GRUPO: Carlos Eduardo, Ricardo Oliveira, Gabriel Candido.

Relatório de Colaboração e Desafios Encontrados

Introdução

Este relatório descreve o processo de desenvolvimento da interface de login, utilizando HTML, CSS e JavaScript, com foco na colaboração entre o grupo no GitHub. O objetivo era criar uma tela de login funcional, integrar o código em um repositório compartilhado e documentar os desafios enfrentados durante o desenvolvimento.

---

1. Criação da Interface de Login

Estrutura da Tela de Login

A interface foi desenvolvida com os seguintes elementos:

- **Campo de Nome de Usuário** : Um campo de entrada (`<input>`) para inserir o nome de usuário.
- **Campo de Senha** : Um campo de entrada (`<input type="password">`) para inserir a senha.
- **Botão de Entrar** : Um botão (`<button>`) que dispara a validação dos campos.
- **Link "Esqueci minha senha"** : Um link (`<a>`) visível, mas sem funcionalidade implementada.
- **Mensagens de Erro/Sucesso** : Exibidas dinamicamente com base na validação dos campos e no resultado do login.

Funcionalidade com JavaScript

A validação foi implementada em JavaScript para garantir que:

1. Ambos os campos estão preenchidos.
2. O nome de usuário e a senha correspondem aos valores predefinidos (`admin` e `1234`, respectivamente).
3. Mensagens de erro ou sucesso são exibidas conforme o resultado da validação.

---

2. Versionamento no GitHub

Estrutura do Repositório

Criamos um repositório no GitHub chamado `Aula01DevOps`. O repositório inclui as seguintes pastas e arquivos:

- `/css`: Contém o arquivo de estilos (`styles.css`).
- `/js`: Contém o arquivo de lógica JavaScript (`script.js`).
- `index.html`: Arquivo principal da interface de login.
- `README.md`: Documentação básica do projeto.

Fluxo de Trabalho

1. **Branching** :
    - Criamos branche principal: `main` (para o código estável) e `branches (nome do aluno)` (para desenvolvimento ativo).
    - Cada aluno trabalhou em sua própria branch de recurso.
2. **Commits Frequentes** :
    - Realizamos commits frequentes com mensagens claras e descritivas, como:
        - `update: atualização do código`
        - `remove: tirar partes do código não utilizados`
3. **Pull Requests** :
    - Após concluir uma funcionalidade, criamos pull requests para integrar as mudanças à branch `main`.
    - Revisamos mutuamente o código antes de aprovar os merges.
4. **Integração Final** :
    - Após testar todas as funcionalidades, realizamos um merge da branch de cada aluno para `main`.

---

3. Integração e Detecção de Dificuldades

Conflitos de Merge

Durante o desenvolvimento, encontramos conflitos ao tentar integrar as alterações de ambos os alunos. Os principais desafios foram:

1. **Conflito no Estilo CSS** :
    - Ambos os alunos editaram o mesmo arquivo de estilo (`styles.css`) simultaneamente.
    - Resolvemos o conflito manualmente, combinando as alterações e garantindo que o layout final fosse consistente.
2. **Sobreposição de Funções JavaScript** :

- Implementamos funções JavaScript em arquivos separados inicialmente, mas ao integrá-las no mesmo arquivo (`script.js`), houve sobreposição de nomes.
- Renomeamos as funções para evitar conflitos e organizamos o código de forma modular.

Testes de Funcionalidades

Após cada merge, realizamos testes manuais para verificar:

- A validação dos campos.
- O comportamento do botão de login.
- A exibição das mensagens de erro/sucesso.

Encontramos um bug onde a mensagem de erro não era limpa após corrigir os campos. Corrigimos isso adicionando uma função para resetar as mensagens antes de cada validação.

---

4. Ferramentas Utilizadas

- **GitHub** : Para versionamento e colaboração.
- **VS Code** : Editor de código utilizado por ambos os alunos.
- **Git** : Ferramenta de controle de versão para gerenciar branches e commits.