**CENTRO UNIVERSITÁRIO BELAS ARTES DE SÃO PAULO**

**GRADUAÇÃO: ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - EAD**

**FELIPE SCHAITEL - 24112424**

**CONTROLE DE VERSÃO DE SOFTWARE - ATIVIDADE 01**

**FLORIANÓPOLIS, SC**

**2025**

**1 INTRODUÇÃO**

Esse relatório contém a Atividade 01 da matéria de Controle de *Software* ministrada pelos professores Márcio Cavalcante e Rafael Murakami no curso de Análise e desenvolvimento de Sistemas.

**2 ENUNCIADO**

**2.1 Objetivo**

Aplicar conceitos de controle de versão em um projeto de software, utilizando um sistema de controle de versão distribuído (Git) e seguindo as práticas de versionamento semântico.

**2.2 O que entregar**

Documento em Word com até duas páginas, contando a capa, constando: Relatório com introdução e descrição do projeto. Link do repositório Git com todo o histórico de commits, e arquivo README.md. Critérios de Avaliação: Clareza e qualidade dos commits. Correta aplicação do versionamento semântico. Uso apropriado de tags. Qualidade da documentação e justificativas apresentadas no relatório. Vale 4 pontos na N1.

**3 DETALHES DA ATIVIDADE**

**3.1 Configuração do Projeto**

Crie um repositório Git para o projeto. Defina um README.md inicial que descreva o projeto. Realize o primeiro commit com a versão inicial (0.1.0).

O repositório chamado “versionamento” com a aplicação simples desenvolvida “rpg dice” está no link: [fschaitel/versionamento: Atividade 1 da disciplina de Controle de Versão de Software, ministrado por Marcio Cavalcante na Belas Artes de São Paulo.](https://github.com/fschaitel/versionamento)

O README.md encontra-se neste link: [versionamento/README.md at main · fschaitel/versionamento](https://github.com/fschaitel/versionamento/blob/main/README.md)

**3.2 Implementação de Funcionalidades**

Desenvolva uma funcionalidade simples (por exemplo, um script que realiza um cálculo de IMC – Índice de Massa Corporal em Javascript ou em qualquer linguagem de programação de sua preferência). Realize commits frequentes a cada etapa de desenvolvimento, seguindo boas práticas de mensagens de commit (escrevendo mensagens claras e descritivas).

O git e github possibilitam a pesquisa e visualização de todos os commits. Essa função é essencial para o funcionamento de um software. Pois permite aos desenvolvedores consultar, por exemplo, quando que começou a aparecer um erro e corrigir ou voltar para a versão anterior rapidamente.

É por conta dessa tecnologia que o Wow Classic está sendo jogável. World of warcraft é um MMO rpg online que está em ativa a 30 anos, com o passar do tempo houve muitas atualizações e modificações. Recentemente a empresa desenvolvedora, Blizzard, disponibilizou a versão antiga de como era o Wow no início para os jogadores. Isso foi possível pois eles tinham os commits iniciais de como era o jogo a 20 anos atrás.

Histórico do versionamento disponível no link: [History for rpgdice - fschaitel/versionamento](https://github.com/fschaitel/versionamento/commits/main/rpgdice)

**3.3 Controle por Numeração e Tags**

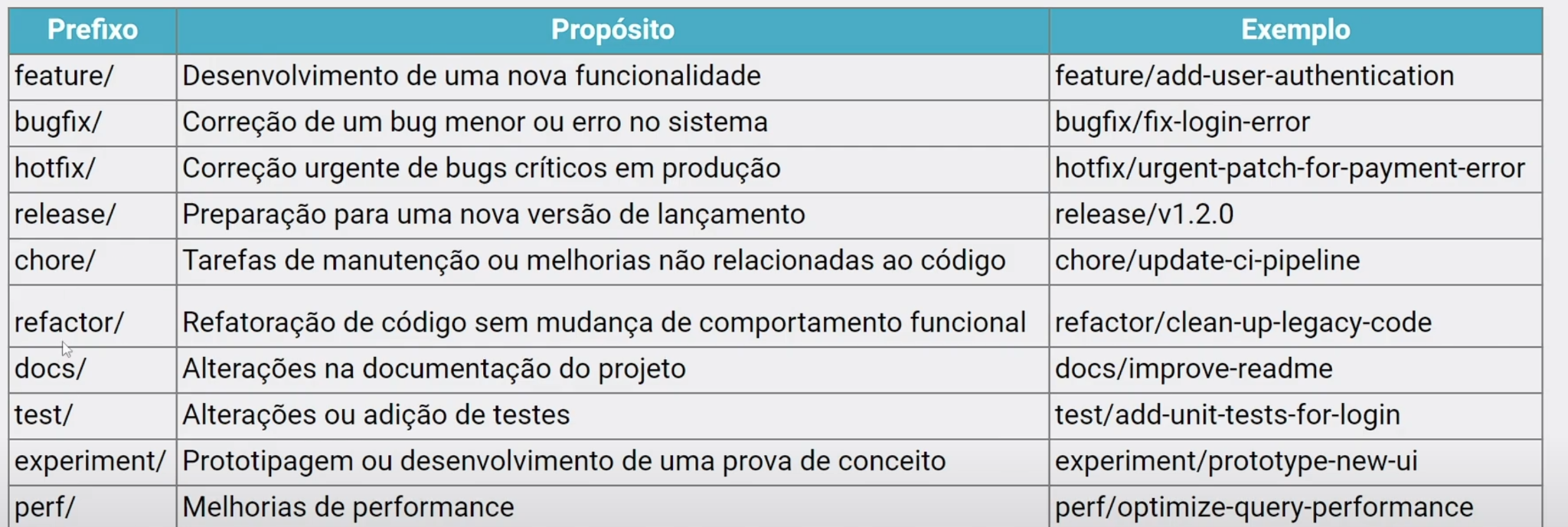
A cada implementação significativa (por exemplo, quando uma nova funcionalidade é adicionada), crie uma tag no repositório utilizando a convenção de versionamento semântico.(MAJOR.MINOR.PATCH).

Justifique a escolha da numeração da versão de acordo com as alterações feitas (incremento de versão major, minor ou patch).

O versionamento semântico é uma maneira de organizar e padronizar as versões dos projetos. Cada parte tem um significado. Major é utilizado quando a nova versão apresenta incompatibilidade, Minor para inserir novas funcionalidades e Patch para correção.

Link para acessar as tags: [Tags · fschaitel/versionamento](https://github.com/fschaitel/versionamento/tags)

Imagem: Prefixos que ajudam a organizar commits



fonte: Devinfra

**3.4 Simulação de Bug Fix**

Introduza um bug simples no código propositalmente. Crie uma nova versão corrigindo o bug, incrementando o patch na versão (por exemplo, de 1.0.0 para 1.0.1).

Link de acesso ao histórico completo dos commits do repositório: [History for rpgdice - fschaitel/versionamento](https://github.com/fschaitel/versionamento/commits/main/rpgdice)

O bug introduzido no código propositalmente está no quinto commit **62951bf**

Link para acessar: [Update rpgdice · fschaitel/versionamento@62951bf](https://github.com/fschaitel/versionamento/commit/62951bff46a01a808a1d26b09456cc0c8f2225d1)

A correção está no sexto commit [**deace9d**](https://github.com/fschaitel/versionamento/commit/deace9d48d914bed05820ca56b7e275a3f9162c5)

Link para acessar: <https://github.com/fschaitel/versionamento/commit/deace9d48d914bed05820ca56b7e275a3f9162c5>

**4 Referências bibliográficas:**

**GitHub no VS Code 08 - Como criar tags no GitHub - YouTube. Disponível em:** [**https://www.youtube.com/watch?v=XXXXX**](https://www.youtube.com/watch?v=XXXXX)**. Acesso em: 24 mar. 2025.**

**Criando Tags com Versionamento Semântico no GitHub | Aula 6. Disponível em: <**[**https://www.youtube.com/watch?v=wPHYhOWw-i8**](https://www.youtube.com/watch?v=wPHYhOWw-i8) **>. Acesso em: 24 mar. 2025.**

**Sobre as versões - GitHub Docs. Disponível em: <**[**https://docs.github.com/pt/repositories/releasing-projects-on-github/about-releases**](https://docs.github.com/pt/repositories/releasing-projects-on-github/about-releases) **>. Acesso em: 24 mar. 2025.**

**Gerenciar tags no GitHub Desktop - GitHub Docs. Disponível em: <**[**https://docs.github.com/pt/desktop/managing-commits/managing-tags-in-github-desktop**](https://docs.github.com/pt/desktop/managing-commits/managing-tags-in-github-desktop) **>. Acesso em: 24 mar. 2025.**

**Controle de versão semântico 2.0.0 | Controle de versão semântico. Disponível em: <**[**https://semver.org/**](https://semver.org/) **>. Acesso em: 24 mar. 2025.**