ShowUp: onde cada linha de código é a prova da sua capacidade.

Tema: Sistema de Gerenciamento de Portfólio

1. Introdução

1.1 Objetivo do Sistema

O **ShowUp** é um sistema de gerenciamento de portfólio desenvolvido para permitir que profissionais e estudantes de tecnologia cadastrem, organizem e exponham seus projetos de forma visual e estruturada. O objetivo principal é consolidar a gestão de dados do usuário e de seus projetos em uma única plataforma.

1.2 Requisitos Acadêmicos e Técnicos Atendidos

Requisito do Curso	Atendimento no Projeto ShowUp
Firebase	Utilização da suíte Firebase: **Authentication** (acesso), **Cloud Firestore** (dados estruturados) e **Cloud Storage** (arquivos).
Modelagem de Entidades	Modelagem NoSQL com duas coleções principais (users e projects) e implementação de relacionamento **1:N** (Um Usuário para Muitos Projetos).
Elaboração de Documentação	Geração deste Documento de Requisitos (DR), Modelo de Dados/DER e Documentação Técnica do Código.
Previsibilidade e Manutenção	Código modularizado em serviços (ex: UserService, ProjectService), garantindo o **Princípio da Responsabilidade Única (SRP)** e facilitando testes e expansões.
Capacidade de "Entrar" Informações	Implementação de funcionalidades de **CRUD** (Criação, Leitura, Atualização e Exclusão) para a entidade *Projeto*.
Capacidade de Upload de Arquivos	Funcionalidade obrigatória de *upload* de mídias visuais (**JPG/PNG**) para o Cloud Storage.
Conhecimento de Código (Clean Code)	Aplicação de nomenclatura semântica, tratamento robusto de erros (try...catch) e otimização de consultas para performance.

---

2. Requisitos Funcionais (RF)

Os Requisitos Funcionais definem as ações que o sistema deve executar para o usuário.

ID	Requisito Funcional	Descrição Detalhada
RF-001	Autenticação Completa	O sistema deve gerenciar o registro de novos usuários e o login de usuários existentes através do Firebase Authentication.
RF-002	Gestão de Perfil (Usuário)	O usuário deve poder visualizar e editar seus dados pessoais na coleção users (nome, e-mail, biografia).
RF-003	Criação de Projetos	O sistema deve permitir a inserção de um novo projeto, associando-o automaticamente ao user_uid do usuário logado.
RF-004	Edição e Exclusão de Projetos	O usuário deve poder modificar ou remover qualquer projeto que seja de sua autoria.
RF-005	Upload de Imagens (Assets)	O usuário deve poder selecionar e enviar um arquivo de imagem (JPG ou PNG) que será armazenado no Cloud Storage e referenciado no Firestore.
RF-006	Listagem do Portfólio	A interface deve exibir todos os projetos do usuário logado, filtrando a coleção projects pelo seu user_uid.
RF-007	Visualização Detalhada	Deve haver uma página para exibir os detalhes completos de um projeto específico, incluindo a imagem de *upload*.

---

3. Requisitos Não Funcionais (RNF)

Os Requisitos Não Funcionais definem os critérios de qualidade e as restrições técnicas.

ID	Requisito Não Funcional	Descrição Técnica (Para a Defesa do Código)
RNF-001	**Arquitetura BaaS**	O sistema deve utilizar **exclusivamente** o Firebase (Authentication, Firestore, Storage) para todas as necessidades de *backend*.
RNF-002	**Segurança (Restrição)**	Deve ser implementado o controle de acesso por meio de **Firebase Security Rules** para garantir que um usuário só possa manipular seus próprios documentos e arquivos.
RNF-003	**Performance (Consulta)**	As queries de listagem de projetos (RF-006) devem utilizar índices e o filtro where('user_uid', '==', currentUserId) para garantir buscas eficientes e de baixo custo.
RNF-004	**Manutenibilidade**	A arquitetura de código deve ser baseada no conceito de **Serviços** (*Clean Code*), isolando as interações com o Firebase da lógica da interface.
RNF-005	**Previsibilidade**	Todas as operações críticas (login, upload, escrita no DB) devem incluir tratamento de erro robusto (try...catch) para prever e lidar com falhas de rede ou permissão.

---

4. Modelagem de Dados (Firestore - Modelo Lógico)

A arquitetura de dados utiliza um modelo **NoSQL de Coleções e Documentos**, estabelecendo uma relação 1:N.

4.1. Coleção users (Entidade Usuário)

- **Finalidade:** Armazena os metadados do perfil.
- **Chave Primária:** UID (ID exclusivo gerado pelo Firebase Auth).**4.2. Coleção projects (Entidade Projeto)**

Atributo	Tipo de Dado	Notas
name	String	Nome completo.
email	String	E-mail do usuário.
bio	String	Pequena descrição/resumo do perfil.

- **Finalidade:** Armazena os dados de cada projeto individual.
- **Chave Primária:** ID auto-gerado do Documento.
- **Relacionamento:** Vinculado à coleção users via Chave Estrangeira.

Atributo	Tipo de Dado	Notas
user_uid	String	**Chave Estrangeira (FK)**: ID do usuário proprietário. Essencial para o filtro 1:N.
title	String	Título do projeto.
description	String	Detalhes técnicos e de desenvolvimento.
technologies	Array of Strings	Lista de tecnologias utilizadas (ex: ["React", "Firebase", "Node.js"]).

main_image_url	String	URL de acesso público do arquivo no Cloud Storage (upload JPG/PNG).
created_at	Timestamp	Data de criação do projeto.