Reunião 29/08

Participantes: Rayane, Nildemar, Gerson e Nathan

Explicado como será a organização do projeto. Que será utilizado para separar o que está em produção, em testes e em desenvolvimento.

Estrutura de Branches no GitHub

- **Main**
  - Ambiente de teste chamado: **Production**
  - Contém o código em produção
  - Só recebe código que já passou por testes da Stable
  - Versão estável e liberada para os usuários finais. (Versão final)

- **Stable**
  - Ambiente de teste chamado: **Staging**
  - Contém código testado para subir para produção
  - Mais atualizada e confiável que *develop*
  - Serve como base para criar novas features ou correções*,* criar novas branches (fazer pull e criar branch a partir dele e etc).
  - Contém menos bugs, validada pelos testes.

- **Develop** : Código em **desenvolvimento**, sujeito a testes
  - Ambiente de teste chamado: **Homologação**
  - Onde o desenvolvimento acontece, mas possui código não testado completamente
  - Local onde novas funcionalidades são integradas antes de serem promovidas para *stable*.

Fluxo de Desenvolvimento

1. **Base inicial**
   - Sempre da branch **Stable**.
   - A *stable* contém o código testado, confiável e próximo da produção.
   - Antes de criar qualquer coisa nova, dar **pull** da *stable* para garantir que está atualizado.
2. **Criar branch de feature/correção**
   - Criar uma branch separada para cada nova funcionalidade ou bug.
   - Essa branch é **criada a partir da stable**, garantindo que já tem o código mais confiável.
3. **Desenvolvimento e commits**
   - Fazer alterações na branch da feature.
   - O Código ainda **não vai para produção**, apenas desenvolvimento.
4. **Merge para develop**
   - Depois de finalizar a feature, abrir um **Pull Request (PR)** para a branch **develop**.
   - A *develop* é o ambiente de **homologação**, onde todas as novas funcionalidades são integradas.
   - Aqui o código passa por **testes automatizados e manuais**.
5. **Testes e ajustes**
   - CI/CD verifica se o código funciona corretamente.
   - Bugs encontrados em Develop são corrigidos na própria branch de feature ou em develop.
6. **Promoção para stable**

    - Quando o código está validado, o PR é feito da branch da feature para a stable.
    - Ambiente **Staging**: testes finais antes da produção.
7. **Promoção para main**
    - Depois de aprovado em stable, o PR vai de **stable para main**.
    - O código chega à produção **(Production)**.

CI/CD – Integração Contínua e Entrega Contínua

- Automatização do processo de atualização da aplicação em cada ambiente.

- **Pipeline**:

    1. Desenvolvedor cria/atualiza branch a partir da *stable*.

    2. O código passa por **verificações automáticas.**

    3. Se aprovado, vai para o ambiente de **homologação** (*develop*).

    4. Depois de validado, pode ser promovido para **Staging** (*stable*).

    5. Quando estiver pronto, é integrado ao **Production** (*main*).

Observação: Sempre ao realizar mudanças, enviar da sua branche criada para stable, nunca enviar direto da Developed para Stable.