Juan Martín García Patiño

Desarrollo Prueba Técnica Ingeniero de Desarrollo Fullstack

Parte 1

Tecnologías utilizadas:

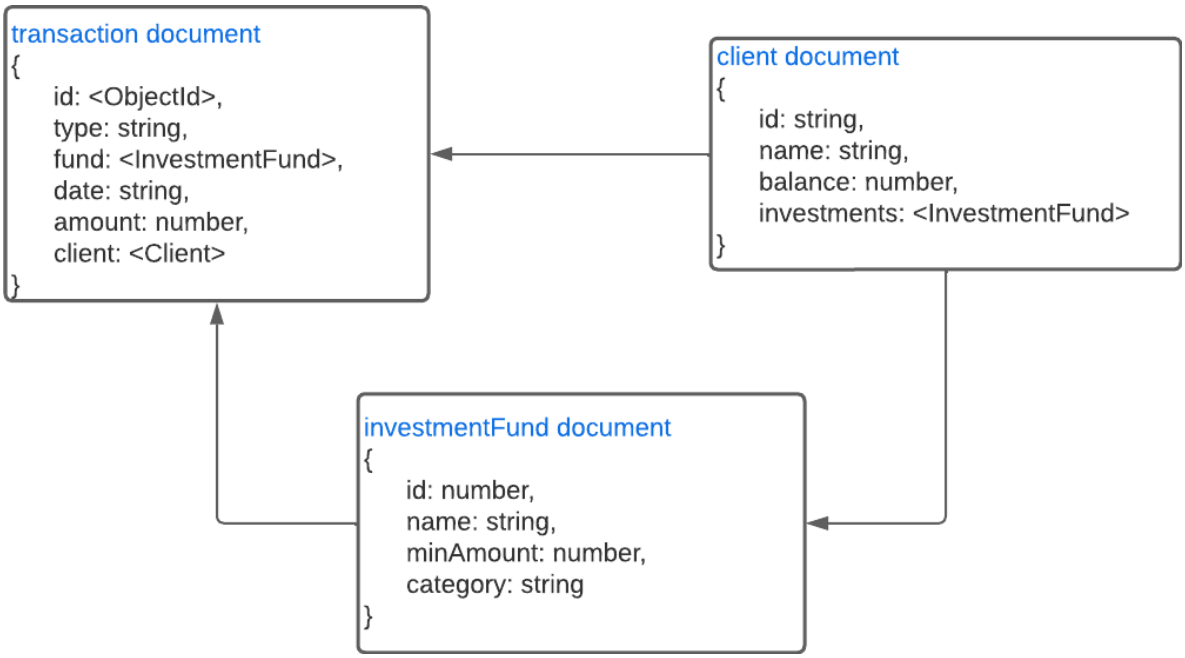
Para el desarrollo de la solución se utilizó **Node.js** en el Backend con **TypeScript** y el framework de **Express**, utilizando una arquitectura onion para garantizar un alto desacoplamiento con respecto a la base de datos y darle más importancia a la lógica de negocio, esto quiere decir que en el momento que se requiera migrar a cualquier base de datos se puede hacer mucho más rápido. Node.js, de la mano con TypeScript, permite crear APIs REST rápidamente gracias a las utilidades compatibles con este lenguaje, en mi caso utilicé **Inversify** para que mis servicios no dependieran de las clases directamente sino de las interfaces, cumpliendo así con el último principio SOLID: **Dependency Inversión**. Este servicio no requiere de mucha configuración y permite tener una estructura mucho más limpia y clara.

Por otra parte, el Frontend se desarrolló en **Angular 14** y **TypeScript**, con el apoyo de **Angular Material** y **Chart.js**, esta tecnología se eligió debido a su capacidad de escalabilidad y compatibilidad con diferentes tecnologías. Posee una estructura basada en componentes, lo que nos permite la reutilización de código y simplificación del desarrollo. Además, permite el desarrollo de **Single Page Application**, lo que facilita la transición a lo largo de la aplicación. En adición, el desarrollo fue a partir de módulos, lo que ayuda inmensamente la escalabilidad y en caso de querer agregar nuevos elementos se agregan como módulos nuevos y la aplicación seguirá funcionando sin problemas.

Finalmente, para el modelo de datos se optó por **MongoDB**, ya que en principio el problema no contiene muchas entidades y estas, a su vez, no son altamente dependientes. MongoDB permite un rendimiento mucho más alto que las bases de datos relacionales. Además, permite tener flexibilidad en los datos que se almacena por lo que no depende de esquemas como en SQL.

Juan Martín García Patiño

Modelo de datos NoSQL:



En realidad, sólo se crearon dos colecciones (transaction y client) mientras que investmentFund se almacena directamente en client como un arreglo. Los fondos disponibles presentados en el enunciado no los almaceno, sino que directamente lo uso en la aplicación como un arreglo constante.

Consideraciones:

- Un cliente no podrá suscribirse a un fondo más de una vez y en caso de querer cancelar una inversión de un fondo al que no está suscrito mostrará el mensaje de error correspondiente. Sin embargo, no se valida al crear un usuario con el mismo id.
- Hay un problema menor de estilos en la vista 'Estadistica' que en algunos casos la gráfica no se muestra y para que vuelva a aparecer es necesario refrescar o minimizar la pantalla. Este error no pudo ser solucionado por cuestiones de tiempo.
- Al invertir en un fondo aparece una ventana modal para ingresar el monto, en caso de querer cancelar sale un error como si se hubiera querido crear la transacción. Por cuestiones de tiempo no pudo ser solucionado.
- Se puede agregar más de un usuario, que se almacena en el localStorage. Sin embargo, en caso de no haber nada siempre utiliza el usuario 'Juan Martin Garcia' con id '1143873318', pero si este no está registrado en la base de datos no se podrá usar hasta crear uno nuevo.
- No se realizaron pruebas unitarias por falta de tiempo y por priorizar las funcionalidades principales de la aplicación. Es caso de haberlas hecho hubiera utilizado **Jest.**
- Para utilizar los endpoints directamente del backend se puede utilizar la siguiente colección de Postman: https://www.getpostman.com/collections/1cc160b34d9b090438c5 , donde la url base es 'http://localhost:3000/api'.
- Los pasos para la configuración del ambiente, instalación y ejecución se encuentran en el archivo 'README.md' del repositorio.

Juan Martín García Patiño

Parte 2

Para obtener los nombres de los clientes que tienen inscrito algún producto disponible sólo en las sucursales que visitan utilicé INNER JOIN de la siguiente forma:

```
SELECT
    DISTINCT "Cliente".nombre,
    "Cliente".apellidos
FROM "Cliente"
    INNER JOIN "Inscripcion" ON "Inscripcion"."idCliente" = "Cliente".id
    INNER JOIN "Producto" ON "Producto".id = "Inscripcion"."idProducto"
    INNER JOIN "Disponibilidad" ON "Disponibilidad"."idProducto" = "Producto".id
    INNER JOIN "Sucursal" ON "Sucursal".id = "Disponibilidad"."idSucursal"
    INNER JOIN "Visitan" ON "Visitan"."idSucursal" = "Sucursal".id
        AND "Visitan"."idCliente" = "Cliente".id;
```