Faculty of Technology

University of Sri Jayewardenepura

ITC 4243

Microservices and Cloud Computing

Assignment 1

Name: G.S. Chamika

Index Number: ICT/20/818

1. What is an API? Compare API VS REST API listing the Features of REST APIs.

API (Application Programming Interface) is a collection of communication protocols, rules, and tools that enable various software systems or programs to communicate with each other. It defines how software components should interact, allowing applications to exchange data and functionality seamlessly. APIs can be used for web services, libraries, operating systems, or hardware interactions.

Feature	API	REST API
Architectural Style	Can follow any design (RPC, SOAP, custom protocols).	Follows REST (Representational State Transfer) principles.
Protocol	May use HTTP, SOAP, FTP, or other protocols.	Uses HTTP/HTTPS exclusively.
HTTP Methods	May not fully utilize standard HTTP verbs.	Uses HTTP methods (GET, POST, PUT, DELETE) for CRUD operations.
Data Formats	Supports XML, JSON, binary, or custom formats.	Uses JSON or XML making it lightweight and easy to parse.
State Management	Can be stateful or stateless.	Stateless: Each request must be independent.
Endpoints	Often uses action-based URLs (/getUser).	Uses resource-based URLs (/users/{id}).
Performance	Can be slower due to heavier payloads (SOAP with XML).	Optimized performance via caching, smaller payloads (JSON), and statelessness.
Flexibility	More flexible in implementation.	Strictly follows REST architectural constraints.
Scalability	Less scalable if stateful.	Highly scalable due to statelessness and caching.

2. Why Spring Boot is Much easier to use than the traditional Spring framework.

Spring Boot simplifies application development by addressing the complexities and boilerplate code associated with the traditional Spring Framework. Basically, the Spring framework does not include a command-line interface (CLI) tool for developing and testing applications. Spring Boot includes a CLI tool for developing and testing applications. Here's a breakdown of why Spring Boot is much easier to use than the traditional Spring framework.

Auto-Configuration

- Spring Boot: Depending on the dependencies (Spring Data JPA, Spring Security), we can automatically configure applications. Additionally, manual Java or XML configuration is not required.
- Spring Framework: Needs specific configuration, usually through XML files or configuration classes, for things like beans, databases, security, etc.

Embedded servers

- Spring Boot: Contains embedded servers such as Tomcat, Undertow, or Jetty. We don't need to set up an external server to run our app as a standalone JAR.
- Spring Framework: Needs WAR files to be deployed to a remote server.

Simplified dependency management

- Spring Boot: Uses a parent POM (pom.xml) to manage dependency versions automatically.
- Spring Framework: Developers must manually specify versions for each dependency.

Ease of testing

- Spring Boot: Simplifies testing with @SpringBootTest, which auto-configures the application context.
- Spring Framework: Testing requires manual setup of context and mocking.

Rapid development

- Spring Boot: Use Spring Initializr (https://start.spring.io) to generate a project skeleton in seconds.
- Spring Framework: Manually create project structure, configurations, and dependencies.

3. Briefly explain the enterprise approach for software development.

The enterprise approach to software development focuses on building scalable, secure, and maintainable software solutions tailored to an organization's needs. It prioritizes long-term maintainability, risk management, alignment with business objectives, and conformity to industry standards. This can include anything from enterprise resource planning (ERP) systems to customer relationship management (CRM) systems. The main goal of developing enterprise software solutions is to help businesses optimize their operations, improve communication and collaboration between employees, better serve their customers, and overall increase their productivity. Here are the key aspects of the enterprise approach for software development.

- Scalability - Systems are designed to handle growing workloads, transactions, and data efficiently.
- Modular architecture - Uses microservices, service-oriented architecture, or layered architecture for better maintainability and flexibility.
- Security - Implements security best practices like encryption, authentication, and authorization and adheres to regulations like GDPR, HIPAA, and PCI-DSS.
- Integration and interoperability - Ensures seamless communication between different applications (ERP, CRM, Databases) via APIs, or middleware.
- Standardized processes - Uses Agile methodologies (Scrum, Kanban) and DevOps for continuous integration and deployment (CI/CD).
- Governance - Formal oversight of budgeting, timelines, and regulatory adherence.
- Data management - Efficient database design, data warehousing, and analytics to support business intelligence and decision-making.
- Cloud and On-Premises deployment - Supports hybrid models using cloud computing (AWS, Azure, GCP) and on-premises infrastructure.
- User-centric design - Focuses on usability, accessibility, and responsive UI/UX for better user experience.

4. **"Microservices Architecture mostly suitable with DevOps", discuss your opinion with this statement.**

I agree with this statement. Microservices align perfectly with the DevOps principles of continuous integration, continuous delivery (CI/CD), and the collaborative approach between development and operations teams. Essentially, microservices enable faster development, independent deployment, and easier scaling, which are key aspects of a successful DevOps workflow.

How microservices and DevOps align perfectly,

- Independent development and deployment - Microservices break applications into smaller, independent services and DevOps enables CI/CD pipelines for frequent updates (automate testing, integration, and deployment) without disrupting the entire system.
- Scalability - Microservices allow independent scaling of services, while DevOps automates resource allocation using containerization like Docker and Kubernetes.
- Faster release cycles - DevOps promotes CI/CD, enabling frequent microservices updates without affecting the entire system.
- Infrastructure as Code (IaC) - DevOps automates microservices deployment using IaC tools like Terraform and Ansible, ensuring consistency across environments.
- Improved fault isolation and recovery - With microservices, failures in one service do not affect the whole application. DevOps uses monitoring tools and automated rollback mechanisms for quick recovery.

Microservices and DevOps work best together because DevOps automates deployment, monitoring, and scaling, while microservices improve agility and fault tolerance. According to the above details, we can say microservices architecture is mostly suitable with DevOps.

5. Spring Boot project

GitHub Repository links

- Project Department: https://github.com/gschamika/department.git
- Project Employee: https://github.com/gschamika/employee.git