Descripción general del proyecto: plataforma de comercio electrónico

Objetivo: Diseñar e implementar parcialmente una plataforma de comercio electrónico simplificada y con funcionalidades básicas. Esta plataforma constará de dos funcionalidades clave: gestión del catálogo de productos y gestión de pedidos. Se anima a los candidatos a elegir la arquitectura más adecuada (microservicios, sin servidor o un enfoque híbrido) en función de sus puntos fuertes y las necesidades del proyecto.

1. Planificación de aplicaciones (arquitectura)

Tarea: Crear un diagrama de arquitectura de alto nivel para la plataforma de comercio electrónico. Los candidatos deben decidir entre utilizar microservicios tradicionales, funciones sin servidor (por ejemplo, AWS Lambda) o una combinación de ambos. La arquitectura debe incluir la aplicación frontend, la elección de los servicios backend (cómo se manejan los listados de productos y la gestión de pedidos) y cualquier recurso de nube simulado por LocalStack.

Entregable: Un diagrama que identifica claramente el enfoque arquitectónico, los componentes y sus interacciones.

2. Diseño de la nube

Objetivo: Con su arquitectura como base, debe simular o delinear teóricamente cómo se implementaría su diseño en un entorno de nube. Tienes la libertad de elegir el método que mejor muestre tus habilidades y conocimientos.

Opciones para completar:

Implementación real en la nube (utilizando cualquier proveedor de nube, como AWS o Azure): si opta por implementar un prototipo, asegúrese de que demuestre los componentes arquitectónicos clave que ha diseñado.

Simulación local: puede simular el entorno de la nube utilizando herramientas o simulaciones personalizadas (por ejemplo, usando contenedores para representar servicios en la nube, directorios para depósitos de S3, etc.).

LocalStack o similar

Entregable: código e instrucciones para ejecutar su simulación local; brinde una explicación detallada de cómo funcionaría en las funcionalidades de la nube del mundo real.

3. Canalizaciones (DevOps)

Tarea: configurar una canalización de CI/CD básica para uno de los servicios utilizando una herramienta de su elección (por ejemplo, GitHub Actions, GitLab CI/CD). El oleoducto debe simular el despliegue. También debería incrementar automáticamente los números de versión y generar un registro de cambios basado en mensajes de confirmación o títulos de relaciones públicas.

Entregable: Archivos de configuración de la canalización y una breve explicación de los pasos de la canalización.

4. backend

Tarea: implementar versiones básicas de dos servicios backend, cada uno con al menos un punto final REST. Si el reclutador o el correo electrónico especifican un idioma en particular, debes usar ese idioma. De lo contrario, eres libre de elegir el idioma que prefieras.

Servicio de catálogo de productos: debe proporcionar listados de productos.

Servicio de gestión de pedidos: debe encargarse de crear y enumerar pedidos.

Entregable: Código fuente de ambos servicios.

5. Interfaz

Tarea: crear una aplicación frontend simple que interactúe con ambos servicios backend. Si el reclutador o el correo electrónico especifican un marco en particular, debes utilizar ese marco. De lo contrario, puede elegir el marco que prefiera (por ejemplo, React, Angular, Vue).

Entregable: Código fuente de la aplicación frontend e instrucciones para ejecutarla localmente.

Criterios de evaluación:

Arquitectura: Claridad y viabilidad de la arquitectura propuesta, incluidas consideraciones de escalabilidad y mantenibilidad.

Diseño de la nube: comprensión de los servicios de la nube y cómo aprovecharlos en entornos de desarrollo y producción. Capacidad de simular estos servicios localmente con LocalStack.

DevOps: eficacia e integridad del proceso de CI/CD, incluido el control de versiones y la generación de registros de cambios.

Backend: Calidad del código, estructura e implementación de principios RESTful en los servicios.

Frontend: Diseño de interfaz de usuario e interacción con servicios backend.

Consejos para los candidatos:

Concéntrese en demostrar conocimiento y toma de decisiones en lugar de completar cada detalle.

Documente las suposiciones, elecciones y atajos realizados debido a limitaciones de tiempo.

Asegúrese de que todos los componentes sean fácilmente ejecutables por los evaluadores, con instrucciones claras.

Lo simple es mejor que lo complejo

Bono (opcional)

Autenticación: implementar funciones de inicio y cierre de sesión de usuario.

Múltiples idiomas: demuestre competencia implementando uno de los servicios backend o un servicio adicional en un idioma diferente.

Actualizaciones en tiempo real: implemente actualizaciones en tiempo real (por ejemplo, actualizaciones de inventario en tiempo real y notificaciones de estado de pedidos) utilizando tecnologías como WebSockets o Server-Sent Events (SSE).

Uso correcto de git: (Microcommits/Branching)

Documentación API: proporcione documentación de Postman o Swagger para sus servicios backend.

Diagrama ER y relaciones: un esquema de base de datos que incluye diferentes tipos de relaciones (por ejemplo, uno a uno, uno a muchos, muchos a muchos).

Envío

Envíe su código a un repositorio de GitHub.

Incluya un archivo README con:

Instrucciones sobre cómo configurar y ejecutar su aplicación.