Versionamento de código

GIT III - Avançado

Professor - Renato Naumann

O que vamos aprender hoje?

- O que é e como configurar **chave ssh**
- O que é e como trabalhar com **commit semântico**
- O que é **code review**
- O que é e como trabalhar com **MR (Merge Request)**
- O que é e como utilizar o **.gitignore**

O que é uma chave ssh?

O SSH é um protocolo de rede que permite a conexão com determinados servidores por meio de uma comunicação criptografada.
As plataformas de hospedagem de código-fonte como GitLab e GitHub permitem que seja criada chaves SSH para que seja possível gerenciar tudo de forma remota, com segurança e sem ficar precisando fornecer o usuário e senha a cada acesso.

Como configurar as chaves ssh

Para realizar a configuração é necessário criar uma chave SSH no computador local e adicioná-la na plataforma de hospedagem de código-fonte.
A geração dessa chave muda de acordo com o sistema operacional, mas o processo é similar e deve ser feito sempre via linha de comando.

Gerando chave ssh no linux

Comando ssh-keygen

ssh-keygen -t rsa -b 4096 -C "SEU-EMAIL"

Criar agente SSH

eval "$(ssh-agent -s)"

Adicionar a chave ao agente

ssh-add ~/.ssh/id_rsa

Adicionar chave no GitLab?

Para adicionar a chave ssh no GitLab basta seguir o passo-a-passo

1. Obter a chave pública a partir do comando: **cat ~/.ssh/id_rsa.pub**
2. Abrir o GitLab
3. No canto superior direito clicar no avatar e selecionar a opção **Preferences**
4. Na coluna da esquerda selecionar a opção **SSH Keys**
5. Inserir o resultado do primeiro comando no campo **Key**

1

Vamos criar nossa chave ssh?

Abra o terminal e vamos fazer juntos

O que é commit semântico?

Commits semânticos é uma convenção para ser utilizada nas mensagens de commit.
Ela define um conjunto de regras para que seja possível criar um histórico de commit explicativo.
Dessa forma, todo commit possui um tipo, que informa a intenção do commit.
Nos próximos slides, vamos ver os tipos e descrições de commit.

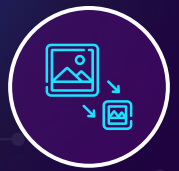
Tipos e descrição

Feat

Incluindo um novo recurso

Fix

Solucionando um problema



docs

Mudanças na documentação.
Não inclui mudança no código

test

Criação/Alteração de testes unitários.
Não inclui mudança no código

build

Modificação em **arquivos de build e dependências**

perf

Alterações relacionadas a performance da aplicação

Tipos e descrição

style

Mudanças na formatação do código.
Não inclui mudança no código

refactor

Refatoração da aplicação que não alterem a funcionalidade atual

2

Vamos ver na prática?

Abram o terminal e GitLab e vamos fazer juntos

O que é Code Review

O **Code review** é uma prática que consiste em fazer revisões no código de uma determinada aplicação.
Essa revisão é feita após a finalização de uma demanda, pode ser feita sob um commit específico ou um conjunto de commits.
Essa revisão deve ser feita por uma pessoa diferente da que desenvolveu o código.
O revisor deve analisar todas as mudanças realizadas, para identificar falhas e pontos de melhoria no código desenvolvido.

Qual a importancia do Code Review?

O code review é importante por vários motivos.

- Eleva a qualidade do software
- Aprimora as habilidades técnicas da equipe
- Auxilia na criação de soluções alternativas
- Identificação de problemas
- Aumenta a interação entre a equipe
- Compartilham melhorias de arquitetura do código
- Membros da equipe conhecem mais a fundo o código

Boas práticas no Code Review

Assim como todas as outras frentes, é fundamental utilizar boas práticas na revisão de código, tais como:

- Criar checklists
- Se atentar às novas ameaças para segurança da aplicação
- Utilizar ferramentas para auxiliar na revisão

O que é e como trabalhar com MRs?

Merge request é uma forma colaborativa de compartilhar a criação ou mudanças de código em um repositório Git.
Com é possível realizar revisão e/ou discussões do código entre os membros da equipe de desenvolvimento

Boas práticas de MR

Para o nosso processo de versionamento de código funcionar bem, temos que seguir algumas boas práticas.
Vamos ver algumas coisas que nós **NÃO** devemos fazer:

- Resolver mais de um problema em um MR
- Implementar mais de uma funcionalidade por MR
- Separar uma funcionalidade em mais de um MR

Boas práticas de MR

Quando seguimos as boas práticas todo o processo acontece de uma forma mais rápida e simples.

- Revisão de código ocorre mais rápida
- Menos conflitos em merges
- Diminuição de correções após revisão de código
- O código é finalizado em produção mais rapidamente
- Maior probabilidade de identificação de erros e melhorias

3

Vamos ver na prática?

Abram o terminal vamos fazer juntos

O que é o .gitignore

O arquivo **.gitignore** é um arquivo de texto oculto que mostra ao Git quais arquivos e pastas **NÃO devem** ser monitorados.
Para criar o arquivo é simples, basta criar um arquivo com o nome **.gitignore**
Cada linha desse arquivo deve listar um arquivo ou pasta que deve ser ignorado.

4

Vamos ver na prática?

Abram o terminal vamos fazer juntos

OBRIGADO!

Duvidas?
Nosso canal oficial de comunicação é o Slack.

academico@geradordedevs.com.br
https://www.geradordedevs.com.br