Challenge

Orquestación de Pagos

ORQUESTACIÓN

DE PAGOS

challenge

📋 Evaluación Técnica – Rol: Full Stack (MID | Senior) Developer

¡Gracias por tu interés! Este desafío está diseñado para evaluar habilidades clave en arquitectura, diseño, testing y autonomía técnica. A continuación te explicamos qué vas a desarrollar, cómo se evalúa, y qué esperamos de un perfil mid | senior.

🎯 Objetivo General

Construir un sistema de orquestación de pagos sencillo, con arquitectura hexagonal, tests automatizados, y frontend desacoplado, que permita reemplazar Astro o NextJs por otro framework sin tocar la lógica de negocio.

Objetivo

🧠 ¿Qué vamos a evaluar?

No queremos una aplicación exhaustiva en funcionalidades, así que no te preocupes si hay pequeños errores, lo que nos interesa es evaluar tus conocimientos y tu capacidad de aportar ideas.

Evaluaremos tu capacidad para diseñar una arquitectura limpia y desacoplada usando principios de la arquitectura hexagonal, tu enfoque en testing efectivo (unitario e integración), la calidad y claridad de tu código, y tu autonomía para tomar decisiones técnicas.

También tendremos en cuenta cómo estructuras el proyecto, documentas tu solución y preparas el entorno para que pueda ser ejecutado fácilmente, así como tu habilidad para construir un frontend desacoplado que pueda ser reemplazado sin afectar la lógica del negocio.

📎 Entregables esperados

1. Repositorio GitHub
2. README con:
a. Setup (docker-compose up)
b. Cómo correr los tests
c. Explicación de arquitectura
d. Qué implementaste y qué no (scope)
3. Tests automatizados
4. Separación de capas (arquitectura hexagonal)
5. Frontend con Astro ó NextJs

Objetivo

⏰ Tiempo estimado

Tienes hasta 10 días desde la fecha de asignación.
Si necesitas más tiempo o tienes dudas, solo avísanos.

🏁 Bonus (no obligatorio)

- Pensamiento escalable ¿Qué puedo hacer para agregar muchos métodos de pago con el menor esfuerzo?

- Almacenar todos los requests a los proveedores, además especificando si fueron exitosos o no, tiempo de duración, entre otras métricas

CHAT GPT?

Te animamos a usar herramientas de inteligencia artificial como apoyo durante el desarrollo —ya sea para generar ideas, automatizar partes repetitivas o validar soluciones— siempre que mantengas un criterio profesional en las decisiones técnicas que tomes.

Usar IA de forma efectiva también es una habilidad valiosa que valoramos.

Challenge
Orquestación de Pagos

Objetivo

Construir un sistema de orquestación de pagos simple, inspirado en el siguiente diagrama C4, El sistema debe seguir la arquitectura hexagonal, ser testeable, y que nos permita cambiar el frontend fácilmente (por ejemplo, reemplazar Astro por otro framework).

Tecnologías a usar

DBs: PostgreSQL ó MySQL ó OracleDB ó SQLite

Front: Astro con (React ó Preact)

Back: Astro ó NextJS

Test: Jest (obligatorio)

tailwind (opcional)

Docker (obligatorio)

NOTA: Se sugiere el diagrama C4 a manera de inspiración, no se debe seguir estrictamente!

Aplicaciones y Proveedores involucrados

Debes crear el servidor en Astro Ó NextJs, que son frameworks webs modernos para crear sitios rápidos y optimizados, que además permiten definir rutas tipo API mediante funciones serverless, combinando contenido estático con lógica dinámica cuando se necesita.

Debes simular o mockear (lo que te parezca más fácil!) la respuesta de un servidor y retornar siempre la siguiente respuesta:

Response
{
"status": "success",
"transaction_id": "12345678-1234-5678-1234-567812345678"
}

Debes simular o mockear (lo que te parezca más fácil!) la respuesta de un servidor y retornar siempre error

Endpoints de la aplicación

1. Crear orden de pago

POST / api / payment_order / :uuid

Endpoint

2. Ver orden de pago

GET / payment_order / :uuid

Page

3. Listar métodos de pago asociados al país

GET / payment_order / :uuid

Page

4. Procesar orden de pago

POST / api / payment_order / :uuid

Endpoint

Challenge
Orquestación de Pagos

Endpoints

1. Crear orden de pago

POST / api / payment_order

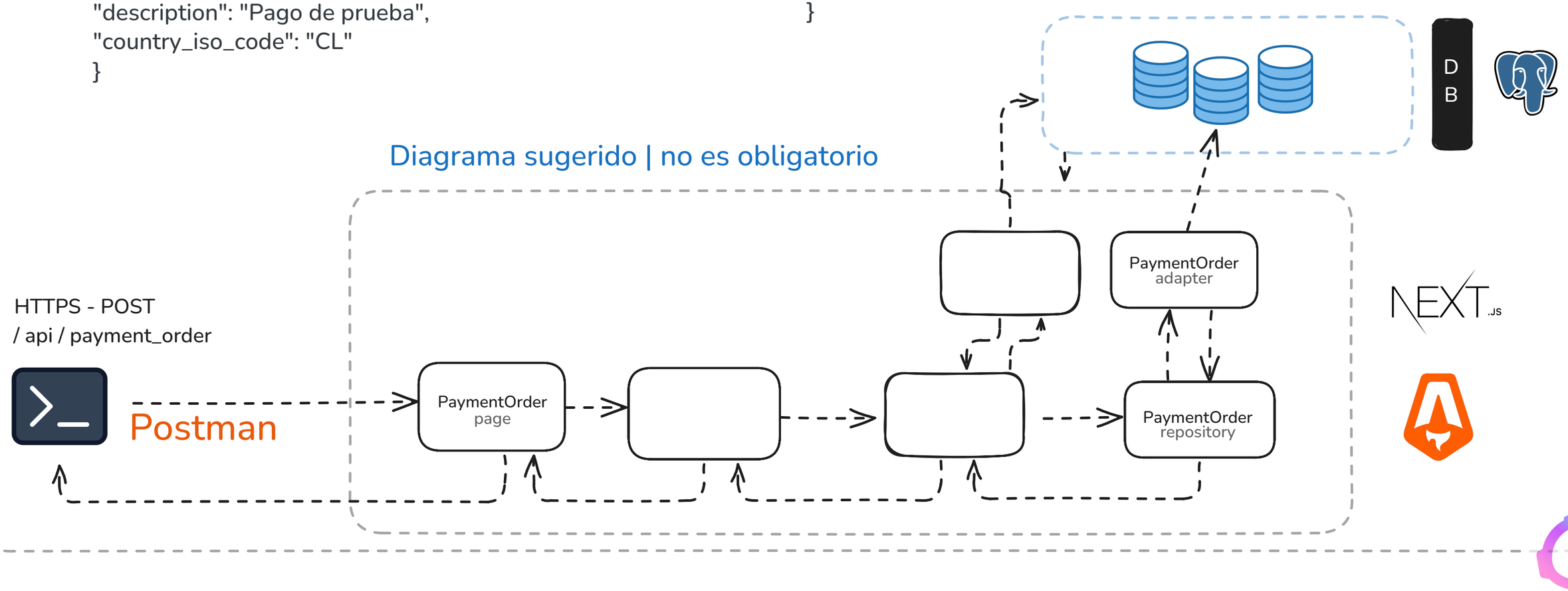
Request

```
{
"amount": 70000,
"description": "Pago de prueba",
"country_iso_code": "CL"
}
```

Response

```
{
"uuid": "f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
"type": "payment_order",
"attributes": {
"amount": 70000,
"description": "Pago de prueba",
"country_iso_code": "CL",
"created_at": "2025-05-27T14:36:22.123Z",
"payment_url": "http://127.0.0.1:4321/api/payment_order/uuid"
}
```

Diagrama sugerido | no es obligatorio



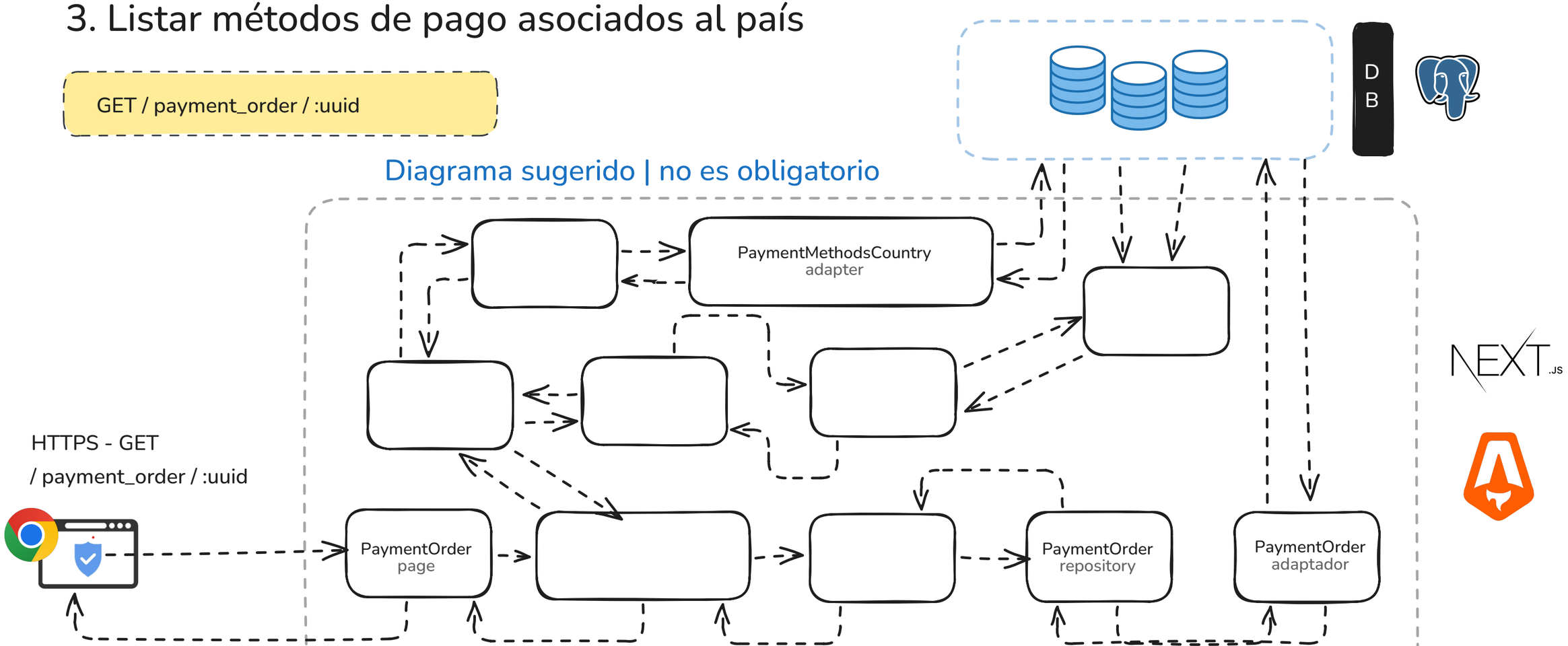
2. Ver orden de pago

GET / payment_order / :uuid

3. Listar métodos de pago asociados al país

GET / payment_order / :uuid

Diagrama sugerido | no es obligatorio



Challenge
Orquestación de Pagos

Pages

Estructura de la pasarela de Pago

El diseño es libre, se valorará la creatividad, transiciones y demás recursos,
pero es obligatorio el uso de React o Preact y que los datos se rendericen desde el servidor!

2. Ver orden de pago

GET / payment_order / :uuid

3. Listar métodos de pago asociados al país

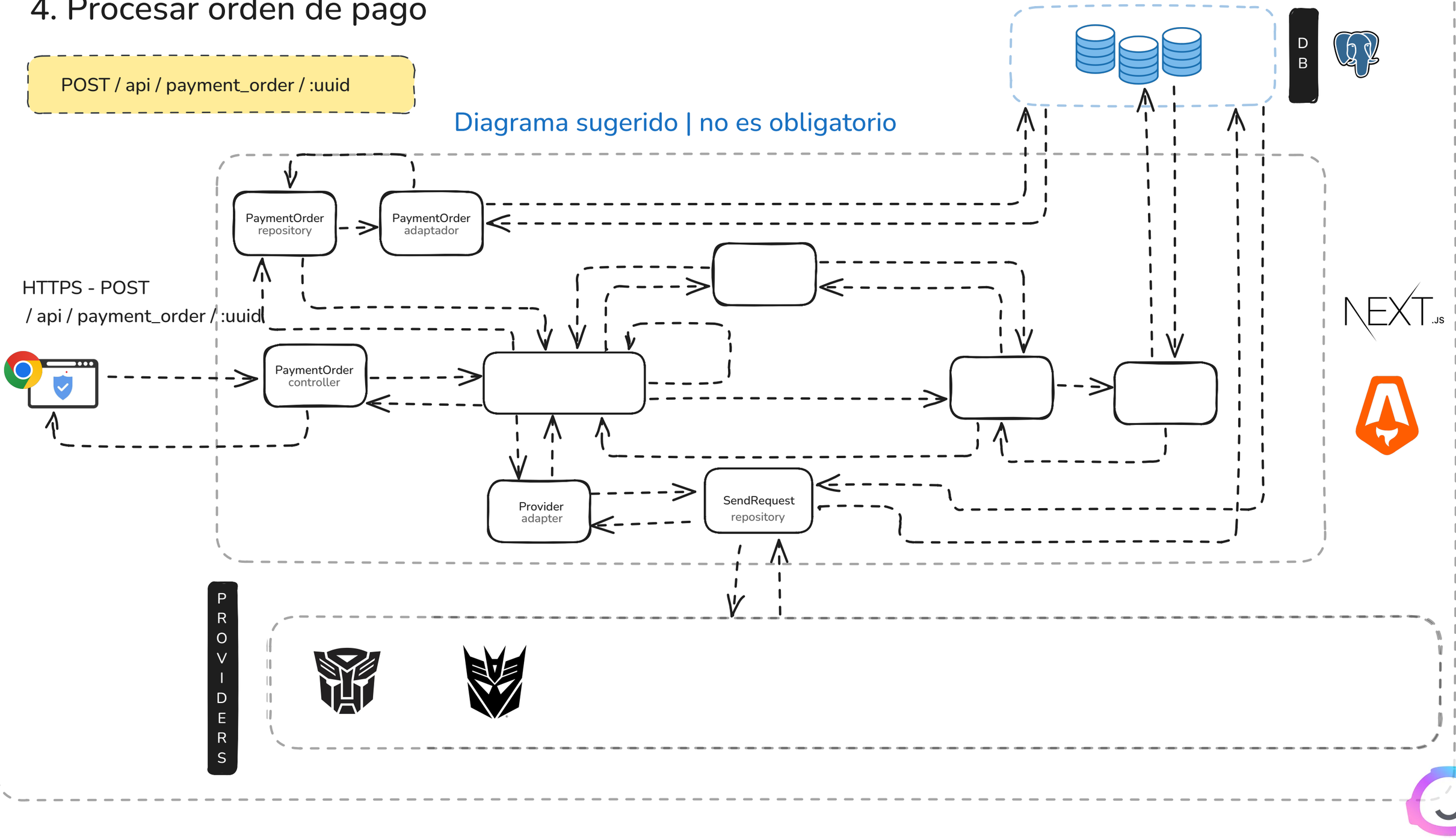
GET / payment_order / :uuid

Endpoints

4. Procesar orden de pago

POST / api / payment_order / :uuid

Diagrama sugerido | no es obligatorio

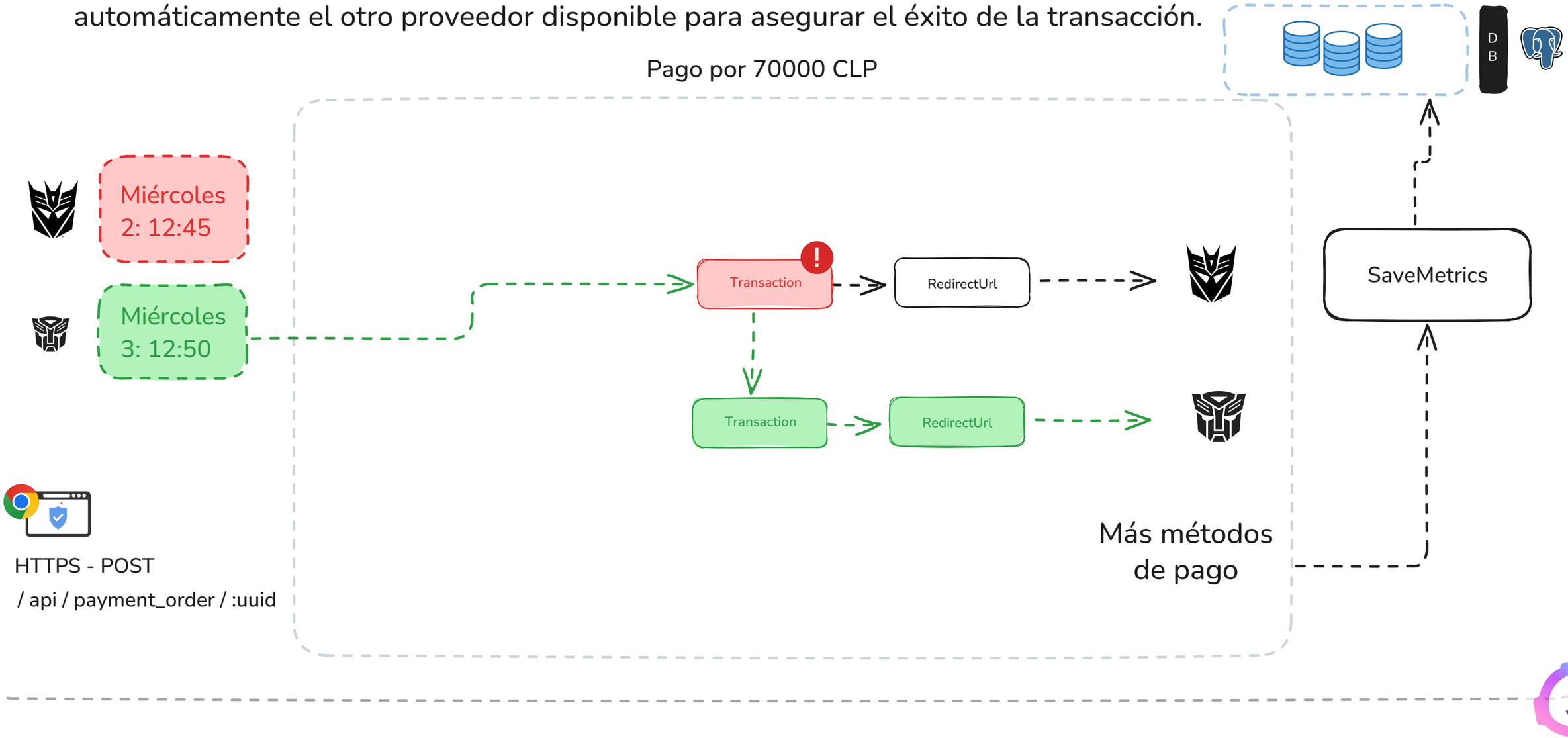


Challenge
Orquestación de Pagos

Endpoints

Procesamiento de una orden de pago

El backend debe manejar un ruteo "inteligente" (entre comillas, aquí nos interesa saber como piensas en escalar a más proveedores) que, ante el fallo de un método de pago, seleccione (como tu lo desees) automáticamente el otro proveedor disponible para asegurar el éxito de la transacción.



Gracias !

Gracias por tomarte el tiempo de realizar esta prueba técnica,
Sabemos que implica un esfuerzo importante y valoramos profundamente tu dedicación,
tu enfoque en la calidad del código y tu forma de resolver problemas.
Independientemente del resultado, agradecemos que hayas compartido tu talento y tu visión con nosotros.

Esperamos poder darte feedback pronto y, si es posible, continuar avanzando en el proceso contigo. ¡Gracias nuevamente!