# 📋 **Relatório Técnico: Sistema de Login e Cadastro com Firebase**

## 🧾 ****Descrição Geral****

Este projeto consiste em um sistema de autenticação de usuários utilizando **Firebase Authentication** com armazenamento de dados no **Firestore**. A aplicação oferece:

* Cadastro de novos usuários com validação de senha;
* Login com verificação de credenciais;
* Redirecionamento para área logada (landing page);
* Feedback visual e mensagens dinâmicas para erros ou sucessos;
* Interface moderna com alternância entre telas (toggle).

## 🔐 ****Funcionalidades de Segurança Implementadas****

| **Recurso** | **Descrição** |
| --- | --- |
| 🔒 **Firebase Authentication** | Utilizado para autenticação por e-mail e senha de forma segura. |
| 🔐 **Validação de Senha** | Senhas precisam ter no mínimo 6 caracteres, letras e números. |
| ❌ **Bloqueio de E-mail Duplicado** | Impede cadastro se o e-mail já estiver registrado. |
| ✅ **Mensagens de Feedback Inteligente**  **Enviam um link para cadastrar nova senha por E-mail** | Informa ao usuário o motivo de falhas no login/cadastro.  Caso o usuário esqueça sua senha, tem a opção de cadastrar uma nova senha. |

## 🛠️ ****Tecnologias Utilizadas****

| **Tecnologia** | **Uso** |
| --- | --- |
| HTML5 | Estruturação das páginas (login, cadastro, recuperação). |
| CSS3 | Estilização moderna e responsiva da interface. |
| JavaScript | Validações, integração com Firebase e controle de eventos. |
| Firebase Auth | Gerenciamento de autenticação segura. |
| Firestore | Armazenamento em nuvem dos dados dos usuários. |

## 🔎 ****Estrutura do Código****

### ****1. Cadastro (****registerForm****)****

* Verifica campos obrigatórios preenchidos.
* Compara senha e confirmação de senha.
* Validação da senha com expressões regulares (RegEx).
* Criação de usuário via firebase.auth().createUserWithEmailAndPassword().
* Armazenamento de dados no Firestore: nome, e-mail e timestamp.

### ****2. Login (****loginForm****)****

* Validação dos campos de entrada.
* Login via firebase.auth().signInWithEmailAndPassword().
* Redirecionamento automático para a landing pagina em caso de sucesso.
* Verificação de e-mail (pendente de implementação obrigatória).

### ****3. Interface (UI)****

* Utiliza container.classList.add/remove("active") para alternar entre os painéis de login e cadastro.
* Estilo visual limpo com ícones da Font Awesome.

## 🚨 ****Oportunidades de Melhoria e Segurança Adicional****

| **Recurso** | **Descrição** |
| --- | --- |
| ✅ **Verificação de E-mail** | Exigir que o usuário verifique o e-mail antes de permitir login. |
| 🔁 **Rate Limiting / reCAPTCHA** | Implementar reCAPTCHA invisível para evitar bots e abusos. |
| 🔐 **HTTPS Obrigatório** | Forçar HTTPS para proteger os dados transmitidos. |
| 🛡️ **Ofuscação de JavaScript** | Reduzir a exposição da lógica ao público para maior segurança. |
| 📋 **Logging Melhorado** | Adicionar logs de tentativas de login mal sucedidas (caso use backend). |

## 🧪 ****Testes Recomendados****

* ✅ Cadastro com e-mails válidos e inválidos.
* ✅ Cadastro com senha fraca.
* ✅ Cadastro com e-mail já existente.
* ✅ Login com dados corretos.
* ✅ Login com senha errada ou e-mail não registrado.
* ✅ Verificação do redirecionamento após login.
* ✅ Exibição de mensagens de erro claras e úteis.

## ✅ ****Conclusão****

O sistema oferece uma **base sólida e funcional para autenticação de usuários**, combinando usabilidade e segurança. Atualmente, o projeto cobre:

* Validação básica de dados;
* Proteção contra duplicidade de usuários;
* Armazenamento de dados em nuvem.

🔒 Com a implementação de recursos adicionais como **verificação obrigatória de e-mail**, **uso de reCAPTCHA**, e reforços no **tratamento de erros**, o sistema poderá evoluir para uma solução ainda mais robusta e pronta para produção.