Backend Development using Spring JPA with PostgreSQL in Kotlin

[1]Samrudhi Narayan Santaji, [2]Sowmyarani C N

[1]Student, [2]Associate Professor,
[1,2]Department of Computer Science & Engineering,
[1,2]RV College of Engineering, Bangalore, India.

Abstract: Every web application needs to save and update data that is accessible via HTTP. This data must be secured. The backend, on the other hand, might be more challenging to get started with since seeing the effects of your work is more challenging. It's more command-line-centric, and it's also more reliant on jargon: everyone can figure out what a paragraph tag does, but GET and POST requests, though essential to backend development and the web, aren't as simple to grasp. We'll go over what back-end development is all about, as well as back-end programming methodologies, in detail in this post. Java with decades of history in professional enterprise development is great choice for any application secure stack. Kotlin similar to Java is used for implementation. Spring within Java ecosystem makes building resources secure. In this paper, we will be developing a backend service with Spring Data JPA to perform CRUD operations with PostgreSQL using Kotlin.

IndexTerms - Spring Data JPA, PostgreSQL, Kotlin.

I. INTRODUCTION

In the area of Web Development, storing and updating data is important and also the data must be secured. Web application technologies are continuously evolving, and there are constant advancements and enhancements. Spring Boot, a Java framework, is used to construct web and corporate applications. It is easy to create and deploy standalone spring application with very little configuration of spring. For any web application, Spring MicroService Architecture can be implemented using Kotlin. Though NoSql is in recent use but RDBMS has numerous advantages, one of them is that the data is structured. In our paper we will be using PostgreSql. JPA (Java Persistence) to connect with databases. Spring JPA helps in the development of code. It decreases the time and effort required for development and maintenance. It allows you to design custom finder methods and interfaces for repositories, and spring framework handles the implementation for you. The next section demonstrates how to utilize PostgreSql with spring JPA. Spring Boot enables programmers to create applications fast utilising conventional RDBMS databases or embedded databases.

Section I contains Introduction of using Spring JPA with PostgreSql, Section II contain the related works, technologies used, Section III explains the methodology, Section IV explains the implementation, Section V summarizes the results and discusses data storage, while Section VI concludes the study with recommendations for further research.

II. LITERATURE REVIEW

The Spring Java Framework is a well-known and frequently used framework for developing web and corporate applications. Spring is a dependency injection container that allows you to configure beans using XML, annotations, and Java configuration. Over time, the Spring framework has grown exponentially by addressing the demands of NoSQL security support. It is a promising platform for developing corporate and online applications[2]. JPA (Java Persistence API) is a specialised technology for interacting with relational databases. It acts as a connector between SQL Tables and Java/Kotlin objects[10]. The persistence layer is necessity for real project in which you need to save and access data[11]. Kotlin like Java is a static typed language and also object oriented language, but unlike Java(until Java 8), it supports functional constructors. Kotlin can be used in functional programming style and in object oriented or mix of both styles[1]. The authors demonstrate how framework integration can increase system development efficiency and reduce coding workload. The paper's weakness was that it didn't describe how to improve system performance, user access speed, or the security of the system framework[6]. It presents the design and implementation of the Spring, Spring MVC, and MyBatis frameworks in the building of Web applications, which reduces the system's development process and burden, allowing for easier growth and deployment[13].

III. METHODOLOGY

The goal of this paper is to create and test a backend for a web application. To do this, we will employ the following methodologies: Spring Boot is a Java-based open source framework for developing microservices. Micro Services is a kind of architecture that enables developers to independently construct and deploy services. Each operating service has its own process, resulting in a lightweight business application support paradigm.

The Spring Web MVC framework which is built on Java, provides a Model-View-Controller (MVC) architecture and components that may be used to create flexible and loosely linked web applications. The MVC architecture isolates the application's input logic, business logic, and user interface logic while allowing for loose coupling between these pieces.

- The Model layer wraps the web application data, which will often be POJO.
- The View layer provides HTML output that the client's browser can comprehend and is responsible for presenting the model data.
- The Controller the third layer, is in charge of handling user requests. This layer creates a model that is then passed to the view layer for rendering.

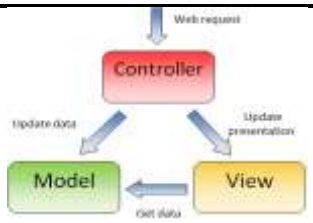

Fig 1: MVC Architecture

Spring Data JPA, which is part of the wider Spring Data family, makes JPA-based repositories simple to construct. This module focuses on improving the support for JPA-based data access layers. It makes it easier to create Spring-based apps that employ data access methods.

Developing a new data access layer for an application has been challenging for a long time. Far too much boilerplate code is required to perform simple searches, pagination, and audits. Spring Data JPA seeks to substantially simplify data access layer implementation by decreasing the amount of effort necessary to a bare minimum. As a developer, you create your repository interfaces, including custom locator methods, and Spring takes care of the implementation.

PostgreSQL, sometimes known as Postgres, is a relational database management system (RDBMS) that focuses on flexibility and SQL conformance.

The implementation language, Kotlin, is a statically typed programming language that supports the Java Virtual Machine (JVM), Android, JavaScript, and Native platforms. It's a free and open programming language. The Kotlin programming language is compiled to Java byte-code, which implies that Kotlin applications may run on the Java virtual machine and are completely compatible with Java. As with Java's complete compatibility, moving tiny sections of code into Kotlin makes it relatively simple to get started with the language. As a result, Kotlin becomes relatively simple to learn for developers.

The entire process is carried out in IntelliJ IDEA, a Java-based integrated development environment (IDE) for creating computer software. It comes with a pre-configured workspace and a plug-in framework for further customization

IV. IMPLEMENTATION

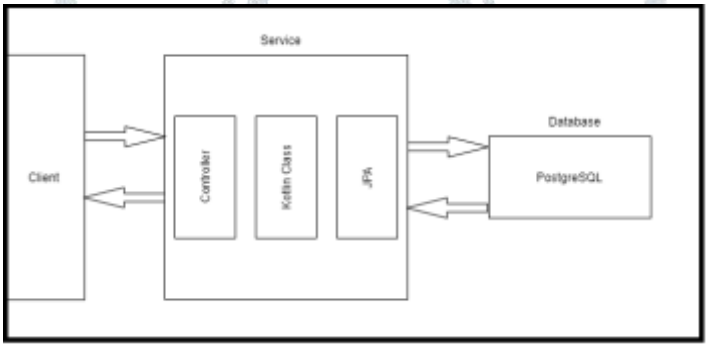

Fig 2: Backend service

In this paper we will focus on building a backend using spring framework and fully configuring the persistence layer which is JPA. For data storage we will be using relational database which is PostgreSQL. Whole implementation is done using Kotlin. This article provides a step-by-step guide on setting up the Spring context using Kotlin-based settings and the project's standard Maven pom. ReactJS may be used in the frontend. The steps to take are as follows:

To begin with , we need to create a project in start.spring.io and add the required dependency like JPA, PostgreSQL etc.

Next step is to Configure Spring JPA in application.properties of the project.

Under package model, we need to create DataModel class. In this class Annotation @Entity indicates that class is an Entity. For example if there is class patient and attributes of that class are first name and last name then Patient table with id, first name, and last name columns will be generated automatically.

Annotations used are:

- @Entity: This annotation specifies that this particular class is hibernate entity which will be mapped to a table in database.
- @TableName: This specifies the name of the table which postgres should map.
- @Id: This specifies that that this particular attribute is the primary key of the table.
- @GeneratedValue represents the primary key's value generating approach.
- @Column: This specifies the name of column that attribute of Java class is mapping to.

We need to design an interface for the package repository that allows us to do all CRUD tasks for the datamodel class. The interface extends CrudRepository and will be autowired in WebController to support repository and custom finder functions.

In Controller layer, we have specified the type of Webservice by using annotation such as @GetMapping, @PostMapping, @PutMapping and @DeleteMapping. We have also specified the end point of the web service. This layer takes the input as request object and send the request to next layer i.e., Service layer.

Save, findall, findbyid, and findbylastname are just a few of the request mapping methods available in the WebController. For this, we'll create a package controller that takes requests from clients, updates/gets data from a repository, and returns results. Create a PostgreSQL database to be mapped in ide.

We utilised certain autowired repository methods that are implemented interface CrudRepository in the web controller methods that are annotated by @RequestMapping. The last step is to run as kotlin project and check for results in the front end or in the backend testing.. . Here we have written logic to set the result data list to our response object which is in the form of JSON.
When we send a request for a resource by specifying the HTTP method, URI and request body, we get the response in the form of JSON along with HTTP status codes.
The four HTTP methods listed below are often utilized.

- GET method is used to access a resource in a read-only mode.
- POST method is used to create and add a new resource.
- DELETE method is used to delete a resource which is already existing.
- PUT method is used to update a resource which is already existing.

V. RESULTS AND DISCUSSION

The response from Postman is in JSON format.

Fig 3: save method

Fig 4: GET method

VI. CONCLUSION

The main conclusions of this paper is that it presents a way to implement backend services Spring JPA with PostgreSQL using Kotlin.We have used RDBMS for implementation but in future NoSQL databases can be used. NoSQL databases provides many advantages. NoSQL databases are generally more scalable and perform better than SQL databases.

REFERENCES

[1] Daniela Gotseva, Yavor Tomov, Petko Danov "*Comparative study Java vs Kotlin*", 27th National Conference with International Participation (TELECOM) 2019

[2] K. Siva Prasad Reddy "*Introduction to Spring Boot*" Publisher(s): Apress ,ISBN: 9781484229316 2017

[3] M. Stonebraker; L.A. Rowe; M. Hirohama "*The implementation of POSTGRES*", IEEE Transactions on Knowledge and Data Engineering 2018

[4] Zhang Lei and Wang Yue, "*Research on MVC Model Based on SpringBoot Microservice Architecture[J]*", Journal of Anhui Vocational College of Electronics and Information Technology, vol. 17, no. 04, pp. 1-9, 2018.

[5] Hatma Suryotrisongko, D. P. Jayanto and A. Tjahyanto, *"Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot*", Procedia Computer Science, vol. 124, pp. 736-743, 2017.

[6] Dandan Zhang, Zhiqiang Wei and Yongquan Yang, *"Research on lightweight MVC framework based on spring MVC and mybatis"*, 2013 Sixth International Symposium on Computational Intelligence and Design, vol. 1, 2013.

[7] HaiTao Wang, BaoXian Jia, *"Research based on web development of Spring Integration Framework",* IEEE.

[8] Zhou Tiecheng, Chen Zhongwen and Xiong Hui, *"The design and implementation of IT service system Based on Struts 2 + JPA + Spring 2 framework"*, Science Technology and Engineering, vol. 9, no. 1297, pp. 1290-1293, 2009.

[9] B. Frey, J. Doddridge and C. Seaman, *"Chasing the AHA! moment: Exploring initial learnability of programming languages"*, Proc. IEEE Symp. Vis. Lang. Human-Centric Comput. VL/HCC, vol. 2017-Octob, pp. 329-330, 2017.

[10] S. Samuel and S. Bocutiu, *"Programming Kotlin"* in Packt Publishing, Birmingham, 2017, ISBN 978-1-78712-636-7.

[11] Michael StonebrakerLawrence A. RoweMichael Hirohama *"The implementation of POSTGRES*" IEEE Transactions on Knowledge and Data Engineering (Volume: 2, Issue: 1, Mar 1990)

[12] Peter Späth *"Adding a Database with JPA*", Apress ISBN: 9781484250792 2019

[13] Daniel Andres Pelaez Lopez *"Modeling Your Entities and Data with JPA"* Berkeley, CA : Apress : Imprint: Apress, 2021.

[14] Yuxiang Hou, *"Design and Implementation of the Framework for Spring+SpringMVC+MyBatis in the Development of Web Applications"*, International Conference of Information Technology, IEEE 2011.

[15] Raj Malhotra *"Common Use Cases with JPA"* Publisher: Apress, Published in: Rapid Java Persistence and Microservices2019

[16] Paul Tepper FisherSoloman Duskis "*Integrating JPA with Spring*" Publisher:Apress ISBN 13:978-1-4302-1878-4 2009.