

Technical Task for Junior Backend Developer

Title: Developing a comprehensive Pizza Delivery System

By: Ritesh

Email: riteshkashyap9999@gmail.com

Index

Abstract

1. Project Overview

1.1.Introduction to Papa Gino's Pizza

1.1.Problem Statement

1.2.Technologies Used

2. Analysis

2.1.System Requirements

2.2.Servlet Requirements

3. Spring Boot Framework

3.1.What is Spring Boot

3.2.Why Use Spring Boot Framework

3.3.Spring Boot Features

3.4.Advantages of Spring Boot

3.5.Limitations of Spring Boot

3.6.Goals of Spring Boot

4. Hibernate Framework

4.1.Introduction

4.2.Hibernate Architecture

4.3.Hibernation Configuration

5. Spring Modules

6. API

6.1.APIs & Database Schemas

6.2.Database Schema

7. Implementation

8. Conclusion

Abstract

This project presents a comprehensive solution for an online website by seamlessly integrating the Spring Boot backend framework and React frontend technology. The key focus is on developing a RESTful API service that incorporates user management, including login and signup functionalities, and ensures that pizza orders are exclusively placed by authenticated users. The Spring Boot backend efficiently handles critical aspects such as user authentication, order processing, and communication with a relational database for storing essential pizza and user information.

On the frontend, React is employed to create a responsive and intuitive user interface. This empowers customers to seamlessly navigate the menu, customize their orders, and complete transactions with ease. The collaborative use of Spring Boot and React aims to deliver a scalable, maintainable, and user-friendly solution for the online pizza ordering system. The RESTful APIs facilitate efficient communication between the frontend and backend, ensuring a smooth and reliable user experience.

The developed system not only meets the technical requirements of a dynamic web application but also prioritizes security and user experience. By emphasizing scalability and maintainability, the project offers a robust foundation for an efficient, secure, and enjoyable online pizza ordering experience, aligning with contemporary standards in web development.

1. Project Overview

1.1. Introduction to Papa Gino's Pizza

This project involves creating a dynamic web application for pizza ordering using the Spring Boot framework for the backend and React for the frontend. The Spring Boot backend will handle user authentication, order processing, and communication with a relational database to store pizza and user information. React will be employed to develop a responsive and intuitive user interface, allowing customers to browse the menu, customize orders, and complete transactions seamlessly. The application will leverage RESTful APIs for efficient communication between the frontend and backend. The use of Spring Boot and React aims to provide a scalable, maintainable, and user-friendly solution for an online pizza ordering system.

1.2. Problem Statement

Develop a RESTful API service for a pizza delivery system, incorporating user management (login and signup) and pizza ordering functionalities. Ensure that placing orders is restricted to logged-in users.

1.3. Technologies Used

- Presentation Layer - ReactJS
- Business Logic- Spring Boot,
- Data Access Layer- Hibernate, MySQL
- Database- MySQL

2. Analysis

2.1. System Requirements

Spring Boot 3.0.2 requires Java 17 and is compatible up to and including Java 19. Spring Framework 6.0.4 or above is also required.

Explicit build support is provided for the following build tools:

Build Tool	Version
Maven	3.5+
Gradle	7.5 or later

2.2. Servlet Requirements

Spring Boot supports the following embedded servlet containers:

Name	Servlet Version
Tomcat 10.0	5.0
Jetty 11.0	5.1
Undertow 2.2 (Jakarta EE 9 variant)	5.0

You can also deploy Spring Boot applications to any servlet 5.0+ compatible container.

3. Spring Boot Framework

3.1. What is Spring Boot

Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications. It is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring Framework. It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.

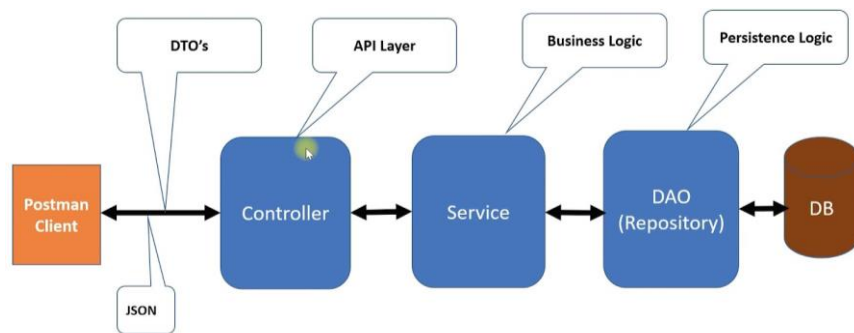


Fig. 3.1: Spring Boot Application Architecture

In short, Spring Boot is the combination of Spring Framework and Embedded Servers. In Spring Boot, there is no requirement for XML configuration (deployment descriptor). It uses convention over configuration software design paradigm that means it decreases the effort of the developer. We can use Spring STS IDE or Spring Initializer to develop Spring Boot Java applications.

3.2. Why Use Spring Boot Framework

We should use Spring Boot Framework because:

- The dependency injection approach is used in Spring Boot.
- It contains powerful database transaction management capabilities.

- It simplifies integration with other Java frameworks like JPA/Hibernate ORM, Struts, etc.
- It reduces the cost and development time of the application

Along with the Spring Boot Framework, many other Spring sister projects help to build applications addressing modern business needs. There are the following Spring sister projects area follows:

Spring Data: It simplifies data access from the relational and NoSQL databases.

Spring Batch: It provides powerful batch processing.

Spring Security: It is a security framework that provides robust security to applications.

Spring Social: It supports integration with social networking like LinkedIn.

Spring Integration: It is an implementation of Enterprise Integration Patterns. It facilitate so integration with other enterprise applications using lightweight messaging and declarative adapters.

3.3. Spring Boot Features

- Web Development
- Spring Application
- Application events and listeners
- Admin features
- Externalized Configuration
- Properties Files
- YAML Support
- Type-safe Configuration
- Logging
- Security

3.4. Advantages of Spring Boot

- It creates stand-alone Spring applications that can be started using Java-jar.

- It tests web applications easily with the help of different Embedded HTTP servers such as Tomcat, Jetty, etc. We don't need to deploy WAR files.
- It provides opinionated 'starter' POMs to simplify our Maven configuration.
- It provides production-ready features such as metrics, health checks, and externalize do configuration. There is no requirement for XML configuration.
- It offers a CLI tool for developing and testing the Spring Boot application. It offers the number of plug-ins.
- It also minimizes writing multiple boiler plate codes(the code that has to be included in many places with little or no alteration), XML configuration, and annotations.
- It increases productivity and reduces development time.

3.5. Limitations of Spring Boot

Spring Boot can use dependencies that are not going to be used in the application. These dependencies increase the size of the application

3.6. Goals of Spring Boot

- The main goal of Spring Boot is to reduce development, unit test, and integration test time. Provides Opinionated Development approach.
- Avoids defining more Annotation Configuration
- Avoids writing lots of import statements
- Avoids XML Configuration.
- By providing or avoiding the above points, Spring Boot Framework reduces Development time, Developer Effort, and increases productivity.

4. Hibernate Framework

4.1.Introduction

Hibernate is an Object-relational mapping (ORM) tool. Object-relational mapping or ORM is a programming method for mapping the objects to the relational model where entities/classes are mapped to tables, instances are mapped to rows and attributes of instances are mapped to columns of table. A “virtual object database” is created that can be used from within the programming language.

4.2.Hibernate Architecture

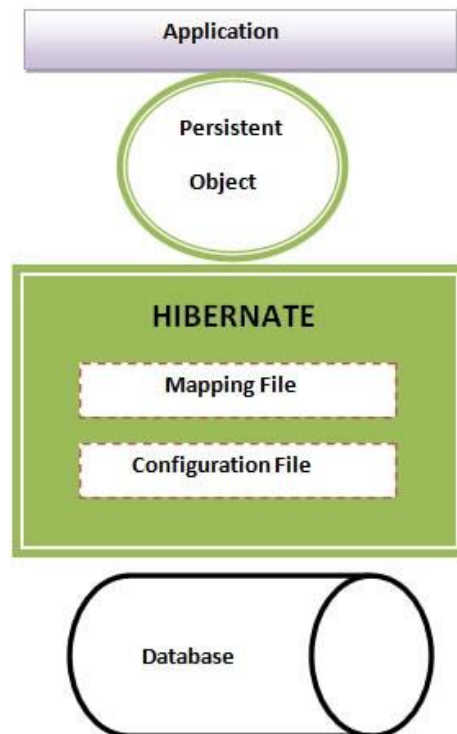


Fig: 4.1: Architecture of Hibernate

4.3. Hibernate Configuration

Hibernate configuration is managed by an instance of `org.hibernate.cfg.Configuration`. An instance of `org.hibernate.cfg.Configuration` represents an entire set of mappings of an application's Java types to an SQL database. The `org.hibernate.cfg.Configuration` is used to build an immutable `org.hibernate.SessionFactory`. The mappings are compiled from various XML mapping files or from Java 5 Annotations.

Hibernate provides following types of configurations

- `hibernate.cfg.xml` – A standard XML file which contains hibernate configuration and which resides in root of application's CLASSPATH
- `hibernate.properties` – A Java compliant property file which holds key value pair for different hibernate configuration strings.
- Programmatic configuration – This is the manual approach. The configuration can be defined in Java class.

Hibernate creates persistent objects which synchronize data between application and database.

5. Spring Modules

Spring framework provides many modules that can be used in development of enterprise grade applications. Here is the list of Spring Modules:

- Inversion of Control container
- Aspect-oriented programming
- Data access (DAO)
- Transaction management
- Model-view-controller (Spring MVC)
- Remote Access framework
- Convention-over-configuration (Spring ROO module)
- Batch processing (Spring Scheduler)
- Authentication and authorization (Spring Security)
- Remote Management (Spring JMX)
- Messaging (Spring messaging)
- Testing - It provides support classes for writing and executing the test cases

6. APIs & Database Schemas

6.1.APIs

a) User Management APIs:

- POST /signUp: Register a new user.
- POST /login: Authenticate a user and return a token.

b) Pizza Management APIs:

- GET /pizzas: List available pizza types and their prices.

c) Order Management APIs:

- POST /orders: Place a new pizza order (only accessible to logged-in users).
- GET /getOrders: Retrieve a list of all orders (user-specific).
- GET /orders/{id}: Retrieve details of a specific order.
- PUT /updateOrder/{id}: Update an existing order (e.g., change address, pizza type).
- DELETE /deleteByOrder/{id}: Cancel an order.

6.2.Database Schema

```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
address	varchar(100)	NO		NULL	
email	varchar(40)	NO	UNI	NULL	
name	varchar(20)	NO		NULL	
password	varchar(20)	NO		NULL	

```
mysql> desc order_table;
```

Field	Type	Null	Key	Default	Extra
orderid	int	NO	PRI	NULL	auto_increment
address	varchar(255)	YES		NULL	
pizzatype	varchar(255)	YES		NULL	
qunatity	int	NO		NULL	
status	varchar(255)	YES		NULL	
totalprice	double	NO		NULL	
user_id	int	YES	MUL	NULL	

```
mysql> desc pizza;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
description	varchar(100)	NO		NULL	
pizzatype	varchar(50)	NO	UNI	NULL	
price	double	NO		NULL	

7. Implementation

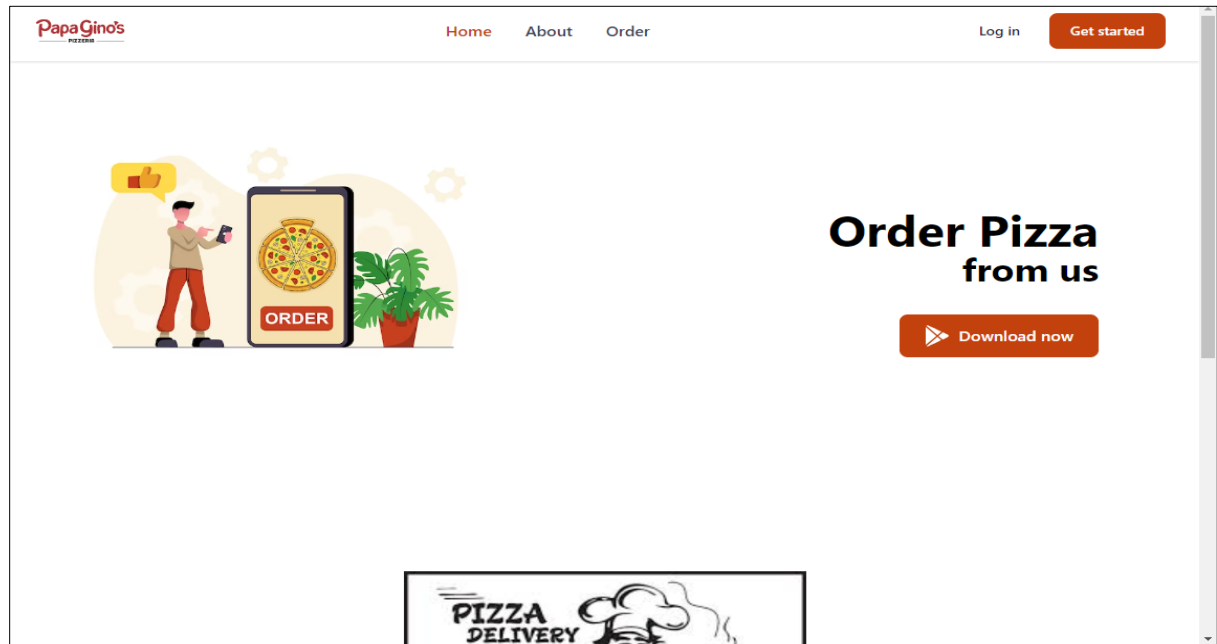


Fig. 7.1: Home Page

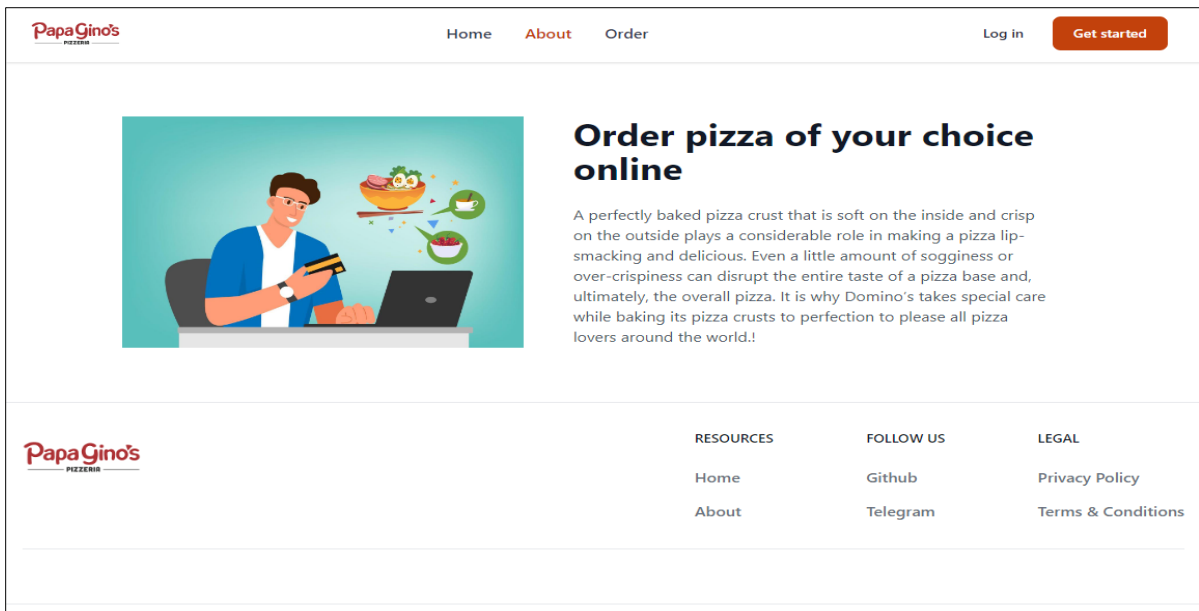
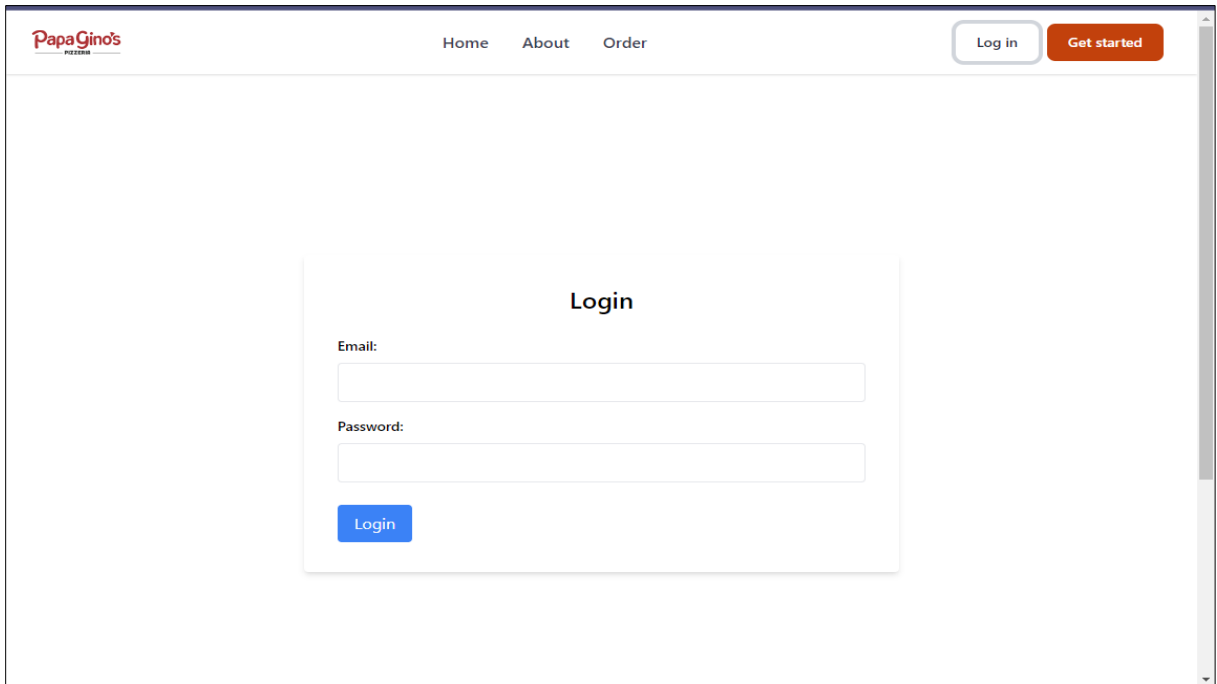


Fig. 7.2: About Page



The screenshot shows the PapaGino's website with a navigation bar at the top containing the logo, 'Home', 'About', 'Order', 'Log in', and 'Get started' buttons. The main content area features a 'Login' form with fields for 'Email:' and 'Password:', and a blue 'Login' button.

PapaGino's
PIZZERIA

Home About Order Log in Get started

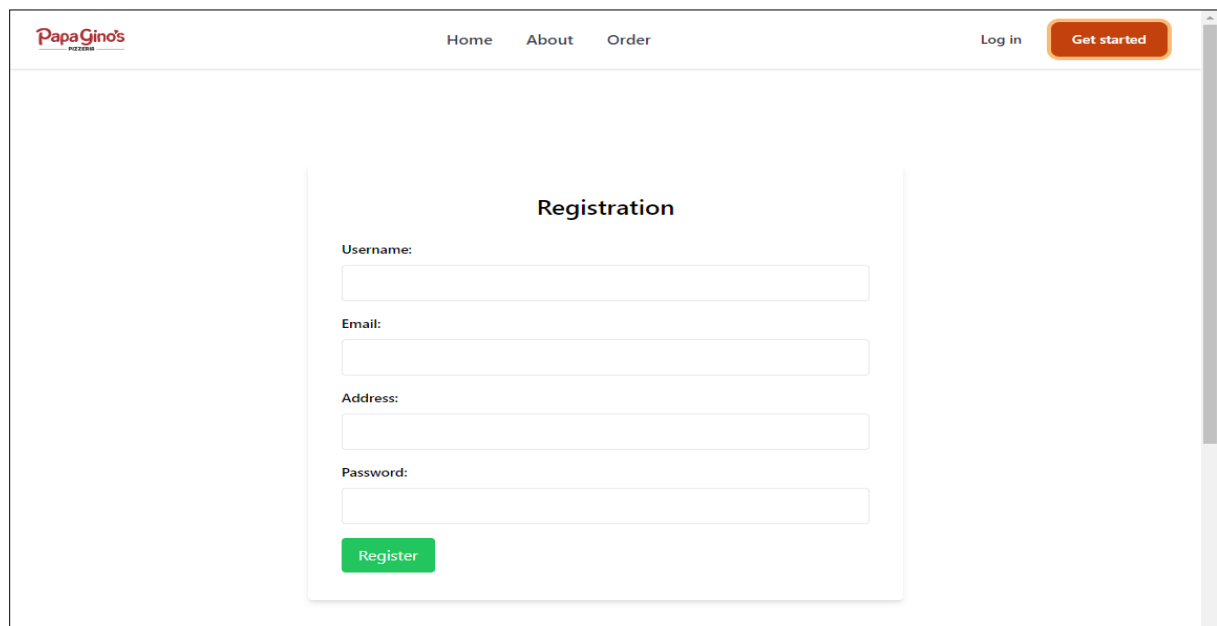
Login

Email:

Password:

Login

Fig. 7.3: Login page



The screenshot shows the PapaGino's website with a navigation bar at the top containing the logo, 'Home', 'About', 'Order', 'Log in', and 'Get started' buttons. The main content area features a 'Registration' form with fields for 'Username:', 'Email:', 'Address:', and 'Password:', and a green 'Register' button.

PapaGino's
PIZZERIA

Home About Order Log in Get started

Registration

Username:


Email:

Address:

Password:

Register

Fig. 7.4: Signup page



[Home](#) [About](#) [Order](#)

[Log in](#) [Get started](#)

Order ID:

Pizza Type:

Small

Quantity:

Address:

Total Price:
₹0

Submit

Fig. 7.5: Order page

8. Conclusion

In conclusion, the developed project seamlessly integrates Spring Boot and React to deliver a robust and user-friendly solution. The RESTful API service ensures secure user management with login and signup functionalities, restricting order placement to authenticated users. Leveraging the Spring Boot framework, the backend efficiently handles user authentication, order processing, and database communication for storing crucial information. Meanwhile, React empowers the frontend with a responsive and intuitive interface, enabling customers to effortlessly browse the menu, customize orders, and complete transactions. This scalable and maintainable application provides a comprehensive online pizza ordering experience, emphasizing efficiency, security, and a delightful user journey.