**Itens de um Plano de Projeto**

**Histórico da Revisão**

| **Data** | **Versão** | **Descrição** | **Autor** |
| --- | --- | --- | --- |
| 20/03/2024 | 1.0 | definição de escopo e arquitetura | Miguel Prates |
| 17/04/2024 | 1.1 | gerência de risco, cronograma | Miguel / Allan / Téo |
| 22/04/2024 | 1.2 | cenários de testes e revisão de terminologias | Téo |
| 22/04/2024 | 1.3 | gerenciamento de configuração de software | Lucas |
|  |  |  |  |

1. **Identificação**
   1. AdoCÃO
   2. 5A
   3. 20.03.2024
2. **Introdução**

O projeto AdoCão consiste no planejamento e no desenvolvimento de um software para permitir o cadastro e a divulgação de animais disponíveis para adoção. Este programa será um aplicativo web que receberá, de grupos de resgates, ONGs e usuários da plataforma, informações detalhadas sobre animais necessitados, e as disponibilizará publicamente, em uma interface intuitiva, para o público geral, de forma a facilitar o processo de busca por possíveis adotantes ou contribuintes para as ONGs cadastradas.

1. **Escopo do projeto**
   1. Requisitos Funcionais

* O usuário deve ser capaz de se cadastrar na plataforma.
* O administrador deve ser capaz de cadastrar uma ONG e seus gerentes.
* O gerente da ONG deve ser capaz de adicionar novos colaboradores a sua ONG.
* O colaborador deve ser capaz de adicionar novos animais à página da ONG.
* O usuário deve ser capaz de cadastrar animais para adoção e incluir características, o que pode incluir fotos e descrições.
* Deve ser possível manifestar interesse em um animal por usuários
  1. Requisitos Não Funcionais
* O custo de hospedagem do projeto não deve exceder 50 reais mensais.

1. **Equipe e infraestrutura**

A equipe é composta por seis membros, dentre os quais Allan Garcia exerce função de Project Owner e Miguel Prates exerce função de Scrum Master. Os nomes dos membros da equipe são mostrados abaixo:

* Arthur Queiroz Moura
* Miguel Prates Ferreira de Lima Cantanhede (SM)
* Allan Garcia Cavalcante e Silva (PO)
* Lucas de Oliveira Ferreira
* Bruno Matheus Foschiani Ricardo
* Téo Sobrino Alves

Em questão de equipamentos e ferramentas, utilizamos computadores/notebooks para o desenvolvimento do software. Não dispomos de equipamento para hospedagem própria da aplicação, porém poderemos alugar em algum serviço terceirizado, como Hetzner, Vultr etc. Após finalizado o desenvolvimento, a gerência de custos será realizada pelo cliente. Para ferramentas de software, priorizamos aquelas atreladas ao desenvolvimento web, tais como:

* Vue.js
* Nuxt.js
* Tailwind CSS
* Express.js

1. **Acompanhamento do projeto**

Esta seção do plano de projeto relaciona os momentos para realização das atividades de verificação do projeto, as quais poderão ser feitas pela equipe técnica das instituições envolvidas (desenvolvedora e cliente), e também a forma como estas atividades serão realizadas. Estas atividades incluem a realização de reuniões e geração de relatórios descrevendo informações sobre o progresso do projeto. Descrever quais reuniões serão realizadas, como o projeto será acompanhado pela equipe e pelo cliente.

1. **Cronograma e Marcos do projeto**

Cronograma disponível em: [Projeto AdoCÃO - Cronograma](https://docs.google.com/spreadsheets/d/1kBATkDuUiH52RlAp6ns5kbXQSAjhq2bCzzpAi686rrs/edit?usp=sharing).

1. **Gerência de Riscos**

| **Descrição** | **Gravidade** | **Contingência** |
| --- | --- | --- |
| Mudança de escopo do projeto durante interações com o cliente. | Baixo, se ocorrido antes do início do desenvolvimento.  Médio, caso contrário. | O documento de *user stories* deve ser alterado. Os protótipos de alta e baixa fidelidade devem acomodar as mudanças.  A fim de evitar, deve-se focar em alinhar as expectativas com o cliente o quanto antes, preferencialmente usando protótipos gráficos, pois uma mudança em estágios posteriores implica em maior atraso. |
| Falta de tempo para finalizar o projeto, especialmente o frontend. | Médio. | As atividades restantes devem ser documentadas no repositório do projeto, a fim de permitir melhor contribuição de futuros contribuidores. |

1. **Testes do produto (ou sistema)**

Cenários de teste disponíveis em: [Cenários de Teste - AdoCão](https://docs.google.com/document/d/1KXtrDI6uCKWDz0ko5ePgtMEJrQ2TJf3N0ss-iy33wiM/edit)

9. **Gerenciamento de configuração de software**

Este item apresenta: i) uma descrição da padronização da nomenclatura associada aos itens de configuração que serão mantidos sob gerenciamento de configuração; ii) o mecanismo utilizado para controle de versão; e iii) as dependências identificadas entre os itens de configuração.

O nosso controle de versão será organizado por meio do nosso repositório no Github, a tabela abaixo é uma versão simplificada de como vai ocorrer o gerenciamento de configuração de software, a versão completa pode ser conferida em [CS - AdoCÃO](https://docs.google.com/document/d/19KbNT-K1lz_4igXvzySxY-D8dyfu6lmsdwJ3dF1Nac0/edit?usp=sharing).

| Item de Configuração | Nomenclatura | Controle de Versão | Explicação dos Graus (Controle de Versão) | Dependências |
| --- | --- | --- | --- | --- |
| Código Fonte | CF | Versão X.X.X | 1º grau: Novos Arquivos  2º grau: Grandes Alterações  3º grau: Pequenas Alterações | CCU |
| Cenários de Caso de Uso | CCU | Versão X.X.X | 1º grau: Completude Momentânea  2º grau: Criação, Remoção e Grandes Modificações de Casos de Uso  3º grau: Pequenas Modificações | US |
| Cenários de Casos de Teste | CCT | Versão X.X.X | 1º grau: Completude Momentânea  2º grau: Alterações Significativas  3º grau: Pequenas Alterações | CCU, PP |
| Plano de Projeto | PP | Versão X.X | 1º grau: Completude Momentânea  2º grau: Preenchimento e Alterações | CCU, CCT, CS |
| User Stories | US | Versão X.X | 1º grau: Remoções e Adições  2º grau: Alterações |  |
| Configuração de Software | CS | Versão X.X | 1º grau: Novos Itens de Configuração  2º grau: Alterações |  |

10. **Plano de Manutenção de software**

O Plano de manutenção de software:

Enquanto não houver código fonte sendo feito o plano de manutenção será de apenas preventivo, no atual momento o plano recaí apenas sobre o sistema, é importante que todo o grupo analise e valide os casos de uso e cenários de teste, a fim de encontrar possíveis omissões ou incompletudes, pois isso fará com que o desenvolvimento do software seja mais robusto, com menor possibilidade de falhas e menor perda de tempo com manutenção.

Desenho do plano de manutenção inicial:

Como minimizar a necessidade de cada outro tipo de manutenção e fazê-las no menor tempo:

Corretiva: Produção de código sem bugs, mantendo sempre estruturas claras que encorajam um desenvolvimento limpo e lógico, pode ser obtido com a escolha das melhores estruturas de controle, fluxo e dados. Para realizar manutenção corretiva o ideal é uso de ferramentas de debug.

Adaptativa: Escolher cuidadosamente o stack de tecnologia e verificar que o projeto pode ser executado com ele, analisando ser um projeto estável, que continuará tendo suporte. A manutenção adaptativa pode ser muito custosa, a melhor forma seria uso de padrões portáveis.

Perfectiva: Recolher os requisitos de forma cuidadosa, para evitar muitas mudanças. A manutenção perfectiva no caso de adicionar/mudar novas features será como o desenvolvimento comum.

As representações gráficas ficarão no git, pois estão relacionadas com o versionamento do software, definindo para cada membro sua posição e papel em cada etapa da manutenção, tal versionamento pode gerar uma tabela de relações, porém isso se dará apenas com o início do desenvolvimento, para isso é interessante padronizar as mensagens de commit no github.

**Referências**

Apresenta-se uma relação dos documentos pertinentes ao projeto.