Treinamento preparatório

Delivoro Sistemas

Assuntos a serem abordados

- Ambiente de desenvolvimento
- Arquitetura do Projeto
- Treinamento
  - Testes Unitários - JUnit
  - WEB - Stripes + JSPs, Spring, MyBatis
- Processo de desenvolvimento
- GIT

Ambiente de Desenvolvimento

- Instalar no ambiente de desenvolvimento
  - JDK 13
  - IntelliJ Idea Community
  - NetBeans
  - Gradle é usado através da interface gráfica do IntelliJ e do NetBeans
  - DBeaver Community
  - Git
  - Google Chrome

- Acesso remoto
  - MySQL
  - Gitlab
  - Amazon AWS

Ambiente de Desenvolvimento

- Configurar JAVA_HOME em variáveis de ambiente no Windows

- Configurar PATH em variáveis de ambiente no Windows
- Configurar acesso ao banco de dados de dev usando DBeaver
- Acessar o Gitlab
- Configurar o IntelliJ

Ambiente de Desenvolvimento

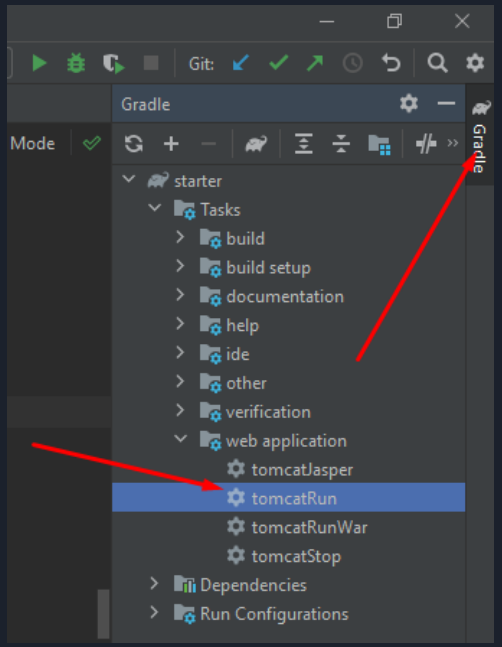
- Configurar acesso ao GIT - criar arquivo config na pasta .ssh do diretório home do usuário usando o Git Bash

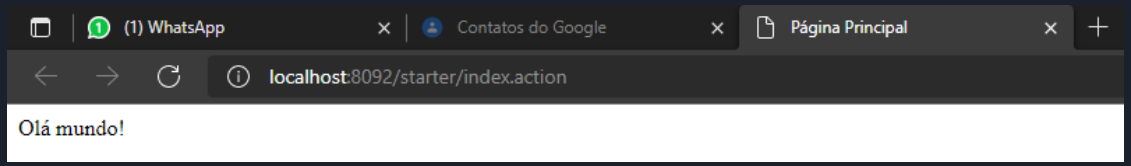
```
# mkdir ~/.ssh
# vi ~/.ssh/config

Host server.delivoro.com.br
        Port 4222
        User treinamento1
        IdentityFile ~/treinamento1.pem
```

- Clonar o projeto starter

Ambiente de Desenvolvimento

- Verificar se o mesmo executa
- Use o target tomcatRun do gradle
- Acessar o projeto em http://localhost:8092/starter





Arquitetura do Projeto

- Multicamadas
- Multi-níveis
- Baseada em padrões de projeto GoF e Java Enterprise
- Utiliza frameworks e tecnologias abaixo
  - JSP/EL/JSTL/DisplayTag
    - Apresentação de informação e lógica na UI
  - Stripes
    - camada Web - navegação - interação com browser
  - Spring Boot
    - Container de Inversão de Dependências, Transações, Integração entre frameworks
  - MyBatis
    - Integração com Banco de Dados - CRUD

Processo de desenvolvimento

- Utilizamos o Gitlab para automação de processos do repositório de software
- Feature branching
  - Cada atividade de cada desenvolvedor deve ser feita em uma branch separada
  - A branch deve partir de uma branch origem: develop/staging/master
  - Features partem da branch develop
  - Bugs podem partir de staging ou de master dependendo quem estiver em produção
- O desenvolvedor cria a própria branch, desenvolve e commita nela
- Ao término faz o push para o repositório remoto
- Cria um Merge Request no Gitlab
- O código é revisado. Caso haja correções a serem feitas, o desenvolvedor as realiza, faz um novo push e comenta no Merge Request o que foi feito
- O código é aprovado e aguarda merge para ser implantado em QA ou PROD

GIT

Usado como repositório e para gerenciamento da configuração de software

Competências a serem dominadas após o treinamento

- Saber usar os comandos básicos
  - add/commit/push/pull/merge
- Fazer um fluxo de desenvolvimento completo de feature branching
  - git fetch origin develop:develop
  - git checkout develop
  - git checkout -b feature/nome-feature
  - git add <arquivos>
  - git commit -m ‘mensagem’
  - git push

GIT

Competências a serem dominadas após o treinamento

- Fazer correções em branch após code review
  - git checkout feature/nome-feature
  - git pull
  - git add <arquivos>
  - git commit -m ‘mensagem’
  - git push
  - Responder comentários no gitlab

Referências - Processos Internos

Utilização de Campos Booleanos

Boas práticas de Código