Documentação Técnica – Projeto Meu Controle Financeiro

Versão: 1.0

Data: 23 de Setembro de 2025

# 1. Visão Geral do Projeto

## 1.1. Descrição

O "Meu Controle Financeiro" é uma aplicação web full-stack concebida para a gestão de finanças pessoais.  
Permite que os utilizadores registem receitas e despesas, criem orçamentos mensais por categoria, definam metas de poupança e visualizem a sua saúde financeira através de gráficos e relatórios.

## 1.2. Arquitetura da Solução

A aplicação segue uma arquitetura cliente-servidor desacoplada:  
  
Backend (Servidor): Uma API RESTful desenvolvida em Node.js com o framework Express. É responsável por toda a lógica de negócio, autenticação de utilizadores e comunicação com a base de dados.  
  
Frontend (Cliente): Uma Single Page Application (SPA) desenvolvida em React. Consome os dados da API backend para renderizar a interface do utilizador e proporcionar uma experiência interativa.  
  
Base de Dados: Uma base de dados relacional MariaDB, executada num container Docker para garantir portabilidade e um ambiente de desenvolvimento consistente.

## 1.3. Tecnologias Principais

|  |  |
| --- | --- |
| Categoria | Tecnologia/Ferramenta |
| Backend | Node.js, Express.js, Sequelize (ORM), JWT, Bcrypt.js |
| Frontend | React (com Vite), React Router, Axios, Tailwind CSS, Recharts |
| Base de Dados | MariaDB |
| Ferramentas | Docker, Git & GitHub, VS Code, ESLint, Nodemon, Sequelize-CLI |

# 2. Backend (API - Node.js & Express)

## 2.1. Estrutura de Pastas

backend/  
└── src/  
 ├── app.js # Configuração central da aplicação Express (middlewares, rotas)  
 ├── server.js # Ponto de entrada que inicia o servidor HTTP  
 ├── config/ # Ficheiros de configuração (ex: base de dados)  
 ├── controllers/ # Lógica de negócio (ex: UserController, TransactionController)  
 ├── database/  
 │ ├── index.js # Inicializador do Sequelize  
 │ └── migrations/ # Alterações no schema da base de dados  
 ├── middlewares/ # Funções pré-rotas (ex: verificação de token)  
 ├── models/ # Definição das tabelas (Sequelize)  
 └── routes/  
 └── routes.js # Endpoints da API

## 2.2. Modelos de Dados (Sequelize)

User: Representa um utilizador. Contém nome, email e senha\_hash. Um User tem muitas (hasMany) Transactions, Budgets e Goals.  
  
Transaction: Representa uma receita ou despesa. Uma Transaction pertence a um (belongsTo) User.  
  
Budget: Representa um orçamento mensal. Um Budget pertence a um (belongsTo) User.  
  
Goal: Representa uma meta financeira. Um Goal pertence a um (belongsTo) User.

## 2.3. Endpoints da API (RESTful)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recurso | Método HTTP | Endpoint | Descrição | Protegido? |
| Autenticação | POST | /api/register | Regista um novo utilizador | Não |
| Autenticação | POST | /api/login | Autentica e retorna token | Não |
| Transações | GET | /api/transactions | Lista as transações do utilizador | Sim |
| Transações | POST | /api/transactions | Cria nova transação | Sim |
| Transações | PUT | /api/transactions/:id | Atualiza transação | Sim |
| Transações | DELETE | /api/transactions/:id | Apaga transação | Sim |
| Orçamentos | GET | /api/budgets | Lista orçamentos | Sim |
| Orçamentos | POST | /api/budgets | Cria novo orçamento | Sim |
| Orçamentos | PUT | /api/budgets/:id | Atualiza orçamento | Sim |
| Orçamentos | DELETE | /api/budgets/:id | Apaga orçamento | Sim |
| Metas | GET | /api/goals | Lista metas | Sim |
| Metas | POST | /api/goals | Cria nova meta | Sim |
| Metas | PUT | /api/goals/:id | Atualiza meta | Sim |
| Metas | DELETE | /api/goals/:id | Apaga meta | Sim |

## 2.4. Autenticação

A autenticação é baseada em JSON Web Tokens (JWT):  
1. O utilizador envia email e senha para /api/login.  
2. O servidor valida credenciais e gera token JWT.  
3. O cliente armazena o token.  
4. Para rotas protegidas, envia no cabeçalho Authorization.  
5. Middleware valida o token antes de permitir o acesso.