Swathi S(210701274)

**CVE data fetch from National Vulnerability**

**Database**

Description :

Cve data fetching and displaying dynamically.

**The layers of the application:**

This application was developed using the microservice based framework.

**Logic:**

The architecture include:

The Entity Layer includes the main entities of the application, that contains the attributes and structures for each of them. Key entities defined are:

- CVE data

- Description data

- CvssMetricV2

- Weaknesses

- References

Service Layer : This layer provides the complete logic of the application that handles how the data is fetched and converted into objects defined by the Data Tranfer Object Layer and finally saving it into the database.

- Data fetching

- Data conversion to DTOs

- Database operations (CRUD)

Repository Layer -The Repository Layer provides an interface for each entity defined in the Entity Layer. These interfaces extend the JpaRepository, enabling seamless interaction with the underlying database. Key responsibilities include:

- Defining repository interfaces

- Extending JpaRepository for CRUD operations

Controller Layer - The Controller Layer manages API integration which is fetched from the [services.nvd.nist.gov/rest/json/cves/2.0](https://services.nvd.nist.gov/rest/json/cves/2.0) that is accessed by the “/fetch-and-store/” in Postman API using GET method, serving as the entry point for external requests.

Data Transfer Object Layer -

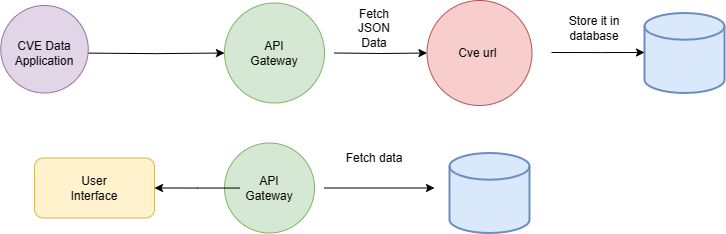
The Data Transfer Object Layer defines the structure of DTOs used for data exchange between different layers of the application. It includes logic for attribute definition and accessor methods.

**Tech stack used**

- Java SpringBoot: For building microservices architecture based application.

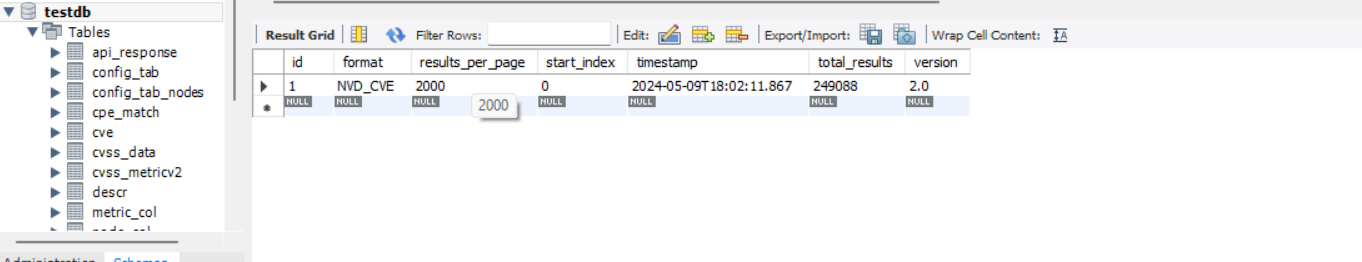
- MySQL: Relational database management system for storing application data.

Flow of data Diagram:

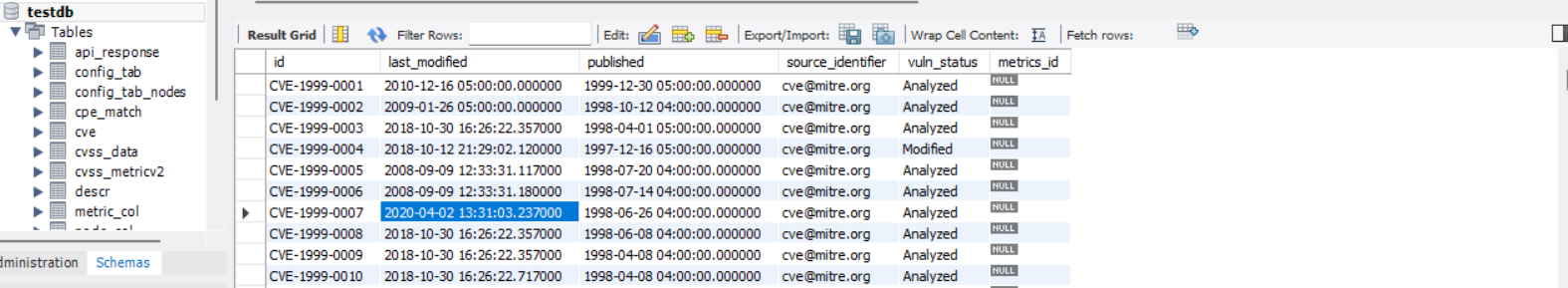


**Database :**

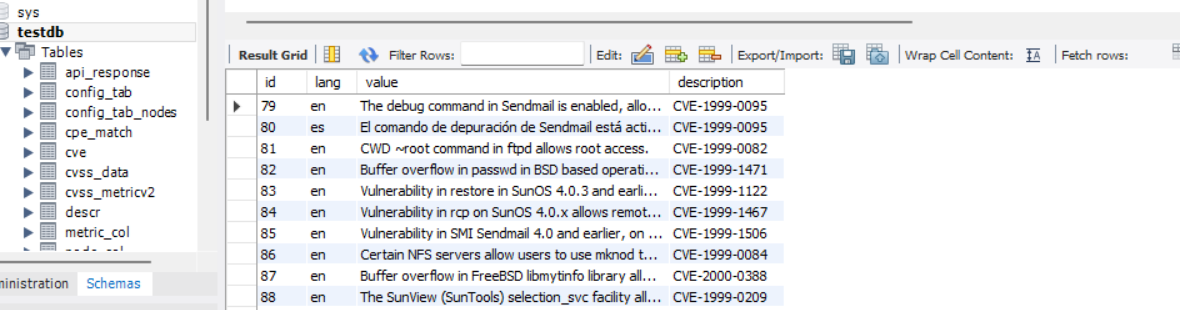
Response table:



Cve table:



Description table:



Business logic of the main methods of Service layer

1. **fetchAndSaveApiResponse(String url)**:

**Process**:

* + - It uses RestTemplate to make a GET request to the provided URL.
    - If the response is successful (2xx status code) and contains a body, it proceeds.
    - The response body is deserialized into an ApiResponse object using ObjectMapper.
    - It iterates over the VulnerabilityItem objects within the ApiResponse.
    - For each CVE object, it checks if the ID is null. If so, it saves the CVE to the cveRepository.
    - It does a similar check for each Description within the CVE, saving it to the descriptionRepository if the ID is null.
    - If there is a CvssMetricV2 object and its ID is null, it saves it to the cvssMetricV2Repository.
    - Finally, it saves the entire ApiResponse object to the apiResponseRepository.
    - It converts the saved ApiResponse entity to a DTO and returns it.

1. **convertToDTO(ApiResponse apiResponse)**:

**Process**:

* + - It creates a new ApiResponseDto object.
    - It sets the properties of the ApiResponseDto with the corresponding values from the ApiResponse entity.
    - It extracts the IDs of the VulnerabilityItem objects and sets them in the ApiResponseDto.
    - It returns the populated ApiResponseDto.

**Output :** 