Sistema de estimación de hotspots urbanos a partir de eventos y contexto temporal

---

1. Idea y planteamiento

Este proyecto parte de una observación sencilla: en una ciudad como Madrid, la actividad no se distribuye de forma uniforme. Hay zonas que, en determinados momentos del día, concentran mucha más actividad debido a eventos concretos.

La intención fue diseñar un sistema que permitiera estimar esas zonas de mayor intensidad urbana para una fecha y hora concretas.

Desde el inicio decidí separar claramente dos conceptos:

- Los eventos reales (dato observable).
- La estimación de hotspots (dato calculado).

Esa separación está presente tanto en la arquitectura como en la API.

2. Experiencia de usuario y control temporal

Un elemento clave del sistema es el control temporal.

El usuario no solo puede seleccionar una fecha, sino que dispone de un control deslizante para elegir la hora del día. Este slider permite ver cómo cambia la intensidad estimada del mapa a lo largo del tiempo.

Esto mejora la experiencia visual porque:

- Permite simular distintos escenarios horarios.
- Hace evidente el impacto temporal de los eventos.
- Facilita entender la dinámica urbana durante el día.

Cada vez que se modifica la hora, el sistema recalcula la estimación del heatmap en tiempo real.

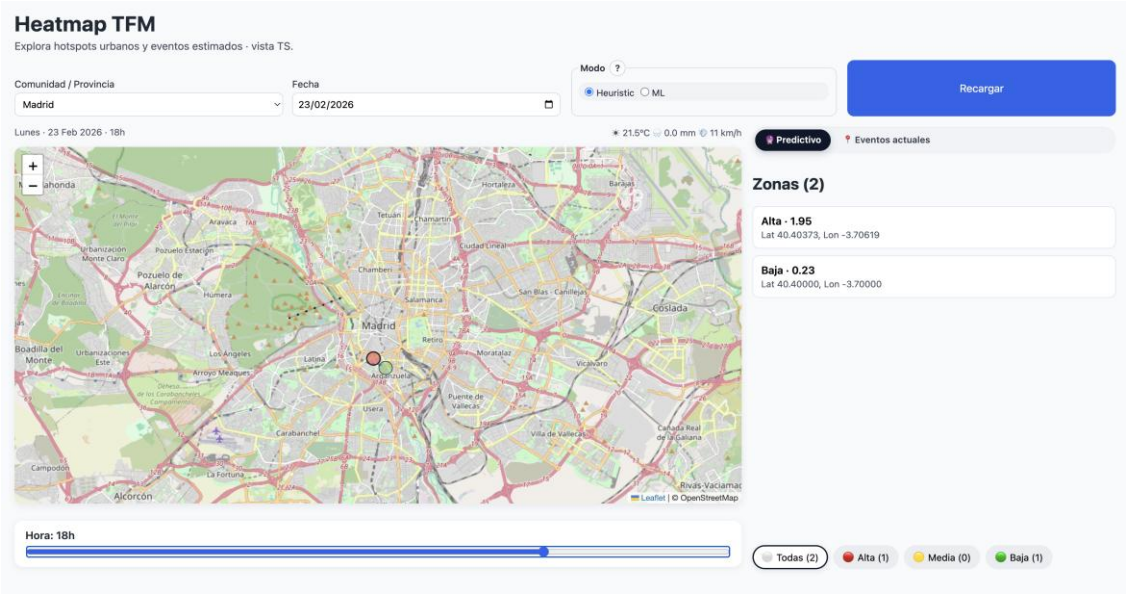
Una de las capturas incluidas en esta memoria corresponde precisamente a esta vista predictiva con el selector horario activo.

3. Capa predictiva: modelo actual

Es importante aclarar que el modelo implementado no es un modelo entrenado con datos históricos reales. No se ha aplicado aprendizaje supervisado en esta fase.

El sistema utiliza un modelo heurístico estructurado que combina variables como:

- Hora seleccionada.
- Tiempo hasta el inicio del evento.
- Categoría.
- Popularidad estimada.
- Proximidad espacial.
- Factores meteorológicos

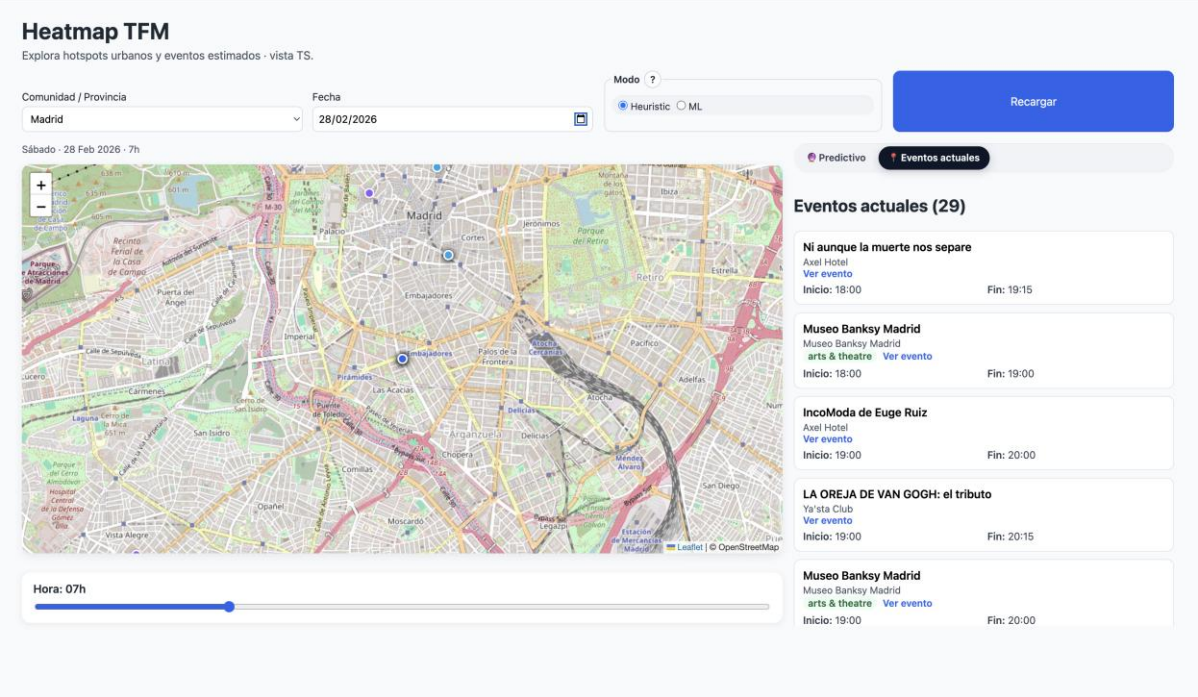


La finalidad no era entrenar un modelo definitivo, sino construir una base técnica preparada para evolucionar hacia un modelo predictivo real en el futuro.

La arquitectura permite generar y almacenar la información necesaria para que, en una siguiente fase, pueda entrenarse un modelo con datos históricos reales.

4. Eventos reales e integración externa

La segunda vista del sistema muestra los eventos reales almacenados en la base de datos.



Actualmente el sistema puede obtener eventos desde APIs externas. En esta versión se ha integrado la API pública de Ticketmaster como fuente principal.

Para facilitar la evaluación:

- El proyecto incluye datos demo que permiten probar el sistema sin depender de servicios externos.
- La integración con Ticketmaster es opcional.

Si el evaluador desea probar la integración real, puede:

1. Crear una API key gratuita en el portal de desarrolladores de Ticketmaster.
2. Añadir la variable de entorno *TICKETMASTER_API_KEY* al entorno Docker.
3. Ejecutar el proceso de sincronización.

En cualquier caso, el sistema funciona completamente con los datos de ejemplo incluidos en el repositorio.

5. Arquitectura y estructura del proyecto

El sistema está dividido en tres bloques principales:

A. Backend
Desarrollado con FastAPI y conectado a PostgreSQL.
Expone dos endpoints diferenciados:

- /api/events → eventos reales.
- /api/heatmap → estimación analítica.

Esta separación evita mezclar datos observados con datos calculados.

B. Frontend
Desarrollado con React y Vite.
Permite seleccionar fecha, ajustar la hora mediante slider y visualizar tanto el mapa como el listado de eventos.

C. Base de datos
Estructurada para almacenar eventos, venues, observaciones meteorológicas y snapshots de variables analíticas.

6. Reproducibilidad y acceso al proyecto

El proyecto es completamente reproducible mediante Docker Compose.

Para ejecutarlo en local:

```
cd docker
docker compose up -d
./init_db.sh
```

Este script crea las tablas necesarias y carga datos de ejemplo.

El código fuente completo está disponible públicamente en:

https://github.com/Viaintermedia-Vicente/urban-demand-heatmap

Además, el sistema puede probarse directamente online en:

https://heatmaps.viaintermedia.com/

Esto permite evaluar su funcionamiento sin necesidad de instalar Docker.

7. Uso de inteligencia artificial durante el desarrollo

La inteligencia artificial se ha utilizado como herramienta de apoyo durante el desarrollo del proyecto.

El proceso seguido fue estructurado:

1. Definición del contrato de la API.
2. Escritura de tests.
3. Implementación progresiva hasta cumplir los tests.

4. Refactorización y mejora de arquitectura.

La IA se utilizó principalmente para:

- Revisar coherencia de arquitectura.
- Detectar inconsistencias entre modelo y base de datos.
- Ajustar el modelo heurístico.
- Resolver errores de integración.
- Preparar el entorno reproducible.

Las decisiones estructurales —separación de capas, modelo heurístico, diseño de endpoints, organización de la base de datos— han sido tomadas de forma consciente y revisadas manualmente.