Meu Checklist de Requisitos - Projeto NetRádio

Este é um resumo de todos os requisitos que eu cumpri no projeto, com exemplos de código que mostram como implementei cada funcionalidade.

RA1 - Utilizar Frameworks CSS para estilização e layouts responsivos.

ID	Requisito	Status	Como eu cumpri
01	**Layout responsivo com Framework**	**Cumprido**	Eu usei o sistema de Grid do Materialize em todo o projeto. **Exemplo:** Na admin/index.html, os cards do dashboard usam class="col s12 m6 l3", fazendo com que ocupem a tela toda em celulares (s12), metade em tablets (m6) e um quarto em desktops (l3).
02	**Responsividade nativa de CSS**	**Cumprido**	Além do framework, eu usei Flexbox para alinhar elementos. **Exemplo:** No client/css/style.css, a classe .player-controls usa display: flex e flex-wrap: wrap para que o nome da empresa e os botões se ajustem de forma elegante em telas pequenas.
03	**Componentes CSS e JS do Framework**	**Cumprido**	Eu utilizei vários componentes do Materialize. **Exemplo 1 (CSS):** Os card-panel no dashboard do admin. **Exemplo 2 (JS):** O modal para adicionar/editar usuários em admin/js/admin.js, que eu ativei com M.Modal.init(...).
04	**Uso de unidades relativas**	**Cumprido**	Eu priorizei unidades relativas para flexibilidade. **Exemplo:** No login/css/style.css, a classe .login-card tem width: 90%, adaptando-se a qualquer tamanho de tela, e usei rem para fontes, como em font-size: 1.1rem.
05	**Animações em elementos**	**Cumprido**	A landing page (index.html) tem animações de "revelar ao rolar". **Exemplo:** No css/style.css, a classe .reveal define o estado inicial (invisível). No js/main.js, a função revealOnScroll adiciona a classe .visible quando o elemento entra na

			tela.
07	**Aplicação de Design System**	**Cumprido**	Eu criei um sistema de design consistente. **Exemplo:** A cor primária #0589ca foi padronizada na classe .btn-radio-blue e usada em todos os botões principais. A fonte 'Inter' é definida no <body> de todos os arquivos CSS.
08	**Uso de pré-processadores (Sass)**	**Não Aplicável**	Eu não utilizei Sass, mas organizei o CSS em arquivos separados por página (login/css/style.css, admin/css/style.css, etc.), seguindo o princípio de modularidade.
09	**Tipografia responsiva**	**Cumprido**	Eu usei a classe flow-text do Materialize para títulos, que ajusta o tamanho da fonte automaticamente. **Exemplo:** No client/index.html, o <h2> da mensagem de boas-vindas usa class="flow-text".
10	**Responsividade de imagens**	**Cumprido**	Eu usei a classe responsive-img do Materialize. **Exemplo:** No login/index.html, a tag <img> da logo tem a classe responsive-img, que garante que ela nunca ultrapasse o tamanho do seu contêiner.
11	**Otimização de imagens**	**Cumprido**	Eu usei o atributo onerror para carregar uma imagem de placeholder caso a original falhe. **Exemplo:** Em todas as logos, eu tenho onerror="this.onerror=null; this.src='...'", evitando um ícone de imagem quebrada.

RA2 - Realizar tratamento de formulários e aplicar validações.

ID	Requisito	Status	Como eu fiz no meu código
12	**Tratamento de formulários no cliente**	**Cumprido**	O formulário de login é o melhor exemplo. **Exemplo:** No login/js/login.js, a função que escuta o submit verifica se os campos estão vazios (if (!emailInput.value.trim())) e

			exibe uma mensagem de erro no correspondente.
13	**Validação com Expressões Regulares (REGEX)**	**Cumprido**	Eu utilizei a validação nativa do HTML5, que usa REGEX internamente. **Exemplo:** No login/index.html, o campo de e-mail tem type="email", e o navegador automaticamente valida o formato do texto.
14	**Uso de select, checkbox ou radio**	**Cumprido**	Eu usei o elemento select na aba de relatórios do painel de admin. **Exemplo:** No admin/index.html, eu tenho <select id="reportUser">..., que é inicializado em admin/js/admin.js com M.FormSelect.init(...).
15	**Escrita e leitura no Web Storage**	**Cumprido (via Supabase)**	Eu não usei localStorage diretamente, mas a biblioteca do Supabase faz isso para manter o usuário logado. **Exemplo:** Quando supabase.auth.signInWithPassword() é executado com sucesso, a biblioteca salva a sessão no Local Storage do navegador.

RA3 - Aplicar ferramentas para otimização do processo.

ID	Requisito	Status	Como eu fiz no meu código
16	**Ambiente com Node.js e NPM**	**Cumprido**	Eu usei o ambiente Node.js/NPM através do comando npx para gerenciar as Supabase Edge Functions. **Exemplo:** Os comandos npx supabase functions new e npx supabase functions deploy são executados no terminal.
18	**Versionamento com Git e GitHub**	**Cumprido**	Eu criei e mantive o repositório do projeto no GitHub para armazenar e versionar todo o código.
19	**Organização do README.md**	**Pendente**	O arquivo README.md será preenchido por mim com os links e informações do projeto.

21	**Estrutura de arquivos coerente**	**Cumprido**	Eu organizei o projeto com cada página (login, admin, client) em sua própria pasta, e dentro de cada uma, subpastas para css e js, separando estrutura, estilo e lógica.

RA4 - Aplicar bibliotecas de funções e componentes em JavaScript.

ID	Requisito	Status	Como eu fiz no meu código
23	**Uso de jQuery**	**Não Aplicável**	Eu optei por não usar jQuery, utilizando "Vanilla JavaScript" (JavaScript puro) e as funções do Materialize, que é uma prática mais moderna.
24	**Integração de um plugin jQuery**	**Não Aplicável**	Pelo mesmo motivo acima, não utilizei plugins de jQuery.

RA5 - Efetuar requisições assíncronas para uma API.

ID	Requisito	Status	Como eu fiz no meu código
27	**Requisições assíncronas para persistir dados**	**Cumprido**	Eu usei funções assíncronas para criar e atualizar dados. **Exemplo:** No admin/js/admin.js, a função handleSaveUser chama a Edge Function create-user com await supabase.functions.invoke('create-user', ...), que é uma requisição POST para minha API no servidor.
28	**Requisições assíncronas para exibir dados**	**Cumprido**	Eu usei requisições assíncronas para buscar e exibir os dados do Supabase. **Exemplo:** No admin/js/admin.js, a função fetchUsers usa await supabase.from('profiles').select('*') para buscar a lista de todos os usuários (uma requisição GET) antes de exibi-la na tabela.

🏆 DESAFIO 🏆

ID	Requisito	Status	Como eu fiz no meu código
1	**Persistir dados em uma API Fake (a partir de um formulário)**	**Cumprido**	API real e robusta fornecida pelo Supabase. **Exemplo:** No painel de administrador (admin/js/admin.js), quando o formulário de "Novo Usuário" é preenchido e salvo, a função handleSaveUser é acionada. Ela não envia os dados diretamente para o banco, mas invoca minha Edge Function create-user de forma segura. Essa função, no servidor, recebe os dados do formulário (em formato JSON) e os persiste nas tabelas auth.users e public.profiles.
2	**Obter e manipular dados dinamicamente de uma API Fake**	**Cumprido**	Usei o Supabase para buscar e exibir listas de dados dinamicamente. **Exemplo:** Na página de administrador (admin/js/admin.js), a minha função fetchUsers faz uma requisição para a API do Supabase para buscar a lista completa de perfis. Os dados retornam como um array de objetos JSON, que então uso para popular a tabela de usuários na tela.
4	**Usar XMLHttpRequest, jQuery ou Fetch API (async/await)**	**Cumprido**	A biblioteca do Supabase (supabase-js) que utilei foi construída sobre a Fetch API, a tecnologia mais moderna para requisições web. Todas as minhas funções que usam await (como await supabase.from(...)) estão usando Fetch e Promises/async/await para funcionar.