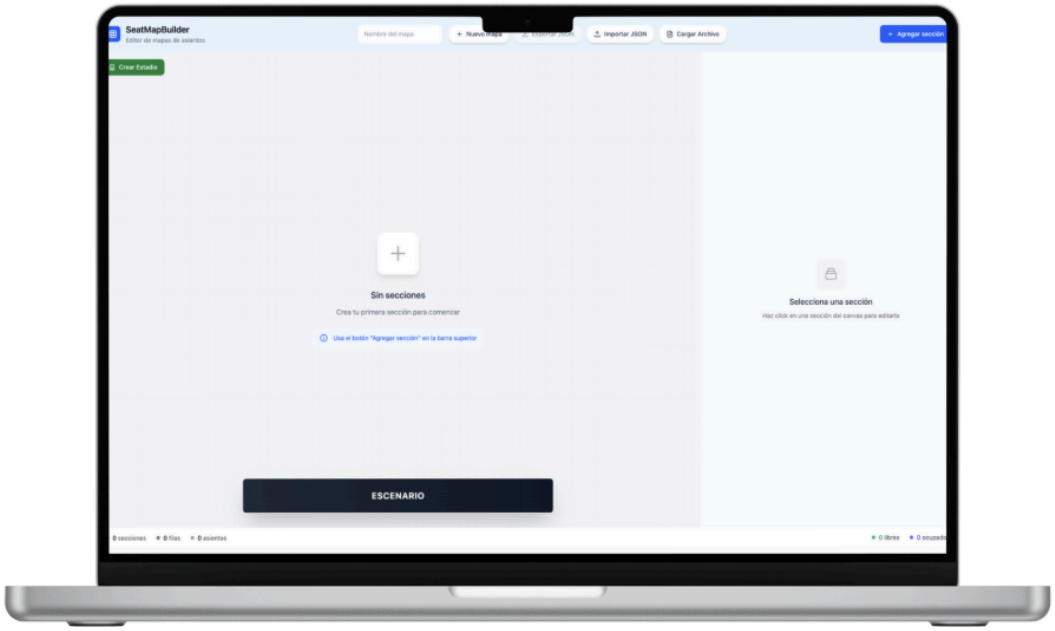
Prueba técnica Fanz

“seatMapBuilder”



by: Pedro González Núñez

Índice:

Desafios encontrados:

Un desafío que encontré en el desarrollo de la aplicación fue el cuestionamiento sobre si usar Seats.io.

Por un lado, según investigué y aprendí, esta tool permite integrar widgets embebibles en su front mediante el SDK/API, para permitir renderizar esos componentes de visualización de estadios.

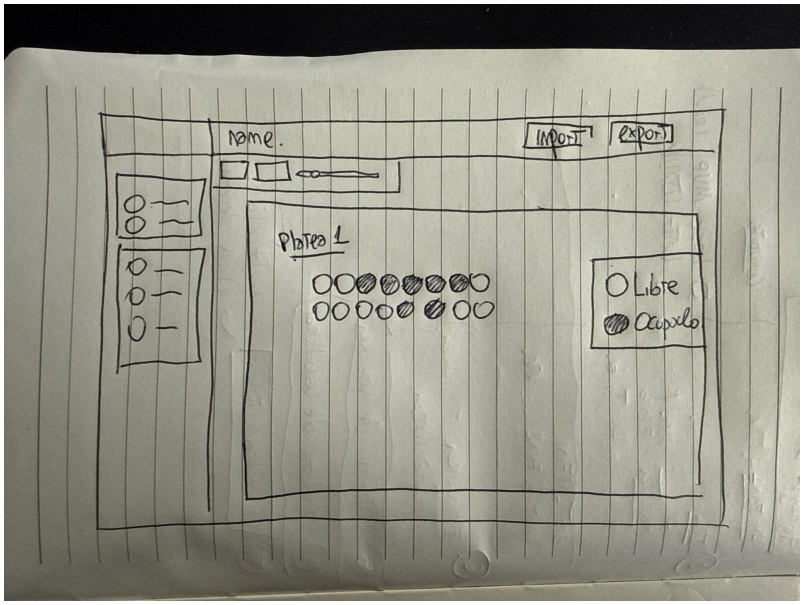
Por otro lado, lo que intuyo, que deben estar usando la API para crear estadios, plateas, asientos, etc; Además para gestionar las reservas y todo eso. Igualmente esta plataforma tiene un dashboard para crear eventos, estadios, etc; Para luego consumirlos directamente con su API. Me imagino que también pueden llegar a usarlo para integrar una pasarela de pagos de por medio.

En este caso decidí **no** usar dicha herramienta, dado que mi aplicación fue realizada sin ningún backend, tal como pedía la consigna. Y asumo que en un entorno real de trabajo, tendré que usarla, pero no sería problema aprenderla bien. Quise mostrar mis capacidades para armar algo muy similar visualmente, y más complejo a nivel de funcionalidades.

Cerca del final de la entrega descubrí que tenía mal el formato de los jsonl. Estaba agregando el output de la IA también.

Prototipado:

Antes de empezar a desarrollar la aplicación opté por hacer un prototipo físico, para tener una idea general de lo que quería lograr. Esto me sirvió mucho para no desviarme de los objetivos planteados, para solucionar el problema propuesto correctamente.



Versiones del MVP + decisiones de usabilidad:

En un principio desarrollé las **versiones 1.X** como una primera opción para resolver el problema. Sin embargo, cuando terminé con el desarrollo me di cuenta que podría extenderlo un poco más con features más interesantes. Por ello, continué desarrollando las **versiones 2.X** con features más interesantes y aplicables en un producto del estilo.

Las **versiones 1.X** se van a comentar a continuación en la sección de *"Primer acercamiento"*, mientras que las otras **versiones 2.X** se van a comentar en la parte de *"Segundo acercamiento".*

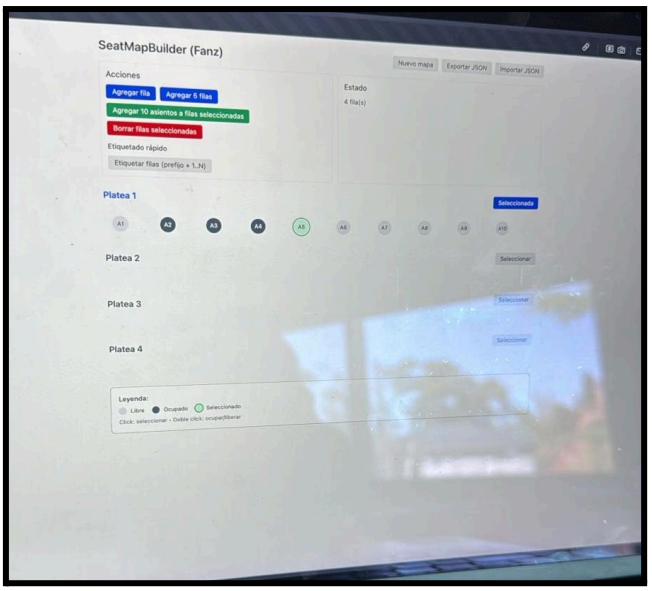
Para ambas series de versiones apliqué heurísticas de Nielsen como marco de referencia, ya que constituyen uno de los estándares más reconocidos en el campo de la usabilidad. La aplicación de estas heurísticas permitió evaluar de manera sistemática distintos aspectos de la experiencia de usuario, tales como la consistencia en la interfaz, la visibilidad del estado del sistema, la prevención de errores y la flexibilidad en el uso.

Este análisis no solo sirvió para identificar posibles problemas de diseño, sino también para proponer mejoras que aporten mayor claridad, eficiencia y satisfacción al usuario final. En este sentido, las heurísticas funcionaron como una guía práctica para orientar decisiones de interfaz y garantizar que las soluciones implementadas no solo cumplieran con criterios estéticos, sino que además respondieron a principios sólidos de usabilidad reconocidos internacionalmente.

Primer acercamiento.

Versión 1.0

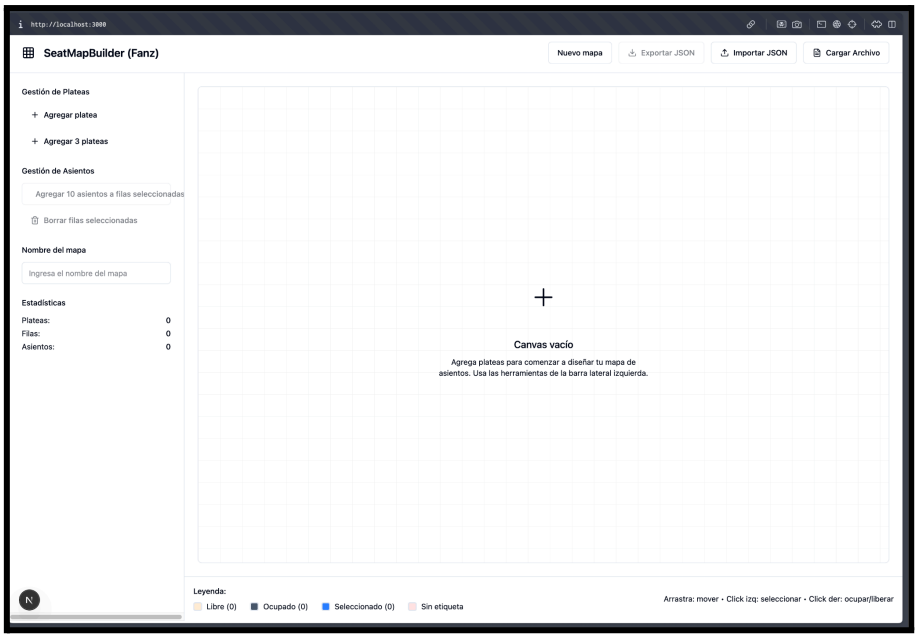
En un principio me enfoqué 100% en resolver el problema pedido de la forma más sencilla posible. Esta **versión 1.0** era muy rústica, bastante desorganizada, pero al final del día 100% funcional, cumpliendo el objetivo.



Versión 1.0

Versión 1.1

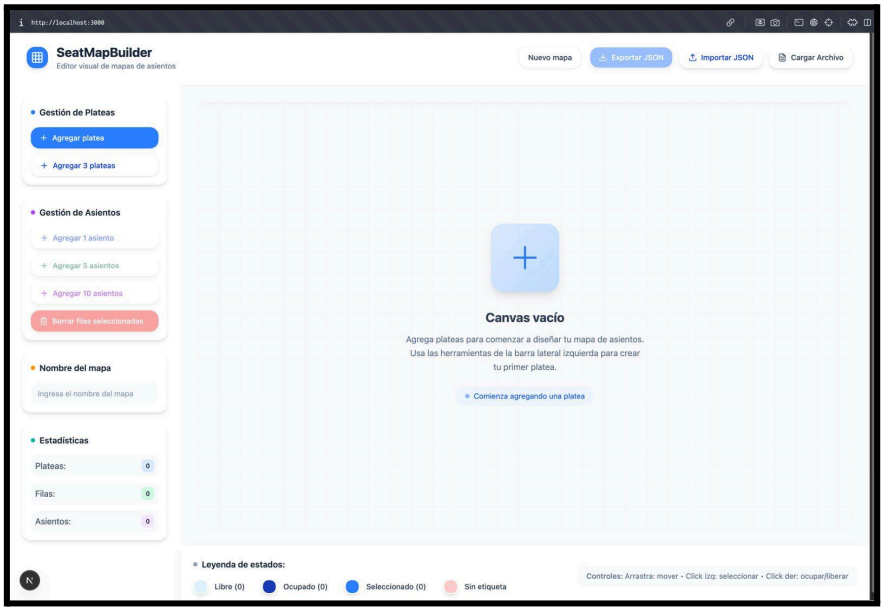
Esta otra versión también era bastante sencilla, pero podemos ver que cada sección está correctamente definida y delimitada, entre ellas, un canvas central, barra de herramientas en la parte superior, un panel lateral para manejar las filas y los asientos. También cuenta con funcionalidades mínimas pero esenciales para utilizarse. Podemos observar que no tiene mucho color, por lo que resulta complicado identificar los botones en la pantalla. Además no sigue ningún patrón de diseño similar a lo que es hoy en día la app web de fanz.



Versión 1.1

Versión 1.2

Esta anteúltima **versión 1.2** ya va tomando color, los botones también tomaron forma, y hay una clara división de jerarquías. La interfaz es mucho más linda visualmente. Sin embargo, quedan muchas cosas para mejorar.

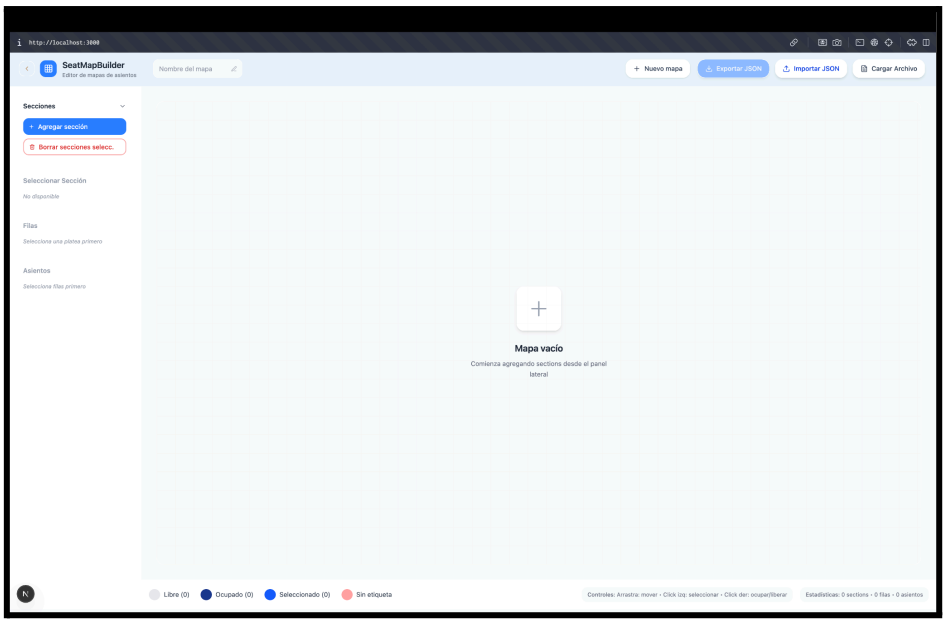


Versión 1.2

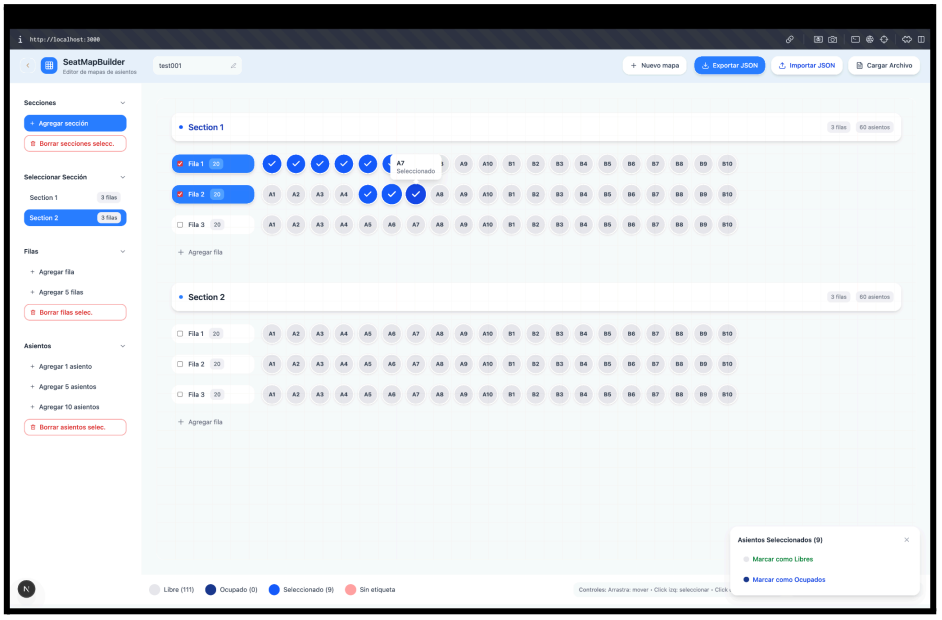
Versión 1.3

Finalmente, se propuso una última **versión 1.3** donde se tomaron dos enfoques principales, para su desarrollo. Estos enfoques son:

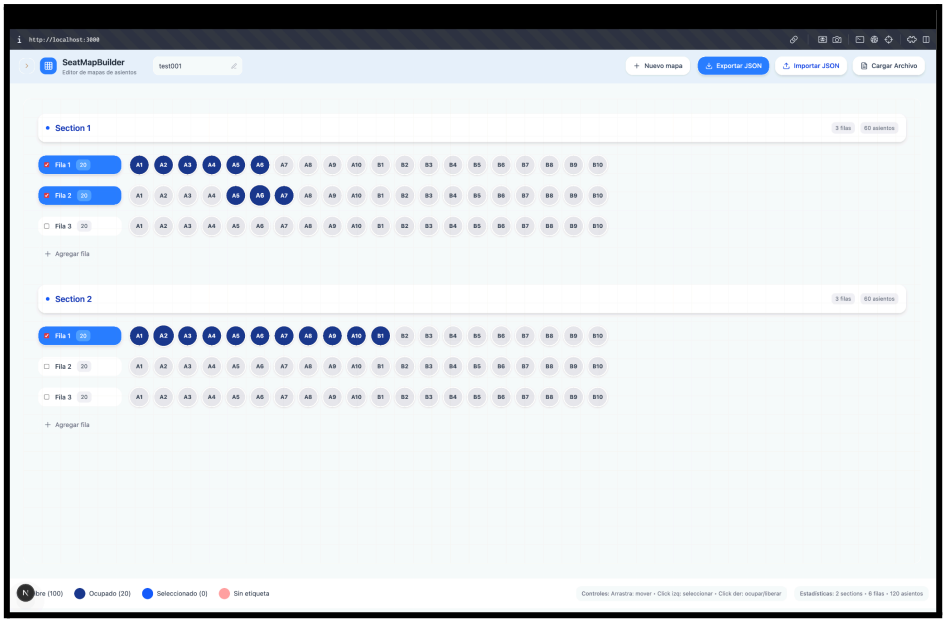
- Mejorar la usabilidad por parte del usuario
- Darle identidad a la aplicación



Versión 1.3 (canvas principal)



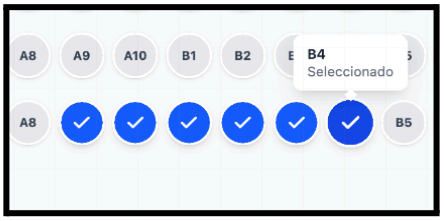
Versión 1.3 (selección de asientos)



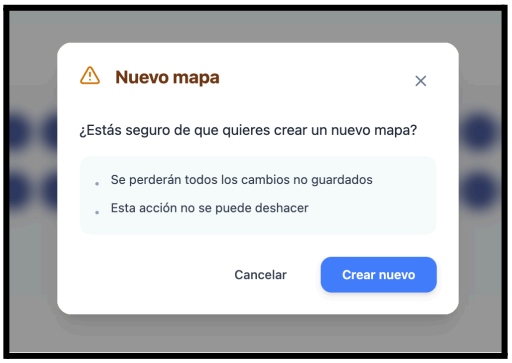
Versión 1.3 (asientos seleccionados)

Dentro de las mejoras de usabilidad en esta versión podemos destacar las siguientes features:

- **Box selection:** muy útil para seleccionar varios asientos y cambiarles el estado a todos juntos.

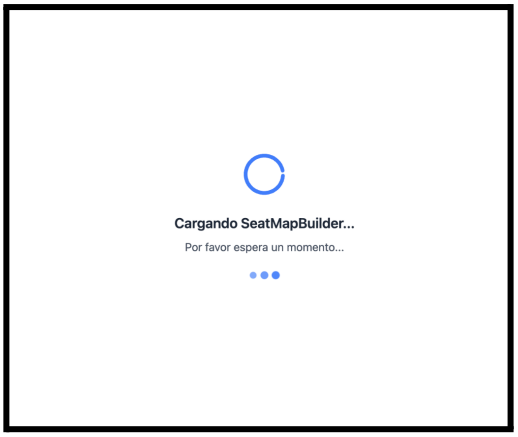


- **Atajos de teclado:** Se puede utilizar la combinación de Clic + cmd para seleccionar una hilera de asientos.
  Por ejemplo: Se hace clic en **A9,** luego se presiona la tecla cmd y se hace clic en el asiento **B4.** Eso generará una selección múltiple en la hilera



- **Confirmaciones de eliminación:** Otra cosa en la que también se hizo énfasis fue en la seguridad de la app para no perder accidentalmente los cambios realizados. Por ello se agregaron diálogos de eliminación. Hay confirmaciones para las eliminaciones de mapas, filas, secciones y asientos

- **Pantalla de carga:**



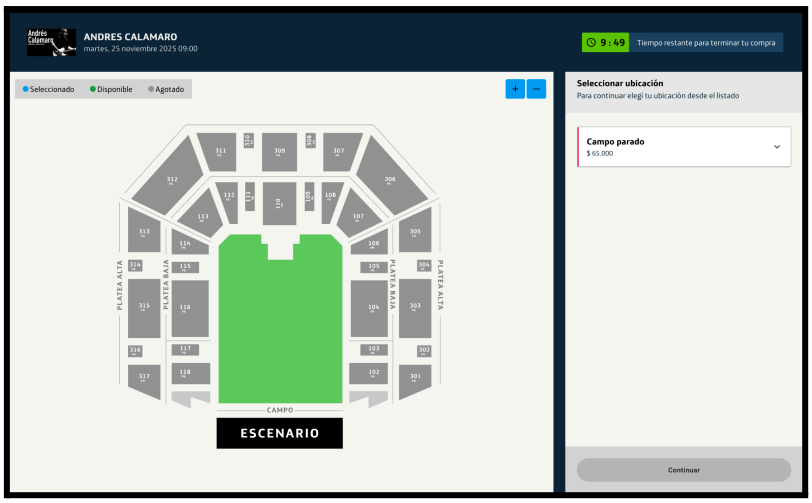
También vale la pena mencionar las siguientes funcionalidades:

- **Side panel colapsable**, para focalizar la atención en las secciones, filas y asientos
- **Oportunidad de importar JSON mediante copy & paste, asi tambien como directamente subir el archivo .json a la app**
- **Atajos de teclado interesantes**: como por ejemplo utilizar la tecla DELETE para eliminar la selección. Así también como **CMD + clic** para seleccionar varios asientos, varias filas, y secciones.

IMPORTANTE: Si se quiere ver esta versión en funcionamiento, se puede hacer viendo la branch 'feat/basicMVP' en el repositorio de github

Segundo acercamiento

Como se mencionó anteriormente, considero que el **Primer acercamiento** es un MVP muy sólido y robusto, para resolver el problema planteado en la consigna. Sin embargo consideré que estaría bueno agregar algunas funcionalidades, basándose en los widgets de seats.io ,y también en páginas de compra de tickets como la del Movistar Arena.



Ejemplo de la UI de la web de Movistar Arena

Me gustó mucho la idea de que se pueda crear no solo un mapa plano (sin sentido de ubicación dentro del propio edificio). Sino que también, se puede ver en qué parte se ubicaría el usuario, que tan cerca/lejos está del escenario, etc.

Por estos motivos se consideró el desarrollo de las versiones 2.X, para lograr un producto muy similar a lo que usan hoy en día gran parte de las páginas de reserva de asientos.

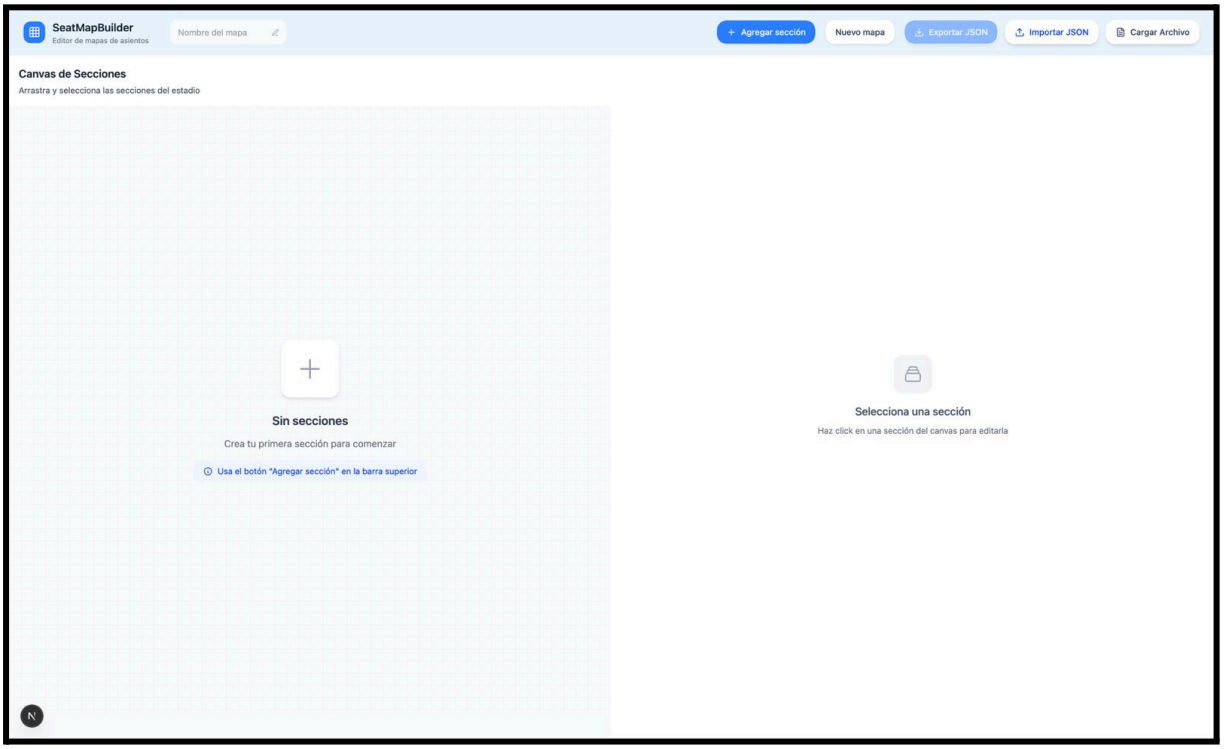
Disclaimer: Esta versión fué decidida para ser entregada.

IMPORTANTE: Si se quiere ver está versión en funcionamiento, se puede hacer viendo la branch 'main' en el repositorio de github

Versión 2.0

En está primera versión del segundo acercamiento, podemos ver una clara división de la pantalla principal en dos partes. A la izquierda se reservó espacio para el mapa “real” de secciones, mientras que a la derecha está la parte para seleccionar asientos.

Al igual que con las primeras versiones de la 1.X, se nota que tiene varias cosas para mejorar en la interfaz.

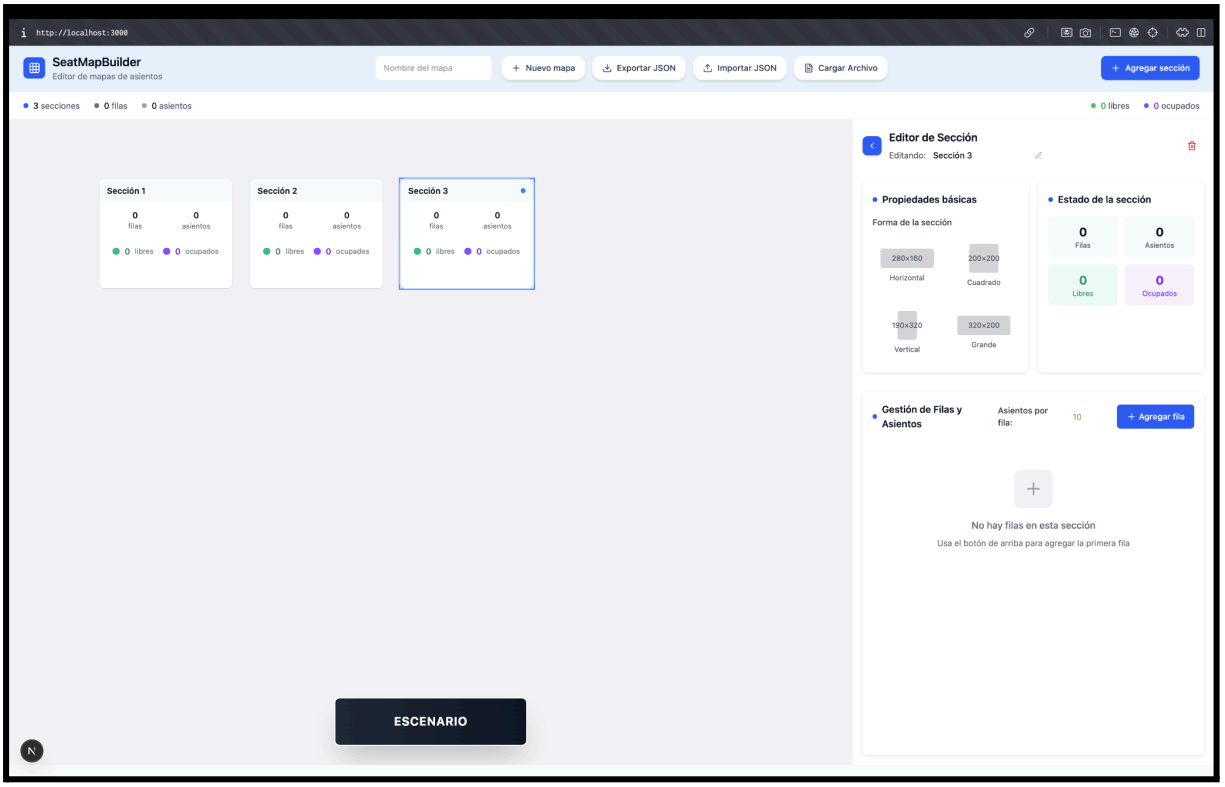


Versión 2.0

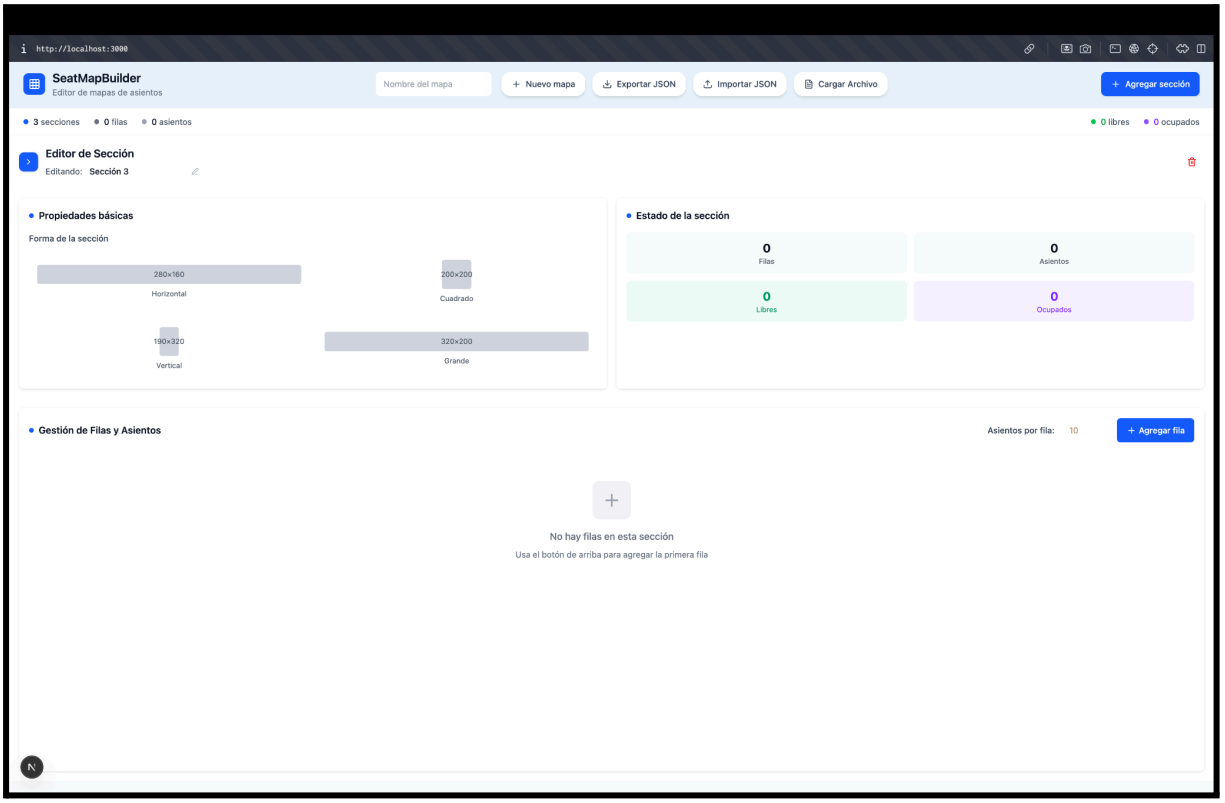
Versión 2.1

En está segunda versión se agregó la posibilidad de agregar secciones en el canvas con ubicación espacial (también un escenario), y con funcionalidad de drag&drop, para ubicarlas donde se necesite.

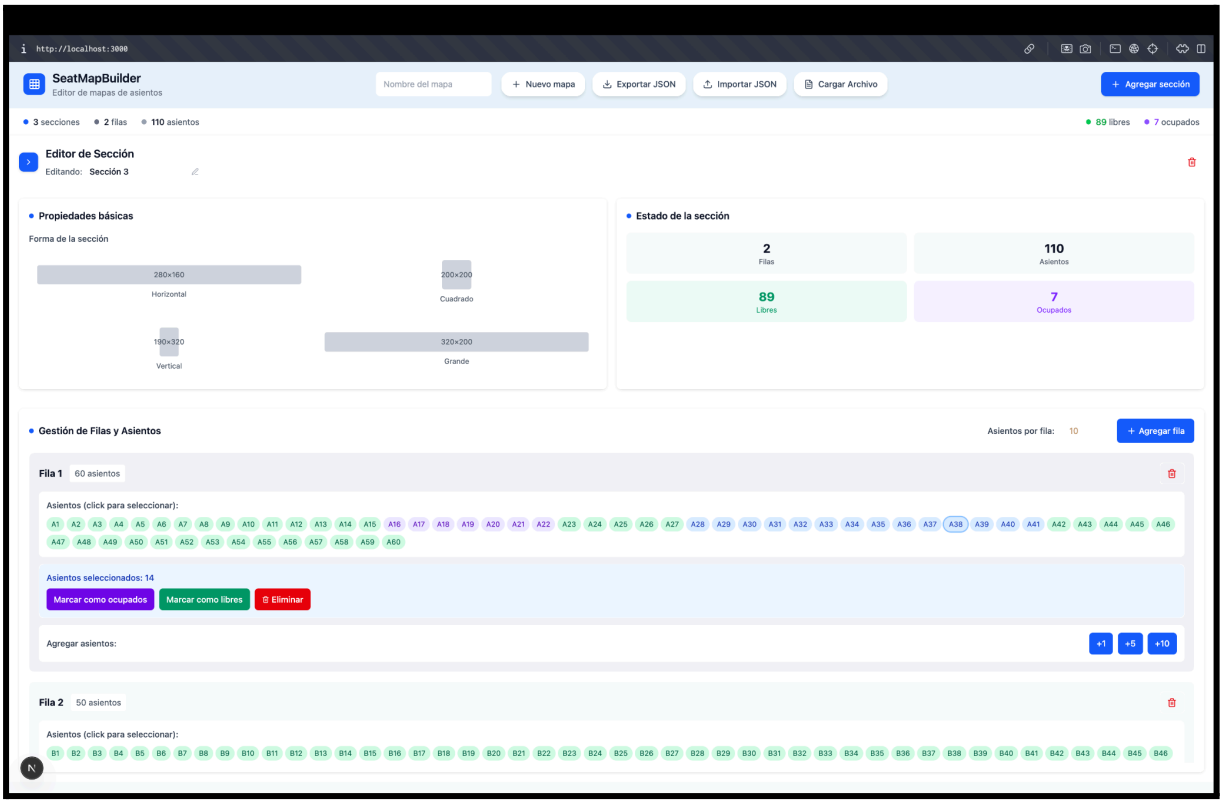
Al igual que las versiones anteriores, se hizo énfasis en la reserva de asientos, por lo tanto el panel izquierdo llamado “Editor de sección” es extensible hacia la izquierda para aprovechar toda la pantalla.



Versión 2.1



Versión 2.1 (panel izquierdo extendido)



Versión 2.2 (panel izquierdo extendido + reserva de asientos)

Dentro de las mejoras que se hicieron vale la pena que las funcionalidades de las versiones 1.X están también disponibles en estas versiones.

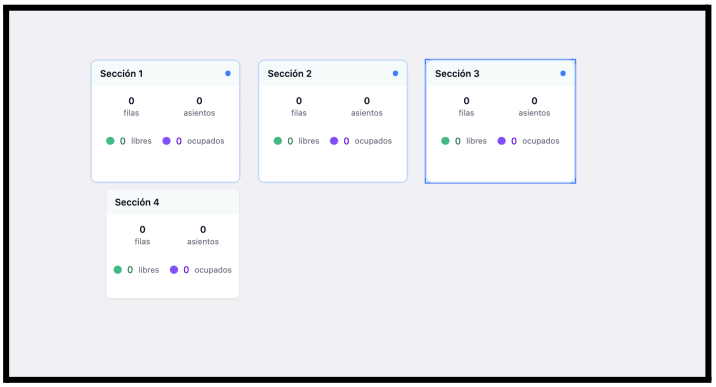
Procedo a comentar las features más relevantes:

- **Validación de mapas:** Para que la aplicación sea más consistente se seteo un formato .json espacial que se debe seguir en caso de querer crear secciones y asientos sin la necesidad de la interfaz.
  Se habla más sobre esto en la sección de “Decisiones de estructuras de datos + formatos”

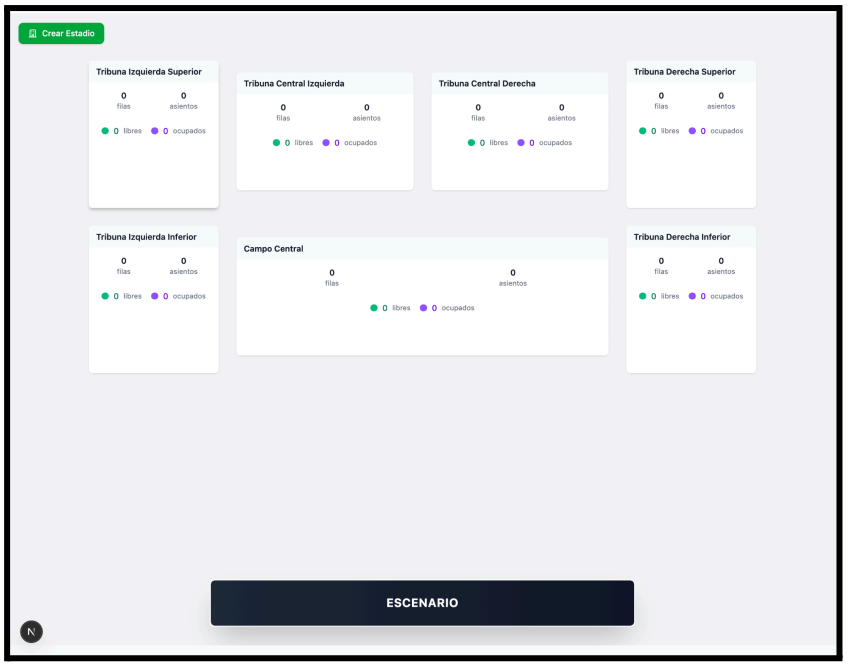
- **Editor de sección:** En esta parte del panel, se pueden revisar/hacer las siguientes cosas:
  - Elegir forma y tamaño de la sección (beta)

- Revisar stats de la propia sección
- Renombrar/eliminar la sección

- **Selección múltiple de secciones:**



- **Generador de estadio (prediseñado):** También consideré útil agregar una función que cree un estadio modelo, para poder ver cómo se vería en un caso real

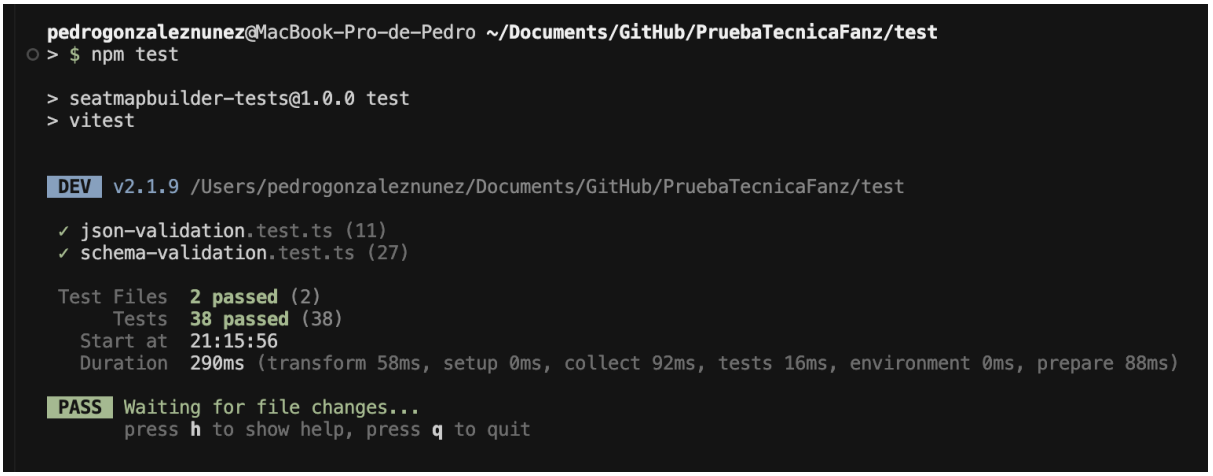


Testing:

También valoré el hecho de hacer tests especialmente para la validación del formato de archivos JSON.
Dentro de los tests que implementé, destaco los siguientes:

1. JSON Validation Tests (11 tests)
   - Validación de archivos JSON completos
   - Estructura de datos válida
   - Rechazo de archivos con errores
   - Casos límite y archivos vacíos
   - Integridad de datos después de validación

2. Schema Validation Tests (27 tests)
   - Validación de esquemas individuales (Seat, Row, Platea, SeatMap)
   - Tipos de datos correctos
   - Rechazo de valores inválidos
   - Campos opcionales y valores por defecto
   - Seguridad de tipos TypeScript
   - Fixtures de prueba:
   - valid-seatmap.json: Teatro completo con 2 plateas, 6 asientos
   - invalid-seatmap.json: Errores intencionales de validación
   - edge-cases.json: Casos límite y especiales

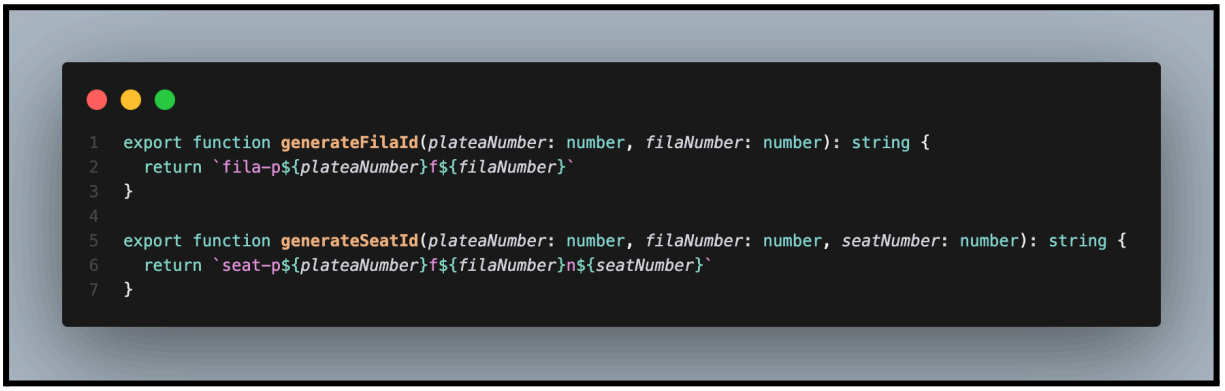


Decisiones de estructuras de datos + formatos:

Ahora veamos un ejemplo del formato del json que se va a exportar:

```json
{
  "name": "Movistar Arena",
  "plateas": [
    {
      "id": "platea-p1",
      "label": "Tribuna Izquierda Superior",
      "x": 135,
      "y": 80,
      "width": 220,
      "height": 250,
      "rows": [
        {
          "id": "fila-p1f1",
          "label": "Fila 1",
          "seats": [
            {
              "id": "seat-p1f1n1",
              "x": 0,
              "y": 0,
              "label": "A1",
              "status": "available",
              "meta": {}
            },
            {
              "id": "seat-p1f1n2",
              "x": 30,
              "y": 0,
```

También se implementaron funcionalidades para crear los IDs de los asientos, filas, etc.

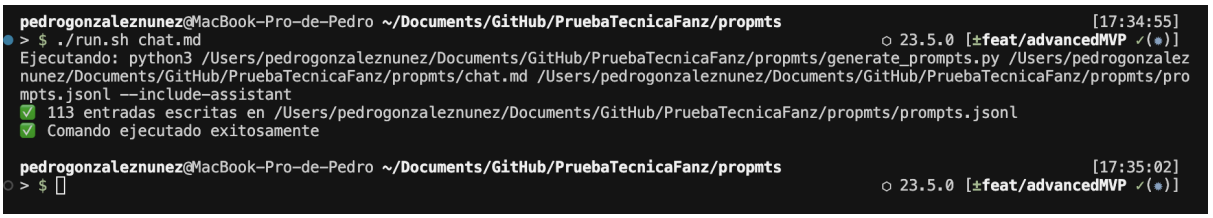


```ts
export function generateFilaId(plateaNumber: number, filaNumber: number): string {
  return `fila-p${plateaNumber}f${filaNumber}`
}

export function generateSeatId(plateaNumber: number, filaNumber: number, seatNumber: number): string {
  return `seat-p${plateaNumber}f${filaNumber}n${seatNumber}`
}
```

Dentro del directorio de extra/prompts en el repositorio de github, se creó un script run.sh para crear parsear los prompts desde .md al formato .jsonl con timestamps, models, etc Para ello se usa como base un código en python, que lo que hace es facilitar aún más el parseo. Para lograr el formato de los prompts en jsonl

Ejemplo:



Organización de componentes

Los estilos se pueden personalizar modificando:

* `app/globals.css` \- Estilos globales
* `tailwind.config.js` \- Configuración de Tailwind CSS

Componentes

Los componentes están organizados por funcionalidad:

* **`components/ui/`** - Componentes de UI genéricos (camelCase) que siguen el patrón de diseño de shadcn/ui
* **`components/seat/`** - Componentes específicos para gestión de asientos (PascalCase)
* **`components/section/`** - Componentes específicos para gestión de secciones

Posibles mejoras futuras

En un futuro se podrían hacer mejoras en la aplicación. Dentro de todas las que hay, destaco las más interesantes:

- Hacer que la app sea responsive, y compatible con todos los dispositivos.
- Soporte multi-evento, para crear varios mapas y estadios.
- Exportación de data más compleja.
- Gestión limitada de estados de asientos → por ejemplo, por ahora solo considero available y reserved, pero no casos como blocked, disabled, VIP.
- Features para dibujar secciones, así como un tablero de miro.
- Niveles o mejor dicho “layers” para los estadios, asi se pueden superponer secciones una encima de otra
- Funcionalidades como desplazarse visualmente en el canvas, zoom in, zoom out, etc

Limitaciones encontradas

- Escalabilidad reducida → la app funciona bien en un caso pequeño, pero no está optimizada para estadios enormes (miles de asientos).
- Sin backend real → no se pudo implementar persistencia de datos en una base de datos; todo quedó en memoria o exportado a JSON.
- Sin integración con Seats.io → los mapas y reservas no están sincronizados con un sistema externo.

Estas limitaciones no afectaron el cumplimiento de la consigna, pero en un entorno real serían necesarias de abordar para asegurar escalabilidad, accesibilidad y mantenibilidad del sistema.

Conclusión:

La verdad es que genuinamente disfruté realizar el desarrollo de este trabajo, sinceramente no me esperaba para nada algo así, y pareció original.

Más allá de lo técnico, me permitió explorar la creación de interfaces desde cero, trabajar con estructuras de datos en JSON y reflexionar sobre cómo se integraría una herramienta profesional como Seats.io en un entorno real.

También me sirvió mucho para hacer cosas que nunca había hecho, como por ejemplo, crear un registro de prompts en formato jsonl

Considero que fue una experiencia enriquecedora, tanto para practicar mis habilidades de desarrollo como para enfrentar decisiones de diseño y usabilidad. Sin dudas, me deja mejor preparado para proyectos futuros y con ganas de seguir mejorando este tipo de aplicaciones.

Gracias por el espacio
Pedro González Núñez