**🗂️ Plan de Implementación – MicroERP**

**1️⃣ Base del Proyecto (Infraestructura mínima)**

* Crear un **repositorio Git** (puede ser monorepo con carpetas /user-service, /product-service, etc., o repos separados).
* Configurar un **Config Server** con un repositorio de configuración.
* Montar un **Eureka Server** (descubrimiento de servicios).
* Levantar un **API Gateway** básico.  
  👉 Con esto ya tendríamos la "columna vertebral" para que los servicios hablen entre sí.

**2️⃣ Seguridad (Keycloak)**

* Instalar y configurar **Keycloak** en local (con Docker es lo más rápido).
* Crear un **realm** y definir roles básicos (ej. ADMIN, USER).
* Configurar el **API Gateway** para validar tokens JWT de Keycloak.  
  👉 Así se garantiza que desde el día 1 la seguridad está lista.

**3️⃣ Primer Microservicio (User Service)**

* Crear el **User Service** con Spring Boot.
* CRUD básico de usuarios (perfil, dirección, etc.).
* Integración con Keycloak para crear usuarios en el realm.
* Base de datos independiente (MySQL/Postgres).  
  👉 Este será el "servicio plantilla" que va a replicar para los demás.

**4️⃣ Servicios de Negocio**

Seguir implementando de a uno, siempre reutilizando la estructura del User Service:

1. **Customer Service** – CRUD de clientes.
2. **Product Service** – CRUD de productos + métricas con Actuator.
3. **Sales Service** – Registro de ventas.
4. **Stock Service** – Se actualiza escuchando eventos de ventas.
5. **Notification Service** – Envía mails al cliente.

**5️⃣ Comunicación Asíncrona (RabbitMQ/Kafka)**

* Levantar RabbitMQ en Docker.
* Configurar **Sales Service** para publicar evento de "venta confirmada".
* Configurar **Stock Service** y **Notification Service** para escuchar el evento.  
  👉 Acá empezás a usar mensajería real en el flujo de negocio.

**6️⃣ Observabilidad**

* Activar **Spring Boot Actuator** en todos los servicios.
* Configurar **Prometheus** para recolectar métricas.
* Montar **Grafana** con un dashboard básico.
* (Opcional) Configurar **ELK** para logs centralizados.
* Integrar **Jaeger/Zipkin** para trazas distribuidas.

**7️⃣ Resiliencia**

* Agregar **Resilience4j** en llamadas REST entre microservicios.
* Configurar **circuit breakers** y **reintentos**.

**8️⃣ Testing**

* **Unit tests** con JUnit + Mockito en cada servicio.
* **Integration tests** con Testcontainers (levantar DB y RabbitMQ en contenedores).
* (Opcional) **Contract tests** con Pact para validar comunicación entre servicios.

**9️⃣ DevOps / CI-CD**

* Crear **Dockerfile** para cada servicio.
* Usar **Docker Compose** para levantar todo junto (ideal en dev).
* Configurar **GitHub Actions** o **GitLab CI** para build + test + deploy.
* (Opcional) Orquestación en **Kubernetes**.

**✅ Resumen del Orden**

1. Infraestructura mínima (Config, Eureka, Gateway).
2. Seguridad con Keycloak.
3. User Service como plantilla.
4. Otros microservicios (uno por vez).
5. RabbitMQ/Kafka y eventos.
6. Observabilidad (Prometheus + Grafana).
7. Resiliencia (Resilience4j).
8. Testing (unit, integration, contract).
9. DevOps (Docker, CI/CD).

👉 La idea es **ir por fases**, no montar todo junto.  
Lo mejor es **arrancar con Config Server, Eureka, Gateway y un microservicio simple** (ej. User Service) → y recién después sumarle Keycloak y los demás.