Software Engineer | Business Case | Control de Mantenimiento de Autos

¡Nos alegra que hayas llegado a esta instancia del proceso!

En esta etapa te invitamos a trabajar sobre un Business Case, que vas a presentar frente a un grupo de líderes del equipo de Kavak. Esta dinámica busca simular un desafío real con el que podrías encontrarte en tu día a día por lo que es una excelente oportunidad para que te acerques aún más a nuestro negocio y a la manera en que trabajamos.

A tener en cuenta:

- Se valorará el uso de herramientas como Cursor, Copilot o similares para potenciar el desarrollo.
- Se evaluará: claridad, buen diseño, respeto de reglas de negocio, código mantenible.
- Puedes agregar funcionalidades adicionales si lo consideras necesario, siempre que no rompan los requerimientos base y estén justificadas en el README.

Entregables:

- Código fuente del proyecto.
- Instrucciones claras para ejecutar la aplicación.
- Breve README que explique:
    - decisiones de diseño,
    - cómo está modelado el dominio,
    - supuestos realizados (incluyendo criterios como cálculo de costo total, manejo de errores, etc.).

Opcional:

- Tests unitarios básicos (JUnit).
- Uso de enums para estados y tipos.
- Separación en capas (domain, service, controller).
- Migraciones o scripts SQL versionados

Requerimientos técnicos

- Java 11 o superior.
- Diseño orientado a objetos.
- Uso correcto de:
    - encapsulación
    - responsabilidades claras
    - validaciones de reglas de negocio
    - excepciones para estados inválidos
- Uso de base de datos PostgreSQL.
    - Se valora reflejar integridad en DB (por ejemplo: unicidad de patente, nullability, etc.)
- Se valora el uso de java.time para fechas.

Contexto

Una empresa que gestiona una flota de autos usados necesita un sistema simple para registrar y controlar los mantenimientos realizados sobre cada vehículo.

La flota se utiliza para operaciones diarias (por ejemplo: test-drives, traslados y preparación para venta). Cuando un auto entra en mantenimiento, el sistema debe reflejar su **impacto en la disponibilidad** del vehículo.

Versión V0: uso interno por operadores**.** La solución puede exponerse como API REST, CLI o similar (a elección del candidato). Se valorará que el diseño sea claro y fácil de ejecutar.

Objetivo

Diseñar e implementar una aplicación en Java que permita:

- Registrar vehículos.
- Registrar mantenimientos asociados a un vehículo.
- Controlar el estado de cada mantenimiento.
- Consultar el historial de mantenimiento de un vehículo.
- Determinar si un vehículo está disponible o no.

Requerimientos funcionales

1. Vehículos

Cada vehículo debe tener al menos:

- id
- patente (**única**)
- marca
- modelo
- año
- kilometrajeActual

El sistema debe permitir:

- Registrar un nuevo vehículo.
- Actualizar el kilometraje de un vehículo.
- Obtener la información de un vehículo por id o por **patente**.

Nota: en el enunciado se usará **patente** (evitar “placa” para no duplicar términos).

2. Mantenimientos

Cada mantenimiento debe estar asociado a un vehículo y contener:

- id
- tipoMantenimiento (por ejemplo: CAMBIO_ACEITE, FRENOS, MOTOR, LLANTAS, TRANSMISION, GENERAL)
- descripcion
- fechaCreacion
- estado (por ejemplo: PENDIENTE, EN_PROCESO, COMPLETADO, CANCELADO)
- costoEstimado
- costoFinal (opcional)

El sistema debe permitir:

- Registrar un mantenimiento para un vehículo.
- Cambiar el estado de un mantenimiento.
- Completar un mantenimiento e indicar su costo final.
- Cancelar un mantenimiento.

3. Reglas de negocio

El sistema debe respetar las siguientes reglas:

3.1 Transiciones válidas de estado

Un mantenimiento solo puede pasar por estados válidos, por ejemplo:

- PENDIENTE → EN_PROCESO → COMPLETADO
- PENDIENTE → CANCELADO
- EN_PROCESO → CANCELADO

Además:

- **No se puede completar** un mantenimiento que esté CANCELADO.
- Un mantenimiento COMPLETADO **no puede volver a cambiar de estado**.

3.2 Disponibilidad del vehículo

Un vehículo se considera **no disponible** si tiene al menos un mantenimiento:

- en estado PENDIENTE o EN_PROCESO.

3.3 Costo total

El sistema debe permitir calcular el costo total de mantenimiento de un vehículo.

- El criterio exacto debe estar definido y justificado por el candidato (por ejemplo: considerar solo COMPLETADOS con costoFinal).

4. Consultas requeridas

El sistema debe permitir:

- Obtener todos los mantenimientos de un vehículo.
- Obtener solo los mantenimientos activos de un vehículo.
- Obtener el costo total de mantenimiento de un vehículo.
- Saber si un vehículo está disponible o no.