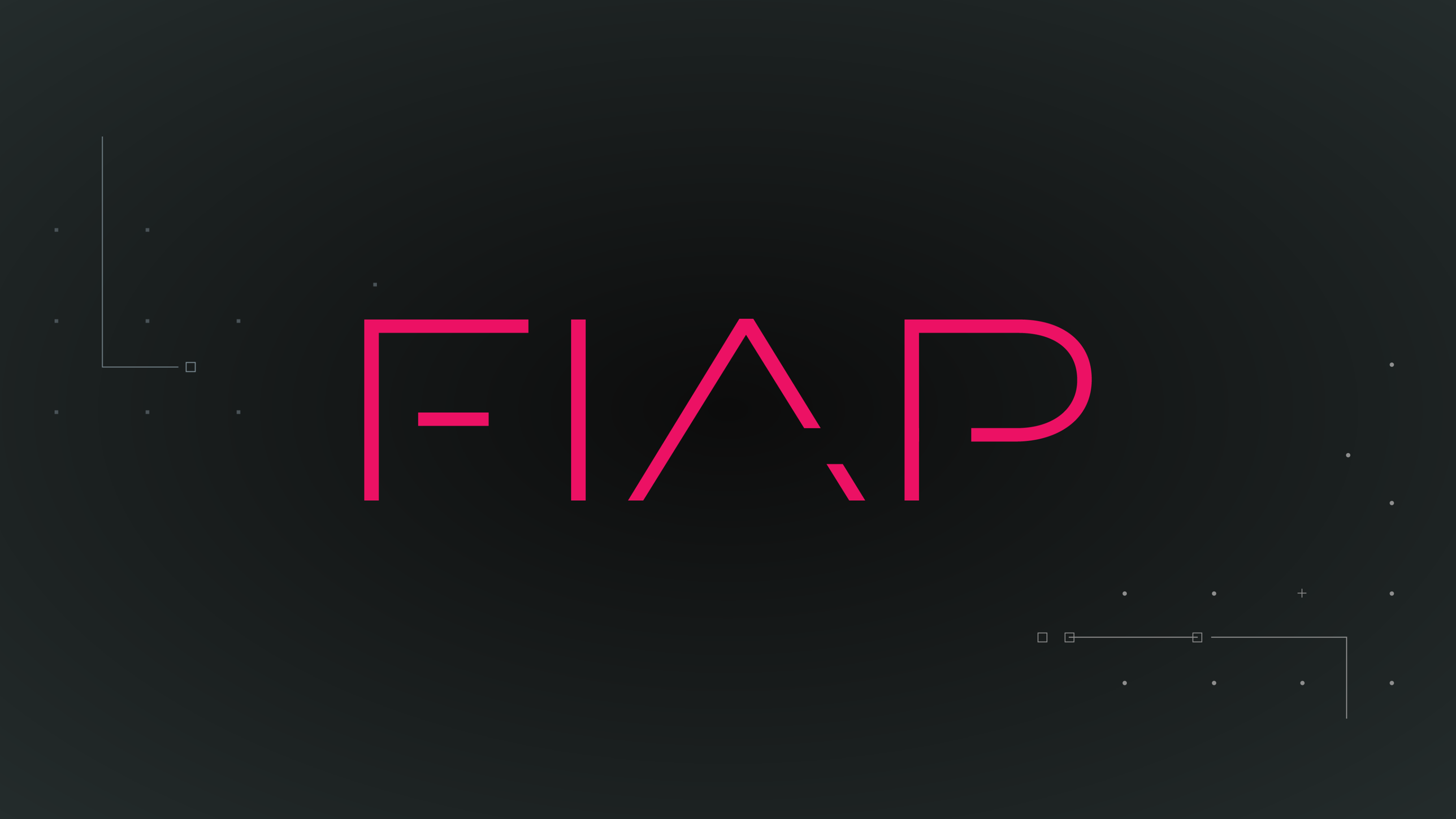
FIAP

HMAD

Hybrid Mobile App Development

Git Flow

Tutorial Completo de Git

Neste tutorial, apresentaremos o Git, uma ferramenta de controle de versão amplamente utilizada no desenvolvimento de software. Vamos dividir o conteúdo em tópicos para facilitar o aprendizado, e faremos exemplos em código em JavaScript para demonstrar as principais funcionalidades do Git.

Tópicos

1. Introdução ao Git
2. Configuração Inicial
3. Repositórios Git
4. Trabalhando com Commits
5. Branches
6. Fusão (Merge)
7. Conflitos de Merge
8. GitHub e Repositórios Remotos
9. Pull Requests
10. Clone e Fork
11. Ignorar Arquivos
12. Desfazendo Alterações
13. Histórico e Logs

1. Introdução ao Git

O Git é um sistema de controle de versão distribuído que permite o rastreamento de mudanças em arquivos e facilita a colaboração entre desenvolvedores. Ele é amplamente utilizado para gerenciar projetos de software e é especialmente útil para controlar o histórico de alterações em código-fonte.

1. Introdução ao Git

Exemplo em JavaScript (Iniciando um repositório Git)

Para começar a utilizar o Git em um projeto JavaScript, abra um terminal na pasta do projeto e execute os seguintes comandos:

```
# Inicializar um repositório Git
$ git init
```

2. Configuração Inicial

Antes de começar a usar o Git, é importante configurar seu nome de usuário e email. Isso é necessário para identificar quem fez cada alteração no projeto.

Exemplo em JavaScript (Configuração Inicial):

```
# Configurar nome de usuário
$ git config --global user.name "Seu Nome"

# Configurar email
$ git config --global user.email "seu.email@example.com"
```

3. Repositórios Git

Um repositório Git é um diretório onde todas as informações sobre o projeto são armazenadas, incluindo o histórico de alterações e branches.

Exemplo em JavaScript (Criando um arquivo):

Crie um arquivo chamado `script.js` com o seguinte código:

```
// script.js
console.log("Olá, mundo!");
```

4. Trabalhando com Commits

Commits são unidades básicas de alterações no Git. Eles são usados para registrar mudanças no projeto e criar um histórico claro de desenvolvimento.

Exemplo em JavaScript (Fazendo um Commit):

```
# Adicionar o arquivo ao índice (staging area)
$ git add script.js

# Criar um commit com uma mensagem descritiva
$ git commit -m "Adicionar saudação em JavaScript"
```

5. Branches

Branches são ramificações do desenvolvimento que permitem que você trabalhe em funcionalidades separadas sem interferir no código principal. Eles são úteis para desenvolvimento paralelo e testes.

Exemplo em JavaScript (Criando e Mudando de Branch):

```
# Criar uma nova branch chamada "feature"
$ git branch feature

# Mudar para a branch "feature"
$ git checkout feature
```

6. Fusão (Merge)

Fusão é o processo de combinar alterações de uma branch para outra. Isso é feito para incorporar as alterações de uma funcionalidade concluída de volta à branch principal.

Exemplo em JavaScript (Fusão de Branches):

```
# Mudar para a branch principal
$ git checkout main

# Fazer a fusão da branch "feature" para a branch principal
$ git merge feature
```

7. Conflitos de Merge

Conflitos de merge ocorrem quando duas branches têm alterações conflitantes no mesmo trecho de código. Nesse caso, é necessário resolver os conflitos manualmente antes de concluir o merge.

Exemplo em JavaScript (Conflito de Merge):

Suponha que a branch "feature" teve uma alteração no arquivo `script.js`, enquanto a branch "main" também teve uma alteração no mesmo arquivo, resultando em um conflito. O Git informará sobre o conflito, e você precisará resolver manualmente antes de concluir o merge.

8. GitHub e Repositórios Remotos

O GitHub é uma plataforma de hospedagem de repositórios Git na nuvem. Ela permite compartilhar projetos com outros colaboradores e facilita a colaboração.

Exemplo em JavaScript (Enviar o Repositório para o GitHub)

1. Crie um repositório vazio no GitHub.
2. Conecte o repositório local ao repositório remoto:

```
# Adicionar o repositório remoto (substitua "seu-nome-de-usuario" e "seu-repositorio" pelo seu nome de usuário e nome do repositório)
$ git remote add origin https://github.com/seu-nome-de-usuario/seu-repositorio.git
```

```
# Enviar o código para o GitHub
$ git push -u origin main
```

9. Pull Requests

Pull requests são solicitações para incorporar alterações de uma branch para outra. Eles são muito utilizados em projetos colaborativos para revisar e discutir mudanças antes de serem mescladas.

Exemplo em JavaScript (Pull Request):

1. Faça uma alteração em um arquivo no GitHub (por exemplo, adicione um novo arquivo chamado `funcao.js` com uma função simples).
2. Crie um pull request para mesclar a branch atual com a branch "main".

10. Clone e Fork

Clonar é fazer uma cópia exata de um repositório Git em um novo diretório, enquanto o fork é fazer uma cópia de um repositório Git na sua própria conta no GitHub.

Exemplo em JavaScript (Clone e Fork):

1. Clone um repositório existente:

```
# Clone um repositório (substitua "url-do-repositorio" pela URL do repositório)
$ git clone url-do-repositorio
```

2. Faça um fork de um repositório no GitHub clicando no botão "Fork" na parte superior da página do repositório.

11. Ignorar Arquivos

Arquivos e pastas que não devem ser rastreados pelo Git podem ser listados em um arquivo `.gitignore`. Isso é útil para evitar que arquivos gerados automaticamente ou contendo informações sensíveis sejam adicionados ao repositório.

Exemplo em JavaScript (.gitignore):

Crie um arquivo chamado `.gitignore` e adicione o seguinte conteúdo:

```
node_modules/
build/
```

12. Desfazendo Alterações

O Git permite desfazer alterações que ainda não foram confirmadas usando o comando `git reset` ou descartar mudanças em arquivos modificados usando `git checkout`.

Exemplo em JavaScript (Desfazer Alterações):

```
# Descartar todas as alterações em um arquivo não rastreado
$ git checkout -- script.js

# Desfazer todas as alterações em um arquivo já rastreado
$ git reset HEAD script.js
```

13. Histórico de Logs

O Git mantém um histórico completo de todas as alterações feitas ao longo do tempo. O comando `git log` mostra um registro detalhado do histórico de commits.

Exemplo em JavaScript (Visualizar o Histórico):

```
# Visualizar o histórico de commits
$ git log
```

Conclusãos

Este tutorial abrange os principais conceitos e funcionalidades do Git, que são essenciais para o desenvolvimento colaborativo de projetos em JavaScript. Com esse conhecimento, vocês poderão trabalhar de forma mais eficiente e segura em seus projetos.

Conclusãos

Lembre-se de que o Git é uma ferramenta poderosa, e a prática é fundamental para se familiarizar com todos os seus recursos.

💻 Divirta-se!

👨‍💻 Profº Vinny Albuquerque

📧 profvinny.albuquerque@fiap.com.br

© @mvalbuquerque