Documentação: Importação de Dados do Sistema Legado ABA (MySQL)

1. Visão Geral

Este documento descreve o processo implementado para importar dados essenciais do sistema legado ABA, que opera em um banco de dados MySQL remoto, para a nova estrutura de banco de dados no Supabase (PostgreSQL).

O objetivo principal é migrar dados de tabelas auxiliares como profissões, locais, salas, usuários específicos do ABA e seus relacionamentos, garantindo a consistência e permitindo que o novo sistema utilize esses dados.

A conexão com o banco de dados MySQL remoto é estabelecida de forma segura através de um **túnel SSH**, configurado utilizando as credenciais presentes no arquivo `.env`.

2. Tabelas Envolvidas

A importação foca nas seguintes tabelas do sistema ABA e suas correspondentes no Supabase:

Tabela Origem (MySQL)	Tabela Destino (Supabase)	Coluna ID Original (Supabase)	Notas
`ws_profissoes`	`profissoes`	`profissao_id`	Importa profissões.
`ps_locales`	`locais`	`local_id`	Importa locais de atendimento.
`ps_care_rooms`	`salas`	`room_local_id` (*)	Importa salas de atendimento.
`ws_users`	`usuarios_aba`	`user_id`	Importa usuários específicos do ABA.
`ws_users_profissoes`	`usuarios_profissoes`	N/A (Usa `usuario_aba_id` -> `usuarios_aba.id` e `profissao_id` -> `profissoes.id`)	Relaciona usuários ABA e profissões.

Tabela Origem (MySQL)	Tabela Destino (Supabase)	Coluna ID Original (Supabase)	Notas
`ws_users_especialidades`	`usuarios_especialidades`	N/A (Usa `usuario_aba_id` -> `usuarios_aba.id` e `especialidade_id` -> `especialidades.id`)	Relaciona usuários ABA e especialidades.
`ps_schedule_professionals`	`agendamentos_profissionais`	N/A (Usa `schedule_id` -> `agendamentos.id` e `professional_id` -> `usuarios_aba.id`)	Relaciona agendamentos e profissionais.

(*) Nota Importante sobre `salas`: O script SQL (`01_criar_tabelas.sql`) define `room_local_id` como a coluna para o ID original na tabela `salas`. No entanto, a lógica de relacionamento pós-importação tenta mapear usando `schedule_room_id` da tabela `agendamentos`. Verificar se `schedule_room_id` no MySQL contém o ID do *local* ou o ID da *sala* (`room_id`). Se contiver `room_id`, a coluna na tabela `salas` ou a lógica de relacionamento pode precisar de ajuste.

Mapeamento de IDs: As tabelas base no Supabase (`profissoes`, `locais`, `salas`, `usuarios_aba`, `especialidades`) possuem uma coluna específica (listada acima) para armazenar o ID original do registro correspondente no MySQL. Isso é **crucial** para que as tabelas de relacionamento (`usuarios_profissoes`, `usuarios_especialidades`, etc.) possam ser populadas corretamente, encontrando os UUIDs correspondentes no Supabase com base nos IDs originais do MySQL.

3. Endpoint Principal (`/importar-tudo-sistema-aba`)

Este endpoint (localizado em `backend/routes/importacao_routes.py`) orquestra todo o processo de importação. Ele executa as seguintes etapas em ordem, utilizando uma única conexão MySQL via túnel SSH:

1. **Importar `profissoes`** (`ws_profissoes` -> `profissoes`)
2. **Importar `locais`** (`ps_locales` -> `locais`)
3. **Importar `salas`** (`ps_care_rooms` -> `salas`)
4. **Importar `usuarios_aba`** (`ws_users` -> `usuarios_aba`)
5. **Importar `usuarios_profissoes`** (Mapeia IDs e insere em `usuarios_profissoes`)
6. **Importar `usuarios_especialidades`** (Mapeia IDs e insere em `usuarios_especialidades`)
7. **Importar `agendamentos_profissionais`** (Mapeia IDs e insere em `agendamentos_profissionais`) - *Atualmente comentado/desativado no código.*

O endpoint retorna um resumo dos resultados de cada etapa.

4. Mapeamento de IDs nas Tabelas de Relacionamento

Para importar tabelas como `usuarios_profissoes` e `usuarios_especialidades`, o processo é o seguinte:

1. Obter a relação do MySQL (ex: `user_id = 100`, `profissao_id = 5`).

2. Buscar na tabela `usuarios_aba` (Supabase) o registro onde a coluna `user_id` (ID original) é `100`. Obter o `id` (UUID) desse usuário.
3. Buscar na tabela `profissoes` (Supabase) o registro onde a coluna `profissao_id` (ID original) é `5`. Obter o `id` (UUID) dessa profissão.
4. Se ambos os UUIDs forem encontrados, inserir um novo registro na tabela `usuarios_profissoes` (Supabase) com os UUIDs encontrados.
5. Se um dos mapeamentos falhar (ex: usuário ou profissão com ID original não encontrado no Supabase), um erro/aviso é registrado e a relação não é inserida.

5. Relacionamento Pós-Importação (`/relacionar-agendamentos-com-tabelas-aba`)

Após a importação das tabelas auxiliares do ABA, este endpoint **separado** deve ser chamado para atualizar a tabela principal `agendamentos` no Supabase.

A tabela `agendamentos` pode conter IDs originais do sistema ABA nas colunas `schedule_room_id`, `schedule_local_id`, `schedule_especialidade_id`.

Este endpoint:

1. Busca todos os registros da tabela `agendamentos`.
2. Para cada agendamento, pega os IDs originais (ex: `schedule_local_id`).
3. Busca na tabela auxiliar correspondente no Supabase (ex: `locais`) o registro que possui aquele ID original na sua coluna específica (ex: `local_id`).
4. Obtém o `id` (UUID) do registro encontrado na tabela auxiliar (ex: o UUID do local).
5. Atualiza o registro do agendamento, preenchendo as colunas de relacionamento do Supabase (ex: `local_id_supabase`) com o UUID encontrado.
6. Repete o processo para salas e especialidades.

Atenção: Este passo depende da correta importação das tabelas auxiliares (`salas`, `locais`, `especialidades`) e da presença dos IDs originais nessas tabelas e na tabela `agendamentos`.

6. Pré-requisitos

- **Arquivo `.env`:** Deve conter as credenciais corretas para:
  - Conexão SSH (`SSH_HOST`, `SSH_PORT`, `SSH_USER`, `SSH_PASSWORD`)
  - Conexão MySQL (`MYSQL_HOST`, `MYSQL_PORT`, `MYSQL_USER`, `MYSQL_PASSWORD`)
- **Banco de Dados MySQL:** O banco de dados especificado (ex: `abalarissa_db`) deve existir no servidor MySQL remoto e ser acessível com as credenciais fornecidas.
- **Schema Supabase:** A estrutura das tabelas de destino no Supabase deve corresponder exatamente ao definido no script `sql/01_criar_tabelas.sql`, incluindo as colunas para armazenar os IDs originais do MySQL.

7. Solução de Problemas Comuns

- **Erros de Conexão SSH/Túnel:** Verifique as credenciais SSH no `.env`, o endereço do host/porta e se o servidor SSH está acessível. Verifique os logs do backend para detalhes.
- **Erros de Conexão MySQL:** Confirme as credenciais MySQL, o nome do banco de dados e se o MySQL está rodando no servidor remoto e acessível via túnel. Verifique os logs.

- **Erro `column ... does not exist`:** Geralmente indica uma divergência entre o nome da coluna esperado pelo código Python e o nome real da coluna no banco de dados Supabase. Atualize o script SQL (`01_criar_tabelas.sql`), aplique as mudanças no Supabase (recriando a tabela se necessário) e/ou corrija o código Python no backend.
- **Erro `Não foi possível mapear IDs para relação...`:**
  - Verifique se os dados base (ex: usuário em `usuarios_aba` ou especialidade em `especialidades`) com o ID original mencionado no erro realmente existem no Supabase.
  - Confirme se a coluna que armazena o ID original (ex: `user_id`, `especialidade_id`) na tabela base está preenchida corretamente para aquele registro.
  - Pode ser necessário re-executar a importação da(s) tabela(s) base correspondente(s).
- **Problemas de Permissão (Supabase):** Embora o script `01_criar_tabelas.sql` tente configurar permissões, erros de `permission denied` podem ocorrer. Consulte a documentação do Supabase sobre RLS (Row Level Security) e gerenciamento de roles.