
IC/UFF

Ponto de Venda
Documento de Arquitetura de Software

Versão <1.3>

Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

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
14/06/22	1.2	Atualizadas sessões 3,4 e 6. Alterações gerais na formatação e espaçamento.	Gabriel Figueiredo
15/06/22	1.3	Atualizada sessão 8.1	Victor Verdan

Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

Índice Analítico

1.	Introdução	4
1.1	Finalidade	4
1.2	Escopo	4
1.3	Definições, acrônimos e abreviações	4
1.4	Visão Geral	4
2.	Metas e Restrições da Arquitetura	4
3.	Suposições e Dependências	4
4.	Requisitos Arquiteturalmente Significantes	5
5.	Decisões, Restrições e justificativas	5
6.	Mecanismos Arquiteturais	5
7.	Camadas da Arquitetura	5
8.	Visões da Arquitetura	6
8.1	Visão Estrutural	6
8.1.1	Visão Geral	6
8.1.2	Estrutura de Pacotes Significativos	6
8.1.3	Estrutura de Classes	7
8.1.4	Visão Lógica dos Dados Persistentes	7
8.2	Casos de Uso	8
8.3	Visão de Classes Participantes (VCP)	9
9.	Qualidade	10

Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

Documento de Arquitetura de Software

1. 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.

1.1 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.

1.2 Escopo

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

1.3 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.

1.4 Visão Geral

Serão apresentadas 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.

2. 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*.

3. Suposições e Dependências

- Sistema de pagamentos externo para uso de cartões de crédito e débito
- Banco de dados com usuário administrador inicial pré-cadastrado

Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

4. Requisitos Arquiteturalmente Significantes

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.

5. 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.

6. Mecanismos Arquiteturais

7. 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.

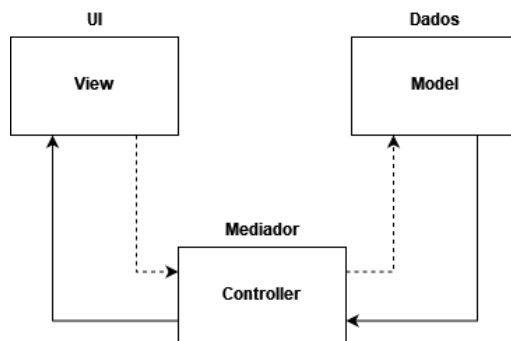
Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

8. Visões da Arquitetura

8.1 Visão Estrutural

8.1.1 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..

8.1.2 Estrutura de Pacotes Significativos

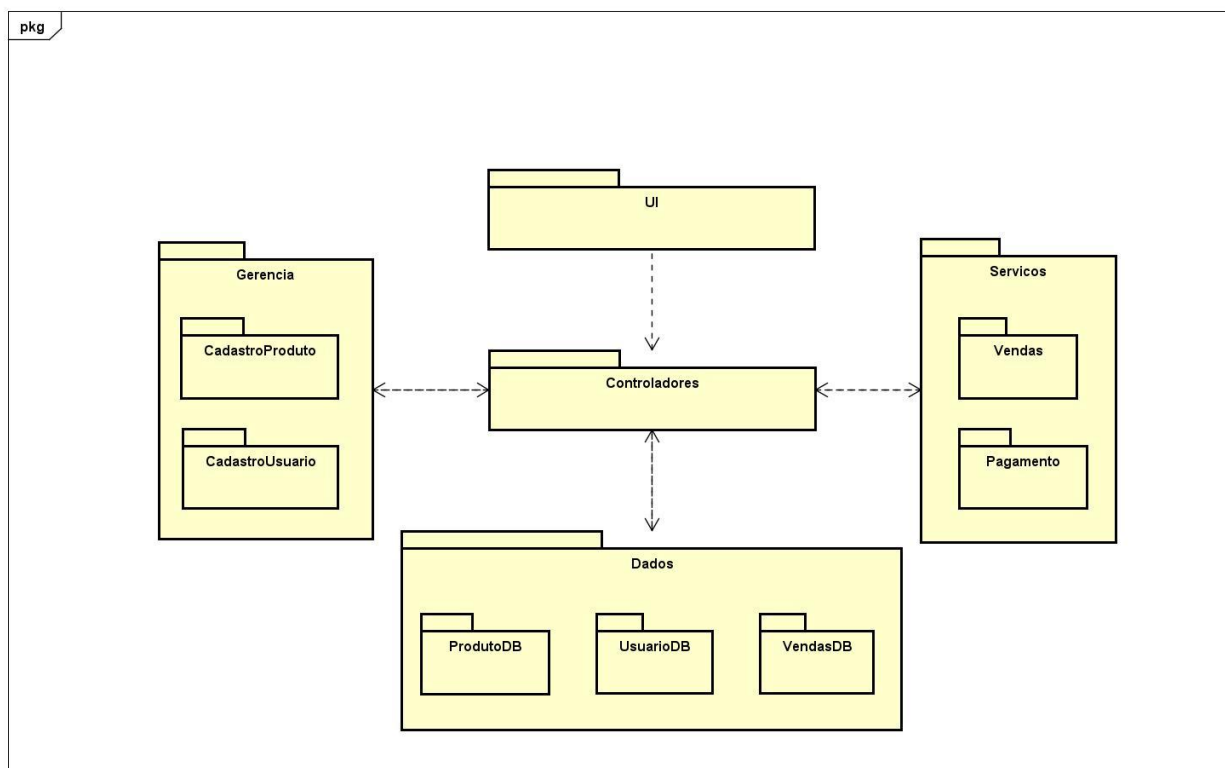


Diagrama de Pacotes do sistema

Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

8.1.3 Estrutura de Classes

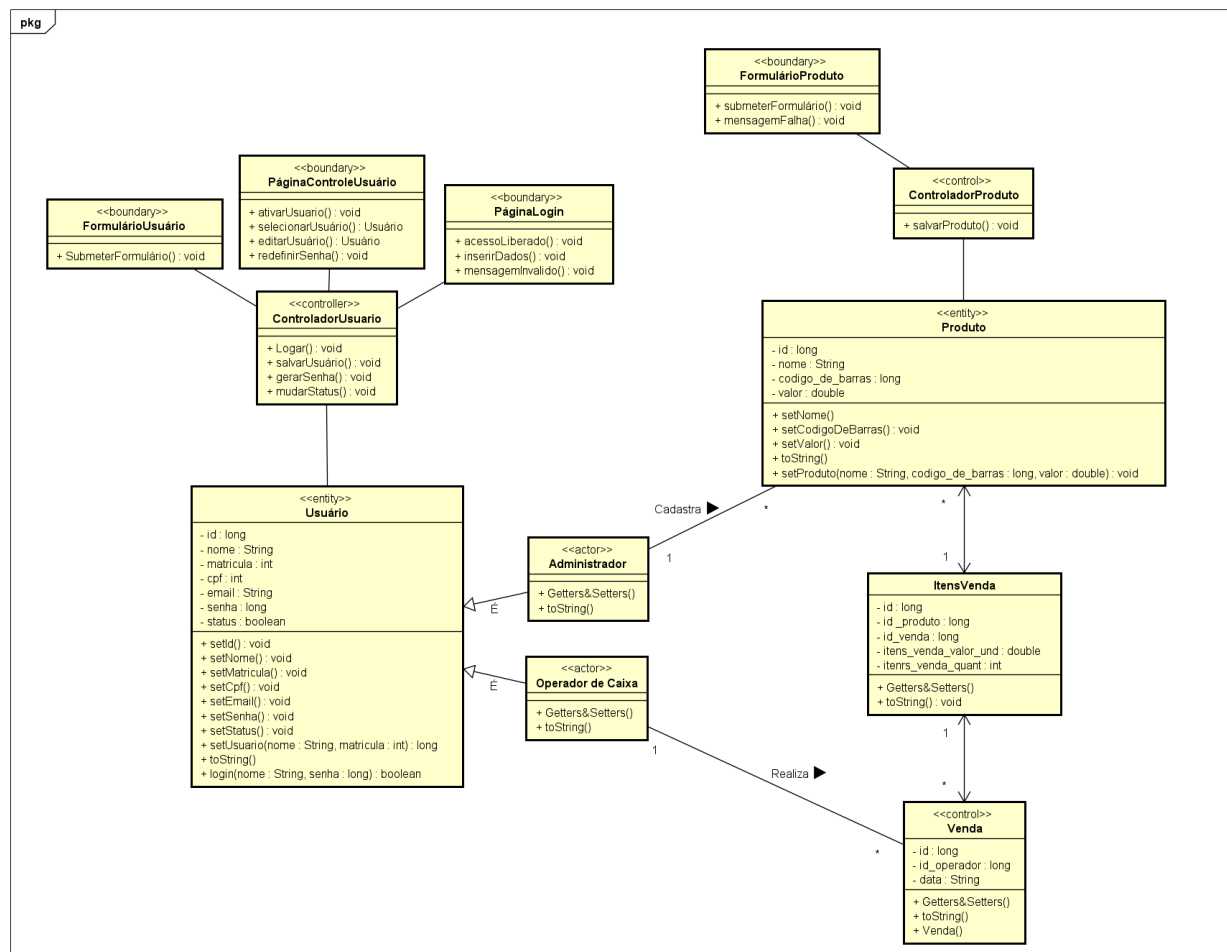
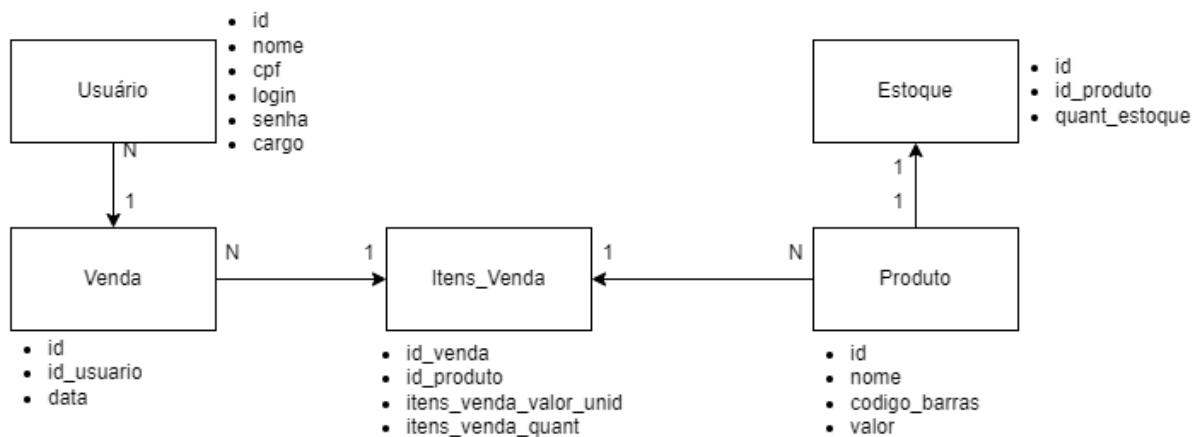


Diagrama de Classe geral do sistema

8.1.4 Visão Lógica dos Dados Persistentes

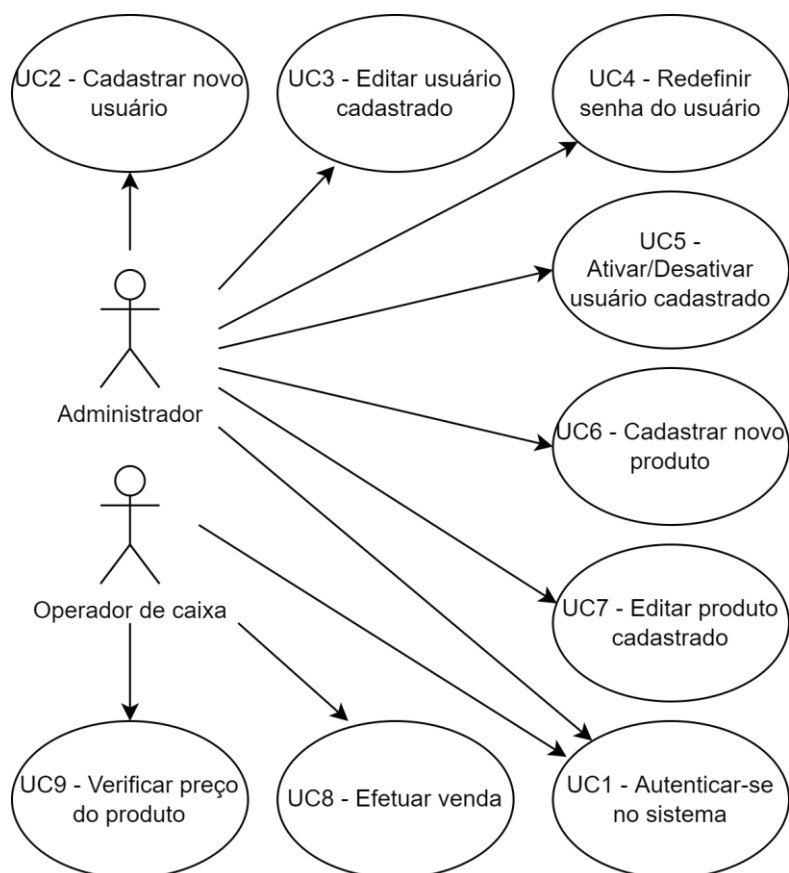


Modelo Lógico

Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

8.2 Casos de Uso

O seguinte diagrama descreve os casos de uso do sistema.



Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

8.3 Visão de Classes Participantes (VCP)

8.3.1 UC1 - Autenticar-se no sistema

8.3.2 UC2 - Cadastrar novo usuário

8.3.3 UC3 - Editar usuário cadastrado

8.3.4 UC4 – Redefinir senha do usuário

8.3.5 UC5 – Ativar/Desativar usuário cadastrado

8.3.6 UC6 – Cadastrar novo produto

8.3.7 UC7 – Editar produto cadastrado

8.3.8 UC8 – Efetuar venda

8.3.9 UC9 – Verificar preço do produto

Ponto de Venda	Version: <1.2>
Documento de Arquitetura de Software	Date: 14/06/22
das@2022.1	

9. Qualidade

O padrão de arquitetura adotado para esse projeto tem como principal objetivo garantir uma boa organização do código fonte, manutibilidade 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.