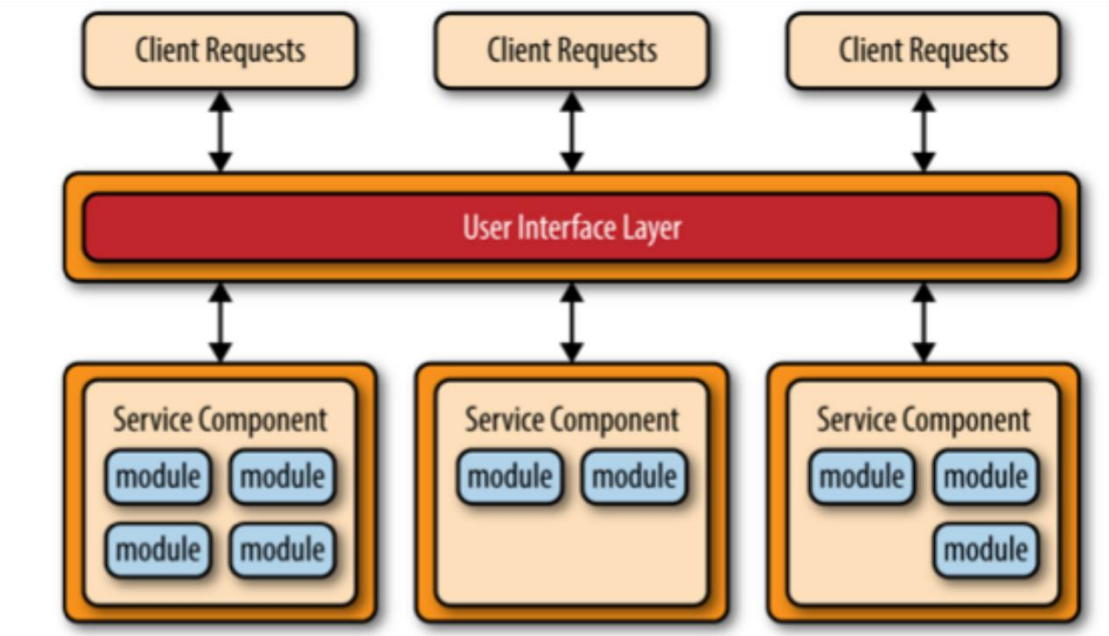
微服务架构：



场景：
在线电子商务平台，该平台允许用户浏览产品、查看详情、下单购买，并管理自己的账户和订单。
微服务架构的每一层及其对应场景：

用户界面层 (User Interface Layer)

场景: 用户通过 Web 浏览器访问电子商务平台，查看商品列表，进行搜索和筛选。
服务组件: 一个 Web 应用服务，提供 RESTful API 供前端使用，允许用户与平台交互。

产品浏览服务 (Product Browsing Service)

场景: 用户浏览不同的产品类别和商品详情。
服务组件: 负责提供产品信息，包括图片、描述、价格等。

购物车服务 (Shopping Cart Service)

场景: 用户选择商品后，添加到购物车，并可以查看购物车内容。
服务组件: 管理用户的购物车，处理商品的添加、删除和购物车状态更新。
相关代码：

```java
@Service
public class CartService {

    // 这里假设你有一个CartRepository来存储购物车数据，可能是一个数据库或缓存
    @Autowired
    private CartRepository cartRepository;

    public void addToCart(String productId, int quantity) {
        // 查找当前用户的购物车（可能需要从认证信息中获取用户ID）
        Cart cart = cartRepository.findByUserId(...);
        if (cart == null) {
            // 如果购物车不存在，则创建一个新的购物车
            cart = new Cart(...);
            cartRepository.save(cart);
        }
        // 向购物车中添加商品
        cart.addItem(productId, quantity);
        // 更新购物车数据
        cartRepository.save(cart);
    }
```

相关配置：

```
# application.properties
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
spring.application.name=cart-service
```

数据持久化层 (Data Persistence Layer)

场景: 所有服务需要持久化数据，如用户信息、订单详情、产品目录等。
服务组件: 每个服务可能有自己的数据库，或者共享某些数据库资源，但通过 API 进行访问，以保持服务间的独立性。

技术实现：
用户界面层：使用 vue 实现，vue 作为一个轻量级框架，能实现逐步应用。
产品浏览服务：使用 springboot 实现，Spring Boot 提供了大量的默认配置和自动配置，从而简化了项目的设置和初始化过程，使用 Spring Boot Starter 项目对象模型（POMs），可以快速地添加和管理项目的依赖关系。

购物车服务：使用 springcloud 实现，理由：Spring Cloud 是 Spring 系列中用于构建微服务架构的工具集，它提供了微服务治理的完整解决方案。在微服务架构中，每个服务都可以独立开发和部署，Spring Cloud 通过其组件和工具，如服务发现、负载均衡、熔断器等，简化了微服务的开发、部署和运维。
数据持久化层：mysql 实现，简单，便捷。