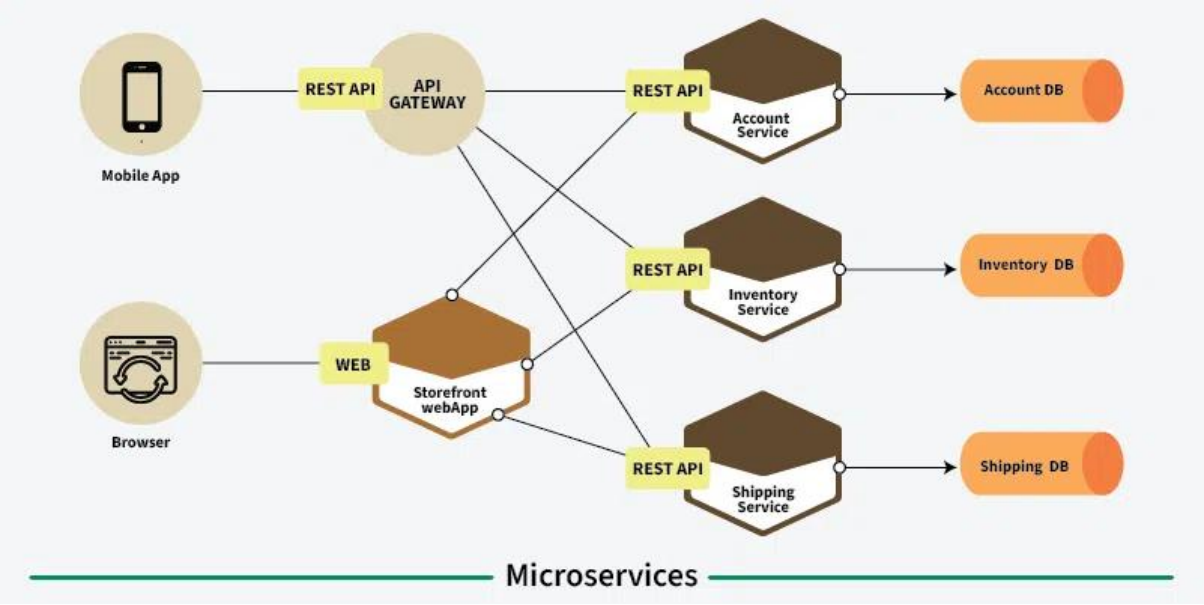
Microservices

What are Microservices?

- Microservices build applications as small, independent services.
- Each service performs a specific business function.
- Services communicate over a network and remain loosely coupled.
- Each microservice can be developed, deployed, and scaled independently.

How do Microservices work?

- Each microservice handles a specific business feature.
- Services communicate through APIs.
- Different technologies can be used per service.
- Independent updates improve reliability.



Main components of Microservices Architecture :-

Main components of microservices architecture include:

1. **Microservices**: Small, independent services for specific business functions.
2. **API Gateway**: Single entry point that routes and secures requests.
3. **Service Discovery**: Helps services find and communicate with each other.
4. **Load Balancer**: Distributes traffic across service instances.
5. **Containerization & Orchestration**: Docker and Kubernetes manage deployment and scaling.

6. **Message Broker**: Enables asynchronous communication between services.
7. **Database per Service**: Each service manages its own data.
8. **Caching**: Improves performance by reducing repeated data access.
9. **Fault Tolerance**: Ensures system reliability during failures.

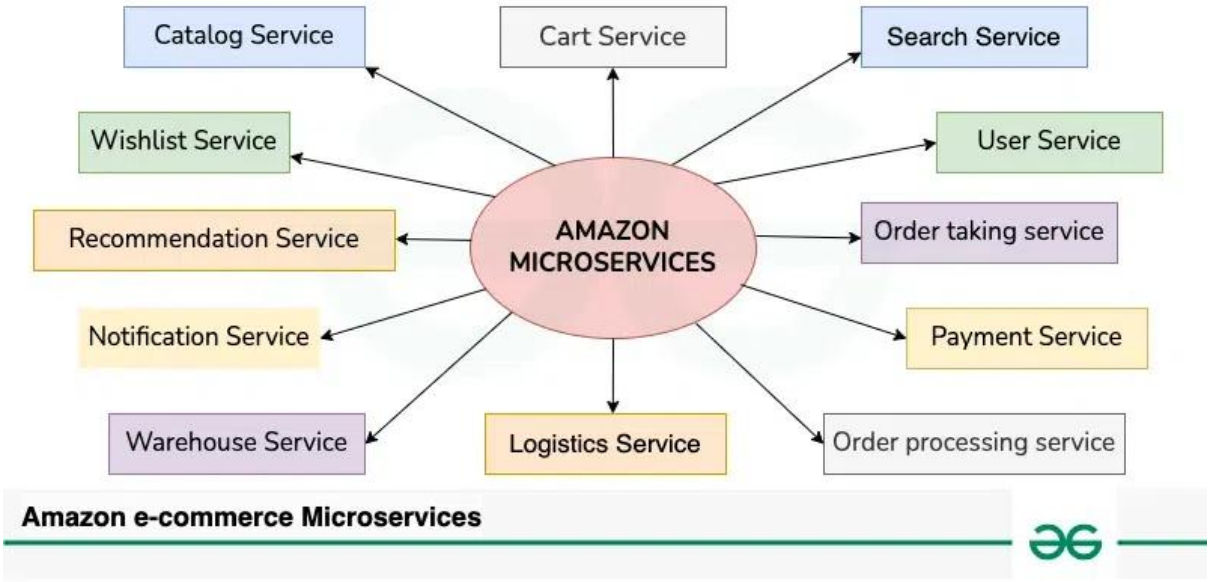
Design Patterns for Microservices Architecture :-

Below are the main design pattern of microservices:

1. **API Gateway Pattern** – Provides a single entry point and handles authentication, logging, and routing.
2. **Service Registry Pattern** – Maintains a dynamic list of services and their locations.
3. **Circuit Breaker Pattern** – Stops requests to failing services to prevent cascading failures.
4. **Saga Pattern** – Manages distributed transactions using compensating actions.
5. **Event Sourcing Pattern** – Stores state changes as events for audit and recovery.
6. **Strangler Pattern –** Gradually migrates a monolith to microservices.
7. **Bulkhead Pattern** – Isolates services to limit the impact of failures.
8. **API Composition Pattern** – Combines data from multiple services into one response.
9. **CQRS Pattern** – Separates read (query) and write (command) operations.

Real-World Example of Microservices :-

- Let's understand the Miscroservices using the real-world example of Amazon E-Commerce Application:
- Amazon’s online store runs on many small, specialized microservices, each handling a specific task

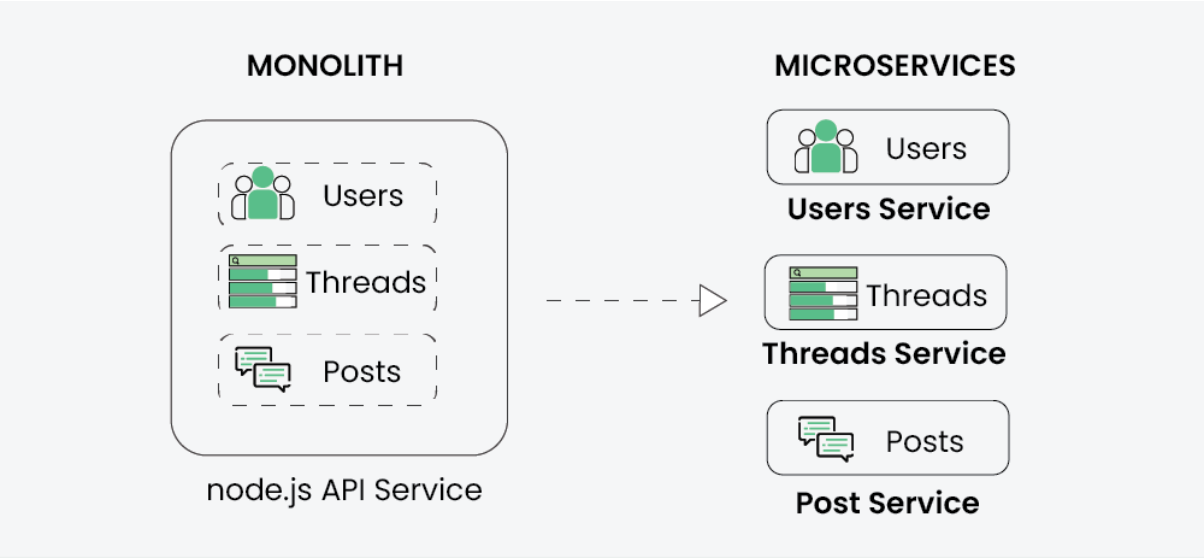


Below are the microservices involved in Amazon E-commerce Application:

1. **User Service:** Manages user accounts and preferences.
2. **Search Service:** Helps users find products quickly.
3. **Catalog Service:** Maintains product details and listings.
4. **Cart Service:** Manages items added for purchase.
5. **Wishlist Service**: Saves products for later.
6. **Order Taking Service:** Validates and accepts orders.
7. **Order Processing Service**: Handles order fulfillment.
8. **Payment Service:** Processes secure payments.
9. **Logistics Service:** Manages shipping and delivery.
10. **Warehouse Service**: Tracks inventory and restocking.
11. **Notification Service**: Sends order and offer updates.
12. **Recommendation Service**: Suggests relevant products.

How to migrate from Monolithic to Microservices Architecture?

Below are the main the key steps to migrate from a monolithic to microservices architecture:



Step 1: Evaluate the monolithic application and identify extractable components.

Step 2: Split the system into business-focused microservices.

Step 3: Use the Strangler Pattern for gradual migration.

Step 4: Define clear APIs for service communication.

Step 5: Set up CI/CD pipelines for automated deployment.

Step 6: Enable service discovery for dynamic interaction.

Step 7: Implement centralized logging and monitoring.

Step 8: Manage security and authentication consistently.

Step 9: Continuously refine services iteratively.

Applications of Microservices :-

1. **E-commerce:** Inventory, payments, orders, and users as separate services.
2. **Banking & FinTech**: Accounts, transactions, fraud detection, and support services.
3. **Streaming Platforms:** User profiles, recommendations, content, and billing services.
4. **Travel & Booking**: Flights, hotels, payments, and notifications services.
5. **Healthcare:** Patient records, appointments, billing, and reports services.
6. **Social Media**: Feed, chat, notifications, and user profile services.

Microservices vs. Monolithic Architecture :-

Aspect	Microservices Architecture	Monolithic Architecture
Architecture Style	Decomposed into small, independent services.	Single, tightly integrated codebase.
Development Team Structure	Small, cross-functional teams for each microservice.	Larger, centralized development team.
Scalability	Independent scaling of individual services.	Scaling involves replicating the entire application.
Deployment	Independent deployment of services.	Whole application is deployed as a single unit.
Resource Utilization	Efficient use of resources as services can scale independently.	Resources allocated based on the overall application's needs.
Development Speed	Faster development and deployment cycles.	Slower development and deployment due to the entire codebase.
Flexibility	Easier to adopt new technologies for specific services.	Limited flexibility due to a common technology stack.
Maintenance	Easier maintenance of smaller, focused codebases.	Maintenance can be complex for a large, monolithic codebase.

Spring Boot Microservices Example :-

Create a New Spring Boot Project in Spring Initializr -

Create a new Spring Boot project using **Spring Initializr** with the following settings:

- **Project:** Maven
- **Language:** Java
- **Packaging:** Jar
- **Java:** 17

Select these dependencies:

- Spring Boot DevTools
- Spring Data JPA
- Postgres Driver
- Spring Web

Generate the project, open it in **IntelliJ IDEA**, and run the application as described in the referenced article.

Create Schema in PostgreSQL and Insert Sample Data -

Open **pgAdmin**, create a database named –

1.userdb table:users - id,username

insert some sample records.

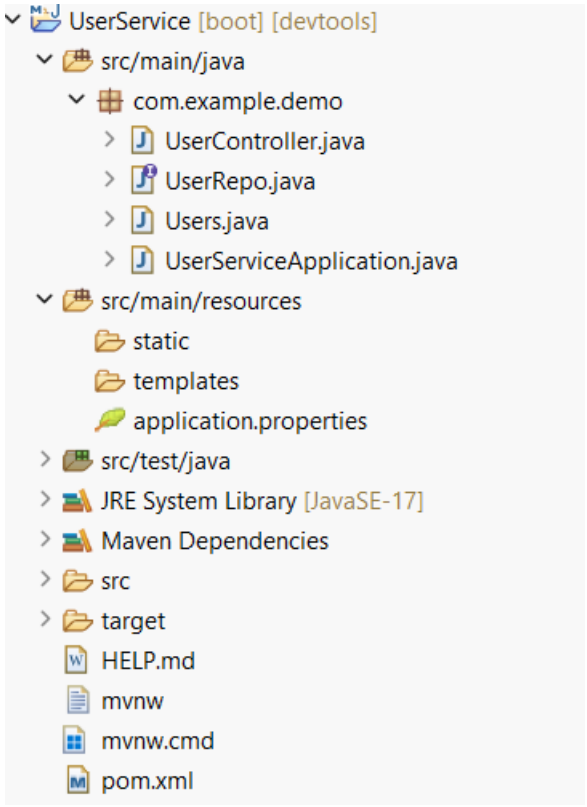
	id [PK] integer	username character varying (50)
1	1	Sagar
2	2	Sahil

2.orderdb table: orders - id,product
insert some sample records.

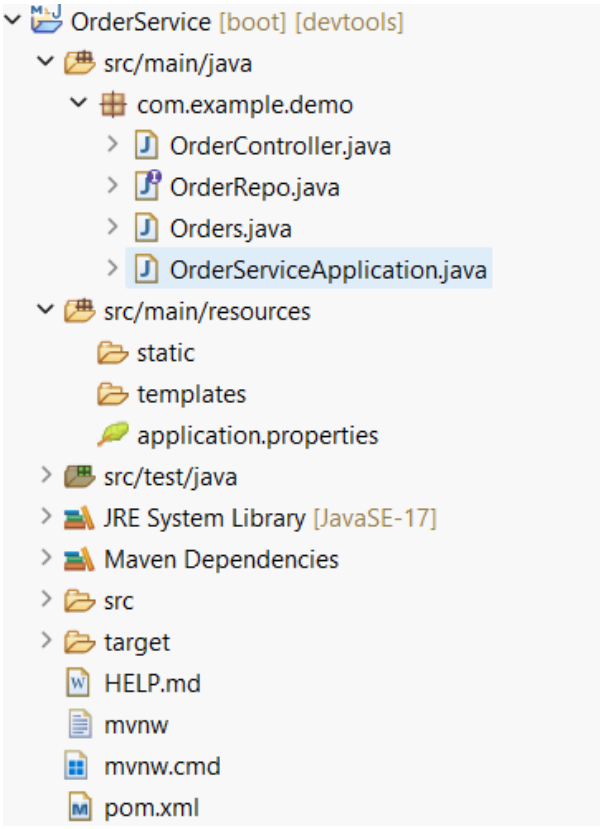
	id [PK] integer	product character varying (50)
1	1	Laptop
2	2	Mobile

Project Structure -

1.UserService



2.OrderService



pom.xml File -

```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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>4.0.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>UserService</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>UserService</name>
    <description>Demo project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies> Add Spring Boot Starters...
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-webmvc</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>

            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-webmvc-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

application.properties File -

Now make the following changes in your application.properties file.

1.UserService

```
spring.application.name=UserService

server.port=8080
spring.datasource.url=jdbc:postgresql://localhost:5433/userdb
spring.datasource.username=postgres
spring.datasource.password=4518

spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

2.OrderService

```
spring.application.name=OrderService

server.port=8081
spring.datasource.url=jdbc:postgresql://localhost:5433/orderdb
spring.datasource.username=postgres
spring.datasource.password=4518

spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Spring Boot Application File -

1.UserServiceApplication.java

```java
package com.example.demo;

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

@SpringBootApplication
public class UserServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(UserServiceApplication.class, args);
    }

}
```

2.OrderServiceApplication.java

```java
package com.example.demo;

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

@SpringBootApplication
public class OrderServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(OrderServiceApplication.class, args);
    }

}
```

Create Your Entity/Model Class -

Go to the **src > main > java > demo** and create a class Employee and put the below code. This is our model class.

1.Users.java

```java
package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="users")
public class Users {
    @Id
    @GeneratedValue
    private int id;
    private String username;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }

}
```

2.Orders.java

```java
package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="orders")
public class Orders {

    @Id
    @GeneratedValue
    private int id;
    private String product;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getProduct() {
        return product;
    }
    public void setProduct(String product) {
        this.product = product;
    }

}
```

Create Your Repository Interface -

Go to the **src > main > java > demo** and create an interface UserRepo and OrderRepo put the below code. This is our repository where we write code for all the database-related stuff.

1.UserRepo.java

```
package com.example.demo;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepo extends JpaRepository<Users,Integer>{

}
```

2.OrderRepo.java

```
package com.example.demo;

import org.springframework.data.jpa.repository.JpaRepository;

public interface OrderRepo extends JpaRepository<Orders,Integer>{

}
```

Create an Controller -

Go to the **src > main > java > demo** and create a class Controller and put the below code.

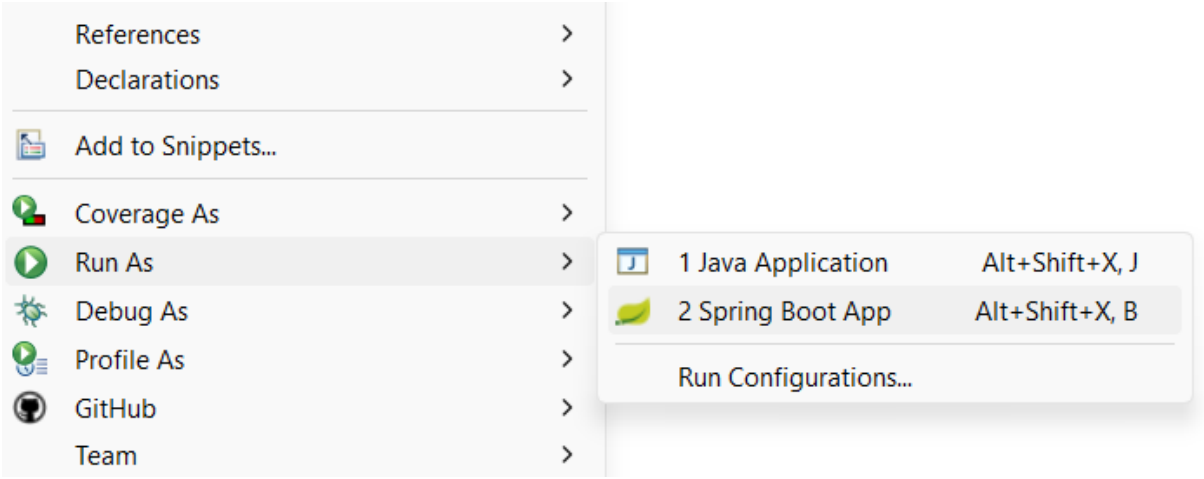
1.UserController.java

```
package com.example.demo;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class UserController {

    @Autowired
    UserRepo ur;

    @GetMapping("/users")
    public List<Users> getallusers(){
        return ur.findAll();
    }

}
```

2.OrderController,java

```
package com.example.demo;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class OrderController {

    @Autowired
    OrderRepo or;

    @GetMapping("/orders")
    public List<Orders> getallusers(){
        return or.findAll();
    }

}
```

Run the application –



1.UserService –

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v4.0.1)

2025-12-20T19:17:04.239+05:30  INFO 73416 --- [UserService] [  restartedMain] c.example.demo.UserServiceApplication
\eclipse-workspace\UserService\target\classes started by sagar in C:\Users\sagar\eclipse-workspace\UserService)
2025-12-20T19:17:04.248+05:30  INFO 73416 --- [UserService] [  restartedMain] c.example.demo.UserServiceApplication
2025-12-20T19:17:04.375+05:30  INFO 73416 --- [UserService] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor
disable
2025-12-20T19:17:04.375+05:30  INFO 73416 --- [UserService] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor
'DEBUG'
2025-12-20T19:17:05.795+05:30  INFO 73416 --- [UserService] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate
2025-12-20T19:17:05.899+05:30  INFO 73416 --- [UserService] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate
2025-12-20T19:17:06.874+05:30  INFO 73416 --- [UserService] [  restartedMain] o.s.boot.tomcat.TomcatWebServer
2025-12-20T19:17:06.905+05:30  INFO 73416 --- [UserService] [  restartedMain] o.apache.catalina.core.StandardService
2025-12-20T19:17:06.906+05:30  INFO 73416 --- [UserService] [  restartedMain] o.apache.catalina.core.StandardEngine
2025-12-20T19:17:07.046+05:30  INFO 73416 --- [UserService] [  restartedMain] b.w.c.s.WebApplicationContextInitializer
2025-12-20T19:17:07.382+05:30  INFO 73416 --- [UserService] [  restartedMain] org.hibernate.orm.jpa
2025-12-20T19:17:07.520+05:30  INFO 73416 --- [UserService] [  restartedMain] org.hibernate.orm.core
2025-12-20T19:17:08.742+05:30  INFO 73416 --- [UserService] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo
2025-12-20T19:17:08.849+05:30  INFO 73416 --- [UserService] [  restartedMain] com.zaxxer.hikari.HikariDataSource
2025-12-20T19:17:09.235+05:30  INFO 73416 --- [UserService] [  restartedMain] com.zaxxer.hikari.pool.HikariPool
2025-12-20T19:17:09.240+05:30  INFO 73416 --- [UserService] [  restartedMain] com.zaxxer.hikari.HikariDataSource
2025-12-20T19:17:09.441+05:30  INFO 73416 --- [UserService] [  restartedMain] org.hibernate.orm.connections.pooling
        Database JDBC URL [jdbc:postgresql://localhost:5433/userdb]
        Database driver: PostgreSQL JDBC Driver
        Database dialect: PostgreSQLDialect
        Database version: 18.0
        Default catalog/schema: userdb/public
        Autocommit mode: undefined/unknown
        Isolation level: READ_COMMITTED [default READ_COMMITTED]
        JDBC fetch size: none
        Pool: DataSourceConnectionProvider
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-12-20T19:17:11.421+05:30  INFO 73416 --- [UserService] [  restartedMain] org.hibernate.orm.core
JTA platform integration)
2025-12-20T19:17:11.445+05:30  INFO 73416 --- [UserService] [  restartedMain] j.LocalContainerEntityManagerFactoryBean
2025-12-20T19:17:11.687+05:30  INFO 73416 --- [UserService] [  restartedMain] o.s.d.j.r.query.QueryEnhancerFactories
2025-12-20T19:17:11.984+05:30  WARN 73416 --- [UserService] [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration
performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-12-20T19:17:12.768+05:30  INFO 73416 --- [UserService] [  restartedMain] o.s.boot.tomcat.TomcatWebServer
2025-12-20T19:17:12.781+05:30  INFO 73416 --- [UserService] [  restartedMain] c.example.demo.UserServiceApplication
```

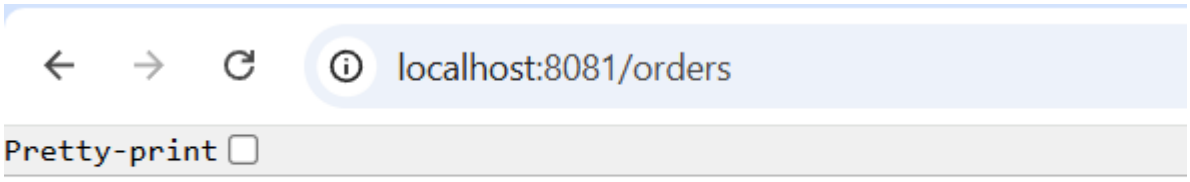
Output –


localhost:8080/users
Pretty-print
[{"id":1,"username":"Sagar"},{"id":2,"username":"Sahil"}]

2.OrderService

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v4.0.1)

2025-12-20T19:19:00.771+05:30  INFO 265080 --- [OrderService] [  restartedMain] c.example.demo.OrderServiceApplication
\eclipse-workspace\OrderService\target\classes started by sagar in C:\Users\sagar\eclipse-workspace\OrderService)
2025-12-20T19:19:00.778+05:30  INFO 265080 --- [OrderService] [  restartedMain] c.example.demo.OrderServiceApplication
2025-12-20T19:19:00.896+05:30  INFO 265080 --- [OrderService] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor
disable
2025-12-20T19:19:00.897+05:30  INFO 265080 --- [OrderService] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor
to 'DEBUG'
2025-12-20T19:19:02.361+05:30  INFO 265080 --- [OrderService] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate
2025-12-20T19:19:02.463+05:30  INFO 265080 --- [OrderService] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate
2025-12-20T19:19:03.565+05:30  INFO 265080 --- [OrderService] [  restartedMain] o.s.boot.tomcat.TomcatWebServer
2025-12-20T19:19:03.594+05:30  INFO 265080 --- [OrderService] [  restartedMain] o.apache.catalina.core.StandardService
2025-12-20T19:19:03.595+05:30  INFO 265080 --- [OrderService] [  restartedMain] o.apache.catalina.core.StandardEngine
2025-12-20T19:19:03.727+05:30  INFO 265080 --- [OrderService] [  restartedMain] b.w.c.s.WebApplicationContextInitializer
2025-12-20T19:19:04.076+05:30  INFO 265080 --- [OrderService] [  restartedMain] org.hibernate.orm.jpa
2025-12-20T19:19:04.208+05:30  INFO 265080 --- [OrderService] [  restartedMain] org.hibernate.orm.core
2025-12-20T19:19:05.446+05:30  INFO 265080 --- [OrderService] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo
2025-12-20T19:19:05.536+05:30  INFO 265080 --- [OrderService] [  restartedMain] com.zaxxer.hikari.HikariDataSource
2025-12-20T19:19:05.902+05:30  INFO 265080 --- [OrderService] [  restartedMain] com.zaxxer.hikari.pool.HikariPool
2025-12-20T19:19:05.907+05:30  INFO 265080 --- [OrderService] [  restartedMain] com.zaxxer.hikari.HikariDataSource
2025-12-20T19:19:06.083+05:30  INFO 265080 --- [OrderService] [  restartedMain] org.hibernate.orm.connections.pooling
        Database JDBC URL [jdbc:postgresql://localhost:5433/orderdb]
        Database driver: PostgreSQL JDBC Driver
        Database dialect: PostgreSQLDialect
        Database version: 18.0
        Default catalog/schema: orderdb/public
        Autocommit mode: undefined/unknown
        Isolation level: READ_COMMITTED [default READ_COMMITTED]
        JDBC fetch size: none
        Pool: DataSourceConnectionProvider
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-12-20T19:19:08.126+05:30  INFO 265080 --- [OrderService] [  restartedMain] org.hibernate.orm.core
enable JTA platform integration)
2025-12-20T19:19:08.151+05:30  INFO 265080 --- [OrderService] [  restartedMain] j.LocalContainerEntityManagerFactoryBean
2025-12-20T19:19:08.353+05:30  INFO 265080 --- [OrderService] [  restartedMain] o.s.d.j.r.query.QueryEnhancerFactories
2025-12-20T19:19:08.622+05:30  WARN 265080 --- [OrderService] [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration
performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-12-20T19:19:09.362+05:30  INFO 265080 --- [OrderService] [  restartedMain] o.s.boot.tomcat.TomcatWebServer
2025-12-20T19:19:09.375+05:30  INFO 265080 --- [OrderService] [  restartedMain] c.example.demo.OrderServiceApplication
```

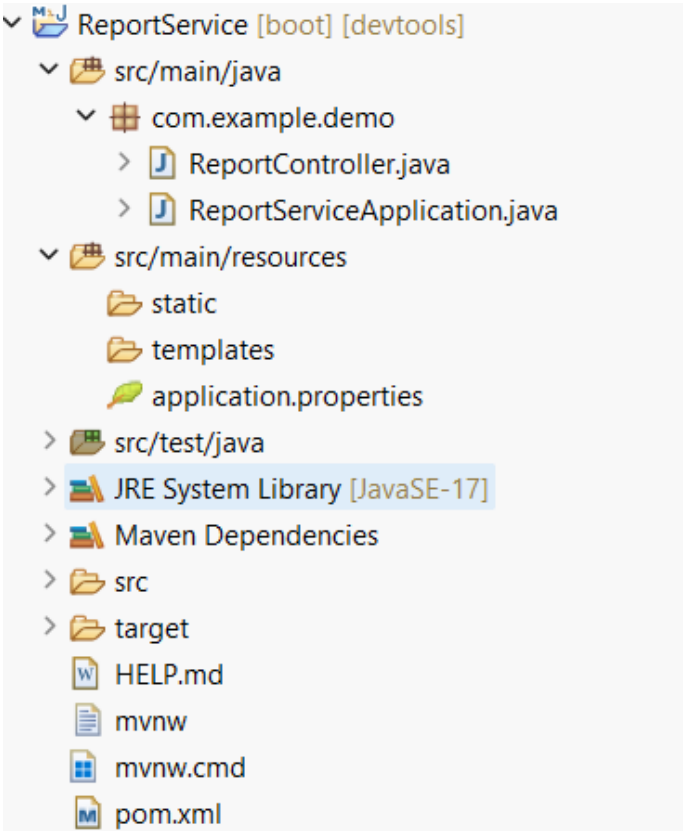
Output -



[{"id":1,"product":"Laptop"},{"id":2,"product":"Mobile"}]

Microservice

Project Structure –



Pom.xml File -

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<?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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>4.0.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>ReportService</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>ReportService</name>
    <description>Demo project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies> Add Spring Boot Starters...
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-webmvc</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-webmvc-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

application.properties File -

```
spring.application.name=ReportService

server.port=8082
```

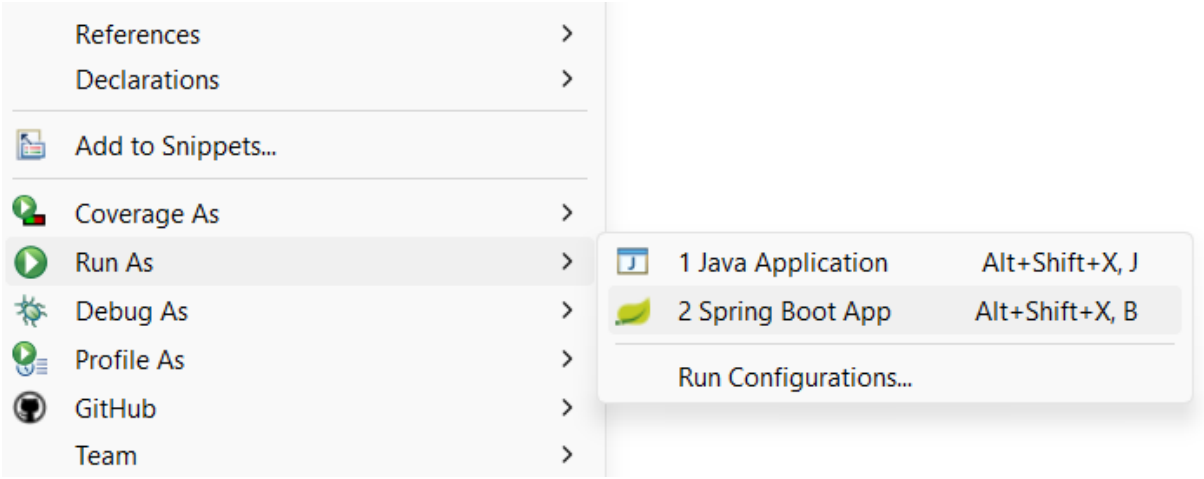
ReportServiceApplication.java File -

```
package com.example.demo;

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

@SpringBootApplication
public class ReportServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(ReportServiceApplication.class, args);
    }

}
```

ReportController.java File -

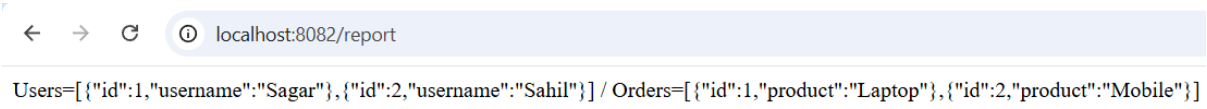
```
package com.example.demo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
public class ReportController {

    RestTemplate rt = new RestTemplate();

    @GetMapping("/report")
    public String getreport() {

        String users = rt.getForObject("http://localhost:8080/users",String.class);

        String orders = rt.getForObject("http://localhost:8081/orders",String.class);

        return "Users="+users+" / Orders="+orders;

    }
}
```

Run the application –




```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v4.0.1)

2025-12-20T19:20:25.527+05:30  INFO 269440 --- [ReportService] [  restartedMain] c.example.demo.ReportServiceApplication
\eclipse-workspace\ReportService\target\classes started by sagar in C:\Users\sagar\eclipse-workspace\ReportService)
2025-12-20T19:20:25.535+05:30  INFO 269440 --- [ReportService] [  restartedMain] c.example.demo.ReportServiceApplication
2025-12-20T19:20:25.652+05:30  INFO 269440 --- [ReportService] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor
disable
2025-12-20T19:20:25.652+05:30  INFO 269440 --- [ReportService] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor
to 'DEBUG'
2025-12-20T19:20:27.496+05:30  INFO 269440 --- [ReportService] [  restartedMain] o.s.boot.tomcat.TomcatWebServer
2025-12-20T19:20:27.531+05:30  INFO 269440 --- [ReportService] [  restartedMain] o.apache.catalina.core.StandardService
2025-12-20T19:20:27.531+05:30  INFO 269440 --- [ReportService] [  restartedMain] o.apache.catalina.core.StandardEngine
2025-12-20T19:20:27.634+05:30  INFO 269440 --- [ReportService] [  restartedMain] b.w.c.s.WebApplicationContextInitializer
2025-12-20T19:20:28.468+05:30  INFO 269440 --- [ReportService] [  restartedMain] o.s.boot.tomcat.TomcatWebServer
2025-12-20T19:20:28.478+05:30  INFO 269440 --- [ReportService] [  restartedMain] c.example.demo.ReportServiceApplication
```

Output –



Users=[{"id":1,"username":"Sagar"},{"id":2,"username":"Sahil"}] / Orders=[{"id":1,"product":"Laptop"},{"id":2,"product":"Mobile"}]