Ponto de Venda

Documento de Arquitetura de Software

Versão <1.0>

Histórico da Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 18/05/22 | 1.0 | Elaboração do Documento | Jair de Lima, Thiago R. da Motta, Victor Verdan, Gabriel Figueiredo, Rodrigo Carvalho, Winne Domingues, Marcio Bedran. |
| 12/06/22 | 1.1 | Atualizado diagrama de casos de uso | Gabriel Figueiredo |
|  |  |  |  |
|  |  |  |  |

Índice Analítico

[**Introdução**](#_gjdgxs) **4**

[Finalidade](#_30j0zll) 4

[Escopo](#_3znysh7) 4

[Definições, acrônimos e abreviações](#_2et92p0) 4

[Visão Geral](#_tyjcwt) 4

[**Metas e Restrições da Arquitetura**](#_3dy6vkm) **4**

[**Suposições e Dependências**](#_1t3h5sf) **4**

[**Requisitos Arquiteturalmente Significantes**](#_4d34og8) **4**

[**Decisões, Restrições e justificativas**](#_2s8eyo1) **4**

[**Mecanismos Arquiteturais**](#_17dp8vu) **5**

[**Camadas da Arquitetura**](#_3rdcrjn) **5**

[**Visões da Arquitetura**](#_26in1rg) **6**

[Visão Lógica](#_lnxbz9) 6

[Visão Geral](#_35nkun2) 6

[Estrutura de Pacotes Significativos](#_1ksv4uv) 6

[Visão Lógica dos Dados Persistentes](#_44sinio) 7

[Casos de Uso](#_2jxsxqh) 7

[Visão de Classes Participantes (VCP)](#_q2wb158d290j) 8

[**Qualidade**](#_sfxajlfovra9) **10**

Documento de Arquitetura de Software

# Introdução

Este documento tem como objetivo descrever o documento de arquitetura do projeto Ponto de Venda. Esse projeto tem como propósito a implementação de um sistema de caixa para mercado que seja de fácil adoção e customização para poder ser adotado a diferentes tipos de negócio.

## Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

## Escopo

Este documento é voltado para o auxílio dos envolvidos no desenvolvimento do projeto Ponto de Venda, captando aspectos arquiteturais do sistema.

## Definições, acrônimos e abreviações

* MVC – Padrão de arquitetura de software que consiste em M – Modelo, V – Visualização e C – Controlador. O modelo é responsável pelas regras de negócio, visualização é responsável pela parte gráfica da aplicação e controladores pelo controle de dados e eventos.
* DAO – Padrão de acesso a dados Data Access Object que tem como objetivo separar a lógica de negócios da lógica de persistência de dados.
* CRUD – Conjunto de operações básicas de um banco de dados relacional. C – Create, R – Read, U – Update e D – Delete, ou traduzindo, criar, ler, atualizar e deletar, respectivamente.

## Visão Geral

Serão apresentados neste documento diferentes visões arquiteturais de como o sistema deverá se comportar em diferentes processos, como deverá ser implementado, justificações pelas escolhas feitas nesse projeto junto a como elas contribuem para todos os recursos.

# Metas e Restrições da Arquitetura

Algumas das restrições de requisito e de sistema terão uma relação fundamental com a arquitetura do projeto sendo elas:

* Sistema deverá ser multiplataforma
* Utilização do paradigma de desenvolvimento Orientado a Objetos;
* Padrão de estrutura MVC;
* Linguagem de programação *Python*;
* Framework *Tkinter*;
* Padrão DAO para operações CRUD;
* Banco de dados *PostgreSQL*.

# Suposições e Dependências

# Requisitos Arquiteturalmente Significantes

# Decisões, Restrições e justificativas

* Utilização da linguagem Python pela sua versatilidade e facilidade para programar, possuindo uma grande gama de bibliotecas e frameworks disponíveis.
* Utilização do framework Tkinter para desenvolvimento de interface multiplataforma. Ele já é nativo da linguagem Python, é relativamente simples, mas oferece uma grande gama de recursos e possui uma baixa curva de aprendizado.
* Utilização do padrão DAO para permitir a separação das regras de negócio das regras de acesso a banco de dados na camada Model. O motivo disso é para tornar as classes mais legíveis e permitir futuras alterações no banco de dados sem interferir nas regras de negócio.
* Escolha do banco de dados PostgreSQL definida com base na facilidade do uso e de conectar-se com o mesmo utilizando o Python, através do Psycopg2. Além disso, a facilidade de subir uma instância do mesmo e gerenciá-la utilizando o pgAdmin.
* A escolha da arquitetura MVC foi definida pela facilidade de utilização em aplicações web ou desktop, o que facilita a sua utilização tanto para um sistema *off-line* quanto para *on-line*. Sua facilidade permite respostas mais rápidas e dinâmicas entre o sistema e o usuário, o que é fundamental para a operação de forma escalável pelos usuários.

# Mecanismos Arquiteturais

**Disponibilidade**

O percentual de tempo em que o sistema deve estar disponível para utilização, incluindo interrupções planejadas como manutenção do sistema.

**Persistência**

Serviços para lidar com a leitura e gravação de dados armazenados.

**Segurança**

Fornecimento de serviços para proteger o acesso a determinadas partes do sistema e restrições em operações CRUD..

**Impressão**

Fornece facilidades para a interface com impressoras.

# Camadas da Arquitetura

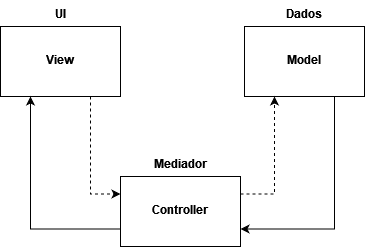
* **MVC**: Com um time de desenvolvedores especializados no front-end e outro time com foco no back-end. Com essa arquitetura separamos a interface gráfica que contará com as interfaces e as regras de negócio, do modelo que contém as entidades de persistência do sistema.
* **Arquitetura Orientada a Mensagem**: Comunicação feita de forma assíncrona entre cliente e servidor, onde o cliente (terminais dos caixas) insere uma mensagem na fila de mensagem do servidor (terminal central) e pode continuar com o seu processamento.
* **Arquitetura Cliente/Servidor:** Centralizar o processamento das transações num servidor. O servidor consistiria de 3 módulos (autenticação/criação de usuários, registro de produtos/vendas, auditoria/relatórios) e o cliente seria a interface do usuário nos terminais dos caixas.

# Visões da Arquitetura

## Visão Lógica

### Visão Geral

A visão lógica define a estrutura da arquitetura. Abaixo será especificado o padrão MVC que foi selecionado para o desenvolvimento do sistema com seus pacotes principais.



* View: Componente que contém toda a interface do projeto.
* Controller: Componente responsável por receber as requisições da *View*.
* Model: Pacote que recebe as requisições e realiza as operações relacionadas ao Banco de Dados.*.*

### Estrutura de Pacotes Significativos

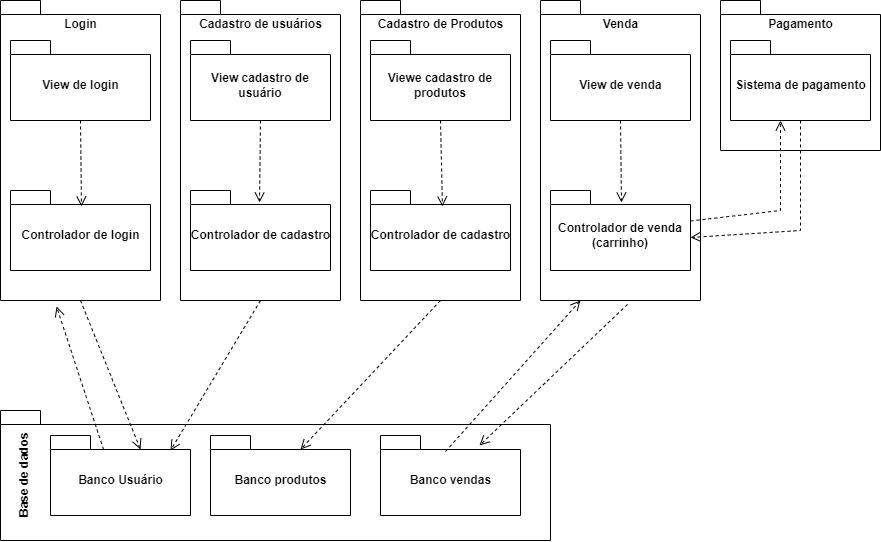
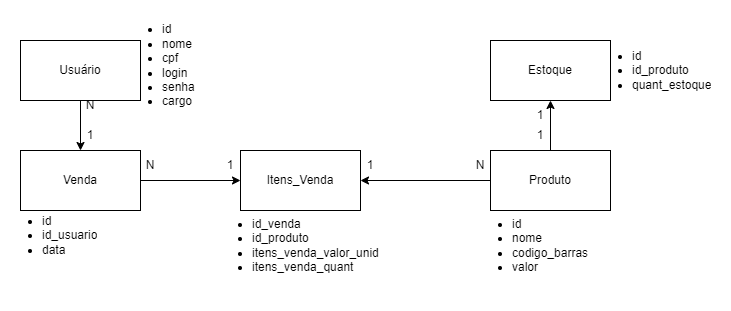


Diagrama de Classe geral do sistema

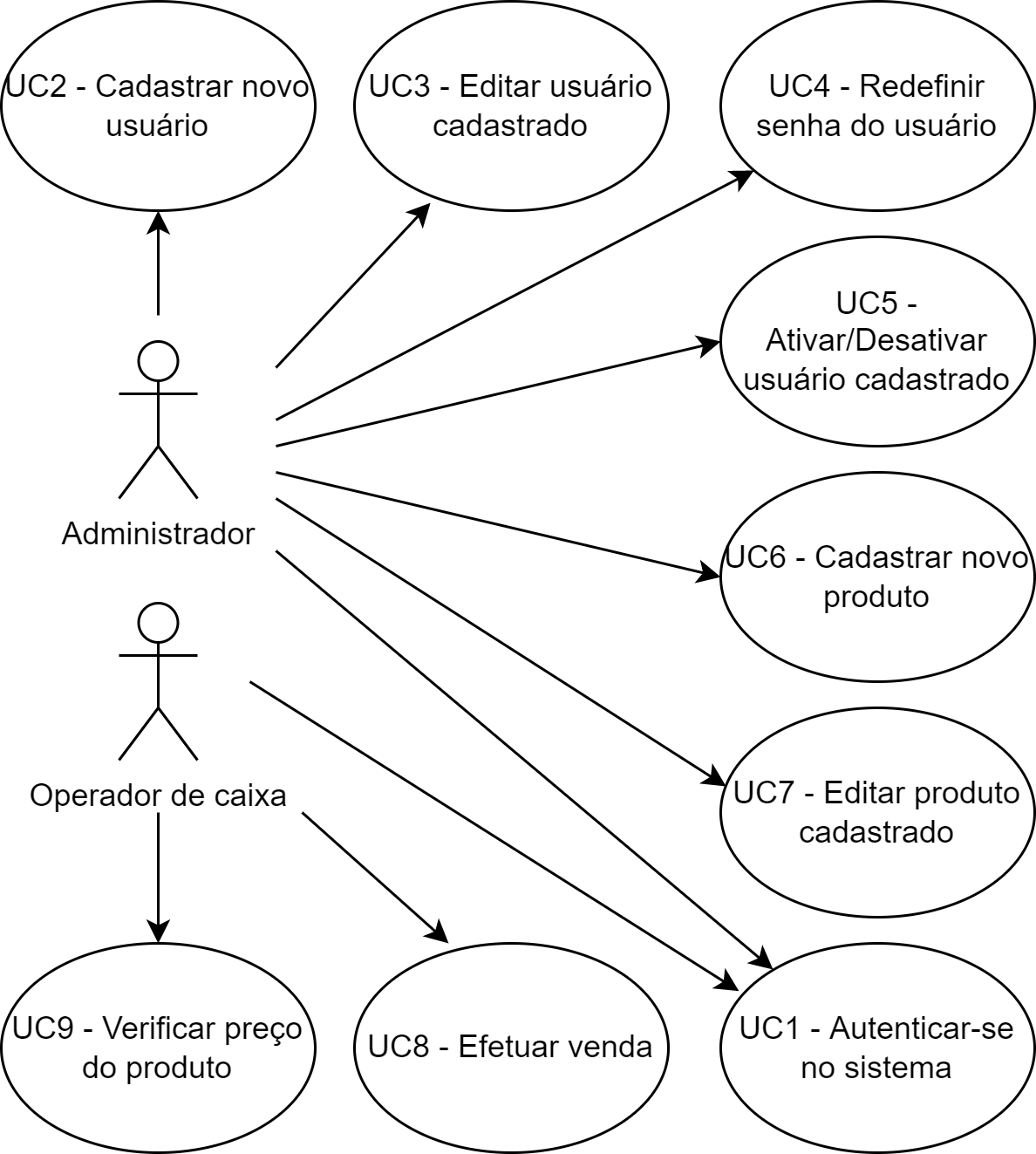
### Visão Lógica dos Dados Persistentes



Modelo Lógico

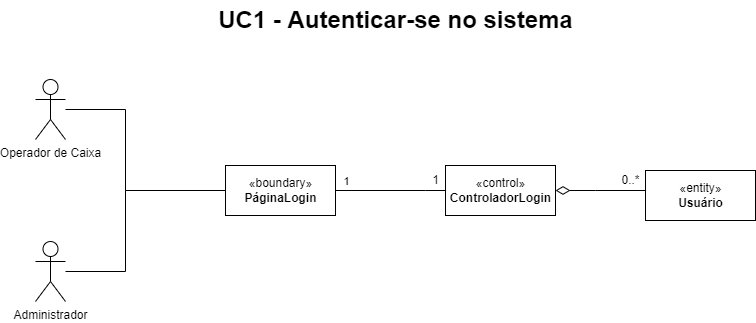
## Casos de Uso

O seguinte diagrama descreve os casos de uso do sistema.

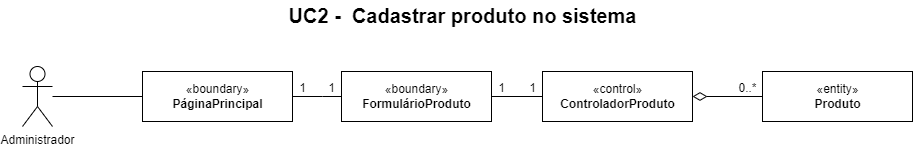


## Visão de Classes Participantes (VCP)

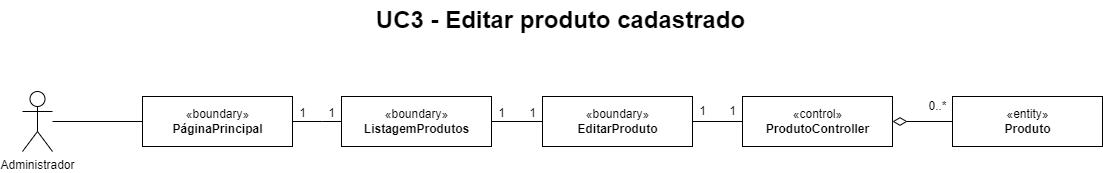
* + 1. UC1 - Autenticar-se no sistema



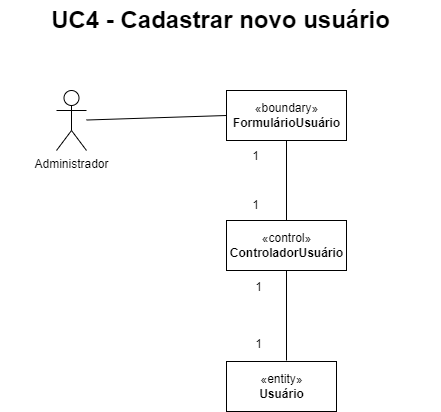
* + 1. UC2 - Cadastrar um produto



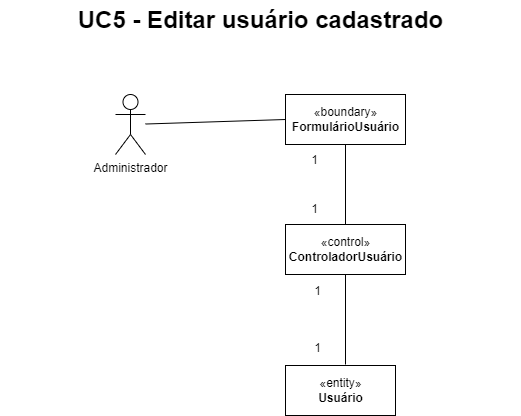
* + 1. UC3 - Editar produto cadastrado



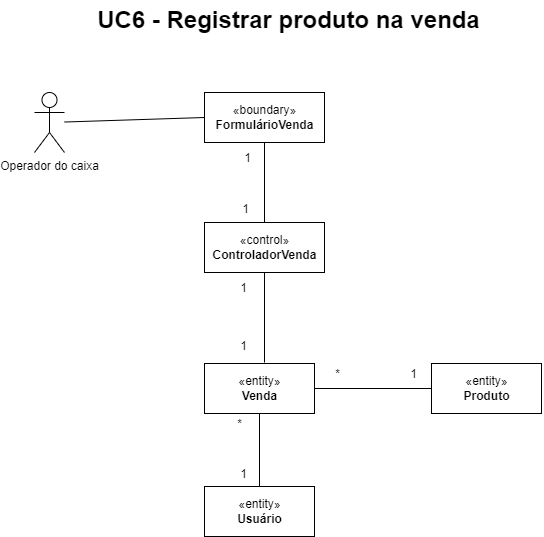
* + 1. UC4 - Cadastrar novo usuário



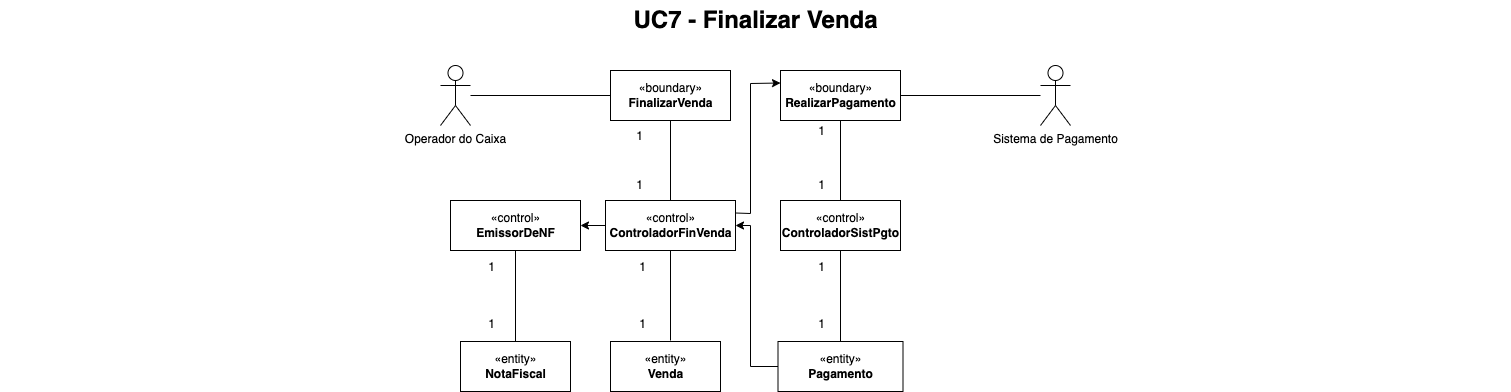
* + 1. UC5 - Editar usuário cadastrado



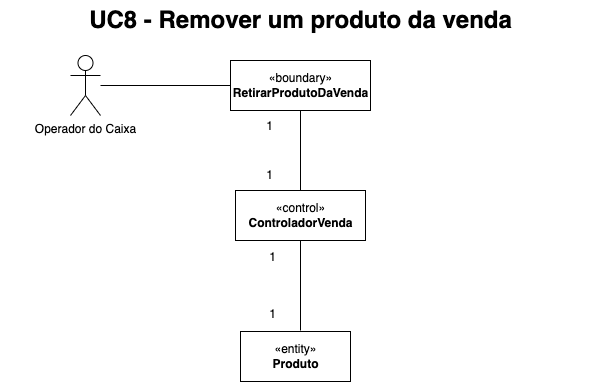
* + 1. UC6 - Registrar produto na venda



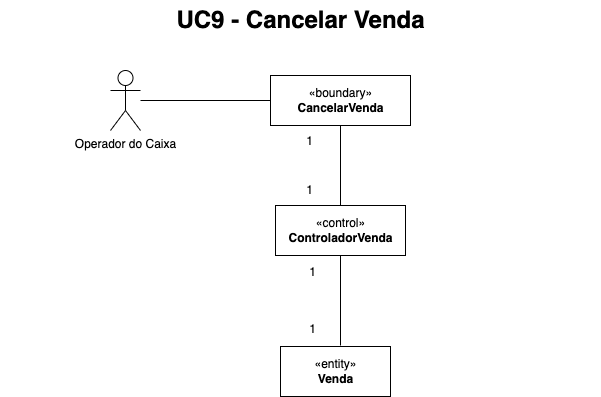
* + 1. UC7 - Finalizar venda



* + 1. UC8 - Remover um produto da venda



* + 1. UC9 - Cancelar venda



# Qualidade

O padrão de arquitetura adotado para esse projeto tem como principal objetivo garantir uma boa organização do código fonte, manutenibilidade e possibilidade de extensão das funcionalidades ou modificação de elementos como a interface gráfica ou banco de dados sem drasticamente afetar o funcionamento do sistema.