Documento de Política de Segurança e Gestão de Acesso do PortfolioHUB

Projeto: PortfolioHUB (Desafio - Entrega Final) **Proprietário:** Vitor Davi

Consultoria de Segurança (IA): Google Gemini

---

1. Objetivo

Este documento detalha as políticas de Gestão de Usuários e Segurança implementadas no repositório GitHub do projeto PortfolioHUB. O objetivo é proteger a integridade do portfólio, garantir que apenas pessoal autorizado possa modificar o conteúdo e proteger informações pessoais ou acadêmicas, seguindo as melhores práticas de desenvolvimento e segurança.

2. Escopo do Projeto

O PortfolioHUB é uma plataforma centralizada que exibe e gerencia projetos e portfólios digitais do estudante Vitor Davi. O repositório contém:

- Projetos acadêmicos (Ex: Projeto MER, Estudo de Banco NoSQL).
- Informações de perfil profissional (links para LinkedIn, currículo).
- Código-fonte do site do portfólio.

3. Gestão de Usuários e Níveis de Acesso

A gestão de usuários no GitHub é configurada para garantir o controle de acesso, conforme o "Princípio do Menor Privilégio".

- **Proprietário (Owner):**
    - **Usuário:** `Vitor5-dotcom` (ou seu nome de usuário GitHub).
    - **Permissões:** Controle administrativo total. Único usuário com permissão para alterar configurações de segurança, gerenciar faturamento (se aplicável) e deletar o repositório.
- **Colaboradores (Colaborators):**
    - **Usuários:** Professores (Ex: Prof. Marcelo Carboni Gomes) ou colegas para revisão.
    - **Permissões:** Acesso de `Leitura` (Read) ou `Escrita` (Write) concedido temporariamente para fins de avaliação ou colaboração. Não podem alterar configurações críticas do repositório.
- **Público (Public):**
    - **Usuários:** Qualquer pessoa na internet.
    - **Permissões:** Acesso de `Leitura` (Read) ao código-fonte. O repositório é público para servir como portfólio. O acesso público *não* permite a modificação de qualquer arquivo.

4. Políticas de Segurança Implementadas

As seguintes políticas foram implementadas para garantir a robustez da segurança do projeto.

1. **Autenticação de Dois Fatores (MFA):**
    - O proprietário do repositório (`Vitor Davi`) deve ter a autenticação de dois fatores ativada em sua conta GitHub. Esta é a barreira mais eficaz contra acesso não autorizado à conta.
2. **Proteção de Branch (Branch Protection) para a `main`:**
    - A branch `main` (principal) é protegida por regras que impedem modificações diretas e destrutivas.
    - **Regra 1: Exigir Pull Requests:** Todo o código deve ser enviado para uma branch secundária (ex: `feature/novo-projeto`) e mesclado na `main` através de um Pull Request (PR). Commits diretos na `main` são bloqueados.
    - **Regra 2: Exigir Revisão:** (Opcional, mas recomendado para colaboração) Exigir que pelo menos 1 (um) colaborador aprove o Pull Request antes do *merge*. Para um projeto individual, o proprietário revisa seu próprio PR.
3. **Gerenciamento de Dados Sensíveis:**
    - **Política:** Nenhuma informação sensível (como senhas, chaves de API, tokens de acesso, números de matrícula ou documentos pessoais) deve ser enviada (*commitada*) ao repositório.
    - **Implementação:** Um arquivo `.gitignore` é utilizado para instruir o Git a ignorar arquivos de configuração local (ex: `.env`, `credentials.json`) que possam conter chaves de API ou dados de conexão.

5. Práticas de Colaboração e Versionamento

Para manter a organização e a segurança, o seguinte fluxo de trabalho (workflow) de colaboração é adotado:

1. **Criação de Branch:** Para qualquer nova funcionalidade ou correção (ex: "Adicionar Projeto MER"), uma nova branch é criada a partir da `main`.
    - *Exemplo:* `git checkout -b feature/add-mer-project`
2. **Desenvolvimento:** O trabalho é realizado e *commitado* nessa nova branch.
3. **Pull Request (PR):** Quando o trabalho está concluído, um Pull Request é aberto no GitHub para mesclar a `feature/add-mer-project` na `main`.
4. **Revisão e Teste:** O PR é revisado para garantir que o código está limpo, funcional e não introduz vulnerabilidades de segurança (como chaves expostas). Os testes automatizados (se configurados) são executados.
5. **Merge:** Após a aprovação, o PR é mesclado na `main`.
6. **Limpeza:** A branch da *feature* é excluída.

Este processo garante que a branch `main` esteja sempre estável, funcional e segura, cumprindo os requisitos de controle de versão e compartilhamento