Backend Development For Platform Backend Team At Tekion

Deep Gajera - 201801258
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India
201801258@daiict.ac.in

Off-Campus Mentor : Nitish K
Tekion India Pvt. Ltd., Bangalore, India

***Abstract*—This document describes my work as a part of the Platform Backend team during my internship at Tekion India Pvt. Ltd. I have worked on Approval service, Approval security service, and Restricted API service of the Platform Backend team. The Procedure Service is a part of the Tekion approval service. Approval Flow Management is a tool used by developers at Tekion for creating approval requests for various ActionTypes, deploying the services, and Managing the properties for different services based on the requirements. When a developer creates an approval request, all the required approvers need to approve the request, and after that, this request goes through the procedure service, in which the primary purpose of the request gets executed. The procedure service handles the approval request after they are approved. Procedure Service processes the task based on the approval request's action types. I have also introduced a new API for approval security service, sonarqube integration for restricted API service, routing change as a part of the backend platform version change in Approval service, and slack notification for the requester in Approval service.**

***Index Terms*—Java, SpringBoot [2], microservices, strategy pattern, testing, sonarqube**

I. Introduction

The procedure service is a part of the Approval flow management backend. The developers need approval from their managers, QA, and a production director to perform some critical and high-priority essential tasks. The purpose of the approval flow management is to automate creating approval requests, review and approve the requests, and execute the task once it gets approved. Every approval request has a unique action type that defines the request's purpose. Once the approval request is approved, the procedure service uses this action type to decide how to execute the request. For some action types, procedure service interacts with databases like MongoDB and elastic search, and for some action types, it uses different microservices of the organization to complete the task. For implementing the procedure service, I have used java and spring boot. After implementing the procedure service based on the given requirements, I have also worked on testing the procedure service. I have written unit tests for procedure service using the Mockito and Junit. I have also worked on integrating the sonarqube tool with the Tekion backend microservices. Sonarqube automatically analyzes the code and generates reports regarding the overall quality of code, issues, security, test coverage, etc. Tekion has multiple versions of their backend platform; I have also worked on migrating the routes of approval service from the old version to the new version. Security is the most crucial part of any organization. The product managers need to ensure that developers do not commit essential credentials of the database, certificates, and tokens to the project repository; To avoid this, microservice called approval security analyzes every developer commit and makes sure that there isn't any security issue. I have also worked on approval security microservice as part of my internship. In an organization, every developer needs their branch to be deployed in different environments like test, dev, and stage. I have worked on one task of implementing a notification service; this service sends a notification to the approval requester about the status of the service build for their branch on the test environment.

II. Training

As a part of internship training, I read two books related to the JAVA language [3], [4]. There were sessions on different technologies taken by company employees to help interns understand how the organization uses these technologies. I also read and understand about MongoDB database. There were sessions for backend interns in which we were briefed about the Tekion backend architecture and industry standards coding practices. After that, I had one task to create a game of cricket; this game needed functionalities like starting a cricket game between two teams with a toss, a live scorecard of the cricket game, and continuous interaction with the MongoDB database after every ball. I understood the technologies thoroughly by implementing this game using java, spring boot, Gradle, and MongoDB.

III. Tools and Technology Used

A. JAVA

Java is a high-level, class-based, object-oriented programming language designed to have as few implementation dependencies as possible. In Tekion, every backend microservice uses JAVA for implementation. Java is an open-source, free, fast, and powerful language.

B. *Spring Boot*

Spring Boot is an open-source Java-based framework used to create a micro Service. It is developed by the Pivotal Team and is used to build stand-alone and production-ready spring applications.

C. *MongoDB*

MongoDB is a document-based NoSQL database. MongoDB is very efficient in storing the schemaless data.

D. *Postman*

Postman is an application used for API testing. It is an HTTP client used to handle HTTP requests and responses.

E. *Bitbucket*

Bitbucket is software built by Atlassian. Many organization uses this software to maintain their repository. It is speedy and efficient and easy to use the software.

F. *Git*

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

IV. Work Involved

A. *Procedure Service Implementation*

1) Introduction: Procedure service is a part of the Approval Service. After the approval requests get approved, the procedure service helps decide and execute specific approval based on the action type. Here, Action Type is a constant value that defines the purpose of a particular approval request. For each action type, the procedure service performs a different task and marks its status based on the results. This service is in the JAVA language.

2) Why: Earlier, there were many if-else statements and many functions in the same file for performing different tasks based on the action type. This type of implementation made code less readable and did not follow industry-standard coding practices. This implementation was needed to replace by procedure service, which uses a strategy pattern [1] in its implementation, which makes code very fast and highly readable.

3) Implementation: The procedure service uses a standard design pattern called strategy pattern. In the strategy design pattern, strategy objects consist of different algorithms that need to be implemented in different contexts, and these objects can be interchangeable. I have used the plugin architecture of the java language to implement the strategy pattern in the procedure service. The plugin architecture has one interface called plugin interface, which takes one argument as a variable to decide the interchangeability of the strategy objects; in our case, this plugin interface has an action type as a generic type. Every strategy class is associated with one and only one action type. After that, I created one interface which extends this plugin interface; this is the main procedure interface that every strategy object class implements. This interface aims to have one function for processing different types of approval tasks. The strategy classes override this function and implement their process associated with the action type. The procedure service uses a plugin registry to fetch the strategy class based on the action type. So Instead of writing many if-else statements and functions for every if-else, the procedure service only use the plugin registry to get a particular class, and then it calls the primary interface method to execute the approval.

4) Testing: Testing is an essential part of the development process. Testing makes sure that the code written is working correctly without any flows. I have written unit tests for procedure service. I have used Mockito and JUnit for writing test cases for procedure service. I was able to cover 70% line coverage for the procedure service.

5) Learnings and Challenges: I was not aware of the design pattern initially, so I read about the strategy design pattern and how to use it in the JAVA language. I got to know about good coding practices in the organization. I also learned to write unit test cases in the JAVA language. I also merged my code with the main project code; before that, my mentor reviewed my code and pointed out necessary changes; Because of that, I got to know about standard coding practices in JAVA for null checks, warnings, solid principles, errors, and annotations.

B. *Properties service use in Approval Service*

1) Introduction: The properties service is a part of the tekion platform backend. Tekion provides automotive retail clouds to the customer. The car companies and the dealers of those companies are the customers to tekion, and the person who buys cars from these dealers are the consumers to tekion. These tenants and dealers use tekion software to maintain their data. The properties service interacts with the database to provide endpoint APIs for fetching the data about dealers and tenants and their properties.

2) Why: The Approval service was using a preference service for fetching data about dealers and tenants, but as a part of the new version of the backend, there was a requirement to use the properties service in the approval service. I needed to change the endpoints routes to use the properties service instead of the older preference service.

3) Implementation: There was severe use of preference service in the approval service, so I changed all the routes from older to the new ones and changed the service details from preference service to properties service. I followed the standard practice of defining the routes as constant in one file and then exporting this constant and using it whenever required to request properties service.

4) Testing: I used the approval flow management tool, which uses an approval service as a backend, to test the implementation. The approval tool makes an HTTP request to the backend for fetching properties of dealers and tenants. This way, I was able to test the routes of the properties service.

5) Learnings and Challenges: I learned how in the organization, one microservice interacts with another microservice, its requirements, and how to implement it. I learned how to use one microservice as a dependency of another microservice by defining its client package in the build.gradle file of the project.

C. API implementation to get security issues

1) Introduction: For any organization, security is a priority. In tekion, there is one microservice called Approval security service to ensure the security of all the microservices of the tekion. During the development, sometimes developers add the credentials and tokens of the database to their code, which is a security breach and could affect the organization in many ways. I had a task to implement an API that takes details for the security check and then returns severe security issues if found.

2) Why: As I mentioned earlier, security is a requirement. In the older version of the approval security service, it was only checking if there were any security issues in the particular branch of any repository; but now, there was a requirement to return the details of the security issue.

3) Implementation: The requirement was to take a list of branch names and the bitbucket URL of the repository in the request body, check for security issues for all these branches, and then return issues if found. I wrote the code to fetch the security issues of the repository and filter out the issues for the given branch name. After implementing this API in the approval security service, I needed to use this API endpoint in the approval service. The developers can not use an approval security service directly because it is a restricted service. So I implemented another API endpoint in the approval service, which uses approval security internally. For this API, I needed to take the list of the branch name and job names in the request body. Here job name is the name of the deployment for a particular microservice. Every deployment has a unique job name assigned to it. First of all, I fetched the bitbucket URL of the repository using the job name. after that, I used this bitbucket URL to get the security issues.

4) Testing: For testing this API, I used the postman tool. I made an HTTP post request multiple times to this API with a different request body. I also tested it for null checks, runtime errors, and database errors. During testing, I found null pointer exception flows in implementation that I fixed later.

5) Learnings and Challenges: I learned the importance of security in the organization, finding these security issues, and using the postman tool for testing.

D. Slack notifications for approval requester

1) Introduction: In the approval flow management, when a requester makes an approval request for service build action type after the request gets approved, procedure service uses Jenkins service build class for further process. This class automates the process of the Jenkins service build and waits for the status of the service build. Tekion uses the slack tool for internal communications. After getting a status for the service build, I needed to return this status as a slack notification to a particular requester.

2) Why: Frequently, developers forget about their service build requests and assume that their service build requests will be successful. However, sometimes it fails, and developers have no idea about it. So to mitigate this flow in the development process, there was a requirement to send a slack notification to every requester for their service build request.

3) Implementation: Requester slack id and approval bot slack id is needed to send slack notifications. I used the requester email id to get a slack id. I created an appropriate slack message template to be used for notification. After that, I used slack API to notify the requester about the service build status.

4) Testing: I first deployed the tekion approval service on the test environment. Later, I used the approval flow tool to create a service build request for another job name. After completing the service build, I got a notification from the slack approval bot of the service build status.

5) Learnings and Challenges: I learned how to integrate external tools like slack with organization microservice. I learned about different slack APIs and how to use them to send notifications.

CONCLUSION

This internship at tekion helps me understand how big organizations work, how the development process at influential organizations works, and the industry-standard coding practices. The tasks given to me during the internship helped me understand the tekion backend architecture very well. There were meetings with mentors and managers on a day-to-day basis, which helped me develop my time management skills. I learned the importance of deadlines, bug fixing, and testing in the organization.

Acknowledgment

This internship at Tekion helped me in many ways, so I would like to thank Tekion for giving me this opportunity. I want to thank my mentor, Mr. Nitish K, for his constant support and supervision throughout the internship. I would also like to thank Mr. Saksham Sethi, Mr. Sarthak Sharan, and Mr. Anirudh Saria for their constant guidance during this internship. I would also like to thank the Btp coordinator for allowing me to do an off-campus internship at Tekion.

References

[1] https://en.wikipedia.org/wiki/Strategy_pattern
[2] https://spring.io/projects/spring-boot
[3] head first java, 2nd edition Book by Bert Bates and Kathy Sierra
[4] effective java, 3rd edition Book by Joshua Bloch