Documento de Arquitetura de Software

Projeto: Aplicativo para Gerenciamento de Finanças Pessoais

Data: 25/07/2025

Versão: 1.0

Equipe: [Preencher com nomes dos integrantes]

1. Introdução

1.1 Objetivo

Este documento descreve a arquitetura do sistema para o aplicativo de gerenciamento de finanças pessoais. A arquitetura orientará as decisões técnicas durante o desenvolvimento e fornecerá uma base para comunicação entre os membros da equipe.

1.2 Escopo

O sistema permitirá que usuários registrem receitas e despesas, classifiquem categorias financeiras e visualizem relatórios mensais.

2. Visão Geral da Arquitetura

2.1 Estilo Arquitetural

O sistema seguirá o padrão MVC (Model-View-Controller) combinado com Arquitetura em Camadas. A API REST será desenvolvida em Java com Spring Boot.

2.2 Camadas

- Apresentação (Front-end): Interface do usuário (ex: React ou Flutter).
- Serviço / Aplicação: Controladores REST e regras de negócio.
- Domínio: Entidades e serviços de domínio.
- Persistência: Repositórios (DAO) com JPA / Hibernate.
- Infraestrutura: Configurações técnicas (banco, autenticação, etc).

3. Componentes Principais

Componente	Descrição
UsuarioController	Expõe endpoints REST para login/cadastro.

TransacaoController	Gerencia endpoints para despesas e receitas.
UsuarioService	Contém lógica de autenticação e regras de usuário.
TransacaoService	Regras de negócio para criação e análise de transações.
TransacaoRepository	Repositório JPA para persistência de dados.
RelatorioService	Gera dados para gráficos e relatórios.

Diagrama de Componentes

Adicionar aqui

4. Tecnologias Utilizadas

Camada	Tecnologias
Front-end	ReactJS ou Flutter
Back-end	Spring Boot, Java 17
Banco de Dados	PostgreSQL
ORM	Hibernate / JPA
Autenticação	JWT
Versionamento	Git + GitHub
Deploy	Docker (opcional)

5. Padrões e Convenções

- DTOs para transporte de dados na API.
- Padrão Repository e Service bem definidos.
- Camadas isoladas via pacotes separados.
- Tratamento de erros com @ControllerAdvice.

6. Requisitos Não Funcionais

Requisito	Descrição
Escalabilidade	A arquitetura deve permitir expansão.
Manutenibilidade	Código modular e reutilizável.
Segurança	Senhas criptografadas, autenticação JWT.
Portabilidade	Compatível com Web e Mobile.

7. Riscos Arquiteturais

- Baixo domínio da equipe sobre padrões REST.
- Integração contínua com front-end mobile/web.
- Dificuldades em testes automatizados.

8. Decisões Arquiteturais

Decisão	**Justificativa**
Uso de Spring Boot	Facilidade de configuração e comunidade sólida.
PostgreSQL como banco de dados	Gratuito, robusto e bem documentado.
Arquitetura em Camadas	Separação de responsabilidades.

9. Diagrama de Domínio