Database Documentation

1. Introduction

The ACT project modernizes legacy **Delphi code** by translating it into **C#**. To support this process, the system is containerized and uses dedicated storage systems for different types of data:

- **CouchDB** – A NoSQL document store for **user translation data**.
- **PostgreSQL** – Managed by **Authentik**, responsible for **user authentication and storage**.
- **Static Assets** – Images, stylesheets, and scripts are stored as files and served over **HTTP/HTTPS**.
- **Data Exchange** – System components communicate using **JSON**.

2. Database Use

2.1 CouchDB for User Translations (NoSQL Document Store)

- **Purpose:** Stores translation documents along with metadata.
- **Document Structure Example:**

```
json

CopyEdit

{
  "_id": "translation:unique-id",
  "user_email": "user@example.com",
  "delphi_code": "begin ... end;",
  "csharp_code": "public void Method() { ... }",
  "timestamp": "2025-03-04T12:00:00Z"
}
```

- **Relationships:**

- Each translation document includes a user_email field linking it to the corresponding user.
- As **CouchDB** is **schema-less**, these relationships are managed at the application level.

2.2 PostgreSQL via Authentik for User Information (Relational Database)

- **Purpose:** Securely manages user authentication and stores user information.
- **Entity-Relationship:** Authentik utilizes **PostgreSQL** to store user details.
- **Simplified Users Table Structure:**

Column	Type	Attributes
user_id	UUID	Primary Key
email	VARCHAR(255)	Unique
password_hash	TEXT	Secure Hash
metadata	JSONB	Additional user info

File Use

Static Files

- **Purpose:** Static assets (e.g., images, CSS, JavaScript) are stored in designated directories.
- **Directory Structure:**
  - `src/assets/Images/` – Stores **PNG, SVG** image assets.
  - `src/assets/stylesheets/` – Contains **CSS** files.
  - `src/pages/` – Holds **HTML** files (e.g., index.html, translation.html).

4. Data Exchange

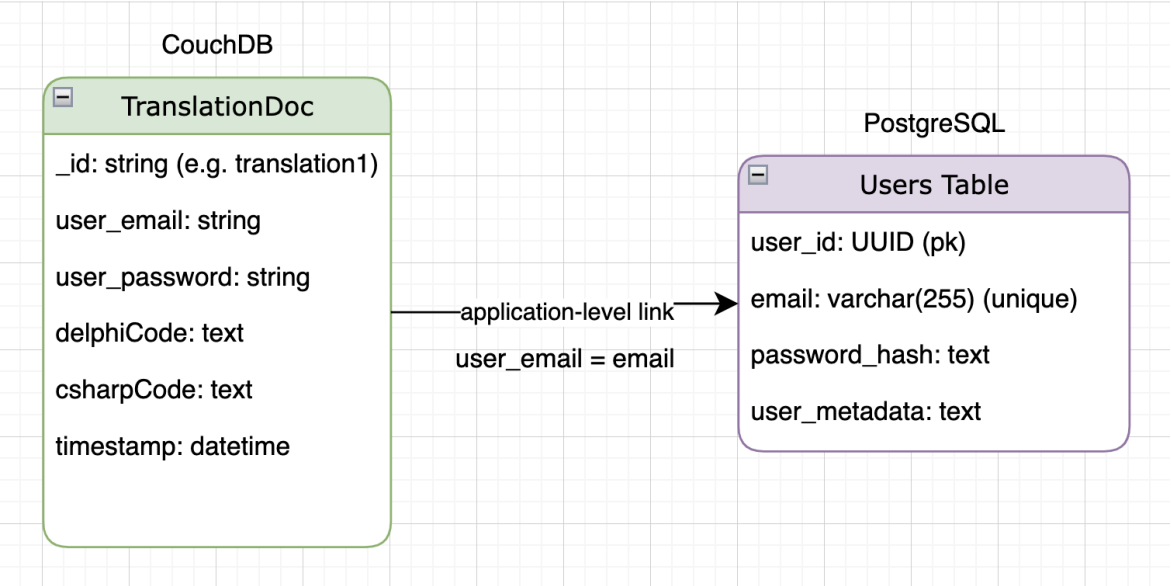
4.1 Format and Protocols

- **Data Format:**
  - **JSON** is used for API responses and token exchanges.
  - **Static files** are served in standard web formats (HTML, CSS, JS).
- **Protocols:**
  - API endpoints communicate via **HTTP/HTTPS**.
  - Authentication follows **OAuth 2.0 / OpenID Connect (OIDC)** protocols.

4.2 Security Considerations

- **Data Encryption:**
  - In **production**, all communication must occur over **HTTPS** to protect data in transit.
- **Authentication:**
  - User authentication is managed via **Authentik**, utilizing **OIDC**.
  - **JWT (JSON Web Tokens)** are used for secure session management.
  - **User credentials are never stored in plain text**.
- **Privacy & Data Exchange Security:**
  - **Personally Identifiable Information (PII)** is handled securely using **encrypted channels** and **strict access controls**.
  - **All API interactions** adhere to best practices for **data integrity and confidentiality**.

5. Diagramming Tools



This combined diagram was chosen to represent our mixed database approach, where CouchDB stores flexible, JSON-based translation documents and PostgreSQL securely manages user authentication data. By using a NoSQL document model for the translation data alongside a SQL table for user information, the diagram logically organizes our system and demonstrates the application-level relationship: the user_email field in each translation document links to the user email recorded in PostgreSQL. This approach highlights the benefits of using each database for its intended purpose while showing the interaction between the two and how they interconnect and store data within the overall system.

6. Summary

- **CouchDB** stores **translation documents** in a flexible **JSON format**, linking them to users via metadata.
- **PostgreSQL (via Authentik)** manages **user authentication** and stores structured user data.
- **Static files** (HTML, CSS, JS, images) are stored in organized directories and served directly.
- **Data exchange** occurs using **JSON over secure HTTP/HTTPS** connections, ensuring **data integrity**.

- **Security Measures** include **JWT authentication, HTTPS enforcement, and encryption** for sensitive data.

This system design ensures both **flexibility** in data storage (**NoSQL for translations**) and **strong security & integrity** for user data (**PostgreSQL and Authentik**).

Migration History

CouchDB:

- As a NoSQL schema-less store, CouchDB does not require formal migration tools.
- Any changes to document structure are handled in application code when documents are created or updated.

PostgreSQL (Authentik):

- Authentik automatically manages its schema using Django's migration system.
- To inspect or apply migrations manually:

```
docker exec -it authentik-worker ak migrate
```

- Current schema version aligns with Authentik v2025.2.0.

Seed Data Instructions

CouchDB:

- No automatic seeding.
- Sample translations can be added manually via a POST request to the backend API:

```
curl -X POST http://localhost:3000/save-translation \
  -H "Content-Type: application/json" \
  -d '{
    "delphiCode": "procedure Hello; begin Writeln(''Hello''); 
end;",
    "csharpCode": "public void Hello()
{ Console.WriteLine(\"Hello\"); }",
    "documentTitle": "Hello Example"
  }'
```

PostgreSQL:

- Seeded through a preloaded Authentik database dump.
- To reset and load:

```
docker stop authentik authentik-worker
docker exec -it postgresql psql -U authentik -d postgres
DROP DATABASE IF EXISTS authentik;
CREATE DATABASE authentik;
\q
docker exec -it postgresql psql -U authentik -d authentik -f 
/authentik_dump.sql
docker compose restart authentik authentik-worker
```