



Aplicativo de facilitação de adoção de animais

Integrantes: David Vilaça, Ícaro Duarte, Lucas Gomes, Tadeu Junior

Documento Técnico

Projeto: Adotei

SUMÁRIO

| | |
|--|----------|
| 1. Controle de versão | 2 |
| 1.1 Versionamento | 2 |
| 2. Automatização e testes | 2 |
| 2.1 Processo de release | 2 |
| 2.2 Testes unitários | 3 |
| 2.3 Testes de integração | 3 |
| 3. Diagramação | 3 |
| 4. Linguagens de programação e frameworks | 3 |
| 4.1 Front-end | 3 |
| 4.2 Back-end | 3 |
| 5. Abordagem de autenticação | 4 |
| 6. Padronizações | 4 |
| 6.1 Commits | 4 |
| 6.2 Linters | 4 |
| 7. Artefatos | 4 |
| 8. Referências | 5 |



1. Controle de versão

Controle de versão é um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa lembrar versões específicas mais tarde. Para este fim utilizaremos o [Git](#). Com o git conseguimos ter um histórico desde o início do projeto, podendo voltar em qualquer versão específica e sabendo quais alterações e quem foi o autor.

Utilizaremos também o [Github](#), que é uma plataforma de hospedagem de código-fonte. Ele utiliza o controle de versão Git e permite que quaisquer usuários cadastrados na plataforma contribuam com projetos privados e/ou de código aberto.

Link do projeto no Github: <https://github.com/Angu-chuleta/adotei>.

1.1 Versionamento

O versionamento é um processo de atribuir um nome único para indicar o estado de um software. Neste projeto adotaremos o [Versionamento Semântico 2.0](#), visto que será muito provável uma API pública já que se trata de um software que tem como propósito contribuir com ONGs.

2. Automação e testes

Automações são necessárias para se obter uma melhor produtividade e uma melhor padronização de processos. Da mesma forma os testes, seja de forma manual ou automatizada, são necessários para garantir uma melhor qualidade em relação ao contexto em que o software deve operar.

Neste projeto iremos aplicar 2 tipos de teste: o unitário e de integração. Ambos serão automatizados utilizando a ferramenta [Jest](#). A fim de automatizar e centralizar essas tarefas iremos utilizar o [CircleCI](#), que é uma plataforma com plano grátis de integração contínua.

2.1 Processo de release

Release, neste contexto, nada mais é que uma liberação de software, ou seja, um lançamento oficial de uma determinada versão deste projeto. Para evitar publicações manuais será desenvolvido uma integração com o CircleCI. Com essa integração vamos automatizar o processo de publicação para que seja publicada uma versão assim que uma [tag](#) for criada no repositório.



2.2 Testes unitários

Os testes unitários serão exclusivos do escopo de business, pois acreditamos que as regras de negócios devem ser tratadas de forma independente no projeto para não sofrer nenhum tipo de efeito colateral.

2.3 Testes de integração

Os testes de integração será de exclusividade do front-end, porque acreditamos que uma boa interação do usuário com o sistema fará total diferença. Para garantir essa boa interação e reduzir as possibilidades de bugs irem para produção, utilizaremos a ferramenta [QA Wolf](#) para testarmos o sistema a partir da interface, simulando um usuário interagindo.

3. Diagramação

Para documentar o sistema, criaremos um diagrama de caso de uso através da plataforma [online.visual-paradigm](#), usando histórias de usuário, Diagrama de classe e um diagrama de arquitetura, tendo esses documentos básicos podemos analisar a viabilidade de outros assim que percebermos a necessidade.

4. Linguagens de programação e frameworks

As linguagens de programação utilizadas serão [TypeScript](#) e [JavaScript](#). O TypeScript é um superset do JavaScript onde é adicionado tipagem, decorators e alguns utilitários que o JavaScript tradicional não suporta nativamente.

4.1 Front-end

O front-end da aplicação será desenvolvido utilizando [JavaScript](#), framework [React](#) e para controle de versão e armazenamento do código utilizaremos o Github.

| Propósito | Nome | Descrição |
|--------------------------|----------------------------|---|
| Linguagem de Programação | JavaScript | JavaScript é uma linguagem de programação de alto |

| | | |
|-------------------|---------------------------|---|
| | | nível criada, a princípio, para ser executada em navegadores e manipular comportamentos de páginas web. |
| Framework | React | Biblioteca JavaScript para criação de interfaces para o usuário, desenvolvida e mantida pelo Facebook. |
| Hospedagem | github.io | Página estática provida pelo github gratuitamente. |

*Os links das ferramentas e frameworks estão linkados com os nomes dos mesmos.

4.2 Back-end

O back-end da aplicação será desenvolvido utilizando a linguagem [TypeScript](#), utilizando a runtime [NodeJS](#), framework [Express](#) para construção de uma API REST. Também utilizaremos [MongoDB](#) para armazenamentos dos dados da aplicação, framework de persistência, usaremos o [Mongoose](#) e como no front-end utilizaremos o Github para armazenar o código fonte e controle de versão.

| Propósito | Nome | Descrição |
|---------------------------------|----------------------------|---|
| Controle de Versão | Github | O GitHub é um site que abriga um software de controle de versão de desenvolvimento através do sistema Git. |
| Linguagem de Programação | NodeJS | Plataforma de aplicação, na qual você escreve seus programas com Javascript que serão compilados, otimizados e interpretados pela máquina virtual V8. |
| Linguagem de Programação | TypeScript | O TypeScript é um superset do JavaScript onde é adicionado tipagem, decorators e alguns utilitários que o JavaScript |

| | | |
|-------------------|---------------------------------|---|
| | | tradicional não suporta nativamente. |
| Framework | <u>Express</u> | O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e mobile. |
| SGBD | <u>MongoDB</u> | MongoDB é um banco de dados que pertence a um grupo de banco de dados não relacionais (NoSQL). No MongoDB, cada registro é armazenado em um documento JSON (JavaScript Object Notation), esses documentos são compostos de campos, agrupados em coleções. |
| Framework | <u>Mongoose</u> | Biblioteca de modelagem de dados do objeto (ODM) que fornece um ambiente de modelagem rigoroso para seus dados. Utilizado para interagir com o MongoDB, assim fornecendo conveniência no gerenciamento dos dados. |
| Hospedagem | <u>Heroku</u> | Heroku é uma plataforma em nuvem como um serviço que suporta várias linguagens de programação. |

*Os links das ferramentas e frameworks estão linkados com os nomes dos mesmos.

5. Abordagem de autenticação

Será adotado como abordagem de autenticação o [JWT](#). Toda requisição HTTP autenticada será por meio de um token JWT no qual se encontrará no cabeçalho (*Authorization*) da requisição.



6. Padronizações

Serão adotados algumas regras e padronizações com o objetivo de se obter um código fonte mais uniforme, sem uma identidade pessoal de cada programador.

6.1 Commits

Utilizaremos uma convenção baseada na [Convenção Angular](#) de commits: o [Conventional Commits](#). Ela separa um conjunto de regras para um melhor histórico de commits e se encaixa perfeitamente com o SemVer, podendo automatizar a criação de [changelogs](#) (registro de alterações).

De forma simples o texto do commit será separado em algumas partes, são elas: tipo, escopo, descrição, corpo e rodapé.

```
<type>[optional scope]: <description>
[<blank line>
optional body]
[<blank line>
optional footer(s)]
```

6.2 Linters

Para assegurar de que o código será escrito com um mesmo estilo por todos os programadores utilizaremos o [ESLint](#), que é uma ferramenta de análise de código estático para identificar padrões problemáticos no código e/ou checar o estilo. O estilo adotado será o [StandardJS](#).

7. Artefatos

Os artefatos estão disponíveis no github através do link:

<https://github.com/Angu-chuleta/adotei/tree/master/Artefatos>



8. Referências

Controle de versão GIT:
<https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Sobre-Controle-de-Vers%C3%A3o>

O que é Open Source:

<https://canaltech.com.br/produtos/O-que-e-open-source>

Versionamento semântico:

<https://semver.org/lang/pt-BR>

Versionamento de software:

https://pt.wikipedia.org/wiki/Versionamento_de_software

CircleCI:

<https://circleci.com/docs/2.0/about-circleci>

CI/CD:

https://medium.com/@gabriel_faraday/o-que-%C3%A9-ci-cd-onde-eu-uso-isso-57e9b8ad8c73

Teste de software:

https://pt.wikipedia.org/wiki/Teste_de_software

Introdução ao TypeScript:

<https://www.devmedia.com.br/introducao-ao-typescript/36729>

Como funciona o JWT:

<https://www.devmedia.com.br/como-o-jwt-funciona/40265>

Arquitetura MVC:

<https://pt.wikipedia.org/wiki/MVC>

O que é changelog:

https://pt.wikipedia.org/wiki/Registro_de_altera%C3%A7%C3%B5es

O que é ESLint:

<https://en.wikipedia.org/wiki/ESLint>

Diagramas de classe:

https://pt.wikipedia.org/wiki/Diagrama_de_classes