# Gestor de Finanças Pessoais com Análises Preditivas

# 1. Introdução

Este projeto visa desenvolver uma aplicação web para gestão de finanças pessoais que ajudará os utilizadores a acompanhar as suas receitas e despesas, gerar relatórios financeiros e utilizar análises preditivas para fornecer previsões sobre futuros gastos e recomendações de poupança. O objetivo é proporcionar uma visão clara da situação financeira, ao mesmo tempo que permite a tomada de decisões informadas com base em previsões.

# 2. Objetivos

Os principais objetivos deste projeto são:

- Desenvolver uma aplicação web intuitiva para a gestão de finanças pessoais.
- Implementar um sistema de previsões financeiras com base em dados históricos.
- Fornecer recomendações de poupança e otimização de despesas.
- Integrar APIs de bancos e plataformas de trading para sincronização automática dos dados financeiros.
- Garantir a segurança e privacidade dos dados financeiros dos utilizadores.

# 3. Funcionalidades Principais

A aplicação oferecerá as seguintes funcionalidades:

- **Gestão de transações**: Permitir que os utilizadores registem receitas e despesas manualmente ou sincronizem com APIs bancárias e plataformas de trading.
- **Relatórios detalhados**: Geração de relatórios sobre a evolução financeira (balanço mensal, categorias de despesas, etc.).
- **Análise preditiva**: Utilização de machine learning para prever futuros gastos.
- **Orçamento pessoal**: Definir orçamentos para diferentes categorias e alertar os utilizadores ao ultrapassar os limites.
- **Sincronização com APIs**: Integração com APIs de contas bancárias e plataformas de trading para importar automaticamente transações e dados financeiros.

# 4. Tecnologias

### Frontend:

- **React/Next.js**: Para o desenvolvimento da interface do utilizador.
- **Tailwind CSS**: Para estilização rápida e responsiva.
- **Framer Motion**: Para animações suaves e transições interativas.
- **Vercel**: Hospedagem do frontend com deploy contínuo através de GitHub Actions.

### Backend:

- **Node.js/Express**: Para a criação do servidor e gestão das APIs.
- **MongoDB**: Armazenamento de dados financeiros com a flexibilidade e escalabilidade de um banco de dados NoSQL.
- **Heroku**: Para a hospedagem do backend e deploy contínuo com integração CI/CD usando GitHub Actions.

### Machine Learning:

- **Python (TensorFlow/scikit-learn)**: Para a implementação de modelos preditivos de machine learning que analisam dados financeiros e oferecem previsões.
- **Flask/FastAPI**: Para integrar o modelo preditivo como microserviços que se comunicam com o backend.

# 5. APIs Selecionadas

**Bancárias**

- **Nordigen API**:
    - **Descrição**: API de Open Banking gratuita que permite a integração de dados bancários (transações, saldos) com várias instituições financeiras em Portugal.
    - **Custo**: Gratuito.
    - **Documentação**: Nordigen API

**Trading/Investimentos**

- **Alpha Vantage API**:
    - **Descrição**: Fornece dados de ações, criptomoedas, forex e indicadores técnicos. Ideal para acompanhar investimentos dos utilizadores.
    - **Custo**: Gratuito até 500 chamadas diárias.
    - **Documentação**: Alpha Vantage API

- **Coin Market Cap API**:
    - **Descrição**: Fornece dados de mercado de criptomoedas em tempo real.
    - **Custo**: Gratuito com um plano básico.
    - **Documentação**: Coin Market Cap API

# 6. Arquitetura do Sistema

### Frontend:

- O utilizador poderá visualizar as suas finanças e acompanhar previsões financeiras numa interface construída em **React/Next.js**.
- O design será simples e eficiente, com utilização de **Tailwind CSS** para garantir uma experiência moderna e responsiva.

### Backend:

- A aplicação será construída com **Node.js/Express** para gerir as APIs e a lógica de negócios.
- **MongoDB** será utilizado para armazenar os dados financeiros dos utilizadores de forma segura.
- O backend será responsável por coletar dados das APIs bancárias e de trading, armazenar as transações e fornecer previsões financeiras.

### Machine Learning:

- **Python** será utilizado para a parte de machine learning, implementando modelos preditivos de gastos com **TensorFlow** ou **scikit-learn**.
- A comunicação entre o backend (Node.js) e o serviço de machine learning (Python) será feita via **APIs REST** ou **microserviços**.

# 7. Uso e Processamento dos Dados

### Dados Bancários

Os dados de transações e saldos bancários serão recolhidos via **Nordigen API** e armazenados de forma segura na base de dados MongoDB. Estes dados serão utilizados para:

- **Categorização automática de transações**: Cada transação será automaticamente categorizada (alimentação, transporte, etc.) para gerar relatórios detalhados.
- **Análises de tendências**: Serão analisadas as tendências de despesas ao longo do tempo para ajudar os utilizadores a perceberem onde podem melhorar o seu planeamento financeiro.

- **Definição de orçamentos**: Com base no histórico de despesas, os utilizadores poderão definir limites de gastos por categoria, sendo notificados quando se aproximarem dos limites.

### Dados de Investimentos e Criptomoedas

Os dados de trading, obtidos através das **APIs Alpha Vantage** e **CoinMarketCap**, serão utilizados para:

- **Monitorizar o portfólio de investimentos**: Os utilizadores poderão visualizar o desempenho dos seus investimentos (ações, criptomoedas) em tempo real.
- **Relatórios de lucros/perdas**: Será calculado o valor atual de cada ativo, com base nas cotações mais recentes, e gerados relatórios sobre lucros ou perdas no portfólio.

# 8. Segurança do Website

A segurança será uma prioridade absoluta, principalmente devido à natureza sensível dos dados financeiros envolvidos. Serão implementadas as seguintes medidas de segurança:

### Criptografia de Dados

- **TLS/SSL**: Toda a comunicação entre o cliente (frontend) e o servidor será criptografada utilizando **TLS/SSL**, garantindo que os dados transmitidos sejam protegidos contra ataques man-in-the-middle.
- **Criptografia em repouso**: Todos os dados armazenados no banco de dados (MongoDB) serão criptografados em repouso, utilizando técnicas de criptografia AES-256.

### Autenticação e Autorização

- **Autenticação via JWT (JSON Web Tokens)**: A aplicação usará tokens JWT para autenticar utilizadores, garantindo que apenas utilizadores autenticados possam aceder aos seus dados financeiros.
- **Autorização baseada em funções**: Implementaremos controlos de acesso baseados em funções para assegurar que os utilizadores só possam aceder aos seus próprios dados.

### Proteção Contra Ataques

- **Proteção contra ataques de força bruta**: Serão implementados mecanismos de limitação de tentativas de login para evitar ataques de força bruta.
- **Prevenção de XSS e CSRF**: Utilização de ferramentas e práticas que protegem contra ataques de Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF).

- **Auditorias e logs**: Toda a atividade no sistema será auditada, mantendo logs detalhados de acessos, modificações de dados e erros, para detecção e análise de potenciais brechas de segurança.

# 9. Plano de Implementação

## Fase 1: Planeamento e Pesquisa (1-2 semanas)

- **Definir os requisitos do utilizador e o MVP (Produto Mínimo Viável)**: Identificar as principais funcionalidades necessárias para o lançamento inicial, incluindo a sincronização de dados bancários, gestão de transações e previsões financeiras.
- **Pesquisar a documentação das APIs e criar contas de desenvolvedor**: Explorar a documentação das APIs escolhidas (Nordigen, Alpha Vantage, CoinMarketCap), criar contas de teste e obter as chaves de API necessárias.
- **Preparar o ambiente de desenvolvimento**:
    - Configurar o servidor backend com **Node.js/Express**.
    - Configurar a base de dados **MongoDB**.
    - Iniciar o frontend com **React/Next.js**.
    - Preparar o ambiente Python para análise preditiva (instalar TensorFlow ou scikit-learn).

## Fase 2: Desenvolvimento do Backend (2-3 semanas)

- **Configurar o servidor em Node.js/Express**:
    - Criar a estrutura da API backend, configurando rotas e controladores para o registo e gestão de utilizadores.
    - Implementar autenticação usando **JWT** para proteger as rotas e garantir que apenas utilizadores autenticados acedem aos dados financeiros.
- **Configurar MongoDB**:
    - Criar os modelos de dados para transações, utilizadores, e previsões financeiras.
    - Implementar operações CRUD para gerir transações (criar, ler, atualizar e apagar).
- **Integração com APIs**:
    - **Nordigen API**: Implementar a integração para importar automaticamente as transações bancárias.
    - **Alpha Vantage API** e **CoinMarketCap API**: Integrar dados de trading e criptomoedas para permitir o acompanhamento do portfólio de investimentos dos utilizadores.

## Fase 3: Desenvolvimento do Frontend (2-3 semanas)

- **Criar a interface com React/Next.js**:
    - Desenvolver a interface do utilizador, com páginas para registo, login, dashboard financeiro e detalhes de transações.

    - Utilizar **Tailwind CSS** para uma experiência de utilizador moderna e responsiva.

- **Dashboards financeiras e relatórios gráficos**:
    - Implementar gráficos interativos utilizando bibliotecas como **Chart.js** ou **D3.js** para visualização de receitas, despesas, e investimentos.
    - Fornecer relatórios detalhados sobre categorias de despesas e tendências financeiras.

## Fase 4: Implementação de Análises Preditivas (2-3 semanas)

- **Criar modelos preditivos em Python**:
    - Utilizar **scikit-learn** ou **TensorFlow** para construir modelos de machine learning que analisam dados históricos e preveem padrões de gastos futuros.
    - Treinar e ajustar os modelos com base nos dados das transações dos utilizadores.
- **Integrar os modelos preditivos no backend**:
    - Expor os modelos de machine learning como uma API REST usando **Flask** ou **FastAPI**.
    - Implementar a comunicação entre o backend (Node.js) e o serviço de machine learning, recebendo previsões e fornecendo recomendações personalizadas aos utilizadores.

## Fase 5: Testes e Refinamento (3-4 semanas)

- **Testar a aplicação**:
    - Realizar testes funcionais e de integração para garantir que todas as funcionalidades estão a funcionar corretamente (sincronização de dados, relatórios, previsões).
    - Testar a aplicação com utilizadores reais (testes beta) para recolher feedback sobre a usabilidade e a experiência de utilizador.
- **Refinamento do design e correção de bugs**:
    - Refinar o design com base no feedback dos testes.
    - Corrigir eventuais bugs e otimizar o desempenho da aplicação, especialmente na integração com APIs externas.

## Fase 6: Lançamento e Documentação Final (2 semanas)

- **Preparar a aplicação para produção**:
    - Implementar medidas de segurança adicionais, como proteção contra ataques de força bruta e garantir que todas as comunicações são feitas via **HTTPS**.
    - Configurar o ambiente de produção, como hospedagem no **Vercel** para o frontend e **Heroku** ou outro serviço de cloud para o backend.

- **Criar documentação detalhada**:
    - Documentar as principais funcionalidades, API endpoints, processos de integração e qualquer configuração necessária para desenvolvedores que possam colaborar no futuro.

# NOTA: Este período pode ser alterado devido a problemas e muitas outras circunstâncias, mas desde que exista 5 horas por dia “35 horas semanais isto poderá ser feito num período de 17 semanas “4 meses”