Backend Architecture Overview

The backend architecture of our Citizen-centric Human Resource Management System in Smart Cities is structured into three primary layers: the Presentation Layer, the Application Layer, and the Data Layer. Each layer has distinct responsibilities and ensures a modular, maintainable, and scalable system.

1. Presentation Layer:

Frontend Technology:

The presentation tier is a single-page application (SPA) developed primarily using Vue.js.

Component-Based Architecture:

The UI is structured as a set of reusable components that are composed into views to implement various use cases.

2. Application Layer:

Backend Framework:

The backend services are implemented using Spring Boot, a robust and widely used framework for building Java applications.

Layered Architecture:

The application tier follows a three-layer architecture:

Controller Layer: Utilizes spring-web-mvc for handling HTTP requests.

Business Layer: Written primarily in plain Java, containing the core business logic.

Data-Access Layer: Implemented via spring-data for database interactions.

Modular Monolith:

Despite being a monolith, the application tier is modular, with each module representing a bounded context. This allows for vertical slicing of each layered architecture.

3. Data Layer:

Database:

The data tier consists of a PostgreSQL database with the TimescaleDB extension enabled. This setup facilitates high-performance analytics ingestion and processing.

Database Configuration:

Managed using Flyway for robust version control and efficient schema migrations.

Domain-Driven Design and Bounded Contexts:

Vertical Slicing:

The application tier is divided into vertical slices based on bounded contexts. Each context represents a distinct domain and operates independently.

Isolation and Loose Coupling:

Each bounded context is isolated to limit coupling, allowing parallel development and easier maintenance.

HTTP API:

Interface Layer:

The HTTP API acts as the bridge between the presentation and application tiers. It provides a set of operations implemented by the application tier and consumed by the presentation tier.

Controllers:

Each bounded context has its own set of controllers, like OccupationController, to handle specific operations.

Backend Services by Context:

Analytics: Handles clickstream analytics and reporting.

Certifications: Manages certifications using datasets like CareerOneStop.
Demand: Manages employment targets for various occupations.
Employment: Tracks employment statistics for occupations.
Job Postings: Aggregates job postings related to occupations.
Learning Material: Manages CMS-based learning content.
News: Aggregates news articles from various RSS feeds.
Occupations: Manages occupation descriptions and metadata.
Unemployment: Tracks unemployment statistics.
Users: Manages user profiles and related data.

Operational Scenarios:
The document outlines various operational scenarios, such as creating and administering accounts, viewing occupation information, selecting job goals, and more. These scenarios are supported by corresponding activity diagrams and describe the interaction flow within the system. This architecture ensures scalability, maintainability, and parallel development efficiency by leveraging a well-structured, modular monolith design and domain-driven design principles.

Architecture Diagram
Here is the architecture diagram illustrating the layered architecture of the backend:
This diagram clearly demonstrates the separation of concerns across the presentation, application, and data layers, ensuring a scalable and maintainable backend system for the Citizen-centric Human Resource Management System.