Sistema de Gestión de Inventario para Retail:

Desarrollo de una aplicación que permite a las tiendas minoristas gestionar su inventario, seguimiento de productos y control de stock en tiempo real.Inventario para Retail:

Problema en que se va a resolver que ayude al cliente o al usuario final.

BASE DE DATO

Usuario

Tabla que almacena los datos de los usuarios que acceden al sistema.

Campo	Tipo de Dato	Descripción
id_usuario	SERIAL PRIMARY KEY	Identificador único del usuario.
nombre	VARCHAR(255)	Nombre completo.
email	VARCHAR(255)	Correo electrónico único.
password	VARCHAR(255)	Contraseña encriptada.
rol	VARCHAR(50)	Rol del usuario (Admin, Empleado).

Tienda

Tabla que almacena la información de cada tienda registrada.

Campo	Tipo de Dato	Descripción

id_tienda	SERIAL PRIMARY KEY	Identificador único de la tienda.
nombre	VARCHAR(255)	Nombre de la tienda.
direccion	TEXT	Ubicacion de la tienda

Producto

Tabla que almacena los productos disponibles en el inventario.

Campo	Tipo de Dato	Descripción
id_producto	SERIAL PRIMARY KEY	Identificador único del producto.
nombre	VARCHAR(255)	Nombre del producto.
descripcion	TEXT	Descripción del producto.
precio	DECIMAL(10,2)	Precio del producto.
categoria_id	ID	Relación con la categoría del producto.

Inventario

Tabla que gestiona el stock de productos en cada tienda.

Campo	Tipo de Dato	Descripción
id_inventario	SERIAL PRIMARY KEY	Identificador único del inventario.

id_tienda	SERIAL PRIMARY KEY	Relación con la tienda.
id_producto	SERIAL PRIMARY KEY	Relación con el producto.
stock_actual	int (stock_actual >= 0)	Cantidad actual del producto en esa tienda.

Clave Única UNIQUE(id_tienda, id_producto) Un producto solo puede tener un stock por tienda.

Categoría

Campo	Tipo de Dato	Descripción
id_categoria	ID	Identificador único de la categoría.
nombre	VARCHAR(255)	Nombre de la categoría.

Movimiento de Inventario

Campo	Tipo de Dato	Descripción
id_movimiento	ID	Identificador único del movimiento.
id_inventario	ID	Relación con el inventario afectado.
usuario_id	ID	Usuario que realizó la acción.

cantidad	INT (cantidad > 0)	Cantidad afectada en el movimiento.
tipo	VARCHAR(50)	Tipo de movimiento (entrada/salida).
fecha	TIMESTAMP	Fecha del movimiento.

REQUISITOS FUNCIONALES

1. **Gestión de Usuarios:**
    - El sistema debe permitir registrar usuarios con nombre, correo electrónico, contraseña encriptada y rol (Admin, Empleado).
    - El sistema debe permitir que los administradores gestionen usuarios (crear, editar, eliminar).
    - Los usuarios deben poder acceder al sistema solo con sus credenciales (correo y contraseña).
2. **Gestión de Tiendas:**
    - El sistema debe permitir registrar nuevas tiendas, asociándolas con una ubicación (dirección).
    - Los usuarios deben poder consultar las tiendas registradas.
    - Los administradores deben poder editar o eliminar tiendas.
3. **Gestión de Productos:**
    - El sistema debe permitir registrar productos, asignarles un precio y una categoría.
    - Los administradores deben poder editar y eliminar productos.
    - Los usuarios pueden consultar el catálogo de productos disponibles.
4. **Gestión de Inventario:**
    - El sistema debe permitir registrar y gestionar el stock de productos en cada tienda.
    - El sistema debe permitir actualizar la cantidad de stock cuando se realice un movimiento (entrada o salida).
    - Los usuarios deben poder consultar el stock de productos en las tiendas.
    - No debe haber duplicados de productos en el inventario de una tienda (por cada tienda y producto).
5. **Movimientos de Inventario:**
    - El sistema debe permitir registrar movimientos de inventario (entradas y salidas) realizados por los usuarios.
    - Los movimientos deben estar asociados a una tienda, producto, cantidad y tipo de movimiento (entrada o salida).

    - Los administradores deben poder consultar todos los movimientos de inventario registrados.
6. **Gestión de Categorías:**
    - El sistema debe permitir crear y gestionar categorías de productos.
    - Los productos deben ser asignados a una categoría al momento de ser registrados.

REQUISITOS NO FUNCIONALES

1. **Escalabilidad:** Capacidad para manejar un alto volumen de transacciones simultáneas.
2. **Disponibilidad:** El sistema debe estar disponible el 99.9% del tiempo.
3. **Seguridad:** Protección de datos sensibles mediante encriptación y OAuth 2.0.
4. **Rendimiento:** Respuesta rápida (<200ms) en operaciones comunes.
5. **Usabilidad:** Interfaz intuitiva y fácil de usar.

Tabla de Casos de Uso

ID	**Nombre del Caso de Uso**	**Descripción**	**Actor(es)**
UC01	Registrar Usuario	Permite a un administrador registrar nuevos usuarios con nombre, correo, contraseña y rol.	Administrador
UC02	Autenticar Usuario	Permite a un usuario ingresar al sistema con su correo y contraseña.	Administrador, Empleado
UC03	Gestionar Usuarios	Permite a un administrador editar y eliminar usuarios.	Administrador
UC04	Registrar Tienda	Permite a un administrador registrar una nueva tienda con nombre y dirección.	Administrador
UC05	Gestionar Tiendas	Permite a un administrador editar o eliminar tiendas existentes.	Administrador

UC06	Consultar Tiendas	Permite a un usuario ver la lista de tiendas registradas.	Administrador, Empleado
UC07	Registrar Producto	Permite a un administrador agregar nuevos productos con nombre, precio y categoría.	Administrador
UC08	Gestionar Productos	Permite a un administrador editar o eliminar productos.	Administrador
UC09	Consultar Productos	Permite a los usuarios ver los productos disponibles.	Administrador, Empleado
UC10	Gestionar Inventario	Permite a un administrador actualizar el stock de productos en cada tienda.	Administrador
UC11	Consultar Inventario	Permite a los usuarios ver la cantidad disponible de un producto en cada tienda.	Administrador, Empleado
UC12	Registrar Movimiento de Inventario	Permite a un usuario registrar una entrada o salida de productos en una tienda.	Administrador, Empleado
UC13	Consultar Movimientos de Inventario	Permite a los administradores ver el historial de movimientos de inventario.	Administrador

Tabla de Microservicios

Microservicio	**Descripción**	**Endpoints principales**

Auth Service	Gestiona la autenticación y autorización con OAuth 2.0.	/login, /register, /token/validate
User Service	Maneja la creación, edición y eliminación de usuarios.	/users, /users/{id}
Store Service	Gestiona las tiendas registradas en el sistema.	/stores, /stores/{id}
Product Service	Gestiona la información de los productos.	/products, /products/{id}
Inventory Service	Administra el stock de productos en cada tienda.	/inventory, /inventory/{store_id}/{product_id}
Movement Service	Registra y consulta movimientos de inventario.	/movements, /movements/{id}
API Gateway	Centraliza las solicitudes y las redirige a los microservicios adecuados.	/api/{service}
Logging & Monitoring	Recolecta logs y métricas del sistema.	/logs, /metrics

Análisis de las Herramientas

Para desarrollar tu aplicación distribuida basada en **microservicios**, con **OAuth 2.0**, **API Gateway**, **Docker** y **despliegue en Azure**, aquí tienes una evaluación de tecnologías y herramientas recomendadas:

---

1. Lenguaje de Programación y Frameworks

Tecnología	Descripción	Ventajas
.NET (C#) con ASP.NET Core	Ideal para aplicaciones empresariales robustas.	Soporte nativo en Azure, rendimiento alto, seguridad integrada.
Node.js con NestJS	Framework modular y escalable basado en TypeScript.	Fácil integración con microservicios, API REST y gRPC.
Spring Boot (Java)	Framework de Java para microservicios.	Soporte robusto para autenticación, integración con Kubernetes.
Golang (Gin/Fiber)	Lenguaje eficiente y ligero para microservicios.	Bajo consumo de recursos, excelente rendimiento.

Recomendación:

---

2. Base de Datos

Tecnología	Descripción	Ventajas
PostgreSQL	Base de datos relacional potente.	ACID, JSONB para datos semiestructurados, escalabilidad.

SQL Server (Azure SQL Database)	Base de datos de Microsoft optimizada para Azure.	Integración nativa con Azure, alta disponibilidad.
MongoDB (NoSQL)	Almacén de documentos flexible.	Escalabilidad horizontal, ideal para datos semiestructurados.
Redis (Caching & Session Storage)	Base de datos en memoria para caché y sesiones.	Alta velocidad, ideal para mejorar rendimiento.

Recomendación: PostgreSQL o **Azure SQL** si trabajas con datos estructurados y necesitas integridad transaccional.

---

4. Autenticación y Seguridad (OAuth 2.0, JWT)

Tecnología	**Descripción**	**Ventajas**
IdentityServer4 (para .NET)	Implementación de OAuth 2.0 y OpenID Connect.	Integración nativa con .NET, personalización.
Keycloak	Servidor de identidad Open Source.	Gestión centralizada de usuarios, compatibilidad con protocolos de autenticación.
Auth0	Plataforma de autenticación en la nube.	Seguridad robusta, fácil integración con APIs.

Azure AD B2C	Servicio de autenticación en Azure.	Autenticación empresarial con Azure.

Recomendación: IdentityServer4 si trabajas con .NET, **Keycloak** para una solución open-source.

---

5. API Gateway

Tecnología	**Descripción**	**Ventajas**
Ocelos (para .NET Core)	Gateway ligero y eficiente para microservicios en .NET.	Integración sencilla con ASP.NET Core.
Kong API Gateway	Solución Open Source para balanceo y seguridad.	Soporte para plugins y autenticación avanzada.
Azure API Management	Servicio nativo de Azure para gestionar APIs.	Monitoreo avanzado y seguridad con OAuth 2.0.

Recomendación: Ocelot si usas .NET, **Azure API Management** si buscas integración con Azure.

---

6. Contenerización y Orquestación

Tecnología	**Descripción**	**Ventajas**
Docker	Contenerización de aplicaciones.	Portabilidad, escalabilidad.
Kubernetes (AKS en Azure)	Orquestación de contenedores.	Escalabilidad automática, gestión eficiente.

Docker Compose	Herramienta para definir y ejecutar aplicaciones multi-contenedor localmente.	Ideal para desarrollo y pruebas.

Recomendación: Docker + AKS (Azure Kubernetes Service) para despliegue escalable.

---

7. Monitoreo y Logging

Tecnología	**Descripción**	**Ventajas**
Azure Monitor + Application Insights	Monitoreo nativo en Azure.	Integración con servicios de Azure, métricas detalladas.
ELK Stack (Elasticsearch, Logstash, Kibana)	Solución de logging en tiempo real.	Análisis avanzado de logs.
Prometheus + Grafana	Monitoreo y visualización de métricas.	Alertas personalizadas, integración con Kubernetes.

Recomendación: Azure Monitor si despliegas en Azure.

---

8. CI/CD (Integración y Despliegue Continuo)

Tecnología	**Descripción**	**Ventajas**
Azure DevOps	Plataforma de CI/CD de Microsoft.	Integración con Azure, pipelines optimizados.
GitHub Actions	Pipelines de CI/CD basados en eventos de GitHub.	Fácil configuración y mantenimiento.

Jenkins	Servidor de automatización Open Source.	Alta personalización, integración con múltiples herramientas.

Recomendación: Azure DevOps para una integración fluida con Azure.

Conclusión y Stack Recomendado

Si buscas **rendimiento y escalabilidad en Azure**, el stack recomendado sería:

- **Backend:** .NET Core con ASP.NET Core
- **Base de datos:** Azure SQL Database o PostgreSQL
- **Comunicación:** REST API con OAuth 2.0 y JWT
- **API Gateway:** Ocelot o Azure API Management
- **Contenedores:** Docker + Azure Kubernetes Service (AKS)
- **Autenticación:** IdentityServer4 o Azure AD B2C
- **Monitoreo:** Azure Monitor + Application Insights
- **CI/CD:** Azure DevOps