FRANCISCO PAULO FAGUNDES FERREIRA

GUILHERME ELVIRO FONSECA

**E-Soluções**

PAU DOS FERROS/ RN – 2022

**Sumário**

1. **Introdução**

1.1 Propósito do documento

1.2 Escopo

1.3 Visão geral

1. **Descrição gereal**

2.1 Funções do produto

1. **Requisitos**

3.1 Requisitos funcionais

3.2 Requisitos não funcionais

3.3 Regras de negócio

1. **Representação arquitetural**
2. **Metas e restrições da arquitetura**
3. **Visão de caso de uso**
4. **Visão funcional**
5. **Modelo lógico**
6. **Endpoints**

**10. Visão de implementação**

**11. Qualidade**

**12. Arquitetura de software utilizada e sua justificativa**

**13. Repositório Github**

**1 - Introdução**

**1.1 Propósito do documento**

O propósito deste documento é definir os requisitos do sistema E-SOLUÇÕES - Sistema de vendas para micro e pequenas empresas, que tem como objetivo vender produtos e permitir que usuários possam vender seus produtos.

**1.2 Escopo do produto**

E-SOLUÇÕES permite cadastrar produtos na seção estoque por categorias com data de entrada e posteriormente de saída, efetua toda a parte de venda relacionada a loja e também permitindo que outros usuários possam cadastrar produtos para venda através de uma API externa.

**1.3 Visão geral do documento**

Este documento apresenta uma visão geral do sistema distribuído, descrevendo suas funcionalidades e delimitações de requisitos, seja pelo contexto no qual será aplicado ou por questões de segurança.

**2 - Descrição Geral**

O E-SOLUÇÕES é um sistema distribuído que tem como objetivo auxiliar a empresa e ao usuário externo a venda do seu produto. O sistema é uma plataforma pronta para todo tipo de Aplicação e que abrange a função multi usuário e trazendo uma relação aos usuários, havendo uma separação por grupo onde cada grupo terá sua função definida a área de acesso ao software. A Empresa principal tem acesso total como: inserção de produtos, visualização, edição,excluir e monitoração. O Usuário externo podendo apenas cadastrar produtos, editar, visualizar, e excluir, isso sendo apenas no seu grupo cadastrado

**2.1 Funções do Produto**

O E-SOLUÇÕES tem por objetivo auxiliar a empresa nas suas vendas online através das seguintes funções:

* Inclusão, alteração e exclusão de produtos.
* Consultar e efetuar vendas de produtos.
* Acompanhamento de vendas online realizadas no sistema
* Gerar relatórios financeiros mensais.

**3 - Requisitos**

**3.1 Requisitos Funcionais**

| **RF001** | **O sistema deve permitir a inclusão, alteração, visualização e remoção de produtos** |
| --- | --- |
| **Descrição** | Nome, endereço, telefone, e-mail, documento de identificação (RG ou CPF) e data de nascimento. |
| **Importância** | **[ X ] Obrigatório** [ ] Importante [ ] Desejável |

| **RF002** | **O sistema deve permitir aos usuários realizar a compra do produto.** |
| --- | --- |
| **Descrição** | Nome, endereço, telefone, e-mail, documento de identificação (RG ou CPF, data de nascimento. |
| **Importância** | **[ X ] Obrigatório** [ ] Importante [ ] Desejável |

| **RF003** | **O sistema deve manter os produtos enquanto o estoque durar.** |
| --- | --- |
| **Descrição** | Nome do produto, valor, sobre o produto. |
| **Importância** | [ ] Obrigatório **[ X ] Importante** [ ] Desejável |

| **RF004** | **O sistema deve remover automaticamente o produto quando o estoque acabar.** |
| --- | --- |
| **Importância** | **[ X ] Obrigatório** [ ] Importante [ ] Desejável |

| **RF005** | **O sistema deve armazenar os dados das compras efetuadas.** |
| --- | --- |
| **Descrição** | Nome, CPF, email, endereço e data da compra |
| **Importância** | **[ X ] Obrigatório** [ ] Importante [ ] Desejável |

| **RF006** | **A API deve permitir que que o usuário possa cadastrar, editar e remover produtos para vendas.** |
| --- | --- |
| **Descrição** | Inserir, deletar e editar. |
| **Importância** | **[ X ] Obrigatório** [ ] Importante [ ] Desejável |

**3.2 Requisitos não Funcionais**

| **RNF001** | O sistema deve responder a consultas online em menos de 5 segundos. |
| --- | --- |

| **RNF002** | O sistema deve ter capacidade para recuperar os dados perdidos da última operação que realizou em caso de falha. |
| --- | --- |

| **RNF003** | Deve ser feito o backup do banco de dados a cada 7 dias |
| --- | --- |

| **RNF004** | O sistema deve fornecer facilidades para a realização de backups dos arquivos do sistema. |
| --- | --- |

| **RNF005** | O sistema deve ser executado em computadores com 2GB de ram ou superior, com sistema operacional Windows 10 ou acima. |
| --- | --- |

| **RNF006** | O sistema deve permitir as seguintes opções de pagamento: Em dinheiro, transferência bancária, pix, cheque ou cartão de crédito e débito. |
| --- | --- |

**3.3 Regras de negócio**

| **RGN001** | O sistema deve permitir logar com as contas do google. |
| --- | --- |

| **RGN002** | O usuário deve conter cadastro na plataforma para inserir produtos e realizar vendas. |
| --- | --- |

| **RGN003** | A compra só será finalizada após a confirmação do pagamento. |
| --- | --- |

| **RGN004** | O endereço de entrega deve ser válido. |
| --- | --- |

| **RGN005** | Após o cadastro na plataforma o usuário terá que aguardar até 2 dias para a confirmação. |
| --- | --- |

**4. Representação arquitetural**

* **Visão de caso de uso**
  + Apresenta as funcionalidades arquiteturais importantes e os usuários do sistema.
* **Visão funcional**
  + Apresenta uma modalidade agrupada por áreas funcionais.

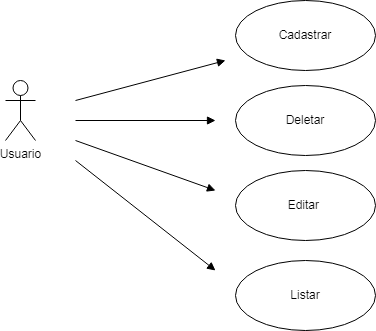
**5. Metas e restrições da arquitetura**

Existem algumas restrições de requisito e de sistema principais que têm uma relação significativa com a arquitetura, sendo elas:

* Utilização do paradigma Orientado a Objetos para o desenvolvimento;
* Estrutura MVC;
* Linguagem de programação PHP e Python
* Framework Flask e Laravel
* O sistema em questão deverá ser distribuído
* Banco de dados sqlite e postgresql
* Scrum

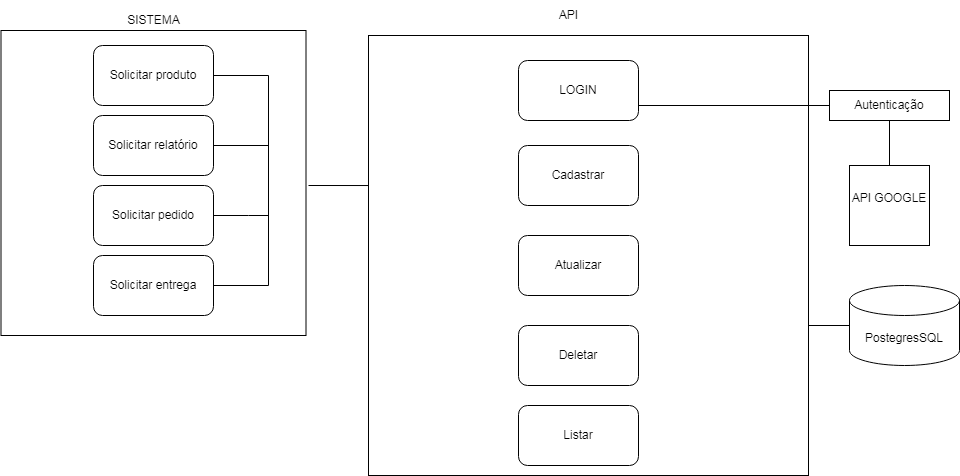
**6. Visão de caso de uso**

Os casos de uso do sistema serão listados abaixo:

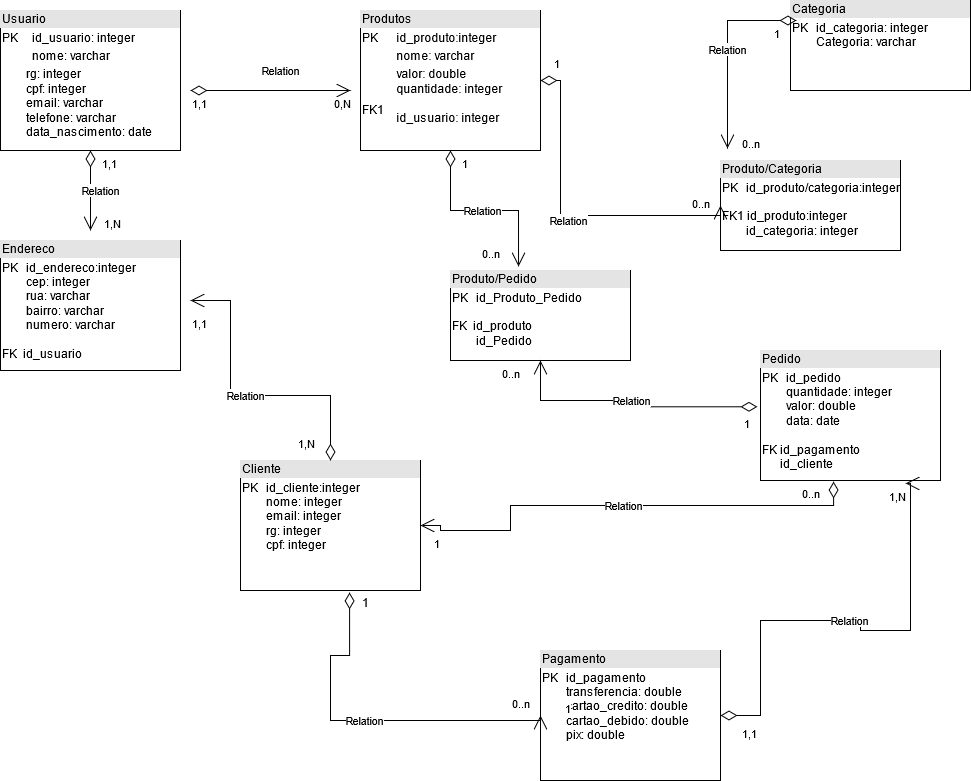


**7. Visão Funcional**

A visão funcional do sistema será listada abaixo:



**8. Modelo lógico**

****

**9. Endpoints**

| **HTTP** | **ENDPOINT** | **DESCRIÇÃO** | **ERRO** | **ACERTO** |
| --- | --- | --- | --- | --- |

| GET | /categorias | Lista todas as categorias | 401,404,405,499 | 200 - OK |
| --- | --- | --- | --- | --- |
| POST | /categorias | criar nova categoria | 400,401,403,405, 499 | 201 - CREATED |
| PUT | /categorias/categoria\_id | Atualiza categoria | 400,401,403,405,  499 | 200 - OK |
| GET | /categorias/<categoria\_id> | Listra único categoria | 401,404,405,499 | 200 - OK |
| DELETE | /categorias/<categoria\_id> | Deleta uma categoria | 400,401,403,405, 499 | 200 - OK |
| GET | /produtos | Lista todos os produtos | 401,404,405,499 | 200 - OK |
| GET | /produtos/produto\_id | Listra único produto | 401,404,405,499 | 200 - OK |
| POST | /produtos | Criar uma ordem para um produto | 400,401,403,405, 499 | 201 - CREATED |
| PUT | /produtos/produto\_id | Atualiza o produtos | 400,401,403,405, 499 | 200 - OK |
| DELETE | /produtos/<produto\_id> | Deleta um produto | 400,401,403,405, 499 | 200 - OK |
| GET | /report/produto\_id | Obter relatório de produtos | 401,404,405,499 | 200 - OK |

| GET | /pedidos | Lista todos os pedidos | 401,404,405,499 | 200 - OK |
| --- | --- | --- | --- | --- |
| GET | /pedidos/pedido\_id | Listra único pedido | 401,404,405,499 | 200 - OK |
| POST | /pedidos | Criar uma ordem para um pedido | 400,401,403,405, 499 | 201 - CREATED |
| PUT | /pedidos/pedido\_id | Atualiza o pedido | 400,401,403,405, 499 | 200 - OK |
| DELETE | /pedidos/<pedido\_id> | Deleta um pedido | 400,401,403,405, 499 | 200 - OK |

**10. Visão da Implementação**

O sistema será implementado utilizando conceitos de programação orientada a objetos através do framework flask e Laravel, estrutura MVC, Web Server

Apache, Linguagem de Programação PHP, Python e Bancos de Dados Sqlite e Postgresql.

**11. Qualidade**

O padrão de arquitetura adotado no projeto tem como finalidade garantir uma melhor organização do código-fonte, o que auxilia na manutenção do software, bem como na portabilidade do mesmo.

**12. Arquitetura de software utilizada e sua justificativa**

* Model-view-controller (MVC)
  + O padrão MVC separa o projeto do software em três camadas independentes: o modelo (manipulação da lógica de dados), a visão (a interface do usuário) e o controlador (fluxo de aplicação). Esta separação facilita a manutenção do código, que pode ser reutilizado em outros projetos.
  + Uma das principais características que fez com que escolhêssemos essa arquitetura foi a **organização**: Esse método de programação permite que um novo desenvolvedor tenha muito mais facilidade em entender o que foi construído, assim como os erros se tornam mais fácil de serem encontrados e corrigidos.
  + Outra característica importante para podermos escolher essa arquitetura foi a **Eficiência**: Como a arquitetura de software é dividida em 3 componentes , sua aplicação fica muito mais leve, permitindo que vários desenvolvedores trabalhem no projeto de forma independente.

**13. Repositório Github:**

* **Repositório da API:**
  + **https://github.com/paulo0264/Api-Integrador**
* **Repositório do Front:**
  + **https://github.com/paulo0264/Integrador-IFRN**

**Pau dos Ferros/RN - 2022**