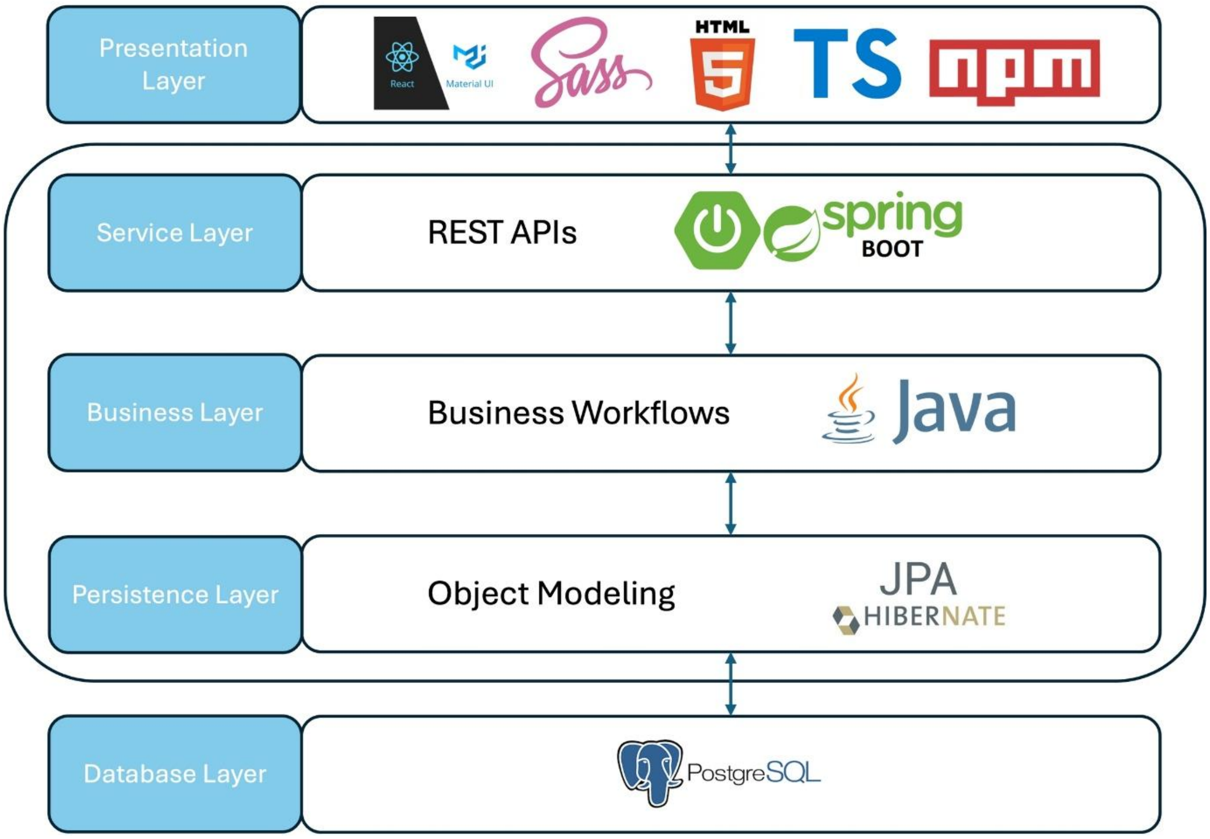
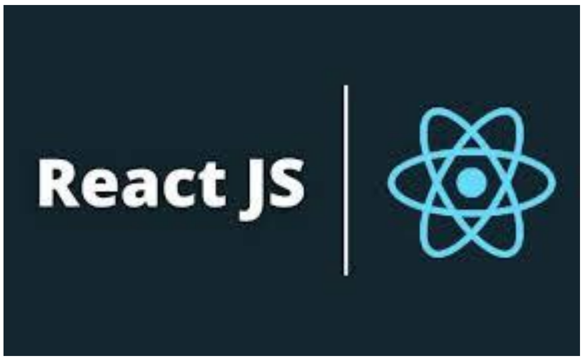
Technology Stack

The Technology Infrastructure of SPICE forms the backbone of its functionality, performance, and scalability.



SPICE Layer Architecture. The SPICE platform utilizes a layered architecture approach with three tiers: 1. Presentation Layer 2. Computer Layers and 3. Database Layers.

Presentation Layer



React

SCSS

Typescript

SPICE's presentation layer relies on a combination of technologies, including React, HTML5, SCSS, TypeScript, and npm. Together, these technologies form a robust and efficient foundation for the application's presentation layer.

- React is a JavaScript library that enables interactive user interfaces
- HTML5 serves as the markup language for structuring the application's content
- SCSS supports writing more maintainable and modular CSS styles.
- TypeScript adds static typing to JavaScript, enhancing code reliability and scalability.
- Npm is a package manager that helps manage and install the necessary dependencies for the project.

Service Layer

Spring Boot

The service layer of the SPICE application leverages the Spring Boot framework to develop REST APIs. Spring Boot simplifies the process of creating and managing these APIs, allowing for seamless communication with other systems or clients. By utilizing features such as dependency injection, security, and database integration, we ensure efficient and scalable services. REST APIs enable SPICE to follow the principles of stateless, client-server communication, facilitating interoperability and flexibility in the SPICE application's architecture. With Spring Boot, we can build reliable and high-performing components for our service layer, providing a solid foundation for SPICE application functionality.

Business Layer

Java

The business layer of the SPICE application is implemented using Java to create business workflows. Java provides a robust platform for developing intricate and efficient business logic, given the fact that it is a versatile and widely-used programming language. Leveraging Java's object-oriented features, we can model complex business processes and incorporate encapsulation, inheritance, and polymorphism to achieve modular and maintainable workflows. Java's extensive libraries and frameworks enhance the capabilities of our business layer, enabling us to handle data transformations, business rules, and workflow orchestration effectively. By coding the business workflows in Java, we ensure a reliable and scalable foundation for the application's core functionality.

Persistence Layer

JPA - Hibernate

The SPICE persistence layer utilizes Object Modeling based on JPA using Hibernate. JPA (Java Persistence API) is a Java specification that provides a standard way to map Java objects to relational databases. Hibernate, an implementation of JPA, simplifies the interaction between the application and the database by handling object-relational mapping. With JPA Hibernate, we can define and manage the persistence of our application's domain objects, enabling seamless storage and retrieval of data. This approach enhances code maintainability, as it abstracts away the underlying database details. Leveraging JPA Hibernate in our business layer ensures efficient and reliable data persistence in our application.

Database Layer

PostgreSQL

The database layer of SPICE is built on PostgreSQL, an open-source, relational database management system (RDBMS). PostgreSQL is known for its stability, scalability, and extensive feature set. It provides robust data storage and retrieval capabilities, ensuring the efficient handling of our application's data. With PostgreSQL, we can create and manage complex database structures, define relationships between entities, and perform efficient queries. It offers advanced features such as transactions, concurrency control, and data integrity mechanisms. Leveraging PostgreSQL in our database layer guarantees reliable and secure storage of our application's data, supporting the overall performance and functionality of our system.

FHIR Service Layer

FHIR Standards

Fast Healthcare Interoperability Resources (FHIR) is a standard developed by HL7 (Health Level Seven) for exchanging healthcare information electronically. It is designed to facilitate interoperability between different healthcare systems by using modern web technologies such as RESTful APIs, JSON, and XML. FHIR provides standardized data models and resources (e.g., Patient, Observation, Medication) that simplify data sharing across healthcare applications.

Key benefits of FHIR include:

- **Interoperability:** Enables seamless exchange of health data between systems.
- **Scalability:** Supports various use cases, from small applications to large healthcare enterprises.
- **Ease of Implementation:** Uses familiar web-based technologies.
- **Modular Design:** Allows developers to use only the necessary resources for their application.

HAPI FHIR JPA Server

HAPI FHIR JPA Server is an open-source implementation of the FHIR standard, built using the **HAPI-FHIR** library. It provides a ready-to-use, RESTful FHIR server that uses a **JPA (Java Persistence API) database backend** for data storage.

Key features of HAPI FHIR JPA Server:

- **Full FHIR Support:** Implements FHIR RESTful APIs for CRUD operations.
- **Database-backed Storage:** Uses JPA/Hibernate to store FHIR resources in relational databases.

- **Security & Authentication:** Supports OAuth2, SMART on FHIR, and custom authentication mechanisms.
- **Extensibility:** Allows customization to fit specific healthcare application needs.
- **Terminology Services:** Supports validation and terminology operations.

HAPI FHIR JPA Server is widely used for building healthcare applications that require standardized FHIR data storage and retrieval while ensuring compliance with HL7 FHIR specifications.