Projeto - Padronização

Git

Código

Todo o código deve ser escrito em inglês utilizando ou *Snake_Case* ou *CamelCase* (não é recomendado utilizar os dois).

Commit

Os *commits* serão estruturados da forma:

```
<tipo>[escopo (opicional)]: <descrição>

[corpo (opcional)]

[rodapé (opcional)optional]
```

E serão padronizados em inglês.

Tipo

Os *commits* podem ser do tipo:

- **feat**: Uma nova *feature* (implementação).
- **fix**: Correção de algum *bug* ou funcionalidade.
- **docs**: Mudança na documentação do projeto.
- **refactor**: Refatoração de código (não corrige algum *bug* nem adiciona uma *feature*, apenas estrutura do código).
- **chore**: Alteração na estrutura de pastas/configurações.
- **style**: Mudança que afeta apenas a estilização do código (espaços em branco, formatação, ponto-vírgula, etc.).
- **test**: Adição/correção/remoção de algum teste.
- **perf**: Mudança que altera a perfomance do código.
- **revert**: Caso precise reverter algum *commit*

Corpo

Caso seja necessário, pode-se detalhar a atividade efetuada, explicando **porquê** do *commit*.

Rodapé

Se o projeto estiver trabalhando com *issue tracker*, a *issue* pode ser fechada na última linha

Exemplos

```
feat: add english language
```

```
docs: correct user manual
```

```
fix: adjust function to parse arrays

A function in controller was throwing an exception.

Resolves: #2
```

Branch

As *branches* serão criadas a partir de uma nova atividade do tipo *feature* ou *fix*, caso as outras atividades sejam grandes também será necessário ter uma nova branch, fora isso elas podem ser *commitadas* diretamente na *master*

O nome da *branch* será estruturada da forma:`

```
<tipo>-<atividade>
```

Exemplos

```
git branch feature-crud-user
```

```
git branch fix-user-controller
```

```
git branch docs-user-manual
```

Pull Request

Quando uma atividade do tipo *feature* ou *fix* é finalizada, a mesma entrará no estado de **Code Review**, conforme estipulado no *Scrum*, e assim será criado um *pull request* para *master* e outra pessoa irá analisá-la e realizar o *merge* quando estiver pronto.

Neste momento é recomendado a criação de testes unitários para estas atividades.

Banco de Dados

Para utilizar o banco de dados com Docker é necessário ter instalado:

- Docker
- Docker-Compose

Após a instalação, a sua utilização é da seguinte forma:

- Navegue para dento do diretório `database`.
- Altere as permissões do diretório `data`
  - `sudo chmod 777 -R data/` (Linux)
- Dentro do diretório `database` execute o comando no terminal
  - `docker-compose up -d`
- O pgAdmin4 rodará na url `http://localhost:5000`
- Use as credenciais para logar no pgAdmin4
  - `username: postgres`
  - `password: 123`
- Use as credenciais para acessar o banco de dados
  - `username: postgres`
  - `password: p0stgr3s`