ESCOLA PROFISSIONAL DA REGIÃO ALENTEJO

Sistemas de Controlo de Versões

UFCD 10789

Profª Betânia Maria Queta

Sumário

- Versionamento aplicacional
- Sistema de Controlo de Versões.
- Sistema Distribuído de Controlo de Versões.

Versionamento aplicacional

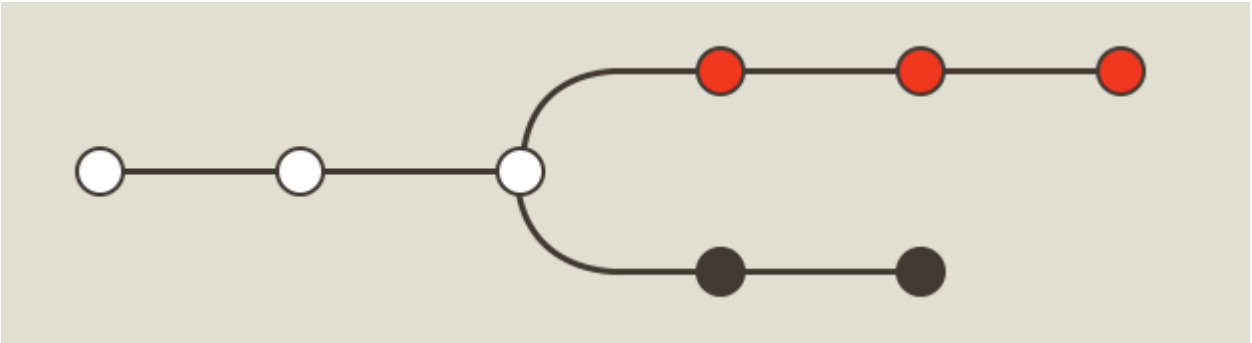
O conceito de **versionamento aplicacional** refere-se à prática de manter diferentes versões de uma aplicação ao longo do seu ciclo de desenvolvimento. Essa prática é essencial para garantir que as atualizações, correções de bugs e novas funcionalidades possam ser feitas de forma controlada, sem comprometer a integridade do software.

Principais conceitos

1. **Controlo de Alterações:**
   - Cada alteração no código é registada, permitindo saber quem fez a mudança, quando foi feita e porquê.
   - Facilita encontrar e corrigir erros ou voltar a versões anteriores, se necessário.
2. ***Branches* e *Merging*:**
   - Branches: Utilizadas para que diferentes programadores possam trabalhar em novas funcionalidades separadamente.
   - *Merge*: Combinar o trabalho feito nas *branches* com o código principal (*branch* principal).
3. ***Tags* e *Releases*:**
   - *Tags*: Usadas para marcar versões específicas, como "v1.0" ou "v2.1".
   - *Releases*: São as versões finais que são disponibilizadas aos utilizadores.
4. **Ambientes de Desenvolvimento:**
   - Desenvolvimento: Onde o código é criado e testado.
   - Homologação: Onde os testes finais são feitos antes do lançamento.
   - Produção: Onde o software está disponível para os utilizadores.
5. **Integração Contínua (CI) e Entrega Contínua (CD):**
   - CI/CD são práticas que automatizam os testes e atualizações. Cada vez que uma nova versão do código é enviada, ela é testada automaticamente e preparada para ser lançada.

Branch

Uma **branch** (ou "ramo", em português) é uma cópia separada do código principal de um projeto. As branches são utilizadas para desenvolver novas funcionalidades, corrigir erros ou testar mudanças sem afetar diretamente o código principal (geralmente chamado de **main** ou **master**).



Exemplo de funcionamento

1. Um programador cria uma nova funcionalidade no branch "feature-nova-funcionalidade".
2. Outro programador corrige um erro no branch "bugfix-corrigir-erro".
3. Eles fundem (merge) as suas mudanças no branch principal quando as alterações estão prontas.
4. Uma nova versão (release) é marcada como "v2.0" e lançada para os utilizadores.

Sistemas de Controlo de Versões

Um Sistema de Controlo de Versões (ou VCS - *Version Control System*) é uma ferramenta que permite guardar, rastrear e gerir todas as mudanças feitas no código de um projeto. Ele é essencial para o trabalho colaborativo, permitindo que múltiplas pessoas trabalhem no mesmo projeto de forma segura e organizada.

Principais Ferramentas de Controlo de Versões:

GitHub

• **O que é**: GitHub é uma plataforma online que usa **Git** (um sistema de controlo de versões distribuído). É amplamente utilizado para armazenar, partilhar e colaborar em projetos de software.

• **Funcionalidades**:

- **Repositórios**: Onde o código é armazenado.
- ***Branches***: Permitem que cada desenvolvedor trabalhe separadamente no mesmo projeto.
- ***Pull Requests***: Usados para sugerir alterações e pedir que outros revejam o código antes de fazer o *merge* com o *branch* principal.

• **Exemplo de uso**: Desenvolvedores de uma equipa usam GitHub para colaborar em diferentes funcionalidades, fazendo *commit* das suas mudanças e sugerindo que o código seja integrado com o código principal através de pull *requests*.

Principais Ferramentas de Controlo de Versões:

SVN (Subversion)

•**O que é**: SVN é um sistema de controlo de versões **centralizado**, o que significa que existe um único repositório central onde todas as alterações ao código são feitas.

•**Funcionalidades**:

- **Repositório Central**: Todas as alterações são feitas a partir deste repositório principal.
- **Checkout e Commit**: Os programadores copiam (fazem checkout) o código do repositório central e enviam (commit) as alterações de volta ao repositório central.

•**Exemplo de uso**: Uma empresa usa SVN para garantir que todo o código-fonte esteja guardado num único lugar, e os programadores têm que sincronizar o código com o servidor central a cada alteração.

Principais Ferramentas de Controlo de Versões:

SourceSafe

•**O que é**: **Microsoft Visual SourceSafe** é um sistema de controlo de versões mais antigo e centralizado, usado principalmente para pequenos projetos internos em empresas.

•**Funcionalidades**:

- **Histórico de Ficheiros**: Permite ver quem fez cada alteração em ficheiros ao longo do tempo.
- **Bloqueio de Ficheiros**: O programador pode "bloquear" ficheiros enquanto trabalha neles, evitando que outros façam alterações até que o bloqueio seja removido.

•**Exemplo de uso**: Numa pequena equipa, cada membro bloqueia os ficheiros que está a modificar, garantindo que outras pessoas não possam fazer alterações nesses ficheiros ao mesmo tempo.

Principais Ferramentas de Controlo de Versões:

Azure DevOps Server

•**O que é**: O ***Azure DevOps Server*** é uma plataforma que integra o controlo de versões com a gestão de projetos e automação de pipelines (CI/CD).

•**Funcionalidades**:

- **Azure Repos**: Controla as versões do código, permitindo colaboração em equipe.
- **Pipelines**: Automação para fazer testes e enviar novas versões do software automaticamente.
- **Boards**: Ferramenta de gestão de tarefas, usada para planear e acompanhar o progresso do projeto.

•**Exemplo de uso**: Numa grande equipa de desenvolvimento, os programadores utilizam o *Azure DevOps* para versionar o código, realizar testes automáticos (CI/CD) e coordenar tarefas de desenvolvimento através do Azure Boards.

Resumo

Cada ferramenta de controlo de versões tem as suas vantagens e características específicas. A escolha entre elas depende do tamanho da equipa, da necessidade de integração com outras ferramentas e do tipo de projeto em que se está a trabalhar.

Questões?

- © Betânia Queta 2024