**Página de Inscrição Inteligente**

**1. Visão Geral e Objetivo**

O objetivo é desenvolver uma única solução para a inscrição pública e inteligente, denominada inscricao.html. Esta página substituirá o sistema atual que exige a criação de múltiplos arquivos HTML idênticos para cada variação de campanha. A nova solução deverá ler parâmetros de URL para capturar dados de tráfego (UTMs), decidir dinamicamente qual fluxo e qual pagina de inscrição apresentar (Simples ou Composto) e salvar todas as informações de forma estruturada no banco de dados Supabase.

**2. Alterações Propostas no Banco de Dados (Supabase)**

Para suportar a nova lógica, as seguintes alterações no esquema do banco são necessárias:

* **Tabela campaign\_settings:**
  + **Adicionar Nova Coluna:** Criar uma nova coluna chamada survey\_id (ou inscricao\_survey\_id).
  + **Tipo:** uuid.
  + **Relação:** Esta coluna será uma chave estrangeira (foreign key) que se referencia à coluna id da tabela pesquisas.
  + **Propósito:** Criar um vínculo explícito e inequívoco entre um canal de tráfego (utm\_content) e a pesquisa específica que deve ser exibida no fluxo de inscrição composto. Isso elimina qualquer ambiguidade sobre qual pesquisa mostrar.
* **Tabela pesquisas:**
  + **Alteração de Dados (não de esquema):** O tipo de dado da coluna categoria\_pesquisa (provavelmente um enum ou text) deve ser atualizado para aceitar um novo valor: **'Inscrição'**.

**3. Alterações no Painel de Controle (index.html)**

Para permitir que o administrador configure a nova lógica, as seguintes alterações são necessárias no painel.html:

* **Formulário 'Criar/Editar Pesquisa' (template-criar-pesquisa):**
  + No dropdown de seleção de categoria (<select id="survey-category">), adicionar a nova opção:

HTML

<option value="Inscrição">Inscrição</option>

* + **Justificativa:** Isso permite que o usuário crie e designe pesquisas específicas para serem usadas no fluxo de inscrição.
* **Dashboard de Performance e Controle (Melhoria Sugerida):**
  + Ao clicar no botão "Alterar Fluxo" e selecionar "Composta", a interface idealmente deveria exibir um novo dropdown.
  + Este dropdown listaria todas as pesquisas com a categoria "Inscrição".
  + Ao selecionar uma pesquisa e salvar, o sistema preencheria a nova coluna survey\_id na tabela campaign\_settings para aquele utm\_content. Isso tornaria a configuração do fluxo completo mais intuitiva.

**4. Especificações Técnicas e Funcionais da Página (inscricao.html)**

* **4.1. Acesso e Captura de Dados:**
  + A página deve ser um arquivo autônomo.
  + Na carga da página, o JavaScript deve ler a URL (window.location.search) e extrair todos os seguintes parâmetros: launch\_id, utm\_source, utm\_medium, utm\_content, utm\_campaign, utm\_term.
  + O launch\_id é **obrigatório**. Se não estiver presente, a página deve exibir uma mensagem de erro.
  + Os parâmetros UTM capturados devem ser armazenados temporariamente para serem salvos junto com os dados do lead.
* **4.2. Lógica de Decisão do Fluxo:**
  + Com o utm\_content capturado da URL, o script deve fazer uma consulta na tabela campaign\_settings.
  + Se um registro para o utm\_content for encontrado, o script lerá o valor da coluna flow\_type.
  + Se flow\_type for **'composta'**:
    1. O script pegará o survey\_id da mesma linha na tabela campaign\_settings.
    2. Com o survey\_id, ele buscará os detalhes e as perguntas da pesquisa correspondente.
    3. A página exibirá tanto o formulário de inscrição quanto a pesquisa.
  + Se flow\_type for **'simples'** (ou se nenhum registro for encontrado para o utm\_content):
    1. A página exibirá apenas o formulário de inscrição básico.
* **4.3. Formulário e Validações:**
  + O formulário deve conter os campos: Nome Completo, E-mail, DDD e Número de Telefone.
  + Todos os campos são obrigatórios.
  + **Validação de Email:** Deve verificar se o formato do e-mail é válido.
  + **Validação de Telefone:** DDD e Número devem aceitar apenas caracteres numéricos.
* **4.4. Lógica de Submissão:**
  + O botão "QUERO MINHA VAGA" deve ser habilitado apenas após o preenchimento dos campos.
  + Ao submeter, o script deve executar as seguintes ações:
    1. Criar um novo objeto leadData contendo: nome, email, telefone (concatenando DDD + número), launch\_id e todos os parâmetros utm\_\* capturados.
    2. Inserir este objeto na tabela leads do Supabase.
    3. Se o fluxo for 'composto', coletar as respostas da pesquisa, criar um objeto JSON e inseri-lo na tabela respostas\_leads, junto com o id do lead recém-criado.
    4. Após o sucesso da submissão, exibir uma mensagem de confirmação/agradecimento na tela.
* **4.5. Requisito Visual:**
  + A página deve ser uma réplica visual exata da imagem de referência image\_77abd1.jpg.
  + O código deve prever a possibilidade de alterar dinamicamente o layout, texto principal e a imagem de fundo com base no utm\_source para se adequar a diferentes redes sociais.

**PROMPT DE DESENVOLVIMENTO: Motor de Inscrição Dinâmica**

**1. Objetivo Geral do Projeto**

Desenvolver uma solução robusta para a captura de leads que consiste em duas partes principais:

1. Uma **página de inscrição pública (inscricao.html)** que seja inteligente, dinâmica e capaz de se adaptar (em funcionalidade e layout) com base em parâmetros de URL.
2. Uma **seção no painel de controle (index.html)** que permita à equipe operacional gerenciar as regras que ditam o comportamento da página de inscrição, de forma ágil e sem a necessidade de alterar o código.

**2. Parte 1: A Página de Inscrição Pública (inscricao.html)**

Este será um arquivo HTML único e autônomo.

**2.1. Requisitos Funcionais:**

* **R.F.1 - Conteúdo Visual Fixo:** A página deve ter um layout visual principal, denominado 'padrao', que seja uma réplica fiel do design de referência (imagem com a professora em sala de aula: https://i.imgur.com/gK1NqGk.jpg). Layouts futuros ('facebook\_v1', etc.) serão adicionados posteriormente.
* **R.F.2 - Lógica de Ativação via utm\_content:** O comportamento da página será determinado pelo valor do parâmetro utm\_content na URL. Este valor atuará como um "identificador de canal".
* **R.F.3 - Fluxo de Inscrição "Simples":** Por padrão, ou se o utm\_content não for encontrado nas regras do banco de dados, a página deve exibir um formulário com três campos: Nome Completo, E-mail e Telefone.
* **R.F.4 - Fluxo de Inscrição "Completo":** Se o utm\_content corresponder a uma regra configurada com flow\_type = 'composta', a página deve exibir, **além** do formulário simples, uma pesquisa de check-in dinâmica.
* **R.F.5 - Pesquisa Dinâmica:** A pesquisa exibida no fluxo "Completo" não é fixa no HTML. Suas perguntas e opções de resposta (formato radio button) devem ser carregadas dinamicamente do banco de dados, com base no id\_pesquisa\_associada definido na regra do canal.
* **R.F.6 - Captura de Dados:** No envio do formulário, todos os dados devem ser capturados: os 3 campos do formulário, todos os 5 parâmetros UTM da URL e, se aplicável, as respostas da pesquisa.
* **R.F.7 - Validação:** O formulário deve ter validação de front-end para garantir que nome, e-mail (formato válido) e telefone (com DDD) sejam preenchidos corretamente. A pesquisa, se visível, também deve ser de preenchimento obrigatório.

**2.2. Especificações Técnicas:**

* **Stack:** HTML5, Tailwind CSS, JavaScript (ES6+), supabase-js v2.
* **Arquivo:** Deve ser um único arquivo inscricao.html.
* **Lógica Principal (handleDynamicFlow):**
  1. Ao carregar a página, a função deve ler o utm\_content da URL.
  2. Se o parâmetro existir, executar uma consulta select na tabela campaign\_settings do Supabase, buscando pela linha onde utm\_content corresponde ao valor lido.
  3. Se uma regra for encontrada e seu flow\_type for 'composta', chamar a função renderSurvey() passando o id\_pesquisa\_associada.
* **Renderização da Pesquisa (renderSurvey):**
  1. Receber um pesquisaId.
  2. Executar uma consulta no Supabase para buscar todas as perguntas e suas respectivas opcoes\_resposta que estão associadas àquele pesquisaId. (Dica: Usar uma RPC (Remote Procedure Call) no Supabase ou um select com inner join pode otimizar isso).
  3. Gerar dinamicamente o HTML para cada pergunta e seus radio buttons, e injetá-lo no div reservado (#checkin-survey).
  4. Tornar o div da pesquisa visível.
* **Envio de Dados:**
  1. No evento de submit, montar um objeto leadData para a tabela leads.
  2. Montar um objeto separado respostasData para a tabela respostas\_leads, contendo o lead\_id (a ser obtido após a inserção do lead) e um campo JSON com as respostas ({ "id\_pergunta\_1": "resposta\_A", "id\_pergunta\_2": "resposta\_C" }).
  3. Executar a inserção na tabela leads e, em caso de sucesso, na respostas\_leads.

**3. Parte 2: Alterações no Banco de Dados (Supabase)**

O objetivo é **ATUALIZAR** a estrutura existente de forma segura, sem perda de dados.

**3.1. Requisitos Funcionais:**

* **R.F.8 - Tabela de Controle Central:** A tabela campaign\_settings deve ser a fonte da verdade para todas as regras de exibição da página de inscrição.
* **R.F.9 - Suporte a Layouts e Pesquisas:** A tabela de controle deve ser capaz de armazenar qual layout visual, qual fluxo funcional e qual pesquisa específica devem ser usados para cada utm\_content.

**3.2. Especificações Técnicas:**

* **Script de Atualização (SQL):** Fornecer um script SQL único e seguro que executa as seguintes alterações:
  1. Na tabela campaign\_settings:
     + ALTER TABLE ... ADD COLUMN IF NOT EXISTS tipo\_layout TEXT DEFAULT 'padrao';
     + ALTER TABLE ... ADD COLUMN IF NOT EXISTS id\_pesquisa\_associada BIGINT REFERENCES public.pesquisas(id);
     + ALTER TABLE ... ADD COLUMN IF NOT EXISTS descricao TEXT;
  2. Na tabela leads:
     + ALTER TABLE ... ADD COLUMN IF NOT EXISTS id\_canal BIGINT REFERENCES public.campaign\_settings(id); (Opcional, mas boa prática para normalização).
* **Segurança:** O script deve garantir que as políticas de RLS (Row Level Security) estejam ativadas e que haja uma política de SELECT pública para campaign\_settings, pesquisas, perguntas e opcoes\_resposta, permitindo que a página de inscrição anônima possa ler as configurações.

**4. Parte 3: Alterações no Painel de Controle (index.html)**

As modificações devem ser integradas ao arquivo index.html existente, respeitando sua arquitetura de Single-Page Application (SPA) baseada em templates.

**4.1. Requisitos Funcionais:**

* **R.F.10 - Acesso à Gestão:** Um novo item de menu, "Gestão de Canais", deve ser adicionado ao painel, preferencialmente sob um novo grupo "Configurações", posicionado acima de "Ferramentas".
* **R.F.11 - Listagem de Regras:** A nova tela deve listar todas as regras existentes da tabela campaign\_settings, exibindo colunas como "Identificador (utm\_content)", "Descrição" e "Fluxo Atual".
* **R.F.12 - Edição Rápida de Fluxo:** A interface deve permitir que a equipe operacional altere o flow\_type de uma regra (simples <-> composta) de forma rápida e intuitiva, idealmente a partir da própria tela de listagem ou de um formulário de edição simples.
* **R.F.13 - Criação de Novas Regras:** Deve haver uma funcionalidade para criar novas regras, onde o usuário define o utm\_content, uma descrição, o tipo de fluxo e, se o fluxo for "composto", qual pesquisa associar.

**4.2. Especificações Técnicas:**

* **Segurança:** A inicialização do cliente Supabase no index.html **deve** usar a chave pública anon, não a chave service\_role.
* **Integração com a SPA:**
  1. **Menu:** Modificar o array menuItems no JavaScript para incluir o novo grupo e link.
  2. **Templates:** Adicionar dois novos blocos <template> ao HTML:
     + id="template-gestao-canais" para a tela de listagem.
     + id="template-form-canal" para o formulário de criação/edição.
  3. **Roteamento:** Adicionar as novas rotas ('gestao-canais', 'criar-canal', 'editar-canal') ao objeto routeMap no JavaScript.
  4. **Lógica (CRUD):** Implementar as novas funções JavaScript (initCanaisList, initCanalForm, etc.) que irão interagir com a tabela campaign\_settings no Supabase para realizar as operações de Listar, Criar, Ler e Atualizar (CRUD). A função de upsert do Supabase é ideal para o salvamento no formulário.