**API - Sistema de Controle Financeiro (Crédito e Débito)**

**Documento de Arquitetura de Software**

**Versão 1.1**

**Integrantes: Adriano**

**Hítallo Oliveira**

**Marcello Alves**

**Reinaldo Albernaz**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 18/05/2019 | 1.0 | Data Inicial | Adriano  Hítallo Flávyo  Marcello Alves  Reinaldo Albernaz |

**Índice Analítico**

[**Introdução**](#_30j0zll) **4**

[Finalidade](#_6tjnbyponkso) 4

[Escopo](#_jtdxheqn6fp3) 4

[Visão Geral](#_icqdg1m1qctw) 4

[**Representação Arquitetural**](#_6k2p1t67i72d) **4**

[**Metas e Restrições da Arquitetura**](#_hejjv8hxln06) **4**

[**Visão de Casos de Uso**](#_9t1h0k6skotp) **4**

[**Visão Lógica**](#_57pvq7izzvj8) **5**

[Visão Geral](#_8jgutxrqvo88) 5

[Pacotes de Design Significativos do Ponto de Vista da Arquitetura](#_bci7wc59z0vs) 5

[**Visão de Processos**](#_o9luau1391hv) **5**

[**Visão de Implantação**](#_bu7ggabrfdgm) **5**

[**Visão da Implementação**](#_l031p13qevbo) **5**

[Camadas](#_z33egzga1ydl) 5

[**Tamanho e Desempenho**](#_107ylok6ti2d) **5**

[**Qualidade**](#_q9zed7e13siz) **6**

Documento de Arquitetura de Software

# Introdução

## Finalidade

Este documento fornece uma visão arquitetural abrangente do sistema, usando diversas visões de arquitetura para representar diferentes aspectos do sistema. Ele pretende capturar e transmitir as decisões arquiteturas significativas que foram decididas no momento da criação do sistema.

## Escopo

O Documento da Arquitetura de Software se aplica à API do Sistema de Controle Financeiro (Débito e Crédito) da AlgaMoney que foi desenvolvido pela AlgaWorks.

## Visão Geral

Este documento está organizado em tópicos relacionados às visões arquiteturais.

# Representação Arquitetural

A API à qual será focada este documento de Arquitetura é uma aplicação web que foi desenvolvida tendo como base a arquitetura REST(ful).

# Metas e Restrições da Arquitetura

* Arquitetura REST(ful);
* Container Web: Spring Boot;
* Banco de Dados: MySQL;
* Protocolo de segurança: auth2.

Acreditamos que os desenvolvedores escolheram o container Spring Boot por ser um framework leve, e extremamente eficiente, tornando o sistema mais ágil num tempo de desenvolvimento pequeno.

Já sobre a escolha arquitetural REST(ful) acreditamos que tenha sido por conta de ter agilidade e Simplicidade, fornecer um uso correto do protocolo HTTP (Por exemplo com o uso de GET), maior produtividade e clareza e por ser mais fácil de dar manutenção no sistema.

# Visão de Casos de Uso

Os casos de uso que influenciarão diretamente na arquitetura do sistema são os que envolve as operações de CRUD em geral, e comunicação com o front end. Os mesmos estão listados abaixo.

* 1. **Efetuar Login**

**4.2 Realizar Ponto**

**4.3 Cadastrar Funcionário**

**4.4 Editar Funcionário**

**4.5 Gerar Relatórios**

**4.6 Consultar Funcionário**

**4.7 Consultar Ponto**

**4.8 Editar Ponto**

**4.9 Incluir Ponto**

**4.10 Excluir Ponto**

# Visão Lógica

Apresentação: Essa camada contém todas as view do projeto, as telas de apresentação, cadastro, relatórios, interface.

Serviço e Negócio: Essa camadas executa as funções: inclusão, exclusão, update, consulta.

Persistência: Essa camada é responsável por acessar o banco de dados recuperar os dados.

## Visão Geral

view: enums

controller: controller, services

model: SQLQuery, repositories

## Pacotes de Design Significativos do Ponto de Vista da Arquitetura

com.br.pontu

com.br.pontu.SQLQuery

com.br.pontu.controller

com.br.pontu.entity

com.br.pontu.enums

com.br.pontu.repositores

com.br.pontu.services

# Visão de Processos

Checar login

Cadastrar, listar, editar e excluir o usuário

Marcar ponto

# Visão de Implantação

O sistema poderá ser utilizado de qualquer dispositivo que tenha conexão com a internet. Computadores, tablets e smartphones, possibilitando assim uma diversidade maior para acessos acesso*.*

# Visão da Implementação

## Camadas

view, controller, persistência.

# Tamanho e Desempenho

O sistema não causará impacto no quesito tamanho, porque ele não ocupa espaço na máquina local, devido a ser uma aplicação online. Em relação ao seu desempenho, esperamos uma resposta considerável do sistema, hospedando em um servidor adequado, para que não ocorra travamentos ou lentidão no sistema. Atendendo assim clientes, que necessitam de suporte para pequenas e grandes escalas.

# Qualidade

Essa arquitetura mvc garante a divisão de funções por papéis, facilitando a manutenção. E será feito o uso de criptografia em sha 256, e perfil exclusivo por usuário. Para garantir a segurança, dar garantia e privacidade aos usuários.