Antes de te enviar um documento técnico, vou explicar alguns itens:

Estamos tratando do processo de acompanhamento de vendas de cursos on line, através de Lançamentos Classicos.

O processo consiste em converter interessados, em leads, e depois efetuar vendas de cursos.  
O processo de captura dos interessados, é efetuado via área de trafego, através de banners e links em redes sociais, Depois que os interessados são transformados em Leads, eles são instruídos a preencherem pesquisas (check ins), para que a área de vendas possa ter orientação de procedimentos assim como a área de trafego, focar nos canais que possuem mais retorno.  
O app que foi desenvolvido e que já esta em produção, faz upload de planilhas googles sheet, com os dados dos leads e das respostas dos check in, através de 15 paginas de dashboards. E algumas paginas de ferramentas operacionais.  
Hoje, esse app, possui login com 2 tipos de perfs (ADM e VIEW), sendo que a área operacional possui perfil ADM e as demais áreas, possuem perfil VIEW.

Com a expansão do projeto para atender outas campanhas, precisamos que esse tipo de acesso, seja além de perfil, tenha um conceito de segregar a visão dos lançamentos, ou seja, a equipe operacional verá os lançamentos cadastrados, assim como as demais áreas verão somente os lançamentos cadastrados ao seu usuário.  
Ou seja, de acordo com o perfil (ADM ou VIEW) o usuário ve ou todo o menu do app, ou somente os dashboras e perguntas.  
E agora precisamos segregar a visualização dos dasboards, para quais lançamentos os perfis VIEWs, podem visualizar.  
Para isso já foram cridas tabelas assim distribuídas:  
profile : Possui o usuário e perfil.

Perfil\_de\_acesso: consta os nomes do perfis de acesso e o ID

Usuários\_perfis\_acesso: consta a amarração dos usuários com os perfis.

Lançamentos\_perfis\_acesso: consta quais perfis tem acesso ao lançamento

Essa é a nossa necessidade.  
Mas não estamos conseguindo efetuar essa divisão de visualização, e todos os esforços efetuados pelos outros chats, foi de que a pagina abre, e não mostra nenhum detalhe.  
  
Como eu não tenho conhecimento em escrita de códigos, não consigo efetuar a codificação necessária. Mas entendo de arquitetura e de análise, levantamento de requisitos e especificações.

SUPABASE\_ACCESS\_TOKEN= sbp\_9a2c0c349f0bfc066804d8d69984c5d5a9fa565a npx supabase gen types typescript --project-id xqsrkvrfvrqjzayrkzsp > src/types/database.ts

set SUPABASE\_ACCESS\_TOKEN= sbp\_533c54b1427d4a28ef426c7ddb9afaf93dbe356e&& npx supabase gen types typescript --project-id xqsrkvrfvrqjzayrkzsp > src/types/database.ts

**Documentação do Sistema de Gestão de Lançamentos e Scoring de Leads**

**Versão:** 1.0 **Data:** 23 de Agosto de 2025

**1. Visão Geral do Projeto**

Este documento descreve a arquitetura e os processos do sistema de gestão de lançamentos e qualificação de leads. O objetivo principal da plataforma é centralizar as informações de múltiplos lançamentos de produtos digitais, permitindo a importação, processamento e análise de leads para identificar os perfis com maior potencial de compra através de um sistema de pontuação (scoring).

**2. Arquitetura de Dados**

A base do sistema é composta por quatro tabelas principais que se relacionam para construir o perfil de cada lead dentro de um lançamento.

**2.1. Tabela public.lancamentos**

* **Conceito Funcional:** Representa um evento de marketing ou venda, como um curso online. É a entidade que agrupa todos os leads, perguntas e resultados associados a uma campanha específica.
* **Descrição das Colunas:**

| Coluna | Tipo de Dado | Descrição Funcional |
| --- | --- | --- |
| id | uuid | Identificador único para cada lançamento (Chave Primária). |
| nome | text | O nome comercial do lançamento (ex: "LC22", "TXT#22"). |
| status | text | O estado atual do lançamento. Os valores possíveis são: |
|  |  | **Planejado:** O lançamento está a ser preparado. |
|  |  | **Em Andamento:** O lançamento está ativo (carrinho aberto, etc.). |
|  |  | **Concluído:** O lançamento foi finalizado. |
|  |  | **Cancelado:** O lançamento foi descontinuado. |
| created\_at | timestamptz | Data e hora em que o registro do lançamento foi criado. |
| ... | ... | Outras colunas para informações adicionais como datas, descrição, etc. |

Exportar para as Planilhas

**2.2. Tabela public.perguntas**

* **Conceito Funcional:** É o catálogo mestre de todas as perguntas que podem ser feitas aos leads, seja na inscrição, no check-in ou após a compra. É aqui que a "inteligência" do scoring é configurada.
* **Descrição das Colunas:**

| Coluna | Tipo de Dado | Descrição Funcional |
| --- | --- | --- |
| id | uuid | Identificador único para cada pergunta (Chave Primária). |
| texto | text | O texto exato da pergunta (ex: "Qual a sua idade?"). |
| tipo | text | O formato da resposta esperada. Ex: multipla\_escolha, sim\_nao, verdadeiro\_falso. |
| classe | text | Define o propósito da pergunta. Usado para separar os scores. |
|  |  | **Perfil:** Pergunta para entender o perfil demográfico (não gera score). |
|  |  | **Score:** Pergunta usada para calcular o Lead Score (score). |
|  |  | **Comprador:** Pergunta específica para quem já comprou, usada para calcular o score\_comprador. |
| opcoes | jsonb | **(MUITO IMPORTANTE)** Um campo JSON que armazena as respostas possíveis e seus pesos. A estrutura é um array de objetos, onde cada objeto é uma opção de resposta. Ex: [{"texto": "Iniciante", "peso": 5}, {"texto": "Avançado", "peso": 20}]. |
| is\_mql | bool | Um campo que pode indicar se esta pergunta é um critério para MQL (Marketing Qualified Lead). |

Exportar para as Planilhas

**2.3. Tabela public.leads**

* **Conceito Funcional:** A tabela central do sistema. Cada linha representa uma pessoa (um lead) que se inscreveu em um determinado lançamento. Armazena suas informações de contato e, crucialmente, suas pontuações calculadas.
* **Descrição das Colunas:**

| Coluna | Tipo de Dado | Descrição Funcional |
| --- | --- | --- |
| id | uuid | Identificador único para cada lead (Chave Primária). |
| email | text | O email do lead, usado como identificador principal. |
| launch\_id | uuid | Identificador que conecta o lead ao lançamento a que pertence (Chave Estrangeira para lancamentos.id). |
| is\_buyer | bool | Um "flag" (marcador) que indica se o lead comprou o produto. false por padrão, muda para true na importação de compradores. |
| score | integer | **Lead Score.** Pontuação calculada com base nas respostas do check-in (perguntas de classe = 'Score'). Mede o potencial do lead *antes* da compra. |
| score\_comprador | integer | **Buyer Score.** Pontuação calculada com base nas respostas da pesquisa de comprador (perguntas de classe = 'Comprador'). Mede o perfil do lead *depois* da compra. |
| mql\_score | numeric | **MQL Score.** Uma pontuação separada que pode usar critérios específicos para identificar um Marketing Qualified Lead. |
| check\_in\_at | timestamptz | Data e hora em que o lead preencheu a pesquisa de check-in. |

Exportar para as Planilhas

**2.4. Tabela public.respostas\_leads**

* **Conceito Funcional:** A tabela que armazena as respostas efetivamente dadas por cada lead. Enquanto a tabela perguntas é o "formulário em branco", esta é a "pasta com os formulários preenchidos".
* **Descrição das Colunas:**

| Coluna | Tipo de Dado | Descrição Funcional |
| --- | --- | --- |
| id | uuid | Identificador único para cada registro de resposta (Chave Primária). |
| lead\_id | uuid | Conecta este conjunto de respostas a um lead específico (Chave Estrangeira para leads.id). |
| respostas | jsonb | Armazena as respostas gerais da pesquisa de check-in. |
| respostas\_perfil | jsonb | Armazena as respostas de perfil da pesquisa de check-in. |
| respostas\_comprador | jsonb | Armazena as respostas da pesquisa de comprador, importadas posteriormente. |

Exportar para as Planilhas

**Estrutura do JSON:** Em todas as colunas jsonb de resposta, a estrutura padronizada é um objeto onde a **chave é o ID da pergunta (uuid)** e o **valor é o texto da resposta** dada pelo lead. Ex: {"5cb34aed-e40c...": "FALSO", "09bfc659-df3e...": "45 a 54"}.

**2.5. Tabela public.utms\_leads**

**Conceito Funcional:** A tabela que armazena as informações do canal de entrada do lead.

| Coluna | Tipo de Dado | Descrição Funcional |
| --- | --- | --- |
| id | uuid | Identificador único para cada registro de resposta (Chave Primária). |

utm\_source TEXT,

utm\_medium TEXT,

utm\_campaign TEXT,

utm\_content TEXT,

utm\_term TEXT,

created\_at TIMESTAMPTZ Data e hora em que o lead preencheu a inscrição

**3. Processos Chave de Importação**

A entrada de dados na plataforma é feita principalmente através de importações, que seguem um fluxo padronizado.

* **Fluxo Padrão:**
  1. **Frontend (React):** O utilizador seleciona um lançamento e um arquivo CSV.
  2. **Supabase Storage:** O arquivo é enviado para uma área de armazenamento temporário.
  3. **Edge Function (Backend):** Uma função serverless é ativada. Ela lê o arquivo CSV, valida os dados, ignora linhas com erros, e prepara um payload JSON padronizado.
  4. **Função SQL (Banco de Dados):** A Edge Function chama uma função SQL, enviando o payload JSON. A função SQL executa a lógica de negócio (inserir, atualizar, calcular scores) de forma eficiente e segura.
* **Fontes de Dados:**
  1. **Arquivo CSV:** O método principal, usado pela interface web.
  2. **Google Apps Script:** Scripts em planilhas Google podem ser programados para enviar dados diretamente para as Edge Functions, seguindo o mesmo fluxo a partir do Passo 3.

**Tipos de Importação**

1. **Importação de Inscrições (Leads):**
   * **Objetivo:** Popular a tabela leads com os registos iniciais de uma campanha.
   * **Arquivo Mínimo:** Uma coluna de email.
2. **Importação de Check-in:**
   * **Objetivo:** Enriquecer os leads existentes com as respostas da pesquisa de perfil/score. É neste passo que o score geral é calculado.
   * **Arquivo:** Uma coluna de email e múltiplas colunas onde cada cabeçalho é o texto de uma pergunta.
   * **Lógica:** O sistema lê as respostas, busca os pesos na tabela perguntas (na coluna opcoes) e atualiza o score do lead. As respostas são salvas nas colunas respostas e respostas\_perfil da tabela respostas\_leads.
3. **Importação de Compradores:**
   * **Objetivo:** Marcar leads existentes como compradores e, opcionalmente, adicionar respostas de uma pesquisa pós-compra.
   * **Arquivo:** Uma coluna de email e, opcionalmente, colunas de perguntas/respostas de comprador.
   * **Lógica:** O sistema encontra os leads pelo email e define is\_buyer = true. Se houver respostas, ele calcula o score\_comprador (usando os pesos das perguntas de classe = 'Comprador') e salva as respostas na coluna respostas\_comprador da tabela respostas\_leads.

**Fluxo Técnico Detalhado**

A "jornada do dado" desde a planilha até o banco de dados segue os seguintes passos:

1. **Gatilho:** Uma edição na "coluna gatilho" (onEdit) ou um clique no menu inicia o processo.
2. **Captura:** O Apps Script lê a linha (ou linhas) correspondente, incluindo os cabeçalhos.
3. **Validação:** Ele verifica se a linha não foi processada antes e se possui um email válido.
4. **Criação do Payload:** A linha é transformada num objeto JSON, onde cada cabeçalho de coluna vira uma chave.
5. **Envio:** O payload JSON é enviado via requisição POST para a Edge Function processar-checkin-compradores.
6. **Processamento no Backend:** A Edge Function recebe o payload, valida a autenticação, encontra o lead e o lançamento, calcula os scores com base nas respostas e salva os dados de forma padronizada nas tabelas leads e respostas\_leads.
7. **Feedback:** A Edge Function retorna uma mensagem de sucesso ou erro.
8. **Atualização da Planilha:** O Apps Script recebe a resposta e escreve o status final na coluna de log da linha correspondente.