**EXPENSETRACK - PLATAFORMA DE GERENCIAMENTO DE DESPESAS CORPORATIVAS**

**Motivação:**

Uma plataforma de gerenciamento de despesas corporativas. Esse tipo de projeto é altamente relevante, uma vez que oferece soluções de pagamento e gestão de despesas aplicáveis em diversas áreas, incluindo combustíveis e frotas.

**Funcionalidades principais:**

* Gestão de despesas:
  + Permitir que funcionários registrem suas despesas (viagens, combustível, refeições, etc.).
  + Upload de recibos com reconhecimento de imagem (poderia usar algum OCR simples para reconhecer dados do recibo).
* Classificação automática de despesas:
  + Usar Machine Learning ou regras automatizadas para classificar as despesas (por tipo, centro de custo, etc.).
* Controle de aprovações:
  + Workflow para aprovação das despesas, onde gerentes podem aprovar ou solicitar mais informações.
* Relatórios e dashboards:
  + Relatórios detalhados de despesas por categoria, data, e por funcionário.
  + Dashboard visual com gráficos sobre tendências de despesas e alertas para despesas fora dos padrões.
* Integração com APIs financeiras:
  + Conectar com APIs de bancos para verificar transações automaticamente ou até mesmo integrar pagamentos automáticos.
* Multiempresa e multicurrency:
  + Funcionalidade para gerenciar várias empresas, útil para empresas que atuam em diferentes países, com suporte para múltiplas moedas.
* Segurança e Compliance:
  + Autenticação com OAuth2/Single Sign-On.
  + Auditoria de ações, registro de logs e verificação de conformidade com normas financeiras (ex: PCI-DSS).

**Stack tecnológica:**

* Back-end: ASP.NET Core com Entity Framework para manipulação dos dados financeiros.
* Front-end: Angular ou React para criar uma interface interativa e fluida.
* Banco de Dados: SQL Server ou Postgres para armazenar as informações de despesas, usuários e relatórios.
* CI/CD: Pipeline no Azure DevOps, GitHub Actions para deploy contínuo.
* Docker: Para containerizar a aplicação e garantir portabilidade.
* Infraestrutura como Código: Usar Terraform para provisionamento de infraestrutura (banco de dados, balanceador de carga, etc.).
* Testes: Aplicação de TDD/BDD com xUnit e integração contínua para garantir a qualidade do código.

**Diferenciais:**

* Integração de microserviços para facilitar a escalabilidade de cada módulo (ex: módulo de aprovação, módulo de relatórios).
* Implementação de um sistema de notificações em tempo real via SignalR para alertar os gerentes sobre novas despesas para aprovação.
* Opção de exportação de relatórios para PDF ou Excel.
* Utilização de boas práticas de segurança e conformidade, como encriptação de dados sensíveis e gerenciamento de identidades.

**Benefícios desse projeto:**

* Aplicabilidade prática: Demonstra que você entende as necessidades reais de empresas financeiras como a WEX.
* Uso de tecnologias modernas: Mostra que você domina o uso de tecnologias como ASP.NET Core, Docker, Terraform, e CI/CD.
* Impacto em performance e escalabilidade: Prova que você pode construir soluções que cresçam conforme a demanda.
* Flexibilidade e adaptação: É um projeto que pode ser facilmente estendido para incluir novas funcionalidades, mostrando sua capacidade de inovação.

**PLANEJAMENTO INICIAL**

**Cronograma Geral:**

* Semana 1: Requisitos e Planejamento.
* Semanas 2-3: Setup de Infraestrutura.
* Semanas 4-6: Desenvolvimento de Autenticação e CRUD de Despesas.
* Semanas 7-9: Fluxo de Aprovações e OCR.
* Semanas 10-11: Dashboards e Relatórios.
* Semanas 12-13: Documentação e Testes Finais.

**Fase 1:**

* Levantamento de Requisitos e Definição do Escopo (Semana 1)
* **Objetivos**:
  + Definir claramente os requisitos funcionais e não funcionais.
  + Identificar as principais partes interessadas e necessidades.
  + Estabelecer uma base de dados de requisitos para o desenvolvimento e testes.
* **Atividades:**
  + Requisitos Funcionais:
    - Sistema de autenticação (OAuth2/Single Sign-On).
    - Gestão de despesas (CRUD de despesas).
    - Aprovação de despesas (gerente e fluxos).
    - Upload de recibos (upload de arquivos e integração OCR).
    - Dashboards e relatórios (visualização de gráficos e geração de PDFs/Excel).
  + Requisitos Não Funcionais:
    - Performance (tempo de resposta < 1s para dashboards, escalar com 500+ usuários).
    - Segurança (dados encriptados, autenticação segura, logs de auditoria).
    - Escalabilidade (infraestrutura de microserviços).
    - Documentação detalhada.
  + Infraestrutura:
    - Definir ambiente de desenvolvimento e produção.
    - Uso de Docker para containerização.
    - Provisionamento de infraestrutura com Terraform (Azure ou AWS).
  + Plano de Gestão de Riscos:
    - Identificação de possíveis atrasos (ex: complexidade na integração OCR).
    - Mitigação: Divisão das funcionalidades críticas (microserviços), cronograma flexível.
* **Entregáveis:**
  + Documento de Requisitos Detalhado.
  + Diagrama da Arquitetura de Infraestrutura.
  + Plano de Riscos.

**Fase 2:**

* Setup de Infraestrutura e Ambientes (Semana 2-3)
* **Objetivos:**
  + Criar e configurar os ambientes de desenvolvimento, testes, e produção.
  + Garantir que a infraestrutura esteja preparada para CI/CD, Docker, e banco de dados.
* **Atividades:**
  + Configuração de CI/CD:
    - Setup no Azure DevOps ou GitHub Actions.
    - Pipeline de build e deploy para ambiente de desenvolvimento.
  + Docker e Containerização:
    - Criação de containers para o front-end, back-end, banco de dados.
    - Definição do docker-compose para ambientes locais.
  + Infraestrutura como Código:
    - Criar scripts Terraform para provisionar o ambiente cloud (servidores, storage, balanceadores).
  + Banco de Dados:
    - Definir schema inicial no SQL Server/Postgres.
    - Criação de tabelas iniciais (Usuários, Despesas, Aprovações, Logs).
* **Entregáveis:**
  + Pipeline de CI/CD funcional.
  + Docker-compose configurado.
  + Infraestrutura provisionada no cloud.
  + Banco de dados estruturado.

**Fase 3:**

* Desenvolvimento do Módulo de Autenticação e CRUD de Despesas (Semana 4-6)
* **Objetivos:**
  + Implementar a base da aplicação com autenticação segura e funcionalidade de CRUD de despesas.
  + Definir modelo de dados e criar as interfaces do usuário.
* **Atividades:**
  + Autenticação:
    - Implementar OAuth2/Single Sign-On com IdentityServer4 ou Auth0.
    - Criar lógica de registro, login, logout e gerenciamento de sessões.
  + CRUD de Despesas:
    - Back-end: Implementar APIs RESTful para CRUD de despesas.
    - Front-end: Interfaces para visualização, adição, edição, e exclusão de despesas.
  + Validações e Testes:
    - Unit tests para garantir a integridade das APIs.
    - Testes funcionais para CRUD completo.
* **Entregáveis:**
  + Sistema de login e registro funcional.
  + APIs RESTful para CRUD de despesas.
  + Testes unitários e funcionais para a primeira versão.

**Fase 4:**

* Implementação de Fluxos de Aprovação e OCR (Semana 7-9)
* **Objetivos:**
  + Adicionar funcionalidade de aprovação de despesas e integração de OCR para análise de recibos.
* **Atividades:**
  + Fluxo de Aprovações:
    - Definir o fluxo de aprovação (gerentes recebem notificações, aprovam ou rejeitam).
    - Implementar lógica de aprovação/rejeição no back-end e front-end.
  + Upload de Recibos e OCR:
    - Implementar funcionalidade de upload de recibos.
    - Integrar API OCR (ex: Tesseract ou uma solução SaaS) para extrair dados dos recibos.
  + Notificações:
    - Utilizar SignalR para notificações em tempo real para gerentes sobre novas despesas.
* Entregáveis:
  + Módulo de aprovação funcional.
  + Upload de recibos com OCR integrado.
  + Notificações em tempo real.

**Fase 5:**

* Dashboards e Relatórios (Semana 10-11)
* **Objetivos:**
  + Criar dashboards e relatórios detalhados para análise de despesas.
* **Atividades:**
  + Dashboard:
    - Utilizar gráficos dinâmicos (com bibliotecas como Chart.js) para exibir tendências de despesas.
  + Relatórios:
    - Gerar relatórios em PDF e Excel com as informações filtradas por período, categoria, etc.
  + Testes de Performance:
    - Verificar tempos de resposta das queries e otimizar o banco de dados, se necessário.
* **Entregáveis:**
  + Dashboard funcional e interativo.
  + Relatórios exportáveis para PDF/Excel.
  + Performance otimizada.

**Fase 6:**

* Documentação e Entrega Final (Semana 12-13)
* **Objetivos:**
  + Garantir que todo o sistema esteja documentado, testado, e pronto para entrega.
* **Atividades:**
  + Documentação:
    - Documentar todas as APIs e módulos (swagger, README, etc.).
    - Documentar infraestrutura e uso de CI/CD.
  + Testes Finais:
    - Realizar testes de aceitação com base nos requisitos levantados.
    - Corrigir eventuais bugs.
  + Deploy Final:
    - Deploy no ambiente de produção provisionado.
* **Entregáveis:**
  + Sistema completo e funcional.
  + Documentação detalhada para todas as partes do projeto.
  + Projeto pronto para ser apresentado como portfólio.