10/3/2025

CACODEV – Congolese Association for Congo Development Management System

By

David Katembo

Fabrice Kadima

Pemphyle Nzuzi

Indiana University Southeast

CSCI P445: Capstone Project I Design Fall 2025

Professor Ronald Finkbine, Ph.D.

Table of Contents

1. Overview

1.1 Purpose

The purpose of this Software Architecture Specification (SAS) is to define the high-level architectural structure of the CACODEV management platform. It serves as a bridge between the system requirements and the detailed design implementation.

1.2 Scope

The CACODEV system is a comprehensive management platform designed to facilitate operations for the Congolese Association for Congo Development. The architecture covers the web-based frontend, the backend API services, and the database persistence layer.

1.3 Architecture Goals

The proposed software architecture follows a **Client-Server** model utilizing a **Layered Architecture** pattern.

- **Web-based:** Accessible via standard web browsers.
- **Modular:** Composed of distinct subsystems for Members, Events, Contributions, and Activities.
- **Secure:** Protecting sensitive member and financial data.
- **Scalable:** Deployment-ready for cloud environments.

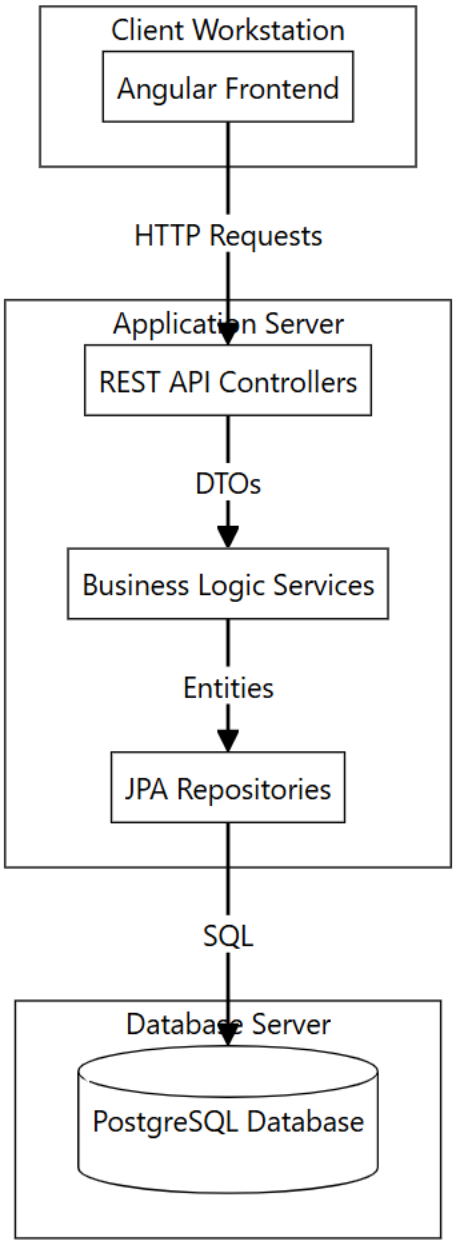
2. Subsystem Decomposition

2.1 Overview

The system is decomposed into three primary layers: Presentation, Business Logic, and Data Access. The responsibilities are allocated as follows:

- **Frontend Subsystem (Angular):** Responsible for user interaction, form validation, and data display.
- **Backend Subsystem (Spring Boot):** Responsible for API endpoints, business rules execution, and request processing.
- **Database Subsystem (PostgreSQL):** Responsible for persistent storage and data integrity.

2.2 Component Diagram



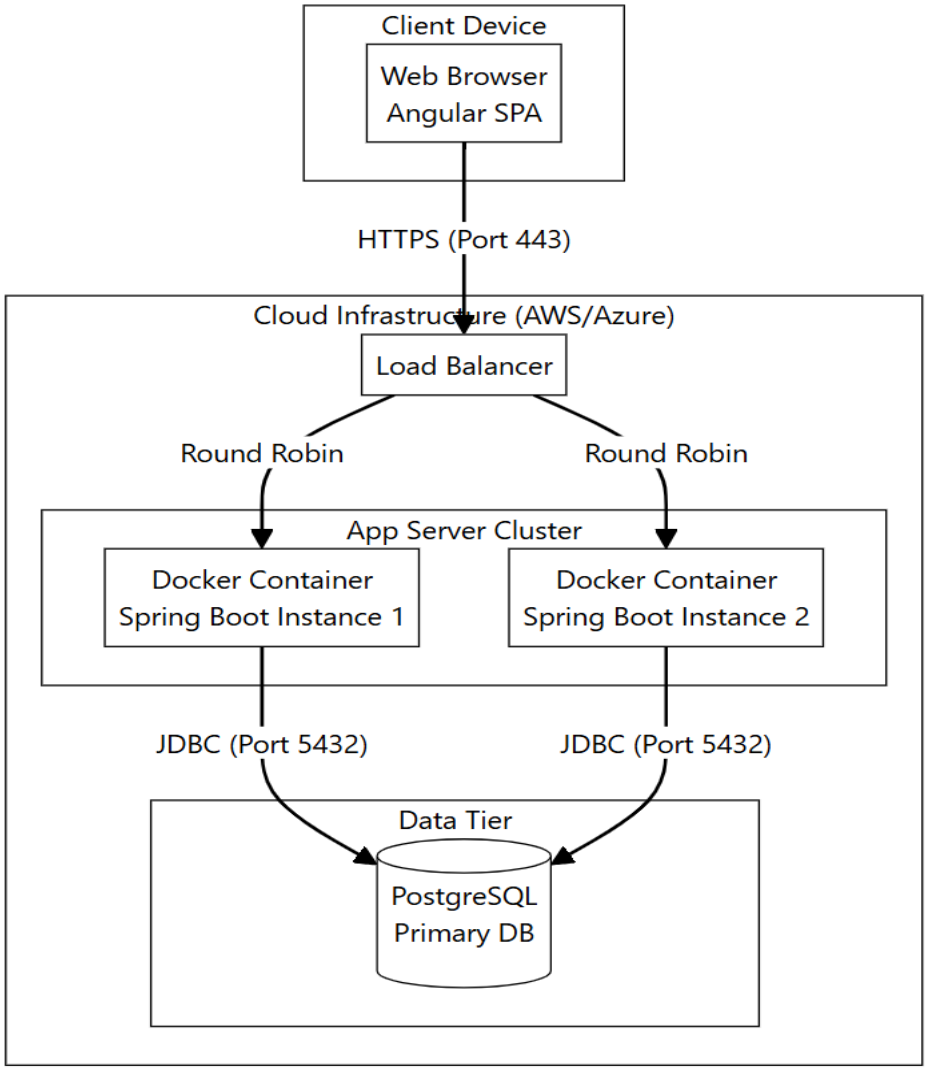
3. Hardware/Software Mapping

3.1 Overview

The subsystems are assigned to hardware nodes as follows:

- **Client Node:** A user's personal computer or mobile device running a modern web browser. It executes the Angular Single Page Application (SPA).
- **Application Server Node:** A cloud-hosted virtual machine or container running the Java Runtime Environment (JRE). It hosts the Spring Boot backend application.
- **Database Server Node:** A dedicated server instance running the PostgreSQL database engine.

3.2 Deployment Diagram



4. Persistent Data Management

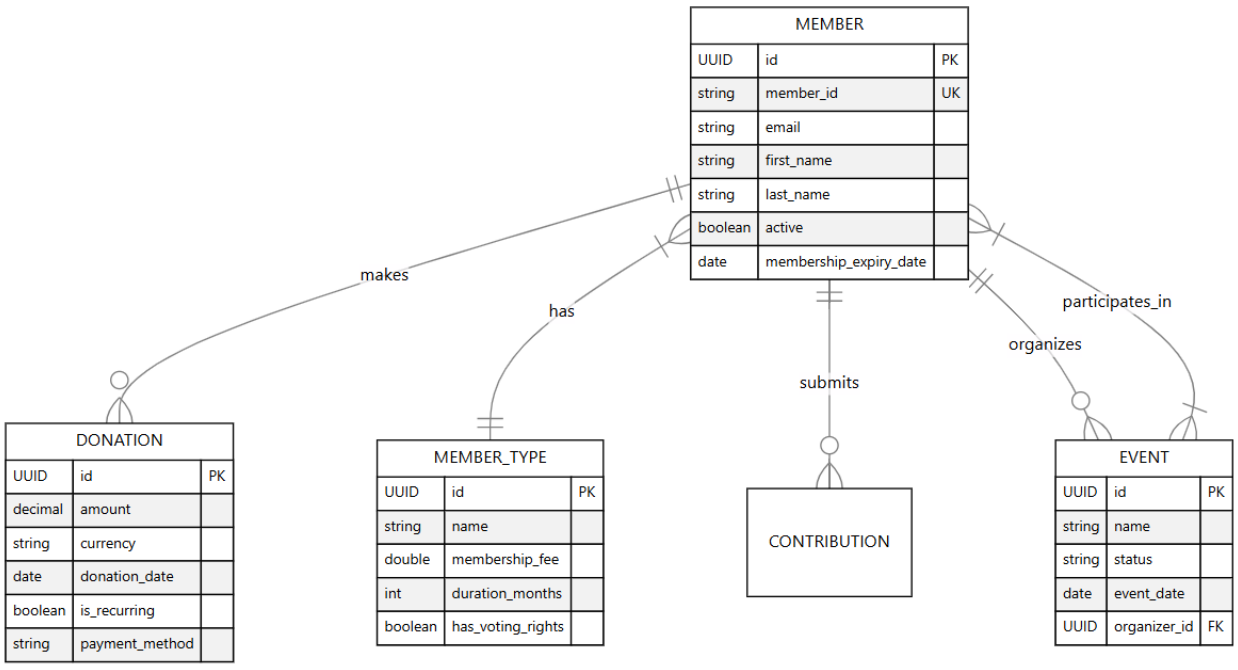
4.1 Data Persistence Strategy

The persistent data for the system includes member profiles, transaction records, event schedules, and system configuration. This data is managed by a **Relational Database Management System (RDBMS)**, specifically PostgreSQL.

- **Storage:** Relational tables with foreign key constraints to ensure referential integrity.
- **Transactions:** All financial operations (Contributions/Donations) are wrapped in ACID-compliant transactions.
- **Backup:** Daily snapshots of the database volume are stored in encrypted object storage (e.g., AWS S3).

4.2 Complete Database Design (ER Diagram)

The following Entity-Relationship Diagram describes the schema used for persistent storage.



5. Access Control and Security

5.1 Actors and Operations

The system distinguishes between several actors, each enabled to perform specific operations:

Actor	Enabled Operations
Administrator (Founder)	• Create/Delete Members • Configure Member Types • View All Financials • Manage System Users
Board Member	• Approve Budgets • View Financial Reports • Manage Events • Edit Member details
Regular Member	• View Own Profile • Pay Contributions • Register for Activities • View Public Events
Public User	• View Landing Page • View Public Events Calendar • Register as new Member

5.2 Authentication and Security Provisions

- **Authentication:** The system uses **JSON Web Tokens (JWT)** for stateless authentication. Upon successful login, the server issues a signed token which must be included in the header of subsequent API requests.
- **Password Security:** User passwords are stored using **BCrypt** hashing with a strength factor of 12.
- **Transmission Security:** All data in transit is encrypted using **HTTPS (TLS 1.2+)**.
- **Input Validation:** The backend implements strict validation logic to prevent SQL Injection and Cross-Site Scripting (XSS).

6. Global Software Control

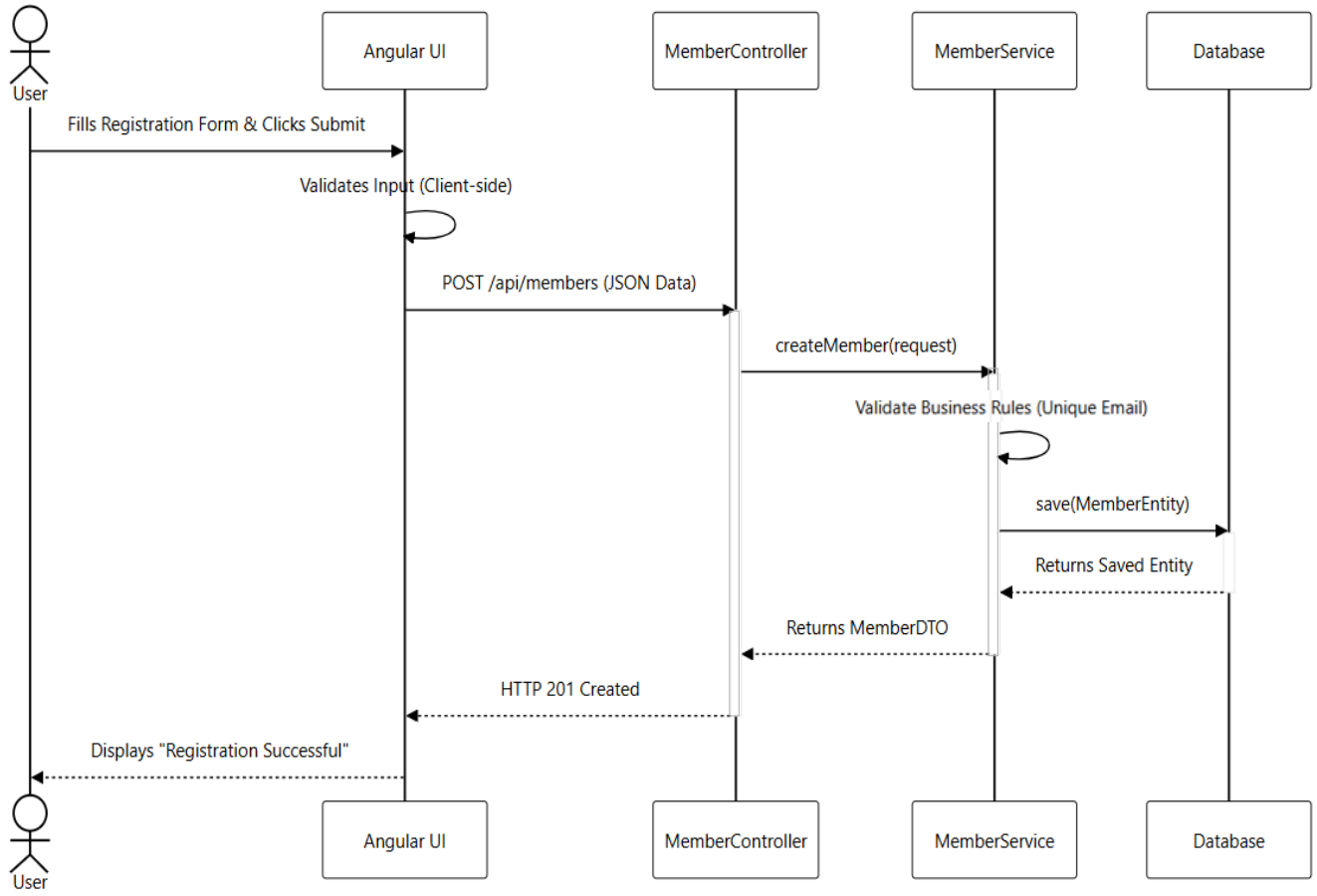
6.1 Control Flow Description

The CACODEV application follows an **Event-Driven (Request-Response)** control flow.

- **Event-Driven:** The user initiates actions on the frontend (clicks, form submissions).
- **Threaded Processing:** The backend server allocates a separate thread for each incoming HTTP request, processes it synchronously, and returns a response.

6.2 Sequence Diagram (Event-Driven Flow)

The following Sequence Diagram illustrates the control flow for the "Member Registration" use case.



7. Boundary Conditions

7.1 Startup and Initialization

- **System Startup:** The backend application is initialized via the Spring Boot framework. On startup, it establishes a connection pool to the database.
- **Data Seeding:** If the database is detected as empty (e.g., first run), a DataInitializer routine runs automatically to populate default Member Types (e.g., "Founder", "Regular") and the initial Administrator account.

7.2 Error and Exception Handling

The system handles errors globally to prevent crashes and information leakage.

- **Exceptions:** Unhandled exceptions are caught by a global @ControllerAdvice handler.
- **Response:** The system returns a standardized JSON error response (e.g., 500 Internal Server Error) to the client, while logging the full stack trace securely to the server logs for developer review.

7.3 Data Migration and Bulk Operations

- **Server Migration:** To migrate to a new server, the system supports database dumps. Administrators can generate a complete SQL dump file.
- **Bulk Load/Dump:** The system includes scripts to execute pg_dump (for export) and psql (for import), allowing the entire application state to be moved to new hardware without data loss.

8. Breakdown of Individual Contributions

Team Member Name	Role	Specific Contributions
David Katembo	**Team Leader / Architect**	• Subsystem Decomposition • Hardware/Software Mapping • Global Control Flow Design
Fabrice Kadima	**Backend Developer**	• Access Control & Security Specification • Persistent Data Management (Schema) • Boundary Condition Definitions
Pemphyle Nzuzi	**Frontend Developer**	• Overview & Scope Definition • Client-Side Architecture • Diagram Generation

9. Key Personnel Information

- **David Katembo** - Team Leader
- **Fabrice Kadima** - Development Team
- **Pemphyle Nzuzi** - Development Team