Joan David Colina Echeverry
Documento de especificaciones técnicas de la Prueba Técnica

Consideraciones en el desarrollo de la solución

1. Para el desarrollo de la solución se consideró que un cliente no puede suscribirse a un mismo fondo dos veces. Por lo cual, un cliente puede pertenecer a muchos fondos, pero solo 1 vez por fondo.
2. Existe un bug en la interfaz gráfica, este consiste en que al ingresar a “HOME” el color del banner cambia a blanco. Esto ocurre por alguna interferencia entre el componente Paper de '@material-ui/core'; y los componentes AppBar, Container, Toolbar de "@mui/material". Este bug no pudo ser solucionado.
3. Es posible que en algunas ocasiones se deba ingresar nuevamente el valor a intertir dado que cuando el componente grafico se renderiza no cambia los valores del componente donde se inserta la inversión.
4. En los documentos README.dm se encuentran los pasos para ejecutar el código.
5. En la configuración de Cors del Back-End se dejó que cualquier origen pueda acceder a la API REST. Esto es una clara vulnerabilidad en la seguridad, pero para el contexto de desarrollo no presenta problemas.

```python
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=False,
    allow_methods=["*"],
    allow_headers=["*"],
    expose_headers=["acces-control-allow-methods"]
    )
```

6. Todos los servicios de la API REST están en un mismo documento .py. Esto podría mejorarse creando un módulo para cada tipo de servicio, en este caso sería bueno separar los servicios por:
    a. Servicios relacionados con Clientes
    b. Servicios relacionados con Fondos
    c. Servicios relacionados con Historia de transacciones
7. La ruta base utilizada en los servicios API REST no debe estar quemada en código, este valor debe obtenerse desde las variables de entorno.

```python
base_path= "/2FVC"
```

Joan David Colina Echeverry
Documento de especificaciones técnicas de la Prueba Técnica

Tecnologías para la solución completa de la necesidad.

A continuación, se presentan las tecnologías que yo utilizaría para solucionar, dado el contexto, la necesidad planteada en la prueba técnica. Las tecnologías están agrupadas por cada una de las capas mínima que debe tener una aplicación web.

Capa Front-End:

- Para el desarrollo básico de la estructura de la interfaz gráfica usaría ***React Hook*** por que permite evitar el uso de clases para el manejo del ciclo de vida del Front-End, permite manejar los estados fácilmente y es posible dedicar mas tiempo al desarrollo de la lógica y no a la configuración de las clases que soportan la lógica del Front-End.

- Aunque este desarrollo es relativamente pequeño y no tiene gran cantidad de estados almacenar, aun así, yo emplearía ***React Redux*** en conjunto con el estilo de trabajo ***Duck***. Dado que favorece al desacoplamiento del código, permite un manejo estándar de la información altamente cambiante e importante de la aplicación. Sin mencionar que favorece la escalabilidad, el entendimiento y mantenimiento del código.

- Con el objetivo de brindar una mejor experiencia de usuario y hacer una interfaz más amigable, yo utilizaría ***Material UI***. Ya que es una tecnología de Google, es de rápido acceso, tiene compatibilidad con la mayoría de los navegadores y es intuitiva.

- Para brindar una capa más de seguridad, yo utilizaría ***Express*** como middleware, puesto que, entre otras cosas, permite ocultas las peticiones al Back-End.

Capa Back-End:

- Yo utilizaría ***Python*** para el desarrollo del back porque es un lenguaje de uso universal y código abierto, tiene compatibilidad con casi todas las tecnologías, no es tipado por lo que es intuitivo. Y específicamente para esta solución porque se pueden realizar API REST de forma sencilla.
- Yo utilizaría la ***FastAPI*** para el desarrollo del API REST porque esta tecnología brinda documentación automática, es altamente configurable, es corto dado que tiene configuraciones automáticas y es muy intuitivo.
- Dado que con ***Pytest*** es más fácil escribir las pruebas y permite ejecutar solo un grupo de pruebas deseado, yo lo utilizaría, en conjunto con ***mocks***, para realizar las pruebas unitarias del código.
- Yo utilizaría ***Pymongo*** para la conexión a la base de datos en MongoDB

Capa de Datos:

- Considerando que el modelo de datos para esta solución es bastante simple, y que las peticiones que se realizarán no son complejas, yo utilizaría MongoDB porque tiene un alto performance y tiene compatibilidad con Python

Otras tecnologías de apoyo:

- Para él versiona miento del código y guardar una copia confiable del código fuente, yo emplearía GitHub como repositorio del código. (Aunque para esta prueba solo hay un desarrollador y no se aprovecha la posibilidad de que varias personas desarrollen a la vez, es importante utilizar un repositorio para guardar el código fuente en un lugar seguro externo al ambiente local del desarrollador)
- Para apoyar, entre toras cosas, la portabilidad, el rápido despliegue, el aislamiento del ambiente donde se ejecuta el código, yo usaría un contenedor ***Docker*** para albergar la interfaz gráfica.