Technical Assessment Documentation

SEPTEMBER 6

EPI-USE

Authored by: Vian Reynecke

EPI-USE®

Employee Hierarchy Management System -

https://vian-epiuse.vercel.app/

Overview

A cloud-hosted application for managing employee hierarchies. It supports CRUD operations, reporting structure control, and visual hierarchy representation. Integrated with Gravatar for avatars and deployed on Vercel with Supabase for data persistence.

Key Features

1. Landing Page
Introduction and feature overview.

2. Home Page

- *2.1 Create Tab (Allows users to create new employees with input validation.*)
    - Reporting Line Manager: The CEO cannot have a reporting manager, and only one CEO is allowed.
    - Validation: Checks for duplicate emails, missing inputs, and correct formats.
    - Error Handling: Displays relevant messages for invalid or missing data.
- *2.2. Manage Tab*
    - **2.2.1 Hierarchy View**
        - Shows the employee hierarchy in a tree structure. Users can search by name, surname, or role, and highlight employees in the tree.
        - Edit Functionality: Users can edit employee data with validations:
            - Cannot change role if others report to them.
            - Cannot assign themselves as their own manager or subordinates as their managers.
            - Cannot delete employees with subordinates.
            - Includes all create tab validations.
        - **2.2.2 List View** (Displays a sortable and filterable table of employees.)

- **Sort & Filter:** Sort by name, role, email, or manager with combined filtering.
- **Search**: Search using the same criteria as the hierarchy view

3. Hierarchy Page
Displays the hierarchy in an expandable and collapsible tree format.

4. About Page:
Project information and links to personal profiles.

5. Dark/Light Mode:
Auto-adapts to user's system theme.

Technical Overview

The application uses an **API Gateway architecture**, centralizing request handling and security while separating frontend and backend concerns.

1. Technologies

- Frontend:
  - **Next.js:** For server-side rendering and static site generation, improving performance.
  - **Next UI:** Provides pre-built components for forms, tables, and hierarchy views.
  - **Tailwind CSS:** Ensures responsive design and supports dark and light modes.
- Backend:
  - **API Gateway:** Centralized entry point for requests, enhancing security and routing.
  - **Supabase:** Manages database operations and real-time updates with PostgreSQL.
  - **Gravatar:** Fetches employee profile pictures based on email hashes.
- Deployment:

- **Vercel**: Facilitates fast, scalable deployment with CI/CD integration. Linked to the main branch of my GitHub repository.

2. Justification for Technology Choices

- **API Gateway:** Ensures centralized request handling, which simplifies security and request routing between the frontend and backend.
- **Next.js:** Chosen for its ability to handle both client-side and server-side rendering, which enhances application performance, especially in a data-driven environment.
- **TypeScript:** Ensures robust type-checking, improving code reliability and preventing common bugs during development.
- **Supabase:** Provides a powerful backend-as-a-service solution, simplifying API creation, database management and real-time capabilities.
- **Gravatar Integration:** Reduces complexity by fetching profile pictures based on employee email addresses.
- **Vercel:** Provides seamless deployment and hosting for Next.js applications, ensuring fast performance, scalability and simple CI/CD integration.

3. Design Patterns

- **3.1 Repository Pattern**
  - Abstracts data access through services or utilities, enhancing maintainability and testability by separating it from business logic.
- **3.2 Observer Pattern**
  - Updates UI components reactively when data changes, ensuring the UI remains synchronized and design stays decoupled.
- **3.3 Factory Pattern**
  - Creates complex components or views for different roles, centralizing creation logic for greater flexibility and consistency.

Database Setup Overview

Database Management System: PostgreSQL

1. Table: *employees*

- **Purpose**: Stores comprehensive information about employees within the organization, including personal details, employment information and role assignments.
- **Table Structure:**
  - **id** (UUID): Unique identifier for each employee, serves as the primary key.
  - **name** (VARCHAR): First name of the employee.
  - **surname** (VARCHAR): Last name of the employee.
  - **birth_date** (DATE): Date of birth of the employee.
  - **employee_id** (VARCHAR): Unique internal identifier for each employee, used for internal tracking.
  - **salary** (NUMERIC or DECIMAL): The salary of the employee.
  - **email** (VARCHAR): Email address of the employee, utilized for Gravatar integration and communication.
  - **updated_at** (TIMESTAMP): Timestamp indicating when the record was last updated.
  - **created_at** (TIMESTAMP): Timestamp indicating when the record was created.
  - **role (role):** The employee's role, constrained by the custom ENUM type role.
  - **reporting_line_manager:** The role of the employee's manager, constrained by the custom ENUM type role.
  - **reporting_id**: (INTEGER): Foreign key referencing the id of the employee's manager, establishing hierarchical relationships.

2. Custom ENUM Type: role

Definition: ENUM type defining a set of roles within the organization.

Values:

- HR Manager

- IT Manager
- Finance Manager
- IT Intern
- HR Specialist
- Financial Analyst
- Accountant
- Senior Developer
- System Administrator
- Database Administrator
- Recruiter

3. Design Considerations

- Role Management: The reporting_line_manager field of type role captures the role of the manager, while reporting_id links to the actual manager's ID, allowing for both role-based and ID-based references in hierarchical relationships.
- Data Integrity: Foreign key constraints ensure that the reporting_id refers to a valid employee, maintaining accurate hierarchical relationships.
- Consistency: The role ENUM type enforces valid role values, ensuring consistency across role assignments.