Database

- Database Setup Documentation

Database Setup Documentation

Overview

The `on-internal-api` service uses a multi-database architecture primarily based on **PostgreSQL**, implemented using the **Sequelize ORM**. It connects to four distinct PostgreSQL databases and one MongoDB instance.

1. Environment Configuration

To run the application, you must configure the following environment variables in your `.env` file to establish connections to the required databases.

Sunshine Database (Main Employee/HR DB)

Variable	Description
`PG_SUNSHINE_HOST`	Database host address
`PG_SUNSHINE_USER`	Database username
`PG_SUNSHINE_PASSWORD`	Database password
`PG_SUNSHINE_DB`	Database name
`PG_SUNSHINE_DIALECT`	`postgres`
`PG_SUNSHINE_PORT`	Port (default: 5432)

Fleet Database

Variable	Description

`PG_FLEET_HOST`	Database host
`PG_FLEET_USER`	Database username
`PG_FLEET_PASSWORD`	Database password
`PG_FLEET_DB`	Database name
`PG_FLEET_dialect`	`postgres`
`PG_FLEET_PORT`	Port

CMS Database

Variable	Description
`PG_CMS_HOST`	Database host
`PG_CMS_USER`	Database username
`PG_CMS_PASSWORD`	Database password
`PG_CMS_DB`	Database name
`PG_CMS_dialect`	`postgres`
`PG_CMS_PORT`	Port

Mitra Database (Partners)

Variable	Description
`PG_MITRA_HOST`	Database host
`PG_MITRA_USER`	Database username
`PG_MITRA_PASSWORD`	Database password
`PG_MITRA_DB`	Database name
`PG_MITRA_DIALECT`	`postgres`
`PG_MITRA_PORT`	Port

(Note: There is also a `MONGO_MITRA` variable for a MongoDB connection).

---

2. Key Database Schemas & Tables

The application uses specific PostgreSQL schemas beyond the default `public`. You may need to create these schemas manually if Sequelize does not create them automatically.

- **Public**: Default schema for most tables.
- **Approval**: Used for `area` and `template_json` tables.
- **CMS**: Used for `ad` and other content management tables.

Sunshine DB Models

This database appears to handle employee management, authentication, and core organizational structure.

Table: `users`

- **Primary Key**: `username` (String)
- **Foreign Key**: `role_code` references `roles(code)`
- **Columns**:
  - `email` (String, Unique)
  - `password` (String)
  - `login_token` (JSONB)
  - `imei` (String)
  - `created_at`, `updated_at` (Timestamp)

Table: `roles`

- **Primary Key**: `code` (String)
- **Columns**:
  - `name` (String, Not Null)
  - `access_menu` (Array of Integers, Not Null)
  - `status` (Boolean)
  - `division` (String)

Table: `positions`

- **Primary Key**: `id` (Integer, Auto-increment)
- **Columns**:
  - `name` (String, Not Null)
  - `parent_id` (Integer) - Likely for hierarchy

Table: `division`

- **Primary Key**: `id` (Integer, Auto-increment)
- **Columns**:
  - `name` (String, Not Null)

- `status` (Boolean, Not Null)

Ticketing Models

Likely located in the Sunshine or Fleet database (referencing `vendors` and `agents`).

Table: `tickets`

- **Primary Key**: `tag` (String) - *Note: `id` is auto-increment but `tag` is marked Primary Key in definition.*
- **Foreign Keys**:
  - `vendor_code` -> `vendors(code)`
  - `agent_code` -> `agents(code)`
  - `problem_type_code` -> `problem_types(code)`
  - `created_by` -> `users(username)`
- **Columns**:
  - `waybill` (String)
  - `status` (Enum: 'Solved', 'Open', 'In Progress', etc.)
  - `description` (Text)
  - `attachments` (Array of Strings) - *Note: Contains logic to handle file paths.*

Approval Schema Models

Table: `approval.area`

- **Primary Key**: `area_code` (String)
- **Columns**:
  - `id` (Integer, Unique, Auto-increment)
  - `area_name` (String)
  - `large_area_group` (String)
  - `status` (String)

Table: `approval.template_json`

- **Primary Key**: `type` (String)
- **Columns**:
  - `template` (JSONB)
- **Timestamps**: False

Mitra DB Models

Table: `driver_login`

- **Primary Key**: `id` (BigInt)
- **Columns**:
  - `email` (String, Not Null)
  - `password` (String, Not Null)
  - `login_token` (String)
  - `online` (Boolean)
  - `disabled` (Boolean)

---

3. Initialization Steps

1. **Create Databases**: Ensure the 4 databases defined in the environment variables exist (`PG_SUNSHINE_DB`, etc.).
2. **Create Schemas**: Run the following SQL in the appropriate databases:

```
CREATE SCHEMA IF NOT EXISTS "approval";
CREATE SCHEMA IF NOT EXISTS "cms";
```

3. **Run Migration/Sync**: The application uses `sequelize.sync()` (implied by model definitions). When the application starts, it will attempt to create these tables if they do not exist. Ensure the database user has `CREATE TABLE` permissions.

4. Notes on Data Types

- **Arrays**: The application makes heavy use of PostgreSQL `ARRAY` types (e.g., `access_menu` in `roles`, `attachments` in `tickets`).
- **JSONB**: Used for dynamic data like `login_token` and `template`.
- **Virtual Fields**: several models (`roles`, `area`) define virtual fields (`code_name`, `formattedName`). These exist in the API layer only and are not columns in the database.