ChatGPT

# SAL Collections Description

This document describes each of the database collections used in the SAL system and their role in the application. The goal is to provide developers with a clear guide on how data is organised so they can integrate the collections into the website and n8n workflows.

## 1. `clients`

Stores information about every end-user (client) served by a consultant. Each record belongs to a specific consultant (`owner_user`), ensuring data isolation between consultants.

| Field | Type/Relation | Notes |
|---|---|---|
| `id` | Integer (primary key) | Unique identifier. Auto-generated. |
| `owner_user` | Many-to-one → `directus_users` | The consultant who created/owns this client. Used for access control. Required. |
| `created_at` | Timestamp | Automatically set on creation. Indicates when the client was added. |
| `name` | String | Full name of the client. Required. |
| `birth_date` | Date | Date of birth of the client. Required. |
| `phone` | String | Contact phone number. |
| `email` | String | Contact e-mail. Mark the field as unique if desired. |
| `source` | String | Where the client came from (e.g. referral, website, social media). |
| `communication_method` | String | Preferred communication channel (phone, Zoom, WhatsApp, etc.). |
| `revenue` | Decimal | Total revenue earned from this client. Can be updated as consultations are completed. |
| `notes` | Text | Free-form notes about the client. |
| `code_personality` | String | SAL code for personality. |
| `code_connector` | String | SAL code for connector type. |
| `code_implementation` | String | SAL code for implementation. |
| `code_generator` | String | SAL code for generator role. |
| `code_mission` | String | SAL code for mission. |

| Field | Type/Relation | Notes |
|---|---|---|
| `linked_clients` | Many-to-many → `clients` | Used to link related clients (for example, family members or partners). Stored via an automatically generated junction table. |

Use this collection to list and search clients, display their details and contact information, and access linked profiles and consultations.

## 2. `profiles`

Represents a SAL profile calculated for a client. A profile contains the raw output from the AI (JSON) and a rendered HTML version for display. Profiles belong to both a client and the consultant who created them.

| Field | Type/Relation | Notes |
|---|---|---|
| `id` | Integer (primary key) | Unique identifier. Auto-generated. |
| `owner_user` | Many-to-one → `directus_users` | Consultant who owns the profile. Required. |
| `client_id` | Many-to-one → `clients` | Client for whom the profile was generated. Required. Cascade delete ensures profiles are removed when the client is deleted. |
| `raw_json` | JSON | Full structured response from the AI containing all profile information. Useful for future re-processing. |
| `html` | Text | Ready-to-display HTML version of the profile. |
| `status` | String (enum) | Status of the profile (`draft`, `ready`, `archived`). Default is `ready`. |
| `version` | Integer | Profile version number. Increment this when generating an updated profile for the same client. |
| `created_at` | Timestamp | Automatically set when the profile is created. |
| `consultation_id` | Many-to-one → `consultations` (optional) | Links the profile to a specific consultation, if the profile was produced as part of a meeting. |

A single client may have multiple profiles (e.g. one per consultation or per time period). Profiles can be used to display client insights on the website and feed context into Q&A.

## 3. `profile_chunks`

Stores small sections of each profile to enable efficient search and context retrieval for the Q&A feature. Each chunk references its parent profile.

| Field | Type/Relation | Notes |
| --- | --- | --- |
| `id` | Integer (primary key) | Unique identifier. |
| `profile_id` | Many-to-one → `profiles` | Parent profile. Cascade delete ensures chunks are removed when a profile is deleted. Required. |
| `section` | String | Name or category of the chunk (e.g. `narrative`, `strengths`, `weaknesses`, `conflicts/1`). |
| `content` | Text | The actual text content of this section. |

This collection allows the Q&A workflow to perform semantic or full-text search on discrete parts of a profile, improving answer relevance.

## 4. `qa`

Tracks questions asked by consultants (or clients) about a profile and the corresponding AI-generated answers. Each entry is tied to a specific profile and consultant.

| Field | Type/Relation | Notes |
| --- | --- | --- |
| `id` | Integer (primary key) | Unique identifier. |
| `owner_user` | Many-to-one → `directus_users` | Consultant who asked the question. Required. |
| `profile_id` | Many-to-one → `profiles` | Profile to which the question relates. Required. Cascade delete ensures Q&A entries are removed when the profile is deleted. |
| `question` | Text | The question text posed by the consultant. Required. |
| `answer` | Text | AI-generated answer. May be null initially and updated later. |
| `created_at` | Timestamp | Timestamp when the question was asked. |

Use this collection to display the history of questions and answers for each profile and to feed context into future questions.

## 5. `consultations`

Represents a consultation session between a consultant and a client. Consultations may have associated profiles and a detailed breakdown of notes or talking points (see `consultation_details`).

| Field | Type/Relation | Notes |
| --- | --- | --- |
| `id` | Integer (primary key) | Unique identifier. |

| Field | Type/Relation | Notes |
|---|---|---|
| `owner_user` | Many-to-one → `directus_users` | Consultant conducting the consultation. Required. |
| `client_id` | Many-to-one → `clients` | Client being consulted. Required. Cascade delete ensures consultations are removed with the client. |
| `created_at` | Timestamp | Date/time when the consultation record was created. |
| `scheduled_at` | Timestamp | Scheduled date and time of the consultation. |
| `type` | String / Enum | Type of consultation ( `base` , `extended` , `target` , `partner` ). Define additional types as needed. |
| `duration` | Integer (minutes) | Planned duration in minutes. |
| `base_cost` | Decimal | Standard cost for the consultation type from your price list. |
| `actual_cost` | Decimal | Actual cost charged (can be modified per client). |
| `status` | String / Enum | Current state of the consultation ( `scheduled` , `completed` , `cancelled` , etc.). |
| `profile_id` | Many-to-one → `profiles` (optional) | Reference to the profile generated during this consultation, if applicable. |

This collection allows scheduling and tracking consultations, including the ability to modify price and duration on a per-consultation basis.

## 6. `consultation_details`

Stores individual sections of content for each consultation. This flexible design allows you to add or remove sections without changing the schema.

| Field | Type/Relation | Notes |
|---|---|---|
| `id` | Integer (primary key) | Unique identifier. |
| `consultation_id` | Many-to-one → `consultations` | Parent consultation. Required. Cascade delete ensures details are removed with the consultation. |
| `section` | String | Name of the content section (e.g. `client_description` , `strengths` , `weaknesses` , `recommendations` ). |
| `content` | Text | Full text for the section. Allows unlimited length. |

Use this collection to capture structured notes and analyses during a consultation. You can display these sections in the client dashboard or export them as part of a report.

## Suggested Next Steps for Implementation

1. **Complete the consultations schema:** After creating the `consultations` collection, add the remaining fields (`type`, `duration`, `base_cost`, `actual_cost`, `status`, optional `profile_id`) in the Directus Data Model using **Create Field in Advanced Mode** → **Standard Field**. For enum fields (`type`, `status`), choose the **String** type and set the interface to **Dropdown** with the desired options.
2. **Create `consultation_details`:** Use the Data Model to add a new collection named `consultation_details`. Add fields `consultation_id` (Many-to-one → `consultations`, required), `section` (String), and `content` (Text). Ensure cascade deletion is enabled so details are removed when a consultation is deleted.
3. **Link profiles to consultations:** In the `profiles` collection, add a field `consultation_id` (Many-to-one → `consultations`) if you wish to tie each profile to the consultation from which it was generated. This field should be optional, as not all profiles are produced via a consultation.
4. **Adjust access policies:** Update the `master` role's access policy to include `consultations` and `consultation_details`. Ensure actions are restricted to records where `owner_user = $CURRENT_USER`, and for `consultation_details` ensure the nested filter `consultation_id.owner_user = $CURRENT_USER`.
5. **Extend the n8n workflows and frontend:** Modify your workflows to create and update `consultations` and `consultation_details` as part of your consultation scheduling process. Extend your website UI to display consultations, create new consultations, and manage consultation details.

This flexible schema is designed to accommodate future expansions—new types, additional fields, or extra sections—without breaking existing data.