|  |  |  |  |
| --- | --- | --- | --- |
|  | **GIT FLOW** | | |
| **MBA** | Engenharia de Dados | | |
| **DISCIPLINA** | *DataOps* | **TURMA** | 11ABD |
| **INTEGRANTES**  **DO GRUPO** | RM 339199 – Bruno Campos de Sousa  RM 338268 – Joyce Miyazato  RM 336926 - Igor Queiroz Ventura  RM 338486 – Marcelo Smid de Campos  RM 338892 – Renato Tavares da Silva  RM 338496 – Victor Ferreira Telezynski Lopes | | |

* Link Repositório GIT: <https://github.com/renatotvs/g5dataops>

**Benefícios**

É fundamental que em qualquer instituição que possua equipe de desenvolvimento tenha uma ferramenta para controle de versão de seus códigos. Não basta apenas ter a ferramenta, também é necessário ter um pattern de utilização da mesma e, por isso, vamos tratar um pouco aqui do Git Flow, um pattern que, além da coincidência do prefixo Git, pode muito bem ser usado sobre sistema de controle de versão homônimo (em relação ao prefixo Git) ou até mesmo em qualquer um outro (Ex.: SVN-Subversion, etc).

O desafio é o mesmo: Como manter um controle estável de versões de arquivos que estão sendo alterados por diversas pessoas ao mesmo tempo? No caso de códigos de software, além manter estáveis é necessário mantê-los funcionais também, certo? O Git Flow vem para ajudar com isso.

O pattern Git Flow define muito bem a utilização do modelo de branches, inclusive nomeando-as, tais como:

* Branch main (também conhecida como master):
  + Branch principal do componente em questão. Reflete o código do software que está em produção;
* Branch develop:
  + Recebe os códigos das features branches e consequentemente contém os códigos da próxima release a ser implantada em produção (branch main), ou seja, quando a funcionalidade está pronta é dessa branch que parte o merge para a master;
* Branch feature:
  + É uma branch criada para desenvolver uma nova funcionalidade e, quando concluída, deve ser excluída após seu merge com a branch develop;
* Branch release:
  + Branch “pré-master”. É a branch que, a partir da develop, faz os ajustes finos de metadados. Essa branch também libera a develop para receber novas features;
* Branch hotfix:
  + Branch para correção de erros identificados no ambiente de produção;

Dado seu modelo muito bem definido de trabalho, muitas empresas adotaram esse pattern o que tornam onboarding de novos desenvolvedores mais facilitado, pois o modelo de trabalho é independente da empresa. Os principais benefícios em usar o Git Flow são:

* A integração fo Git Flow com outros patterns de desenvolvimento possibilita melhor controle de qualidade de código. Ex.: Testes automatizados, esteiras CI/CD, etc;
* Gera independência no desenvolvimento de novas features, dado que seu modelo de feature branch cria um repositório exclusivo para esse fim;
* Melhor organização e, por ser um pattern de mercado, mais fácil na hora de fazer o onboarding de novos desenvolvedores;
* Gera maior visibilidade para problemas em produção e isola o ambiente de desenvolvimento da correção destes por uma equipe de sustentação, se for o caso;
* Reforça a abordagem de code review nos processos de PR (pull requests) entre as branchs;

**Justificativa**

Diante de todos os benefícios listados anteriormente, neste projeto vamos utilizar o pattern Git Flow para melhor gestão do códigos. Não foi falado nos benefícios acima, mas nosso time atua na abordagem ágil e temos diversas features a serem desenvolvidas, portanto entendemos também que esse modelo ajuda bastante o time de desenvolvimento na zeladoria do código para melhor qualidade do software.