ValueMomentum

Cloud Native Application Development - Java

Contents

[1 Problem Statement 2](#_Toc168557181)

[2 Solution Approach 2](#_Toc168557182)

[3 Resources 3](#_Toc168557183)

[4 Tech Stack Requirements 3](#_Toc168557184)

[5 Tentative Timelines 3](#_Toc168557185)

# Problem Statement

In standard project implementations, teams dedicate a significant amount of time to establishing the project base setup for spring boot applications.

This includes:

1. **Project Structure**: Determining the placement of various implementations within the project.
2. **Dependent libraries**: Effectively performing dependency injection.
3. **Infrastructure Services**: Implementing the base around infrastructure services.
4. **Startup Code Setup**:
   * Loading configurations from external configuration providers.
   * Loading logging-related configurations.
   * Loading commonly used dependencies/frameworks such as Spring Boot, JPA etc.

This overhead requires developers to devise an approach to address these concerns and ensure compliance throughout the project implementation timeline along with the feature/functionality implementation.

# Solution Approach

To address these concerns, we propose leveraging pre-built project templates. These templates come pre-equipped with:

1. **Standard Project Structure**: Aligning with CLEAN architecture principles.
2. **Comprehensive Startup Code**: Covering all startup code concerns.
3. **JPA Integration**: Leveraging Dapr for infrastructure-related implementations and base implementations.
4. **Logging Capabilities**: Utilizing Spring Boot inbuilt slf4j.
5. **12 Factor Application Methodology:** implement templates aligning to this methodology.

These templates cover the following use cases:

1. **Web API Template**
2. **System API Template**
3. **Domain API Template**
4. **Experience API Template**
5. **Batch Application:** leveraging Spring Batch
6. **Message driven Application:**  supporting Azure Service Bus (ASB), Azure EventHub, Kafka

To ensure that developers are fully aligned with these templates, a playbook will be provided. This will detail the standards and principles followed for project template implementation and how to align with them during development activities. Which also includes, a checklist that should be followed by reviewers for enforcing required governance.

These templates will be developed aligning to the Java coding standards and best practices. If there is a need, these templates can be tweaked to align with an enterprise level coding standard.

# Resources

|  |  |
| --- | --- |
| Skill Set | No Of Resources |
| Tech lead – Java 17+, Spring Boot 3.x, JPA/Hibernate, ECS/EKS/K8S | 1 |
| Developer – Java 17+, Spring Boot 3.x | 1 |

# Tech Stack Requirements

* Java
* Spring & Spring Boot
* JPA/Hibernate
* AWS/Azure App Configuration
* AWS Parameter store for secure parameters
* Docker
* Junit
* SLF4J
* Spring Cloud
* Spring Batch

# Tentative Timelines

|  |  |
| --- | --- |
| Activity | Duration (In Weeks) |
| Building Technical Competency & Planning | 2-4 |
| Understanding Domain API Template | 2 |
| Domain API Template – Documentation | 1 |
| Web API (System) Template - Design | 1 |
| Web API (System) Template – Development & Unit Testing | 3 |
| Web API (System) Template – Documentation & Training Material | 1 |
| Experience API Template - Design | 1 |
| Experience API Template – Development & Unit Testing | 3 |
| Experience API Template – Documentation & Training Materials | 1 |
| Batch App Template - Design | 1 |
| Batch App Template – Development & Unit Testing | 3 |
| Batch App Template – Documentation & Training Material | 1 |
| Message Driven App Template - Design | 1 |
| Message Driven App Template – Development & Unit Testing | 2 |
| Message Driven App Template – Documentation & Training Material | 1 |