Documentação Técnica do Projeto: StreamAll

Versão: 1.0

Tipo: Plataforma Full-Stack de Streaming Multimídia

---

1. Visão Geral do Projeto

O **StreamAll** é uma aplicação web que unifica o consumo de diferentes tipos de mídia (Filmes, Séries, Livros e Músicas) em uma única interface intuitiva. O sistema simula uma plataforma de streaming completa, gerenciando autenticação de usuários, personalização de listas e consumo de dados via APIs externas.

1.1. Objetivo

Demonstrar competência em desenvolvimento Full-Stack, integrando um frontend responsivo e dinâmico com um backend em Python (Flask) robusto, persistência de dados e consumo de serviços externos.

1.2. Stack Tecnológica

- **Frontend:** HTML5, CSS3, JavaScript (ES6+).
- **Backend:** Python 3, Flask (Microframework).
- **Banco de Dados:** SQLite.
- **ORM:** SQLAlchemy.
- **Autenticação:** Flask-Login, Werkzeug Security.
- **APIs Externas:**
    - *Filmes/Séries:* The Movie Database (TMDB).
    - *Músicas:* Spotify Web API.
    - *Livros:* Google Books API.

---

2. Arquitetura do Sistema

O sistema segue a arquitetura **MVC (Model-View-Controller)** adaptada para API RESTful. O Flask atua como um servidor de API e Proxy reverso para proteger as chaves de acesso.

Fluxo de Dados

1. **Cliente (Browser):** Solicita dados via fetch() (AJAX).
2. **Servidor (Flask):**
    - Verifica a sessão do usuário.
    - Consulta o Banco de Dados local (SQLite) para dados do usuário.
    - Requisita dados às APIs externas (TMDB, Spotify, Google) injetando credenciais seguras.
3. **Resposta:** O servidor processa os dados e retorna um JSON unificado para o cliente renderizar.

---

3. Requisitos Funcionais (RF)

3.1. Módulo de Autenticação e Usuários

- **[RF001] Cadastro:** O sistema deve permitir criar conta com Nome, E-mail e Senha.
- **[RF002] Login Seguro:** O sistema deve autenticar usuários validando o hash da senha e gerenciar sessões.
- **[RF003] Edição de Perfil:** O usuário logado deve poder alterar seus dados cadastrais e senha.
- **[RF004] Persistência de Sessão:** A sessão deve permanecer ativa até o usuário efetuar Logout.

3.2. Módulo de Catálogo e Conteúdo

- **[RF005] Agregação de Mídia:** O sistema deve exibir Filmes, Séries, Livros e Músicas em carrosséis separados.
- **[RF006] Busca Global:** Deve haver uma barra de pesquisa que consulta todas as categorias simultaneamente (Live Search).
- **[RF007] Filtros:** Permitir filtragem por Gênero e Ordenação (Mais Populares, Lançamentos) dentro de cada categoria.
- **[RF008] Detalhes do Item:** Ao clicar em um item, exibir sinopse, capa, autor/diretor e nota.

3.3. Módulo de Personalização

- **[RF009] Minha Lista:** O usuário deve poder adicionar e remover itens de uma lista pessoal ("Assistir/Ler Mais Tarde").
- **[RF010] Progresso de Consumo:** O sistema deve registrar onde o usuário parou (ex: tempo do filme ou página do livro) ao fechar o item.
- **[RF011] Histórico de Atividade:** Exibir uma seção "Continuar Consumindo" na página inicial com base no histórico recente.

---

4. Requisitos Não Funcionais (RNF)

- **[RNF001] Segurança de Dados:** Senhas devem ser armazenadas exclusivamente como *hashes* (PBKDF2/SHA256 via Werkzeug).
- **[RNF002] Proteção de API Keys:** As chaves das APIs externas devem residir em variáveis de ambiente (.env) no servidor, nunca expostas no frontend.
- **[RNF003] Responsividade:** A interface deve se adaptar fluidamente a desktops, tablets e smartphones (Mobile First).
- **[RNF004] Performance:** O tempo de carregamento da página inicial não deve exceder 2 segundos (usando cache simples no backend se necessário).

---

5. Modelo de Dados (Schema Simplificado)

O banco de dados utilizará o SQLite gerenciado pelo SQLAlchemy.

Tabela: User

Campo	Tipo	Descrição
id	Integer (PK)	Identificador único.
name	String(100)	Nome completo.
email	String(120)	E-mail único (Login).

password_hash	String(128)	Hash da senha.
created_at	DateTime	Data de cadastro.

Tabela: UserLibrary (Lista Pessoal)

Campo	Tipo	Descrição
id	Integer (PK)	Identificador do registro.
user_id	Integer (FK)	Referência à tabela User.
media_id	String	ID externo do item (ex: ID do IMDB ou ISBN).
media_type	String	'movie', 'series', 'book', 'music'.
title	String	Título (para cache rápido).
image_url	String	URL da capa (para cache rápido).
status	String	'planned', 'watching', 'completed'.
progress	String	Marcador de progresso (página ou tempo).

---

6. Rotas da API Interna (Endpoints)

O Frontend se comunicará com o Backend através das seguintes rotas:

Autenticação

- POST /auth/register - Criar novo usuário.
- POST /auth/login - Iniciar sessão.
- GET /auth/logout - Encerrar sessão.
- PUT /auth/profile - Atualizar dados do usuário.

Dados do Catálogo

- GET /api/media/trending - Retorna destaques de todas as categorias.
- GET /api/media/search?q={termo} - Busca global.
- GET /api/media/{type}/{id} - Detalhes de um item específico.

Usuário

- GET /api/library - Retorna a lista pessoal do usuário.
- POST /api/library - Adiciona item à lista.
- DELETE /api/library/{id} - Remove item da lista.
- PATCH /api/library/{id}/progress - Atualiza o progresso de consumo.