Data Platform Engineer Challenge

En Spot2, estamos redefiniendo el futuro del Real Estate Comercial, aprovechando la tecnología para conectar de manera efectiva espacios comerciales con proyectos empresariales ambiciosos, todo a través de nuestra avanzada plataforma. Nuestros clientes confían en nosotros para identificar y asegurar los espacios ideales que materialicen sus visiones de crecimiento y éxito. En la dinámica y competitiva industria actual, el análisis avanzado de datos es clave para desarrollar herramientas que generen insights valiosos para la toma de decisiones, así como productos de datos que impulsen un impacto tangible.

Nuestro equipo de Data promueve una cultura impulsada por los datos, con el objetivo de posicionar a Spot2 como líder en investigación de mercado e innovación, utilizando Machine Learning e IA. El rol del Data Platform Engineer es fundamental, ya que diseña productos de datos escalables y de alta calidad, implementando soluciones que reflejan nuestra excelencia y sólidas prácticas de programación.

Buscamos personas apasionadas por la tecnología que compartan estos principios para liderar juntos la innovación en el mercado del real estate comercial.

Objetivo

Búsquedas Geoespaciales con PostgreSQL y Docker

Este challenge tiene como objetivo evaluar tus habilidades en **Django**, **GeoDjango**, **PostgreSQL (PostGIS)** y **Docker**, desarrollando una API REST capaz de exponer datos geoespaciales y realizar consultas espaciales eficientes.

Durante el ejercicio, deberás:

- Diseñar un modelo de datos geoespacial.
- Cargar un dataset inicial en la base de datos mediante un **custom command de Django**.
- Exponer una API REST con endpoints que soporten búsquedas espaciales.

Como *extra opcional*, podrás trabajar con un segundo dataset que requiere normalización antes de ser cargado.

Tiempo de desarrollo aprox.: Está pensado para ser completado en **un fin de semana** (8 a 12 horas de trabajo total).

Instrucciones

Se deberá cargar un dataset inicial en la base de datos mediante un **custom command de Django**.

Además se deberá Exponer una API REST con endpoints que soporten búsquedas espaciales. Los endpoints a implementar son los siguientes

1. **Listar todos los spots**

   GET /api/spots/

2. **Búsqueda geoespacial – spots cercanos**

   GET /api/spots/nearby/?lat=19.4326&lng=-99.1332&radius=2000

3. **Filtrado por atributos**

   GET /api/spots/?sector=9&type=1&municipality=Álvaro Obregón

4. **Búsqueda geoespacial – spots dentro de un polígono**
   POST /api/spots/within/

   {

   polygon": {

   "type": "Polygon",

   "coordinates": [your_coordinates]}

}

5. **Consulta agregada – precio promedio por sector**

   GET /api/spots/average-price-by-sector/

6. **Detalle de un spot específico**

   GET /api/spots/{spot_id}/

7. **Ranking de spots por precio total de renta**

   GET /api/spots/top-rent/?limit=10

Requerimientos técnicos

Tecnologías obligatorias:

- Python 3.11+
- Django 5+
- Django REST Framework
- GeoDjango
- PostgreSQL + PostGIS
- Docker y docker-compose

Datos

1. **lk_spots.csv**
   - Datos ya limpios y normalizados.
   - Este dataset **debe cargarse obligatoriamente** a la base de datos mediante un comando de Django.

2. **props_list.json (opcional)**
   - Datos desnormalizados y con inconsistencias.
   - Puedes realizar un proceso de transformación y carga como un **extra opcional**.
   - Si decides hacerlo, documenta brevemente tu proceso en el README.

A continuación se detalla los campos de LK_SPOTS.csv

Nombre	Descripción
Spot ID	El ID identificador del spot
Spot Sector Id	9 → Industrial 11 → Office 12 → Retail 15 → Land
Spot Type Id	1 → Single 2→ Complex 3 → Subspace
Spot Settlement	La Colonia donde pertenece el spot
Spot Municipality	La Municipalidad donde pertenece el spot
Spot State	El Estado donde pertenece el spot
Spot Región	La Región donde pertenece el spot
Spot Corridor	El corredor donde pertenece el spot; Los corredores segmentan característico socio demográficas
Spot Address	La dirección del Spot
Spot Title	El título del Spot
Spot Description	La descripción del Spot
Spot Latitude	
Spot Longitude	
Spot Area In Sqm	Área en m2
Spot Price Sqm Mxn Rent	Precio por m2 MXN para Renta
Spot Price Total Mxn Rent	Precio total MXN para Renta
Spot Price Sqm Mxn Sale	Precio por m2 MXN para Venta
Spot Price Total Mxn Sale	Precio total MXN para Venta
Spot Maintenance Cost MXN	Costo de mantenimiento en pesos mexicanos.
Spot Modality	Los spots pueden estar publicados para renta, venta o ambos
User Id	El ID identificador del Usuario
Spot Created Date	Fecha de creación del Spot

Criterios de evaluación

Criterio	Descripción
Diseño del modelo	Correcto uso de campos geoespaciales (PointField, PolygonField, etc.)
Carga de datos (obligatoria)	Implementación de un *management command* funcional (python manage.py load_data)
Consultas espaciales	Correcta implementación y eficiencia
Calidad del código	Organización, legibilidad y buenas prácticas
Uso de Docker	Proyecto reproducible con docker-compose up
Tests	Cobertura básica de pruebas unitarias o de integración para el API y lógica de carga de datos.
Documentación	README claro con instrucciones y ejemplos.
Extras (opcional)	Normalización del segundo dataset, paginación, filtros avanzados, índices geoespaciales, documentación con Swagger o despliegue en la nube.

Extras (Opcional)

- Normalizar y cargar el prop list.json
- Implementar paginación, ordenamiento o filtros avanzados.
- Añadir pruebas unitarias o de integración.
- Agregar índices geoespaciales.
- Documentar la API con **Swagger** o **drf-spectacular**.

Entrega

1. Publica tu código en un **repositorio público de GitHub**.
2. Incluye en tu README.md:
    - Instrucciones de ejecución (docker-compose up, load_data).
    - Descripción de las decisiones técnicas.
    - Ejemplos de endpoints con parámetros y respuestas
3. Proyecto con uso de Docker para entorno reproducible.
    - Código limpio, modular y bien documentado.

- API funcional con ejemplos de consultas en el README.
- Carga de datos realizada mediante un **management command** de Django.
- (Opcional) Carga de los datos raw mediante un **management command** de Django.
-

4. Envía el enlace del repositorio al equipo técnico para revisión.

Dudas o preguntas

Si tienes alguna duda, escribe a Estefania Louzau José Castellanos