UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ

Universidade Federal do Ceará - *campus* Quixadá

Projeto Integrado em Engenharia de Software I

Documento de Mudanças - Sistema de Gestão Financeira

Docente:
Jefferson Kennedy

Discentes:

Antônio Rewelli de Oliveira, 554047;
Giliardy Alves da Silva, 552752;
Lucas Ferreira Nobre, 554590;
Miqueias Bento da Silva, 553972.

Av. José de Freitas Queiroz, 5003, Quixadá - CE, 63902-580

14.09.2024

1. Planejamento do Processo

- **Replanejamento do processo de desenvolvimento:**

  Implementamos práticas ágeis para melhorar o fluxo e a organização do projeto, adotando as seguintes cerimônias do Scrum:

  - **Sprint Planning:** Planejamento das tarefas a serem desenvolvidas em cada sprint, sendo realizada no sábado a tarde, com prioridade definida no **Product Backlog**.
  - **Daily Meetings (Dailies):** Reuniões diárias rápidas para acompanhar o progresso, discutir bloqueios e alinhar os próximos passos, realizadas nas terças e quintas-feiras.
  - **Product Backlog:** Lista completa de funcionalidades priorizadas com base nas necessidades do cliente e requisitos levantados.
  - **Sprint Backlog:** Seleção de itens do **Product Backlog** que serão desenvolvidos durante a sprint atual.

  Foi definido que o tempo para cada spring de desenvolvimento do sistema deve durar duas semanas,

  Distribuição das Atividades: A equipe foi dividida em:

  - **Backend:** Dois membros responsáveis pela implementação da API, lógica de negócio e integração com o banco de dados.
  - **Frontend:** Dois membros responsáveis pelo desenvolvimento da interface do usuário (UI) e integração com a API.

- **Planejamento do Quadro Kanban no Trello:**
  - O **Product Backlog** foi mapeado no Trello com base nas prioridades e entregas esperadas.
  - Criamos colunas no quadro Kanban para gerenciar o progresso:
    - **Backlog do projeto - Documentação e validação**
    - **Backlog do projeto - Desenvolvimento**
    - **Sprint Backlog**
    - **Desenvolvimento**
    - **Concluído - Sprint atual**
    - **Concluído - Anterior**

- **Planejamento do Cronograma:**

  O cronograma do projeto foi estruturado a partir do quadro Kanban, com prazos definidos para cada sprint. Cada atividade do **Product Backlog** foi ou será associada a responsáveis e deadlines, garantindo uma visão clara do progresso e entregas.

---

2. Tecnologias

- **Uso do Electron para construção da aplicação:**

  Após avaliação das tecnologias disponíveis, decidimos **definitivamente** utilizar **Electron** e **React.js** para a construção do frontend. As principais razões para essa escolha incluem:

  - **Qualidade e quantidade de documentação:** A extensa documentação facilita a resolução de problemas e aprendizado contínuo.
  - **Suporte a bibliotecas de dashboards:** Grande número de bibliotecas disponíveis, permitindo a construção de interfaces mais **interativas** e **intuitivas**, essenciais para um sistema de gestão financeira.

- **CSS Modules (Substituindo o Tailwind):**

  Optamos por **CSS Modules** para o gerenciamento de estilos no frontend devido às razões:

  - Facilidade de uso e configuração, o que reduziu o tempo gasto com problemas de ambiente enfrentados anteriormente com o **Tailwind CSS**.
  - Melhor encapsulamento e modularidade dos estilos, facilitando a manutenção e evitando conflitos de classes.

- **Banco de Dados e Docker Compose:**
  - Decidimos usar o **Docker Compose** com o **PostgreSQL** ao invés de configurar o PostgreSQL isoladamente.
  - **Motivo:** Cada desenvolvedor pode subir o ambiente do banco de dados de forma rápida e uniforme, sem necessidade de criar o banco localmente ou configurar o PostgreSQL manualmente. Isso garante consistência entre os ambientes de desenvolvimento.
  - Decidimos também criar dois ambientes diferentes com o Docker, os ambientes de ***developer*** e ***staging***, o motivo foi tentar diminuir a possibilidade de conflitos durante *merges* das branchs com diferentes configurações.

- **Hospedagem da API:**

  Decidimos que a API será hospedada em um serviço de **cloud computing**, no caso os serviços de nuvem da Oracle, devido razões como:

  - **Facilidade de escalabilidade;**
  - **Disponibilidade e acessibilidade;**
  - **Integração simplificada com Docker;**

---

3. Requisitos

- **Requisito de Transação Fixa:**

  Definimos que a **Transação Fixa** será totalmente fixa, oferecendo ao cliente as seguintes opções:

  - Excluir apenas uma transação específica.
  - Excluir as próximas transações associadas.
  - Excluir todas as transações recorrentes referentes à transação fixa.

  Vale ressaltar que para realização desse requisito, será necessário um script para identificar o novo mês e cadastrar automaticamente a transação.

- **Requisito de Exclusão de Categoria:**

  Não será possível excluir uma **Categoria** enquanto ela estiver sendo utilizada por uma ou mais transações, garantindo a **integridade dos dados** e evitando inconsistências no sistema.

- **Requisito de Exclusão de Conta:**

  Similar à exclusão de categoria, não será permitido excluir uma **Conta** que esteja associada a transações existentes.
  No entanto, o usuário terá a opção de:

  - Excluir a conta e, **automaticamente**, todas as transações vinculadas a ela.
  - Esse comportamento foi definido para manter a integridade dos dados e oferecer mais flexibilidade ao usuário.

- **Requisito de Metas:**

  A partir de reuniões com os stakeholders e usuários que testaram o sistema e até por complicações no desenvolvimento frontend, decidimos que para a primeira implementação do sistema o requisito de “metas” não será implementado. Contudo, para as disciplinas de Gerência e Configuração e Projeto de Detalhado de Software tivemos que fazer sua implementação no backend, a fim de utilizar padrões de projeto, pedido por Projeto Detalhado, e requisito de implementação em Gerência de Configuração.

- **Requisito de Orçamentos:**

  Por questões de perda de desempenho quando precisávamos atualizar o progresso de um orçamento decidimos atualizar somente quando o usuário o solicitava na tela da aplicação. Antes tínhamos que a cada nova transação ou atualização, verificar se havia um orçamento destinado para aquela transação, podendo fazer um grande percurso na base de dados do sistema, e depois tenho que iterar todas as transações do orçamento para que seus valores fossem atualizados.

- **Transações de Metas:**

  A fim de não interferir na estrutura de uma transação comum, tivemos que criar uma nova transação para “Metas”.