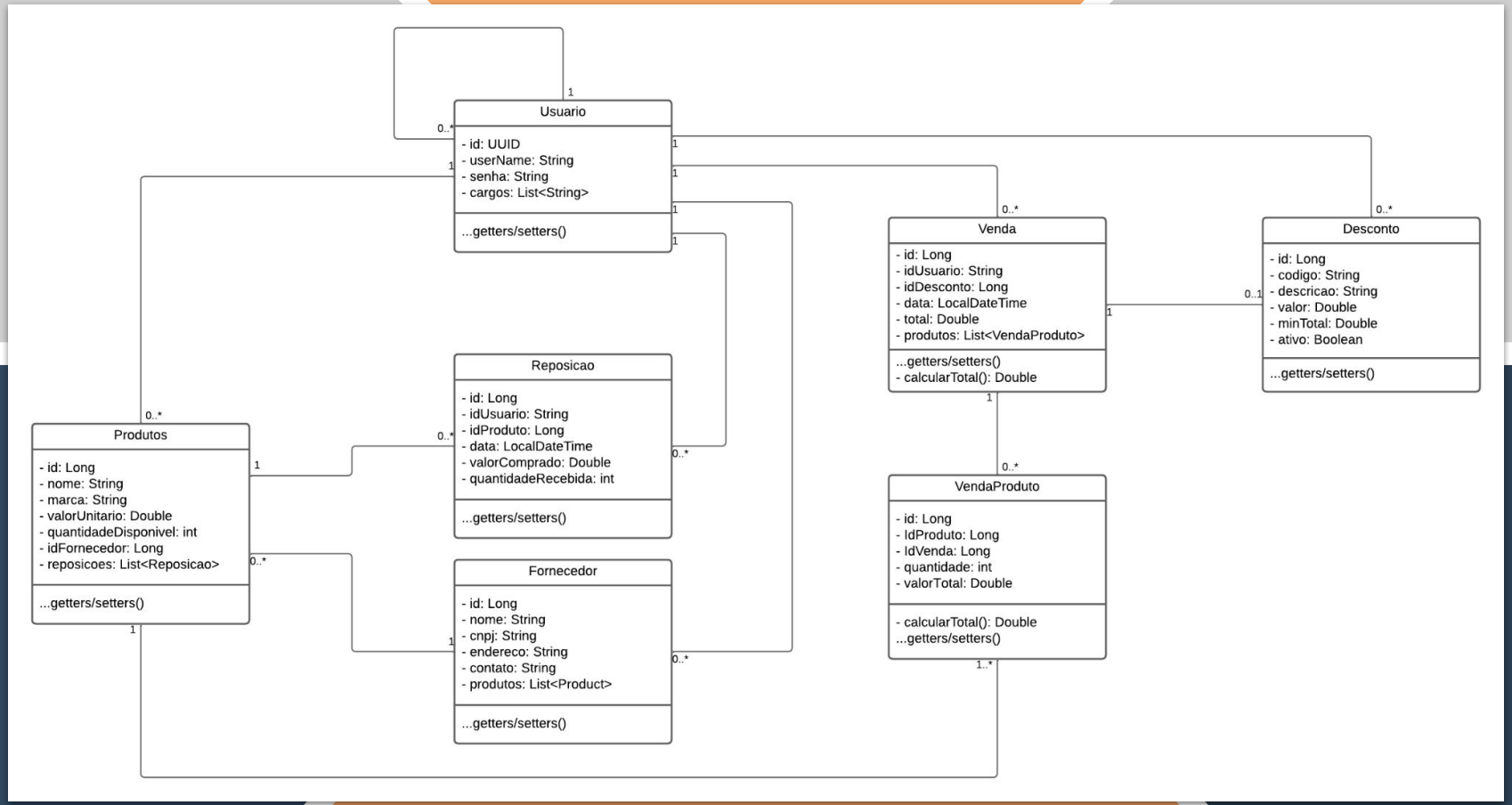
Atividade
Mostra o fluxo de atividades e decisões.

Conceitual
Planeja a estrutura de dados abstrata.

Classe
Define a estrutura e relacionamento de classes.

Caso de uso
Ilustra a interação do usuário com o sistema.

CLASSE


Usuario
- id: UUID
- userName: String
- senha: String
- cargos: List<String>
...getters/setters()
Venda
- id: Long
- idUsuario: String
- idDesconto: Long
- data: LocalDateTime
- total: Double
- produtos: List<VendaProduto>
...getters/setters()
- calcularTotal(): Double
Desconto
- id: Long
- codigo: String
- descricao: String
- valor: Double
- minTotal: Double
- ativo: Boolean
...getters/setters()
Produtos
- id: Long
- nome: String
- marca: String
- valorUnitario: Double
- quantidadeDisponivel: int
- idFornecedor: Long
- reposicoes: List<Reposicao>
...getters/setters()
Reposicao
- id: Long
- idUsuario: String
- idProduto: Long
- data: LocalDateTime
- valorComprado: Double
- quantidadeRecebida: int
...getters/setters()
Fornecedor
- id: Long
- nome: String
- cnpj: String
- endereco: String
- contato: String
- produtos: List<Product>
...getters/setters()
VendaProduto
- id: Long
- IdProduto: Long
- IdVenda: Long
- quantidade: int
- valorTotal: Double
- calcularTotal(): Double
...getters/setters()

Diagramas

Componentes
Representa módulos e suas dependências no sistema.

Sequencial
Exibe a interação entre objetos ao longo do tempo.

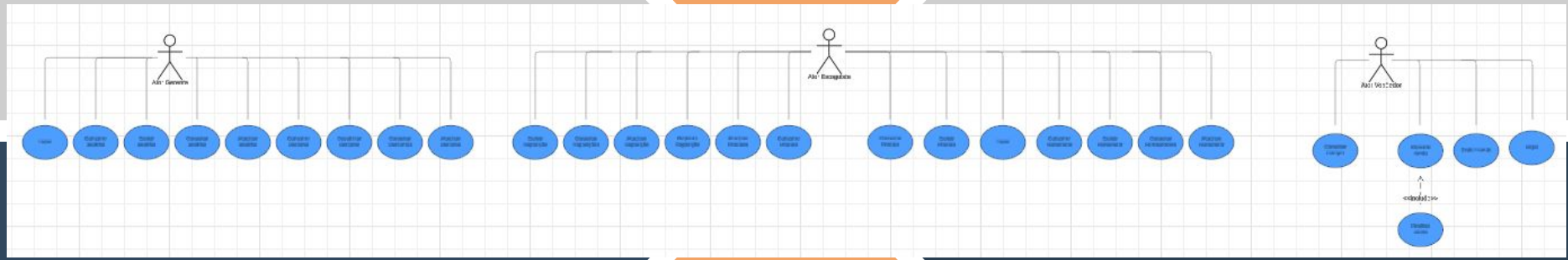
Atividade
Mostra o fluxo de atividades e decisões.

Conceitual
Planeja a estrutura de dados abstrata.

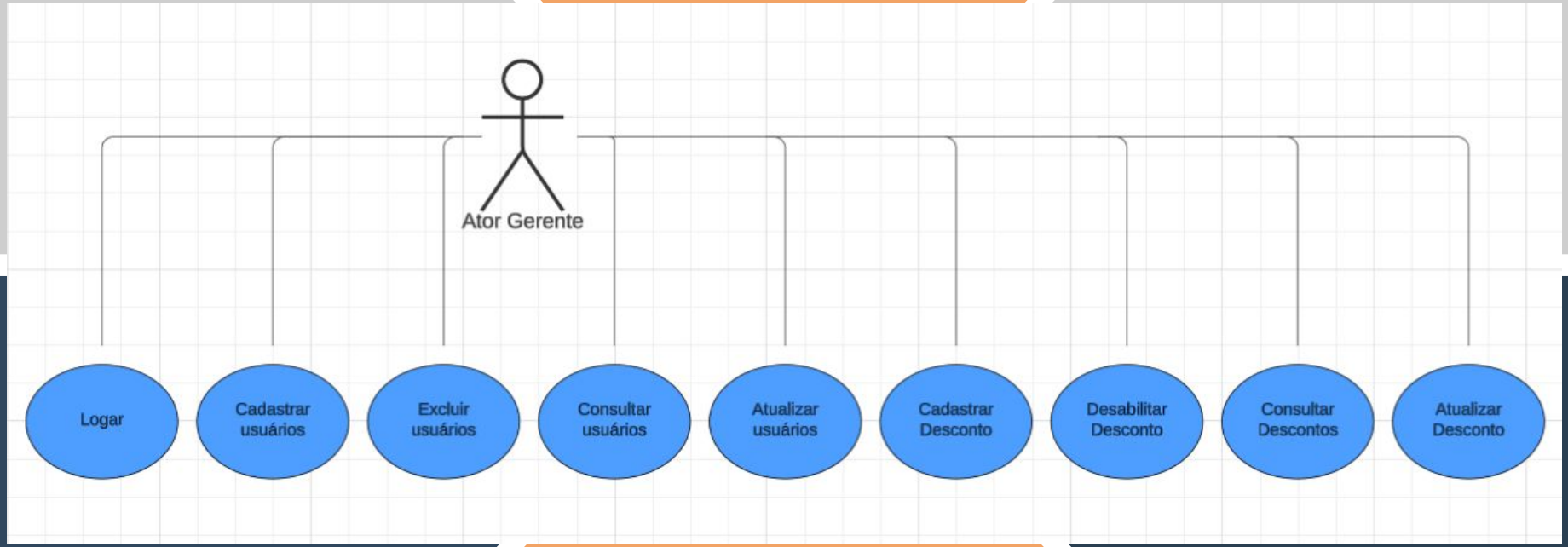
Classe
Define a estrutura e relacionamento de classes.

Caso de uso
Ilustra a interação do usuário com o sistema.

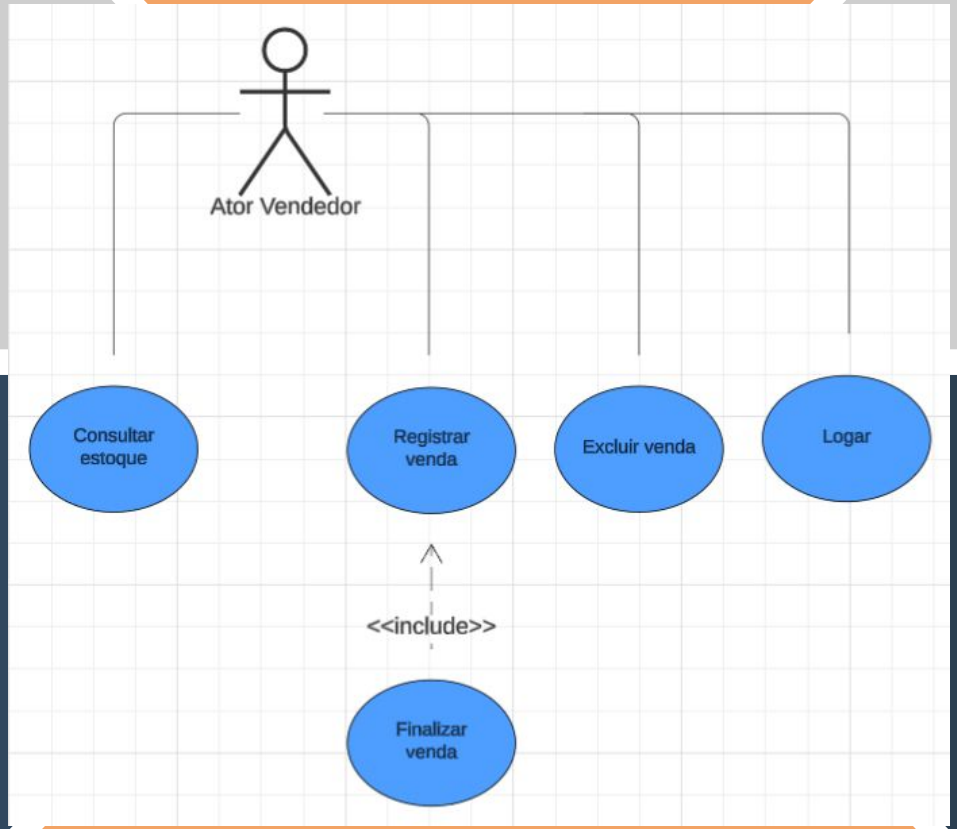
CASO DE USO



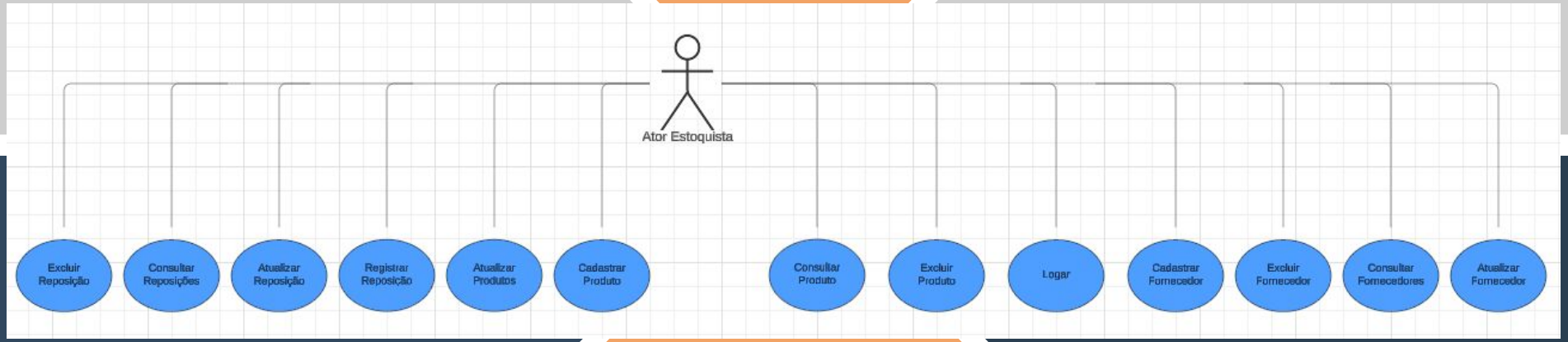
CASO DE USO


Ator Gerente
Logar
Cadastrar usuários
Excluir usuários
Consultar usuários
Atualizar usuários
Cadastrar Desconto
Desabilitar Desconto
Consultar Descontos
Atualizar Desconto

CASO DE USO


Ator Vendedor
Consultar estoque
Registrar venda
Excluir venda
Logar
<<include>>
Finalizar venda

CASO DE USO


Ator Estoquista
Excluir Reposição
Consultar Reposições
Atualizar Reposição
Registrar Reposição
Atualizar Produtos
Cadastrar Produto
Consultar Produto
Excluir Produto
Logar
Cadastrar Fornecedor
Excluir Fornecedor
Consultar Fornecedores
Atualizar Fornecedor

Diagramas

Componentes
Representa módulos e suas dependências no sistema.

Sequencial
Exibe a interação entre objetos ao longo do tempo.

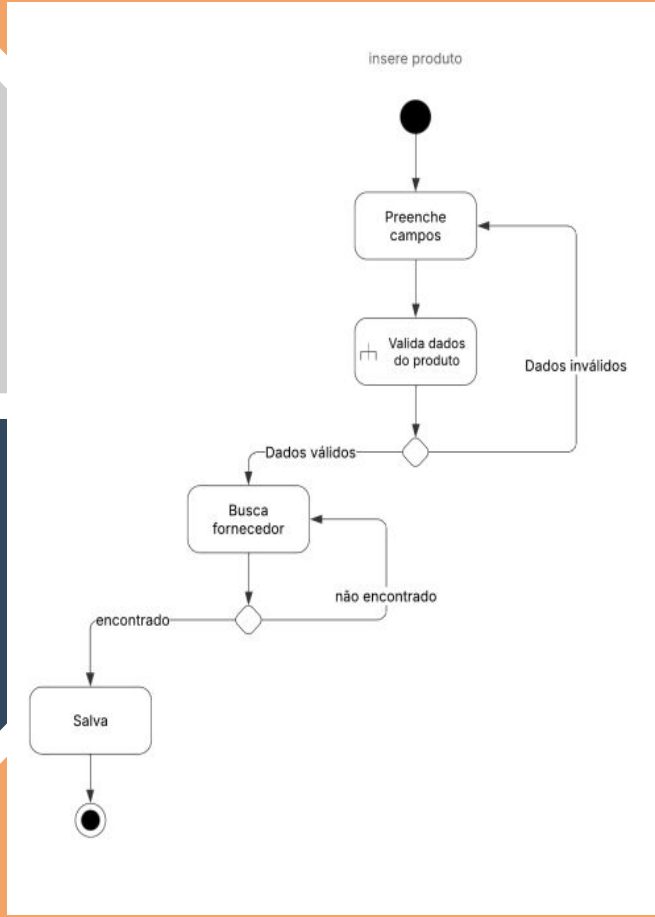
Atividade
Mostra o fluxo de atividades e decisões.

Conceitual
Planeja a estrutura de dados abstrata.

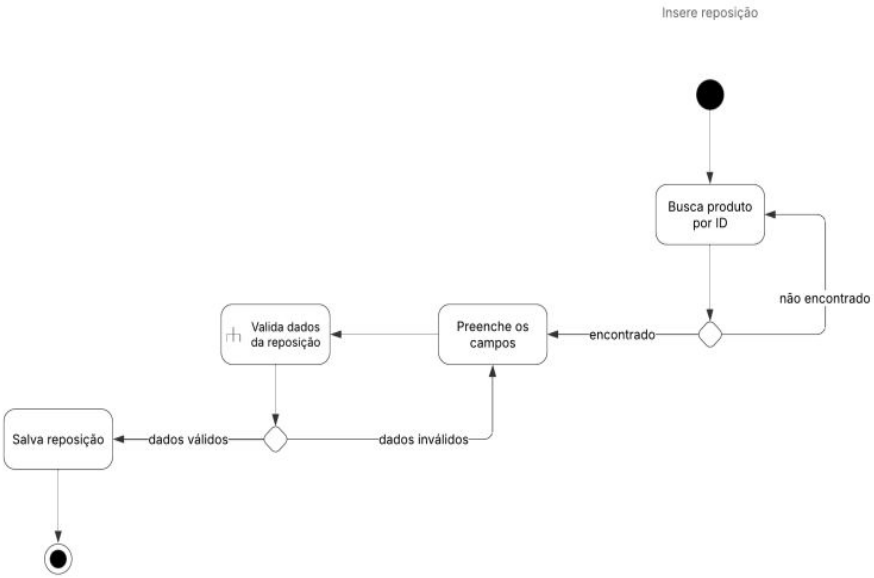
Classe
Define a estrutura e relacionamento de classes.

Caso de uso
Ilustra a interação do usuário com o sistema.

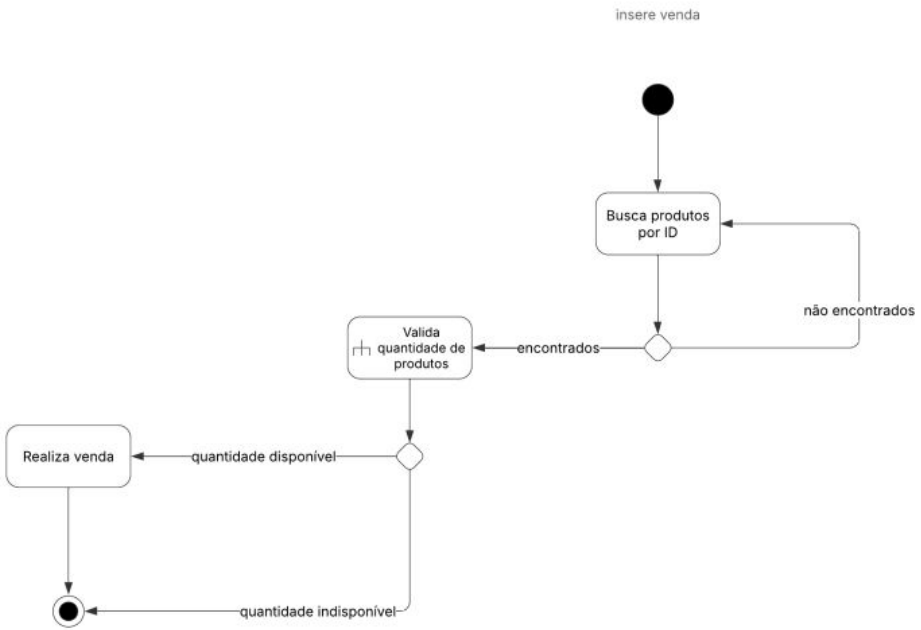
ATIVIDADE


insere produto
Preenche campos
Valida dados do produto
Dados inválidos
Dados válidos
Busca fornecedor
não encontrado
encontrado
Salva

ATIVIDADE


Insere reposição
Busca produto por ID
não encontrado
encontrado
Preenche os campos
Valida dados da reposição
dados inválidos
dados válidos
Salva reposição

ATIVIDADE


insere venda
Busca produtos por ID
não encontrados
Valida quantidade de produtos
encontrados
Realiza venda
quantidade disponível
quantidade indisponível

Diagramas

Componentes
Representa módulos e suas dependências no sistema.

Sequencial
Exibe a interação entre objetos ao longo do tempo.

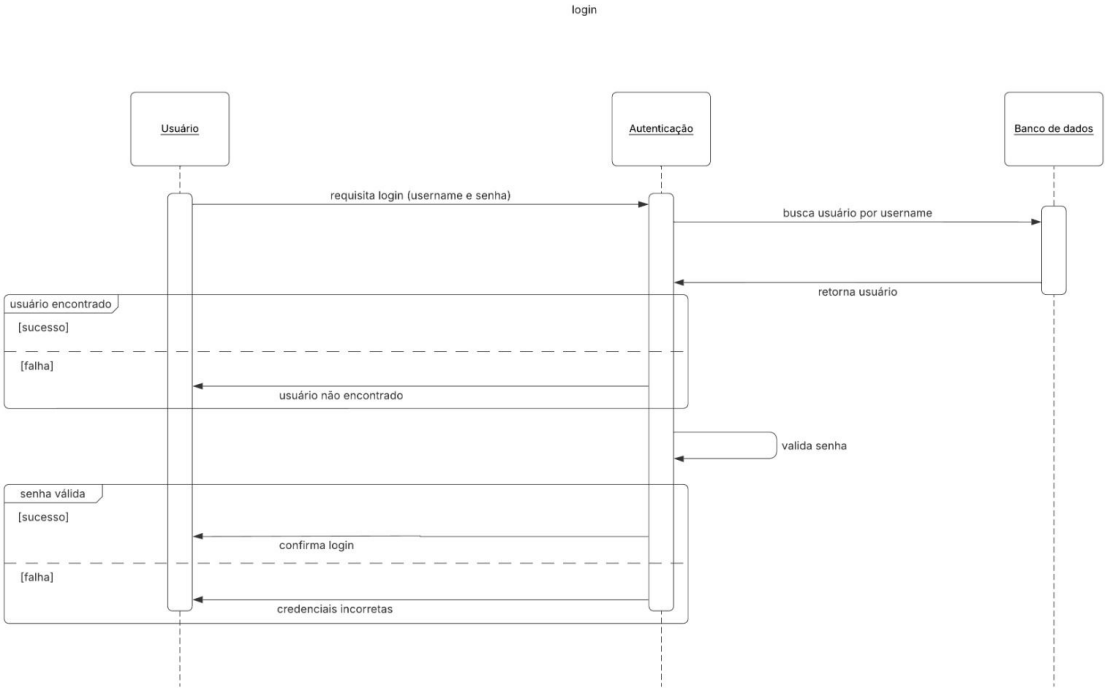
Atividade
Mostra o fluxo de atividades e decisões.

Sequencial
Exibe a interação entre objetos ao longo do tempo.

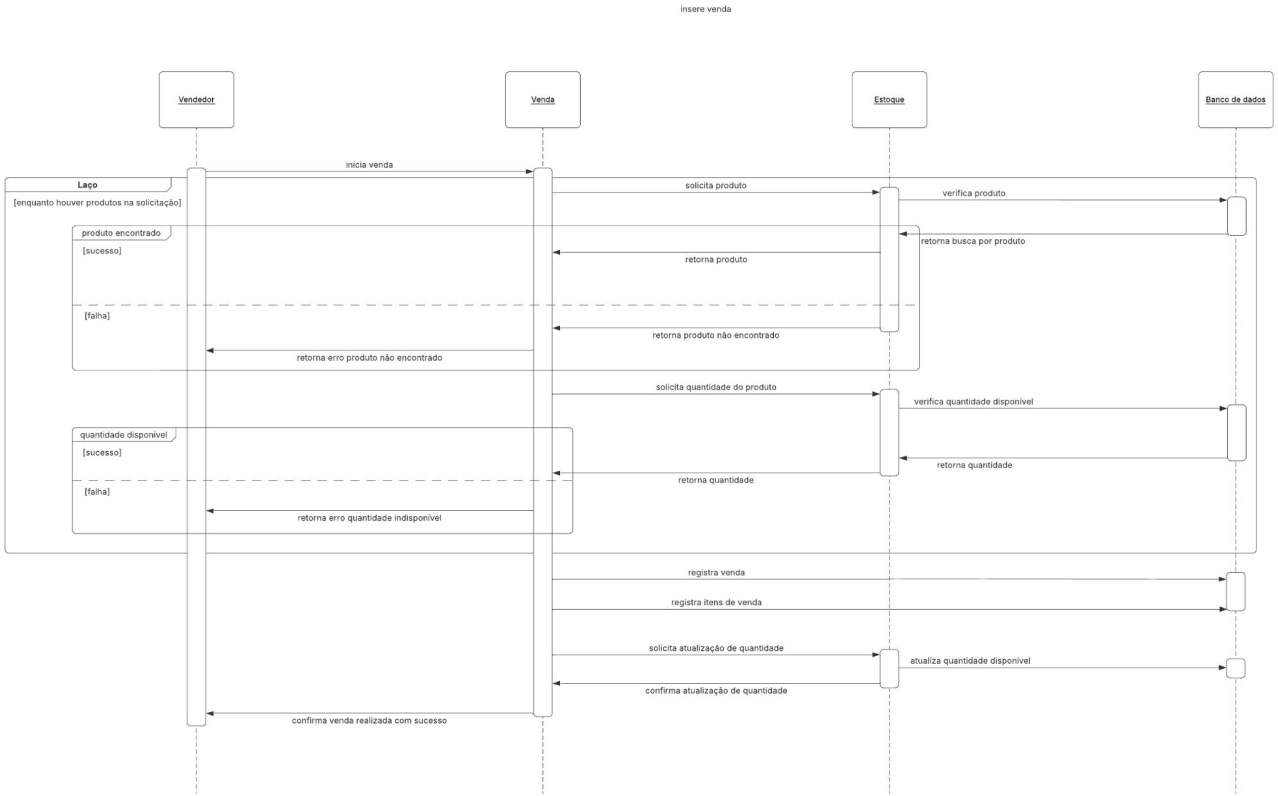
Atividade
Mostra o fluxo de atividades e decisões.

Caso de uso
Ilustra a interação do usuário com o sistema.

SEQUENCIAL


login
Usuário
Autenticação
Banco de dados
requisita login (username e senha)
busca usuário por username
retorna usuário
usuário encontrado
[sucesso]
[falha]
usuário não encontrado
valida senha
senha válida
[sucesso]
confirma login
[falha]
credenciais incorretas

SEQUENCIAL


insere venda
Vendedor
Venda
Estoque
Banco de dados
inicia venda
Laço
[enquanto houver produtos na solicitação]
solicita produto
verifica produto
produto encontrado
[sucesso]
retorna busca por produto
retorna produto
[falha]
retorna produto não encontrado
retorna erro produto não encontrado
solicita quantidade do produto
verifica quantidade disponível
quantidade disponível
[sucesso]
retorna quantidade
retorna quantidade
[falha]
retorna erro quantidade indisponível
registra venda
registra itens de venda
solicita atualização de quantidade
atualiza quantidade disponível
confirma atualização de quantidade
confirma venda realizada com sucesso

CONSISTÊNCIA DE DIAGRAMAS

CheckList

1. Estrutura Geral

☑ O diagrama possui classes? Caso esteja vazio e desnecessário, deve ser descartado.

☑ Nenhuma classe está vazia (sem atributos ou métodos) sem motivo justificável.

☑ Não existem classes com o mesmo identificador no diagrama.

2. Validação de Classes Abstratas e Concretas

☑ Classes abstratas possuem pelo menos um método abstrato (próprio ou herdado).

- Nosso diagrama não possui classes abstratas

☑ Classes concretas não possuem métodos abstratos (nem próprios, nem herdados não sobrescritos).

3. Atributos e Métodos

☑ Não existem atributos duplicados na mesma classe ou na hierarquia de herança.

☑ Não existem métodos com o mesmo identificador e mesmo comprimento de lista de parâmetros sem que caracterizem sobrecarga ou sobrescrição.

☑ Todos os atributos, parâmetros e retornos de métodos possuem um tipo definido.

4. Relacionamentos e Herança

☑ Não há ciclos em relacionamentos de herança.

☑ Se uma classe implementa uma interface, ela fornece implementação para todos os métodos declarados na interface.

- No nosso diagrama não possui interface

☑ Métodos abstratos herdados são sobrescritos no nível mais baixo da hierarquia de herança, a menos que se trate de um framework orientado a objetos.

- No nosso diagrama não possui métodos abstratos

CÓDIGO

OBRIGADO PELA ATENÇÃO!