

# Documentación:

## Guía técnica explicando:

- Estructura del proyecto.
- Flujo de datos.
- Configuración e integración del backend.
- Instrucciones para ejecutar la aplicación completa (frontend y backend).

## Estructura del proyecto

La estructura del proyecto se basó de manera pensada que se separa el Frontend del backend lo cual tomé la decisión de llamarle client-financial-reports

En la imagen se podrá encontrar el contenido donde dentro de src podemos hallar

src/

assets/ => Donde cree una carpeta llamada css

css/ => Para colocar las hojas de estilo que se ocupen en el sistema

tailwind.css => Archivo donde importe el manejo de tailwind

components/ => Aquí coloque los diferentes módulos que ocupe en el desarrollo del sistema

CategoryAnalysis.vue => Componente de gráfica de Análisis por categoría

CustomerSummary.vue => Tabla de sección de gráficas de Resumen por cliente

MainDashboard.vue => Componente del Dashboard / Plantilla

ThemeSwitcher.vue => Componente para toggle del tema del sitio

TransactionForm.vue => Componente de formulario

TransactionList.vue => Componente que muestra el listado de Transacciones

TransactionManager.vue => Componente principal main de la sección de Gestor de Transacciones

TransactionStates.vue => Tabla de sección de gráficas de Estados de transacciones

TypeAnalysis.vue => Componente de gráfica de Evolución temporal

ViewingMetrics.vue => Componente principal main de la sección de Visualización de métricas

Interfaces/

ITransaction.ts => Manejo de interfaces de las Transacciones

stores/

transactions.ts => Manejador de estado de las transacciones

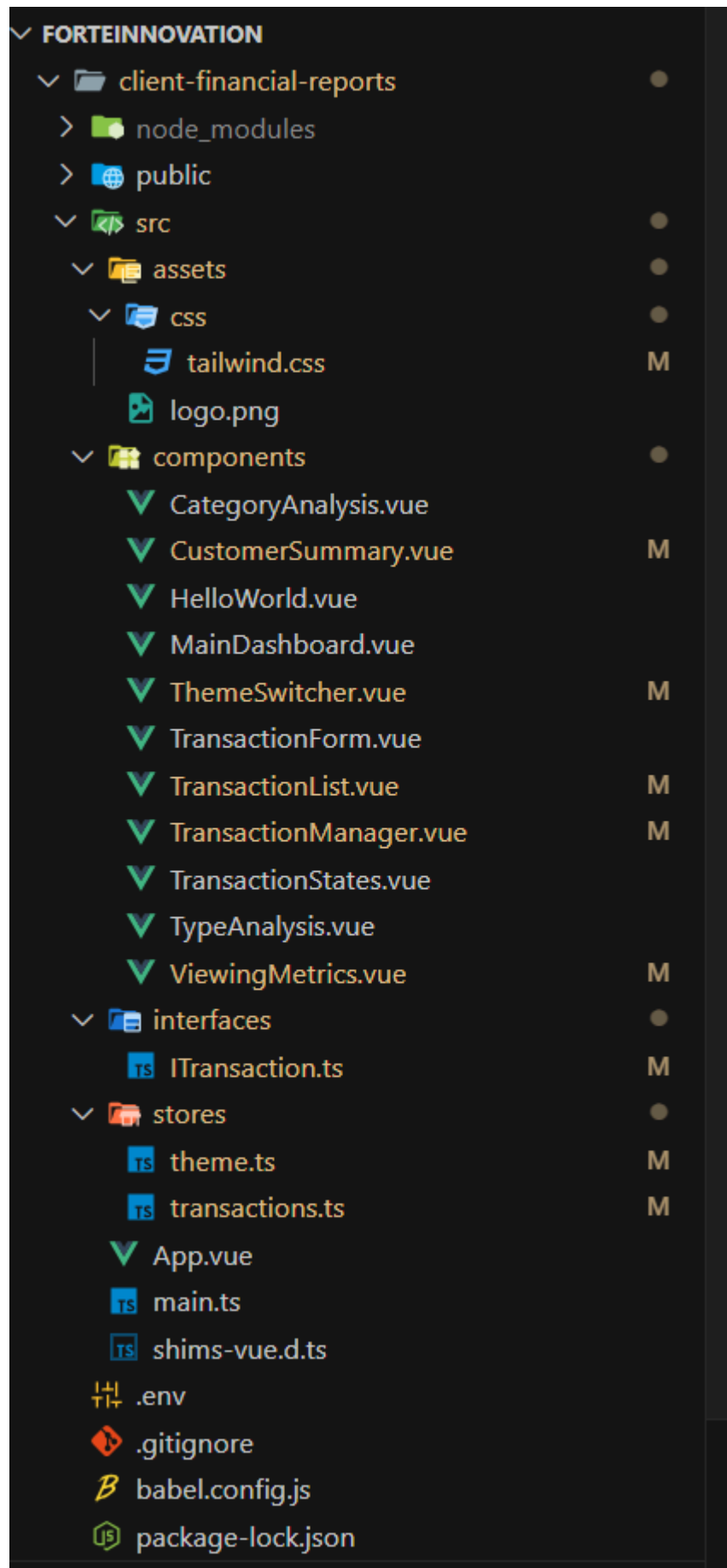
theme.ts => Manejador del estado del tema de la aplicación

App.vue => Renderización de la APP

main.ts => Coloque el tema para ser global

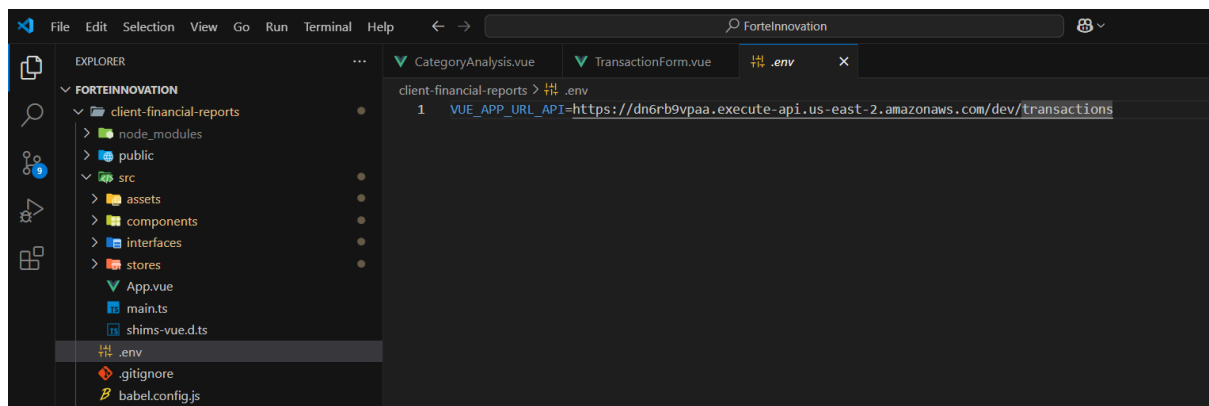
.env => Para variables de entorno aquí está la url del endpoint en la cual se conecta la aplicación

tailwind.config.js => la configuración para usar tailwind y parte del tema



## Flujo de datos / Configuración e integración del backend

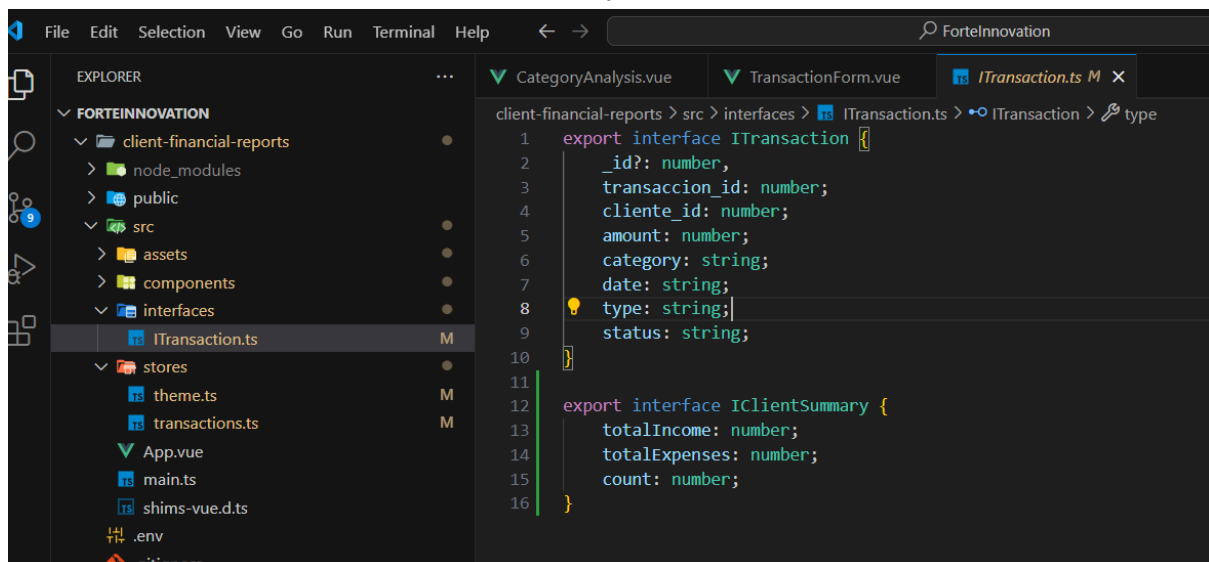
Cree un .env para colocar las variables de entorno en este caso ahí coloque mi conexión de la API que usa el sistema



The screenshot shows the Visual Studio Code interface with the Explorer panel on the left displaying the project structure. The file explorer shows a folder named 'client-financial-reports' containing subfolders like 'node\_modules', 'public', and 'src', along with files like 'App.vue', 'main.ts', 'shims-vue.d.ts', '.env', '.gitignore', and 'babel.config.js'. The main editor area shows the '.env' file with the following content:

```
1 VUE_APP_URL_API=https://dn6rb9vpaa.execute-api.us-east-2.amazonaws.com/dev/transactions
```

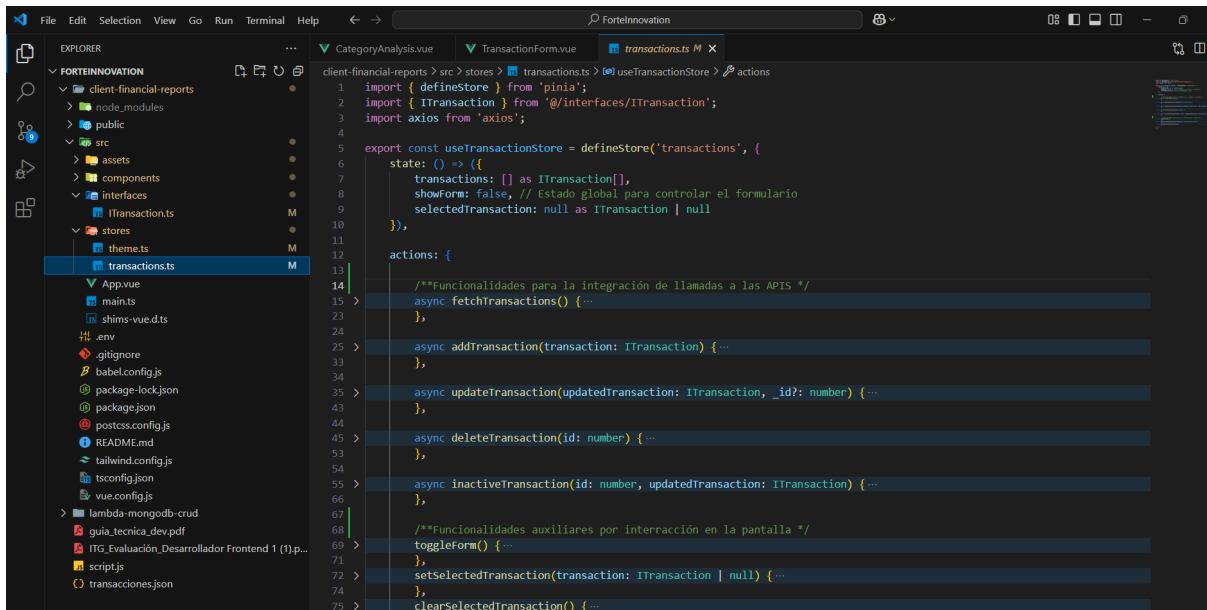
Creé e hice uso de interfaces ITransaction.ts donde coloque tanto interfaz de Transacciones como también el que hice uso para el summary de clientes.



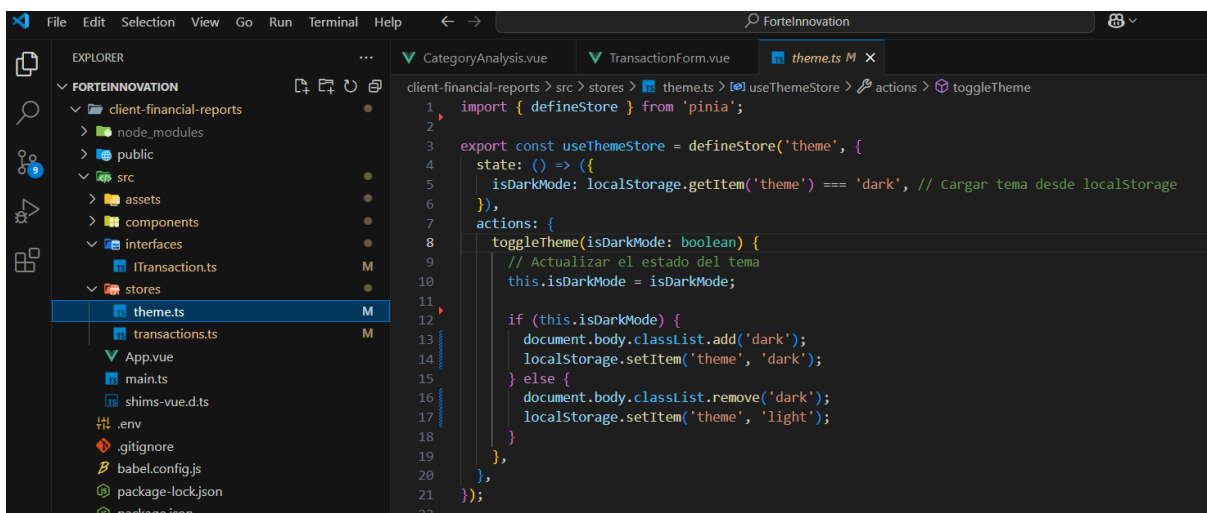
The screenshot shows the Visual Studio Code interface with the Explorer panel on the left displaying the project structure. The file explorer shows a folder named 'client-financial-reports' containing subfolders like 'node\_modules', 'public', and 'src', along with files like 'App.vue', 'main.ts', 'shims-vue.d.ts', '.env', '.gitignore', and 'babel.config.js'. The main editor area shows the 'ITransaction.ts' file with the following content:

```
1 export interface ITransaction {
2   _id?: number;
3   transaccion_id: number;
4   cliente_id: number;
5   amount: number;
6   category: string;
7   date: string;
8   type: string;
9   status: string;
10 }
11
12 export interface IClientSummary {
13   totalIncome: number;
14   totalExpenses: number;
15   count: number;
16 }
```

Para el flujo de datos manejo estos stores: transactions.ts => Este store me sirve para el control del estado de Transacciones donde manejo los datos creados, actualizados. Como también la integración de los endpoint dentro de actions. state => Control de estado de Transacciones actions => Integración de mis llamadas a las APIs

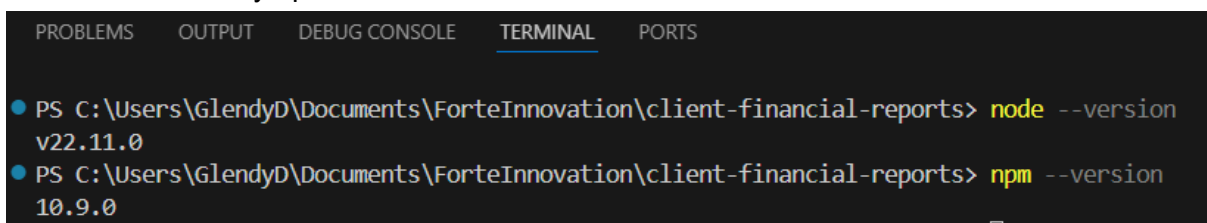


theme.ts => Para el control del estado de el tema a usar en el sistema



Instrucciones para ejecutar la aplicación completa (frontend y backend).

Versiones de node y npm



1. Instalar los node módulos
  - a. **npm install**
2. Correr el proyecto
  - i. Correr en entorno dev (desarrollo)
    1. **npm run serve**
  - ii. Correr en entorno producción

## 1. npm run build

Nota: Si adicional se requiere revisar el código se puede usar el siguiente comando **npm run lint** para checar la reglas de cumplimiento en código  
el npm install => Descargar los módulos para ejecución del proyecto  
el npm run serve o build => Ejecutará la aplicación

Información adicional con respecto al documento de **ITG\_Evaluación\_Desarrollador Frontend 1 (1).pdf**

### 2.Base de datos:

- Proveer un script para crear una base de datos de ejemplo en MongoDB : script.js
- Poblar datos de prueba con un archivo JSON: transacciones.json

### 2.Datos de Prueba:

- JSON para poblar MongoDB con transacciones de ejemplo : transacciones.json

### 3.Integración Backend:

- Script para poblar MongoDB con datos de prueba: script.js

## Ejecución del proyecto

1. Clonar el proyecto
  - a. <https://github.com/Glendy-Covarrubias/client-financial-reports.git>
2. Instalar las dependencias
  - a. **npm install**
3. Levantar el proyecto
  - a. Modo desarrollador
    - i. **npm run serve**
  - b. Modo producción
    - i. **npm run build**

Complementos del entregable en el siguiente repositorio

ITG\_Evaluación\_Desarrollador Frontend 1.pdf

[https://github.com/Glendy-Covarrubias/documents\\_client-financial-reports](https://github.com/Glendy-Covarrubias/documents_client-financial-reports)