i2i Systems
innovation to integration

i2i Academy

Training Document

Topic	SWAGGER
Document Name	SWAGGER

Document Difficulty Level			
Beginner	**Junior**	**Senior**	**Expert**
☐	■	☐	☐

Document History

Date	Author	Ver	Comments
05.03.2025	Murat Kağan Temel	1.0	Initial Draft

SWAGGER

Exercise SWAGGER:

Definiton: Create a layered Spring Boot RESTful application that provides basic CRUD operations for managing Customer data. The application should include OpenAPI-compliant API documentation using the `springdoc-openapi-starter-webmvc-ui` dependency. This dependency also enables Swagger UI, so no additional configuration is required.

You may choose to simulate database operations using in-memory data structures such as Map or List. Using a real database is not mandatory.

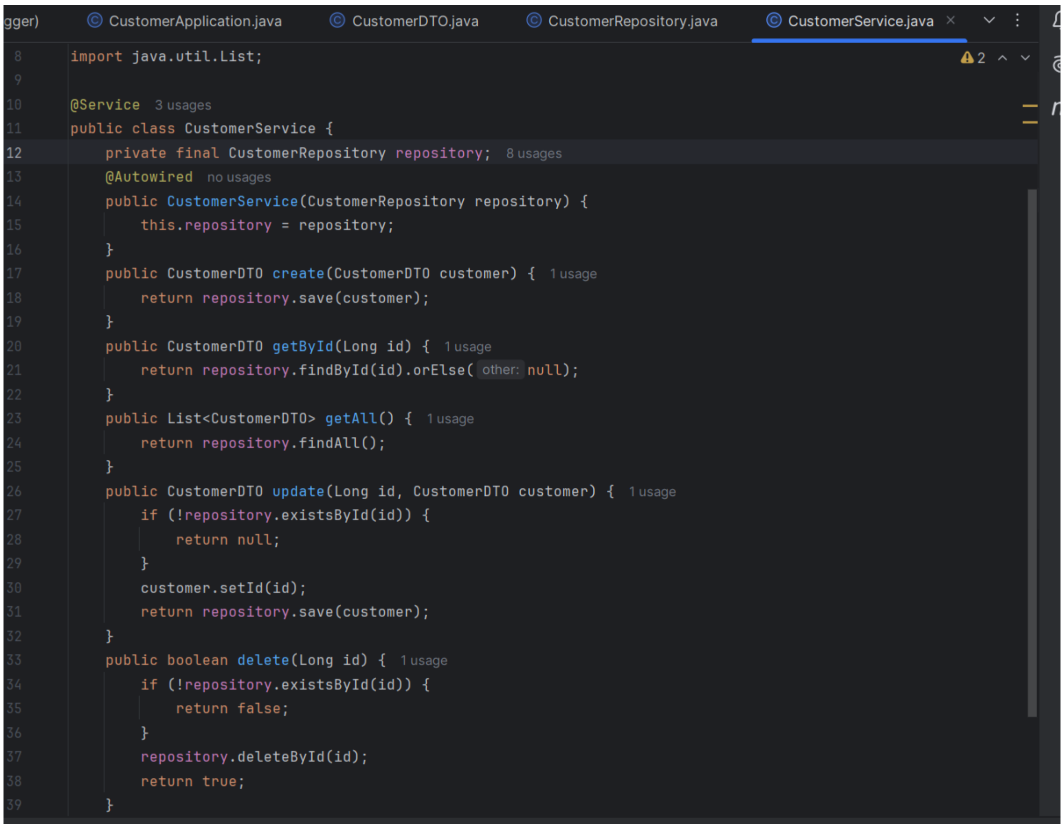
The application should expose endpoints for the following operations:

- **Create a new customer**
- **Retrieve a customer by ID**
- **Retrieve all customers**
- **Update an existing customer**
- **Delete a customer**

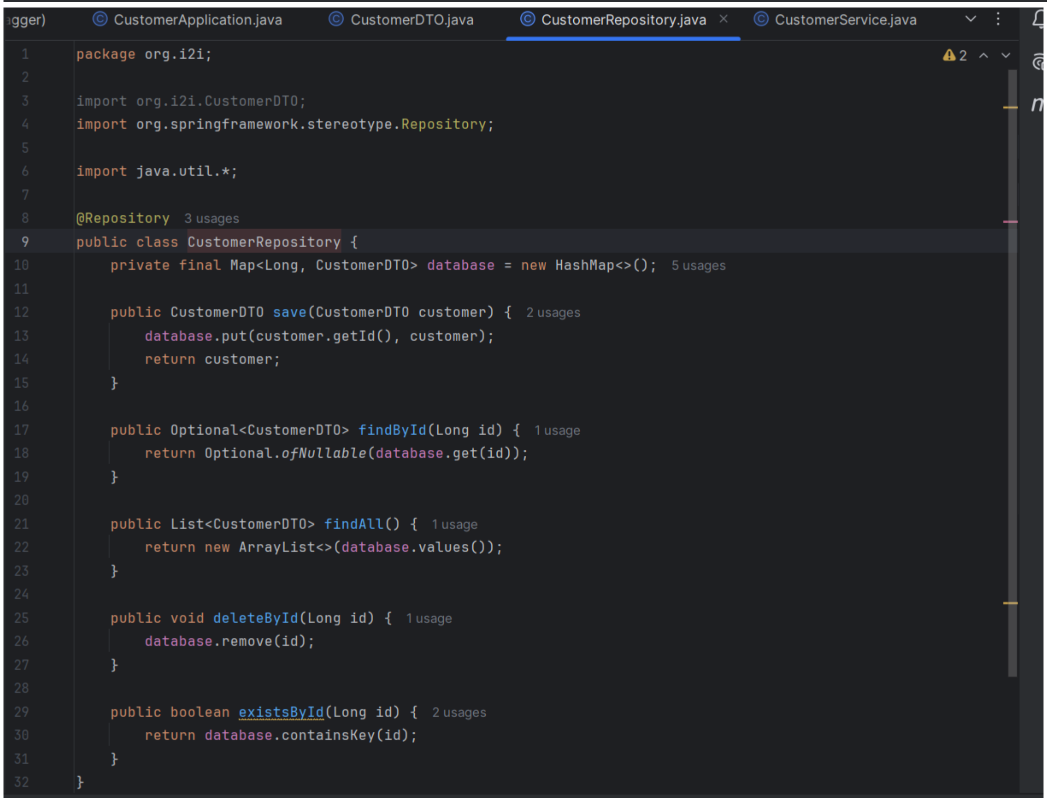
After completing the implementation, open the Swagger UI in your browser and share screenshots of the generated API documentation and the available endpoints. A sample CustomerDTO (DTO refers to Data Transfer Object) is shared in the next page.

```
public class CustomerDTO {

    @Schema(description = "Unique ID of the customer", example =
"1")

    private Long id;



    @NotBlank

    @Schema(description = "Full name of the customer", example =
"John Doe")

    private String name;



    @Email

    @Schema(description = "Email address", example =
"john.doe@example.com")

    private String email;

}
```

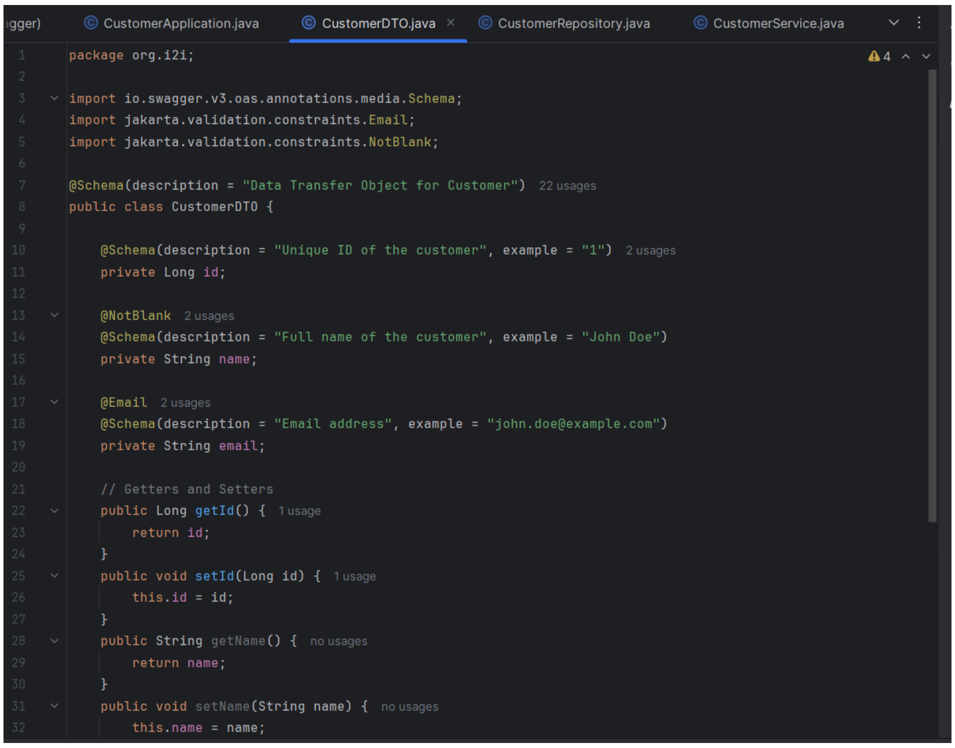
SWAGGER:



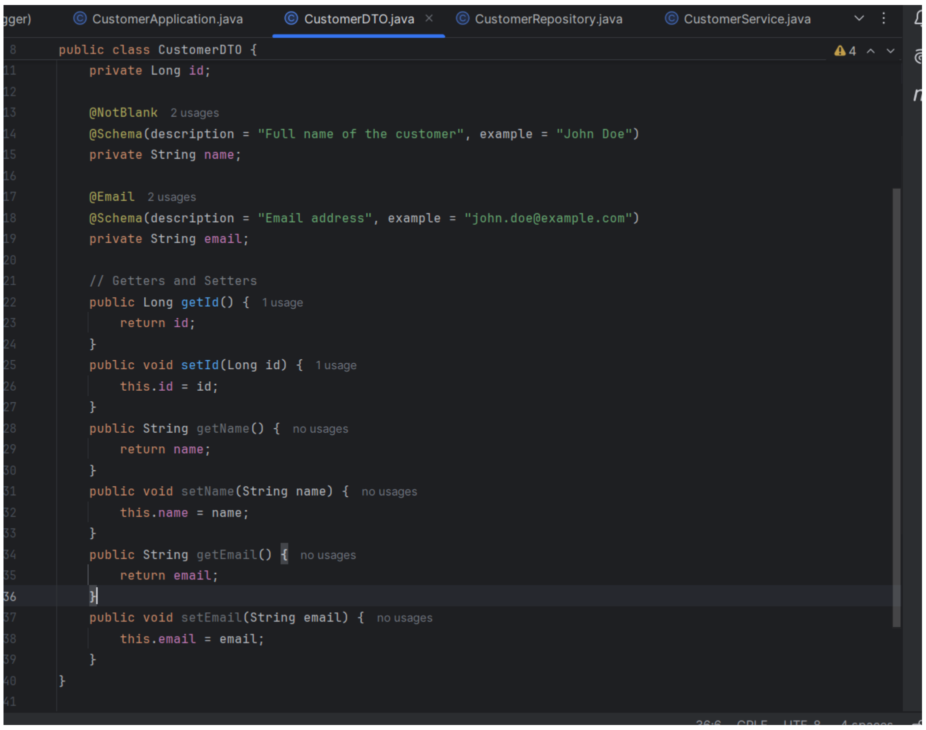
```java
import java.util.List;

@Service  3 usages
public class CustomerService {
    private final CustomerRepository repository;  8 usages
    @Autowired  no usages
    public CustomerService(CustomerRepository repository) {
        this.repository = repository;
    }
    public CustomerDTO create(CustomerDTO customer) {  1 usage
        return repository.save(customer);
    }
    public CustomerDTO getById(Long id) {  1 usage
        return repository.findById(id).orElse( other: null);
    }
    public List<CustomerDTO> getAll() {  1 usage
        return repository.findAll();
    }
    public CustomerDTO update(Long id, CustomerDTO customer) {  1 usage
        if (!repository.existsById(id)) {
            return null;
        }
        customer.setId(id);
        return repository.save(customer);
    }
    public boolean delete(Long id) {  1 usage
        if (!repository.existsById(id)) {
            return false;
        }
        repository.deleteById(id);
        return true;
    }
```



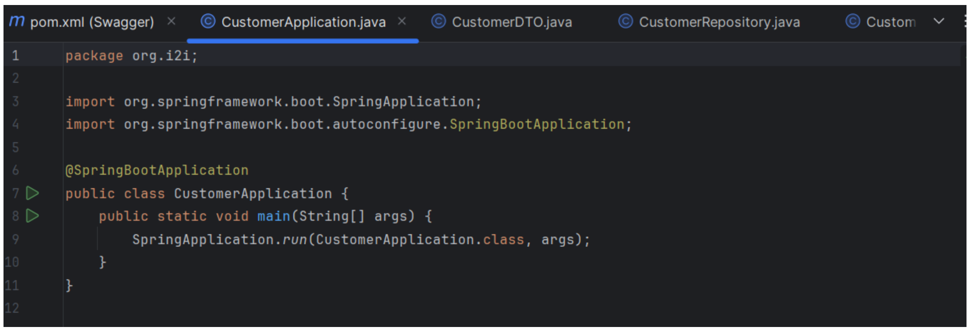
```java
package org.i2i;

import org.i2i.CustomerDTO;
import org.springframework.stereotype.Repository;

import java.util.*;

@Repository  3 usages
public class CustomerRepository {
    private final Map<Long, CustomerDTO> database = new HashMap<>();  5 usages

    public CustomerDTO save(CustomerDTO customer) {  2 usages
        database.put(customer.getId(), customer);
        return customer;
    }

    public Optional<CustomerDTO> findById(Long id) {  1 usage
        return Optional.ofNullable(database.get(id));
    }

    public List<CustomerDTO> findAll() {  1 usage
        return new ArrayList<>(database.values());
    }

    public void deleteById(Long id) {  1 usage
        database.remove(id);
    }

    public boolean existsById(Long id) {  2 usages
        return database.containsKey(id);
    }
}
```



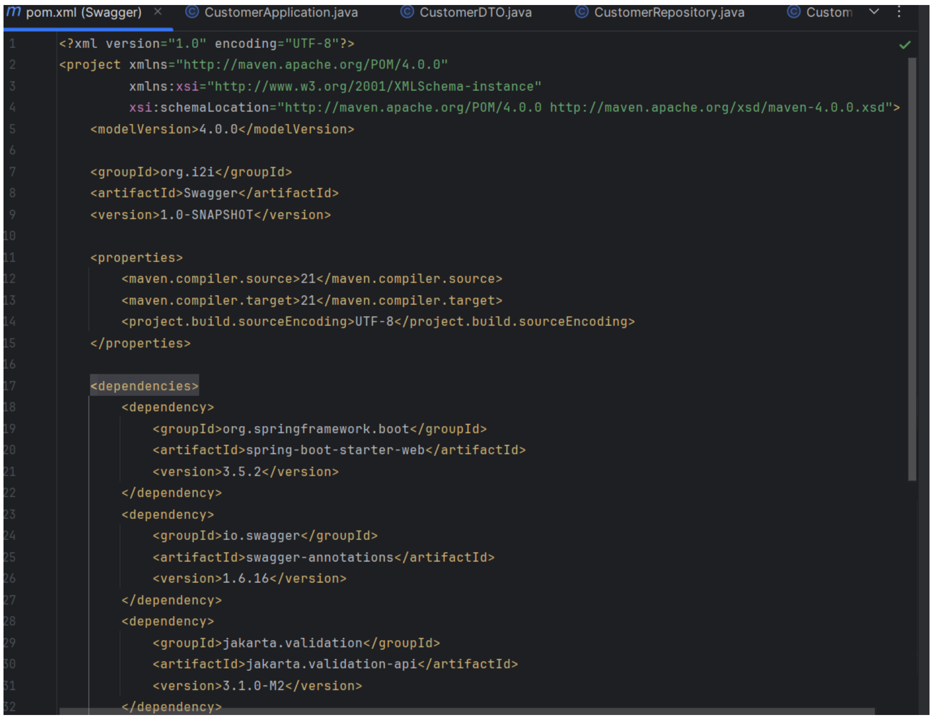
```java
package org.i2i;

import io.swagger.v3.oas.annotations.media.Schema;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;

@Schema(description = "Data Transfer Object for Customer")  22 usages
public class CustomerDTO {

    @Schema(description = "Unique ID of the customer", example = "1")  2 usages
    private Long id;

    @NotBlank  2 usages
    @Schema(description = "Full name of the customer", example = "John Doe")
    private String name;

    @Email  2 usages
    @Schema(description = "Email address", example = "john.doe@example.com")
    private String email;

    // Getters and Setters
    public Long getId() {  1 usage
        return id;
    }
    public void setId(Long id) {  1 usage
        this.id = id;
    }
    public String getName() {  no usages
        return name;
    }
    public void setName(String name) {  no usages
        this.name = name;
```



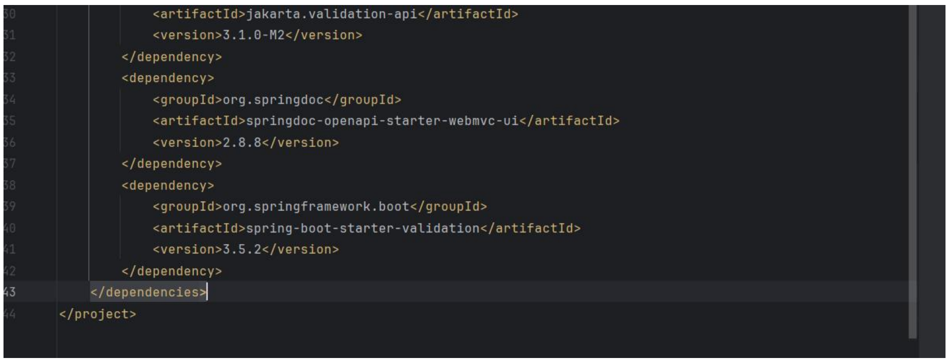
```java
public class CustomerDTO {
    private Long id;

    @NotBlank  2 usages
    @Schema(description = "Full name of the customer", example = "John Doe")
    private String name;

    @Email  2 usages
    @Schema(description = "Email address", example = "john.doe@example.com")
    private String email;

    // Getters and Setters
    public Long getId() {  1 usage
        return id;
    }
    public void setId(Long id) {  1 usage
        this.id = id;
    }
    public String getName() {  no usages
        return name;
    }
    public void setName(String name) {  no usages
        this.name = name;
    }
    public String getEmail() {  no usages
        return email;
    }
    public void setEmail(String email) {  no usages
        this.email = email;
    }
}
```



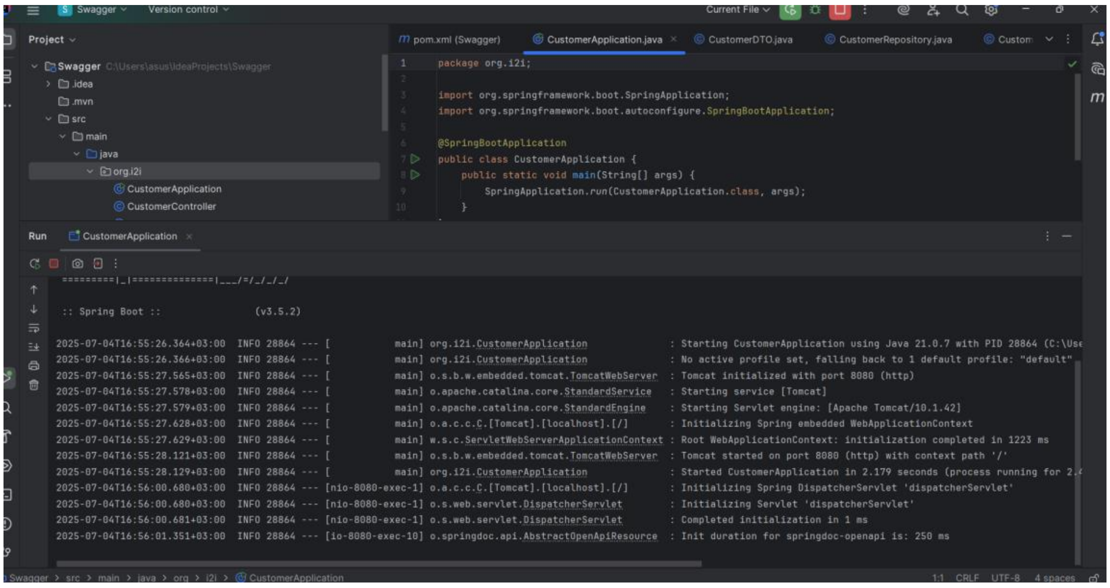
```java
package org.i2i;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CustomerApplication {
    public static void main(String[] args) {
        SpringApplication.run(CustomerApplication.class, args);
    }
}
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.i2i</groupId>
    <artifactId>Swagger</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <version>3.5.2</version>
        </dependency>
        <dependency>
            <groupId>io.swagger</groupId>
            <artifactId>swagger-annotations</artifactId>
            <version>1.6.16</version>
        </dependency>
        <dependency>
            <groupId>jakarta.validation</groupId>
            <artifactId>jakarta.validation-api</artifactId>
            <version>3.1.0-M2</version>
        </dependency>
```

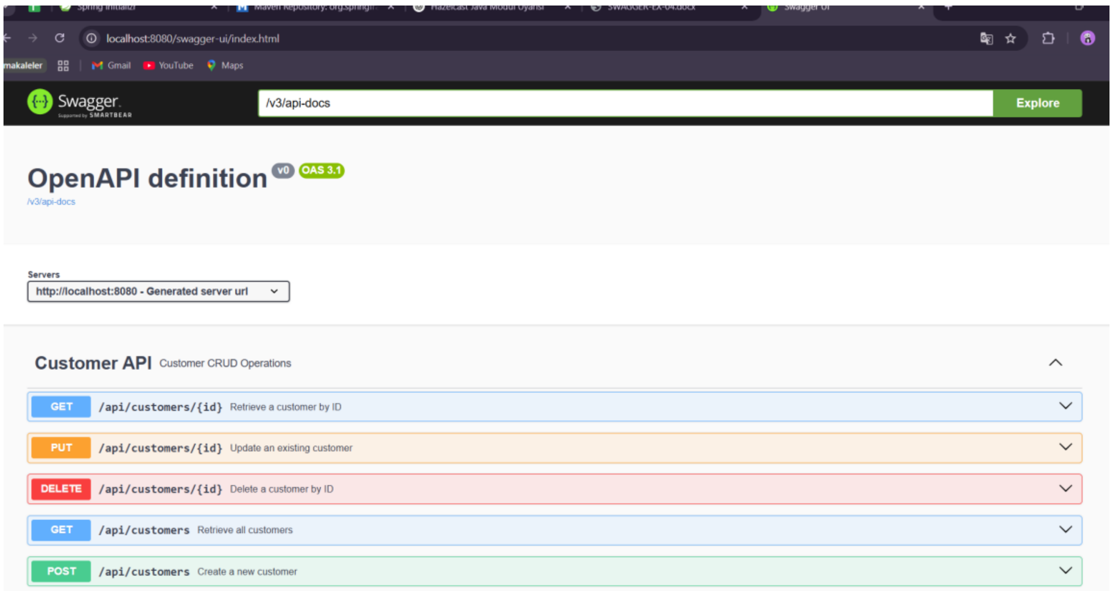


```xml
            <artifactId>jakarta.validation-api</artifactId>
            <version>3.1.0-M2</version>
        </dependency>
        <dependency>
            <groupId>org.springdoc</groupId>
            <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
            <version>2.8.8</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
            <version>3.5.2</version>
        </dependency>
    </dependencies>
</project>
```



```java
package org.i2i;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CustomerApplication {
    public static void main(String[] args) {
        SpringApplication.run(CustomerApplication.class, args);
    }
```

```
=========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.2)

2025-07-04T16:55:26.364+03:00  INFO 28864 --- [           main] org.i2i.CustomerApplication              : Starting CustomerApplication using Java 21.0.7 with PID 28864 (C:\Use
2025-07-04T16:55:26.366+03:00  INFO 28864 --- [           main] org.i2i.CustomerApplication              : No active profile set, falling back to 1 default profile: "default"
2025-07-04T16:55:27.565+03:00  INFO 28864 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-07-04T16:55:27.578+03:00  INFO 28864 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-04T16:55:27.579+03:00  INFO 28864 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-04T16:55:27.628+03:00  INFO 28864 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-07-04T16:55:27.629+03:00  INFO 28864 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1223 ms
2025-07-04T16:55:28.121+03:00  INFO 28864 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2025-07-04T16:55:28.129+03:00  INFO 28864 --- [           main] org.i2i.CustomerApplication              : Started CustomerApplication in 2.179 seconds (process running for 2.
2025-07-04T16:56:00.680+03:00  INFO 28864 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-07-04T16:56:00.680+03:00  INFO 28864 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-04T16:56:00.681+03:00  INFO 28864 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2025-07-04T16:56:01.351+03:00  INFO 28864 --- [io-8080-exec-10] o.springdoc.api.AbstractOpenApiResource  : Init duration for springdoc-openapi is: 250 ms
```



OpenAPI definition v0 OAS 3.1

/v3/api-docs

Servers

http://localhost:8080 - Generated server url

Customer API Customer CRUD Operations

GET `/api/customers/{id}` Retrieve a customer by ID

PUT `/api/customers/{id}` Update an existing customer

DELETE `/api/customers/{id}` Delete a customer by ID

GET `/api/customers` Retrieve all customers

POST `/api/customers` Create a new customer