

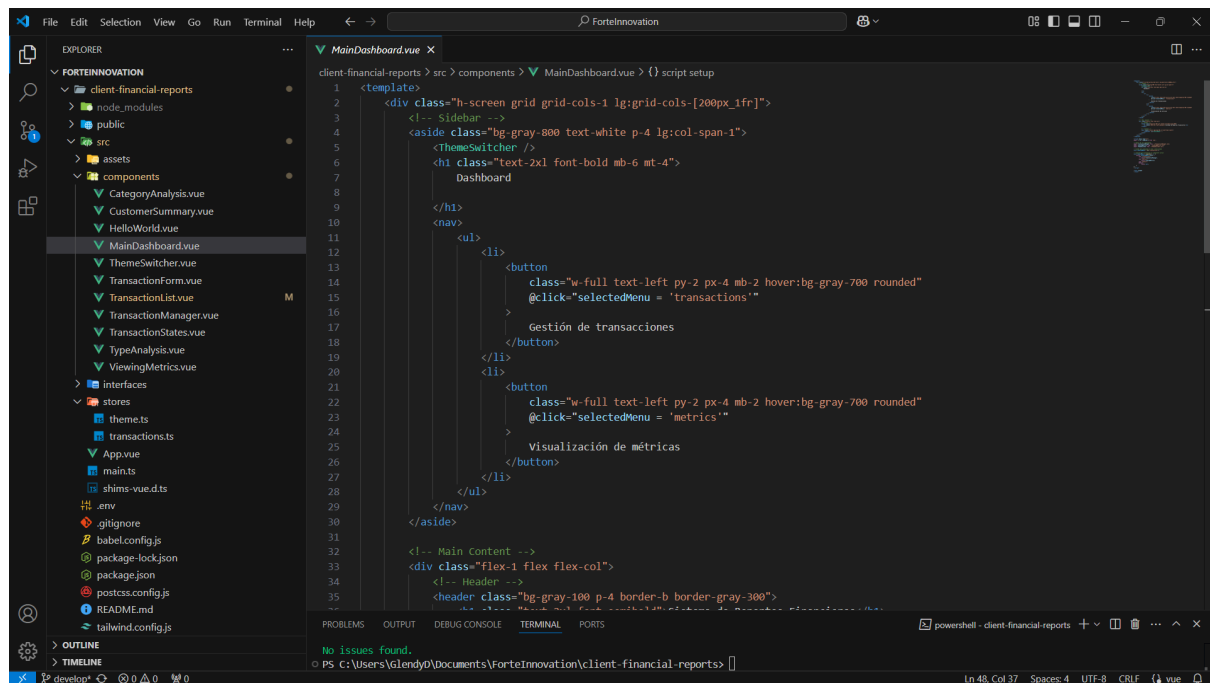
# Documentación:

## Frontend

- Crear una guía técnica para desarrolladores con detalles de:
  - Componentes creados.
  - Flujo de datos.
  - Pruebas unitarias e integración con el backend.

## Componentes creados

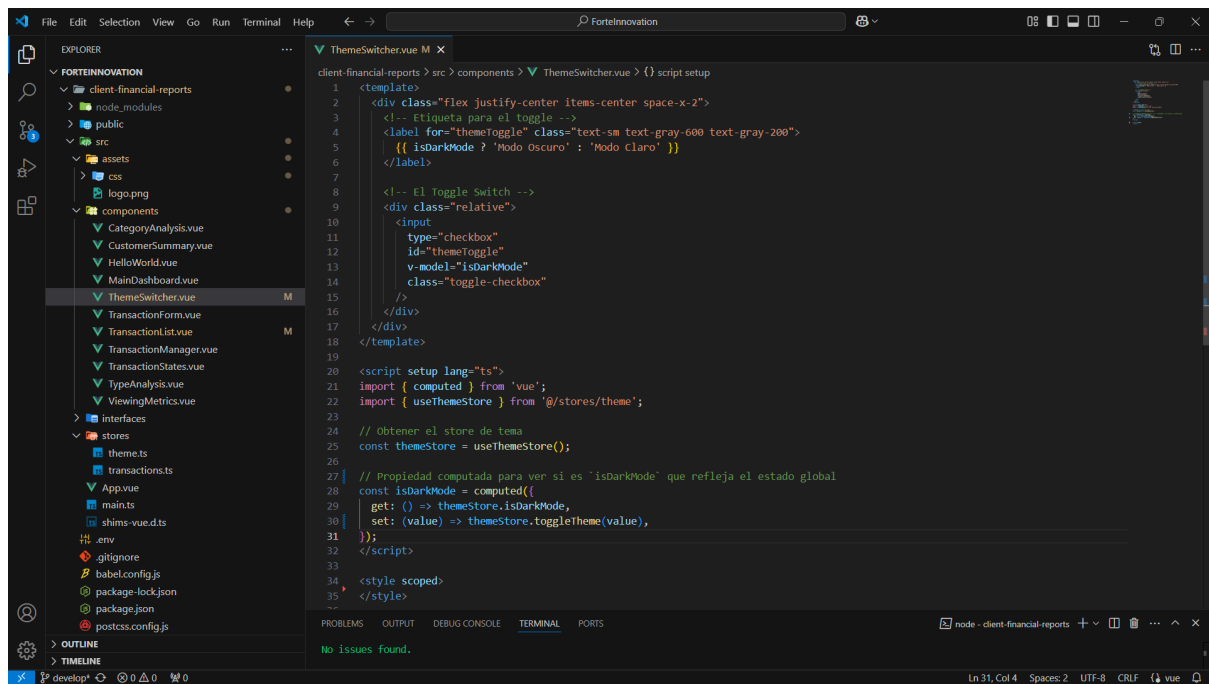
MainDashboard.vue => Componente principal para montar la plantilla del proyecto creado. Donde incluye el Menú lateral izquierdo y el contenido para mostrar el apartado correspondiente seleccionado.



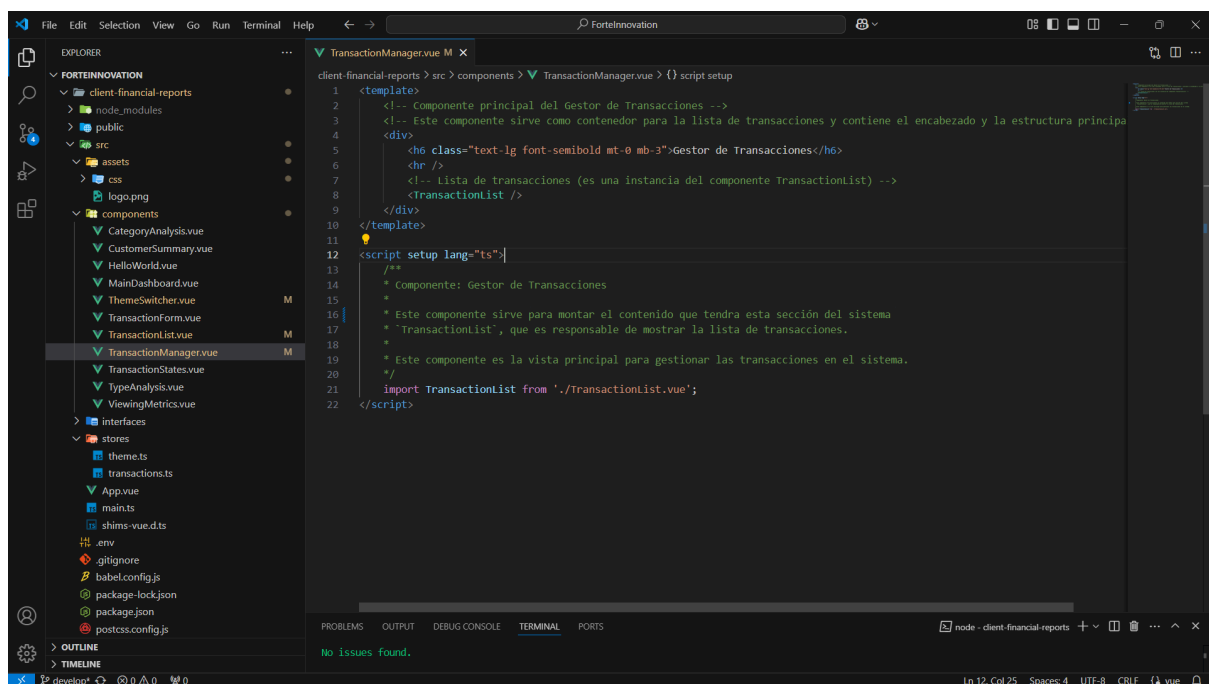
Dentro del MainDashboard.vue se encuentra otro componente ThemeSwitcher.vue.

ThemeSwitcher.vue => Se encarga de customizar el tema oscuro y claro.

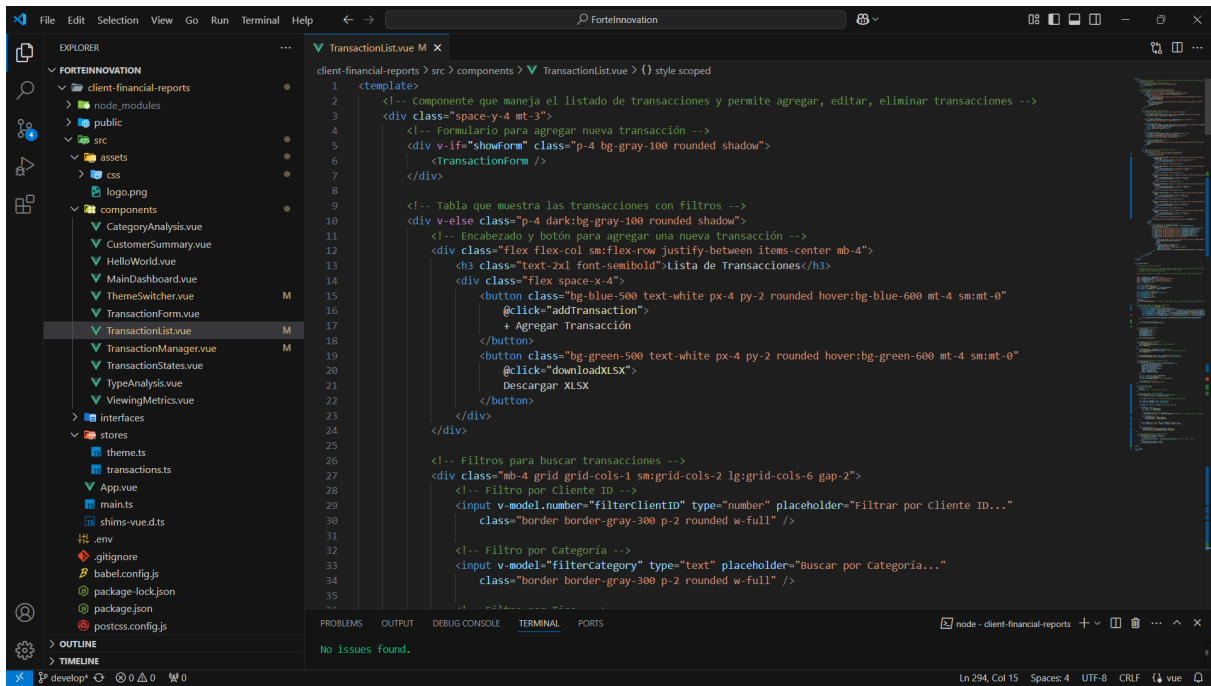
Nota: Esta parte solo lo integre en contenedor de la tabla de Lista de Transacciones



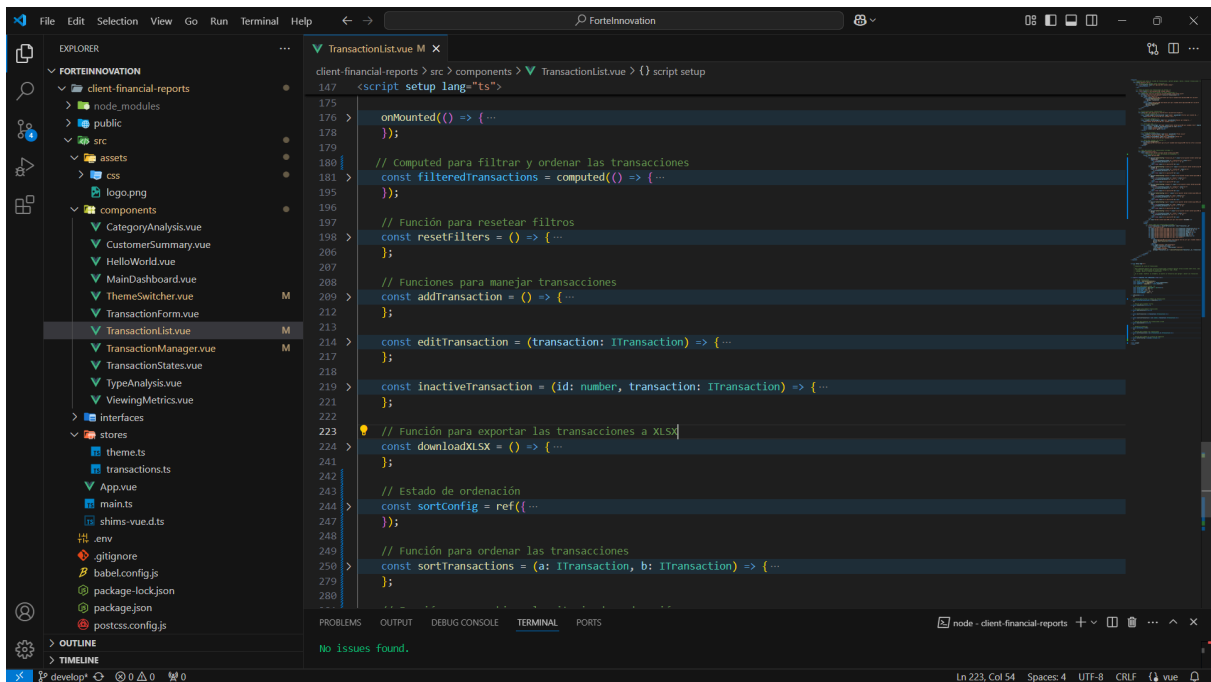
TransactionManager.vue => Es el componente principal para el módulo de Gestor de Transacciones donde se manda a llamar el componente que trae la tabla de movimientos. Es donde se va montar que contiene esa sección. En este caso solo monte una tabla de movimientos.



TransactionList.vue => Es el componente que maneja toda la interacción con la tabla de movimientos. Se podrá encontrar dentro de este las funcionalidades que llevan a agregar, editar, consultar, filtrar, limpiar, descargar excel.

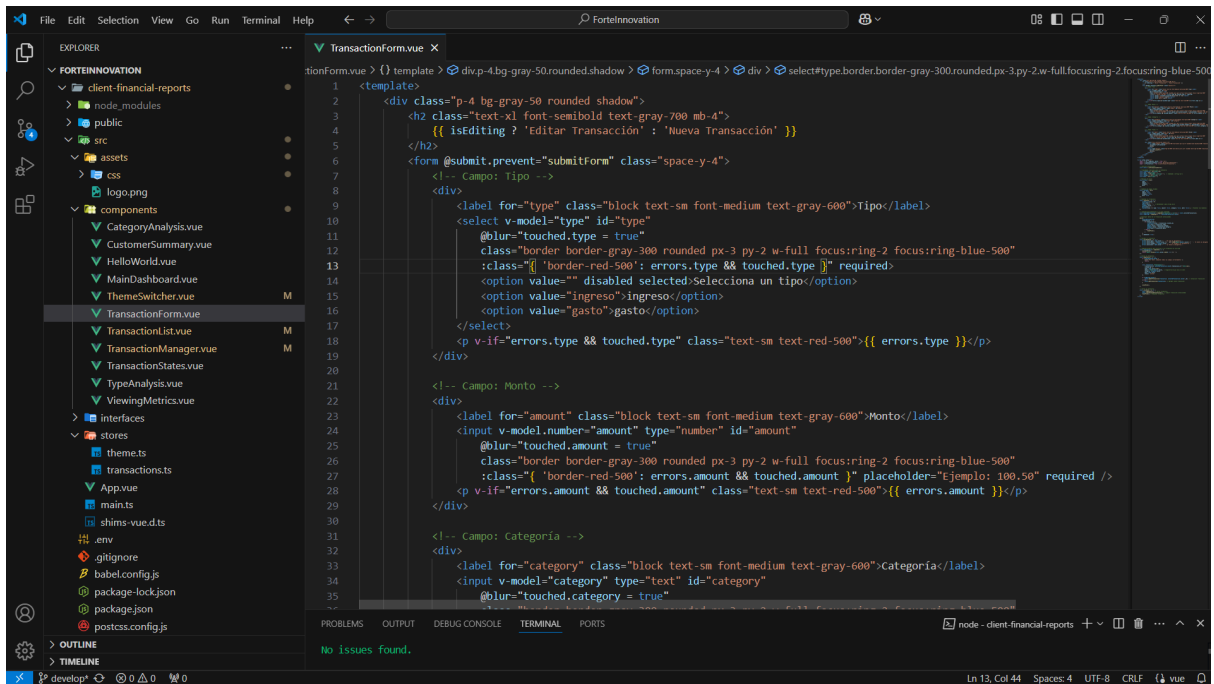


```
1 <template>
2   <!-- Componente que maneja el listado de transacciones y permite agregar, editar, eliminar transacciones -->
3   <div class="space-y-4 mt-3">
4     <!-- Formulario para agregar nueva transacción -->
5     <div v-if="showForm" class="p-4 bg-gray-100 rounded shadow">
6       <TransactionForm />
7     </div>
8
9     <!-- Tabla que muestra las transacciones con filtros -->
10    <div v-else class="p-4 dark:bg-gray-100 rounded shadow">
11      <!-- Encabezado y botón para agregar una nueva transacción -->
12      <div class="flex flex-col sm:flex-row justify-between items-center mb-4">
13        <h3 class="text-xl font-semibold">Lista de Transacciones</h3>
14        <div class="flex space-x-4">
15          <button class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600 mt-4 sm:mt-0"
16            @click="addTransaction">
17            + Agregar Transacción
18          </button>
19          <button class="bg-green-500 text-white px-4 py-2 rounded hover:bg-green-600 mt-4 sm:mt-0"
20            @click="downloadXLSX">
21            Descargar XLSX
22          </button>
23        </div>
24      </div>
25
26      <!-- Filtros para buscar transacciones -->
27      <div class="mb-4 grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-6 gap-2">
28        <!-- Filtro por Cliente ID -->
29        <input v-model="filterClientID" type="number" placeholder="Filtrar por Cliente ID..."
30          class="border border-gray-300 p-2 rounded w-full" />
31
32        <!-- Filtro por Categoría -->
33        <input v-model="filterCategory" type="text" placeholder="Buscar por Categoría..."
34          class="border border-gray-300 p-2 rounded w-full" />
35      </div>
36
37      <table>
38        <thead>
39          <tr>
40            <th>ID</th>
41            <th>Cliente</th>
42            <th>Categoría</th>
43            <th>Monto</th>
44            <th>Fecha</th>
45            <th>Acciones</th>
46          </tr>
47        </thead>
48        <tbody>
49          <tr>
50            <td>1</td>
51            <td>Juan Pérez</td>
52            <td>Alimentación</td>
53            <td>15000</td>
54            <td>2023-10-26</td>
55            <td><button @click="editTransaction(1)">Editar</button> <button @click="deleteTransaction(1)">Eliminar</button>
56          </tr>
57          <tr>
58            <td>2</td>
59            <td>María Gómez</td>
60            <td>Transporte</td>
61            <td>8000</td>
62            <td>2023-10-25</td>
63            <td><button @click="editTransaction(2)">Editar</button> <button @click="deleteTransaction(2)">Eliminar</button>
64          </tr>
65          <tr>
66            <td>3</td>
67            <td>Carlos Ruiz</td>
68            <td>Salud</td>
69            <td>12000</td>
70            <td>2023-10-24</td>
71            <td><button @click="editTransaction(3)">Editar</button> <button @click="deleteTransaction(3)">Eliminar</button>
72          </tr>
73          <tr>
74            <td>4</td>
75            <td>Ana López</td>
76            <td>Educación</td>
77            <td>9000</td>
78            <td>2023-10-23</td>
79            <td><button @click="editTransaction(4)">Editar</button> <button @click="deleteTransaction(4)">Eliminar</button>
80          </tr>
81          <tr>
82            <td>5</td>
83            <td>Diego Martín</td>
84            <td>Ocio</td>
85            <td>7000</td>
86            <td>2023-10-22</td>
87            <td><button @click="editTransaction(5)">Editar</button> <button @click="deleteTransaction(5)">Eliminar</button>
88          </tr>
89          <tr>
90            <td>6</td>
91            <td>Sofía Vargas</td>
92            <td>Alimentación</td>
93            <td>11000</td>
94            <td>2023-10-21</td>
95            <td><button @click="editTransaction(6)">Editar</button> <button @click="deleteTransaction(6)">Eliminar</button>
96          </tr>
97          <tr>
98            <td>7</td>
99            <td>Luis Flores</td>
100           <td>Transporte</td>
101           <td>6000</td>
102           <td>2023-10-20</td>
103           <td><button @click="editTransaction(7)">Editar</button> <button @click="deleteTransaction(7)">Eliminar</button>
104         </tbody>
105       </table>
106     </div>
107   </div>
108 </template>
```

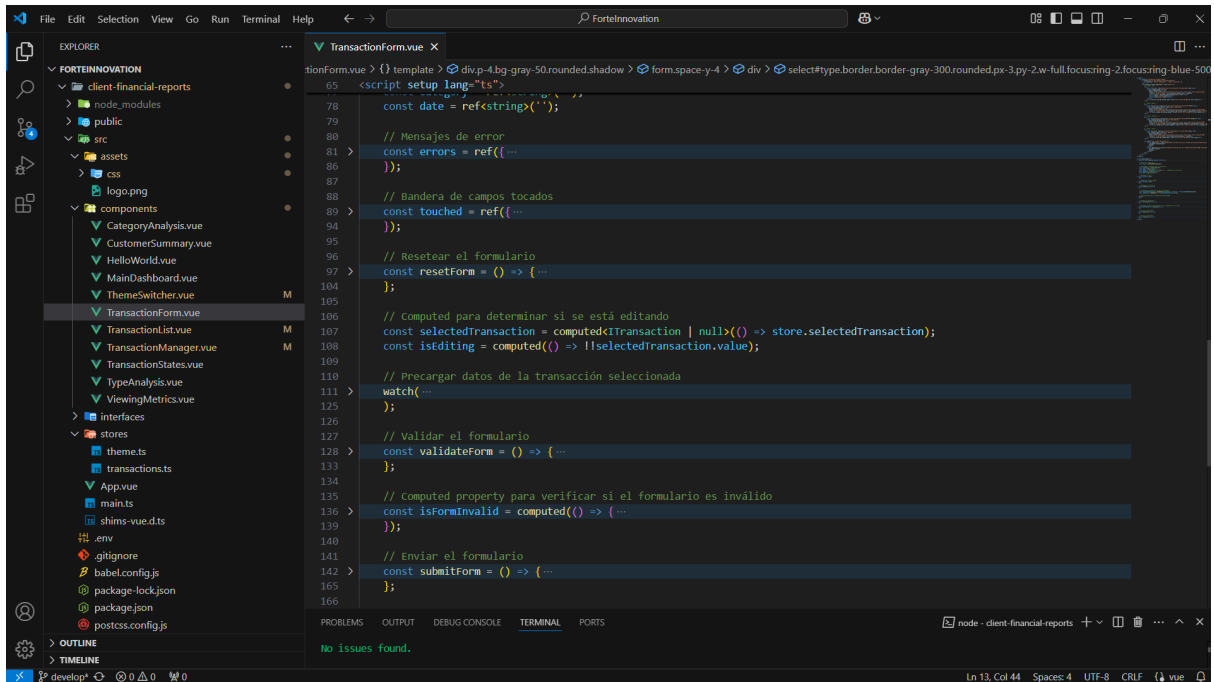


```
147 <script setup lang="ts">
148
149   // Lifecycle hooks
150   onMounted(() => {
151     // Inicialización de datos y filtros
152   });
153
154   // Computed para filtrar y ordenar las transacciones
155   const filteredTransactions = computed(() => {
156     // Lógica de filtrado y ordenamiento
157   });
158
159   // Función para resetear filtros
160   const resetFilters = () => {
161     // Lógica para resetear los filtros
162   };
163
164   // Funciones para manejar transacciones
165   const addTransaction = () => {
166     // Lógica para agregar una nueva transacción
167   };
168
169   const editTransaction = (transaction: ITransaction) => {
170     // Lógica para editar una transacción existente
171   };
172
173   const deleteTransaction = (id: number) => {
174     // Lógica para eliminar una transacción
175   };
176
177   // Función para exportar las transacciones a XLSX
178   const downloadXLSX = () => {
179     // Lógica para generar y descargar el archivo XLSX
180   };
181
182   // Estado de ordenación
183   const sortConfig = ref({
184     // Configuración inicial de ordenamiento
185   });
186
187   // Función para ordenar las transacciones
188   const sortTransactions = (a: ITransaction, b: ITransaction) => {
189     // Lógica para comparar y ordenar las transacciones
190   };
191 </script>
```

TransactionForm.vue => Este componente es dedicado al formulario reutilizable tanto para crear como para editar. Aquí se maneja los errores del formulario y toda la funcionalidad de recuperar la información capturada para la creación o en su caso si es edición setear los valores anteriores para mostrar el llenado del formulario.



```
1 <template>
2   <div class="p-4 bg-gray-50 rounded shadow">
3     <h2 class="text-xl font-semibold text-gray-700 mb-4">
4       {{ isEditing ? 'Editar Transacción' : 'Nueva Transacción' }}
5     </h2>
6     <form @submit.prevent="submitForm" class="space-y-4">
7       <!-- Campo: Tipo -->
8       <div>
9         <label for="type" class="block text-sm font-medium text-gray-600">Tipo</label>
10        <select v-model="type" id="type"
11          @blur="touched.type = true"
12          class="border border-gray-300 rounded px-3 py-2 w-full focus:ring-2 focus:ring-blue-500"
13          :class="{ 'border-red-500': errors.type && touched.type }" required>
14          <option value="" disabled selected>Selecciona un tipo</option>
15          <option value="ingreso">ingreso</option>
16          <option value="gasto">gasto</option>
17        </select>
18        <p v-if="errors.type && touched.type" class="text-sm text-red-500">{{ errors.type }}</p>
19      </div>
20      <!-- Campo: Monto -->
21      <div>
22        <label for="amount" class="block text-sm font-medium text-gray-600">Monto</label>
23        <input v-model.number="amount" type="number" id="amount"
24          @blur="touched.amount = true"
25          class="border border-gray-300 rounded px-3 py-2 w-full focus:ring-2 focus:ring-blue-500"
26          :class="{ 'border-red-500': errors.amount && touched.amount }" placeholder="Ejemplo: 100.50" required />
27        <p v-if="errors.amount && touched.amount" class="text-sm text-red-500">{{ errors.amount }}</p>
28      </div>
29      <!-- Campo: Categoría -->
30      <div>
31        <label for="category" class="block text-sm font-medium text-gray-600">Categoría</label>
32        <input v-model="category" type="text" id="category"
33          @blur="touched.category = true"
34        </input>
35      </div>
36    </form>
37  </div>
38</template>
```



```
65 <script setup lang="ts">
66   const date = ref<string>("");
67   // Mensajes de error
68   const errors = ref({});
69   // Bandera de campos tocados
70   const touched = ref({});
71   // Resetear el formulario
72   const resetForm = () => { ... };
73   // Computed para determinar si se está editando
74   const selectedTransaction = computed<Transaction | null>(() => store.selectedTransaction);
75   const isEditing = computed(() => !!selectedTransaction.value);
76   // Precargar datos de la transacción seleccionada
77   watch(selectedTransaction, () => { ... });
78   // Validar el formulario
79   const validateForm = () => { ... };
80   // Computed property para verificar si el formulario es inválido
81   const isFormInvalid = computed(() => { ... });
82   // Enviar el formulario
83   const submitForm = () => { ... };
84</script>
```

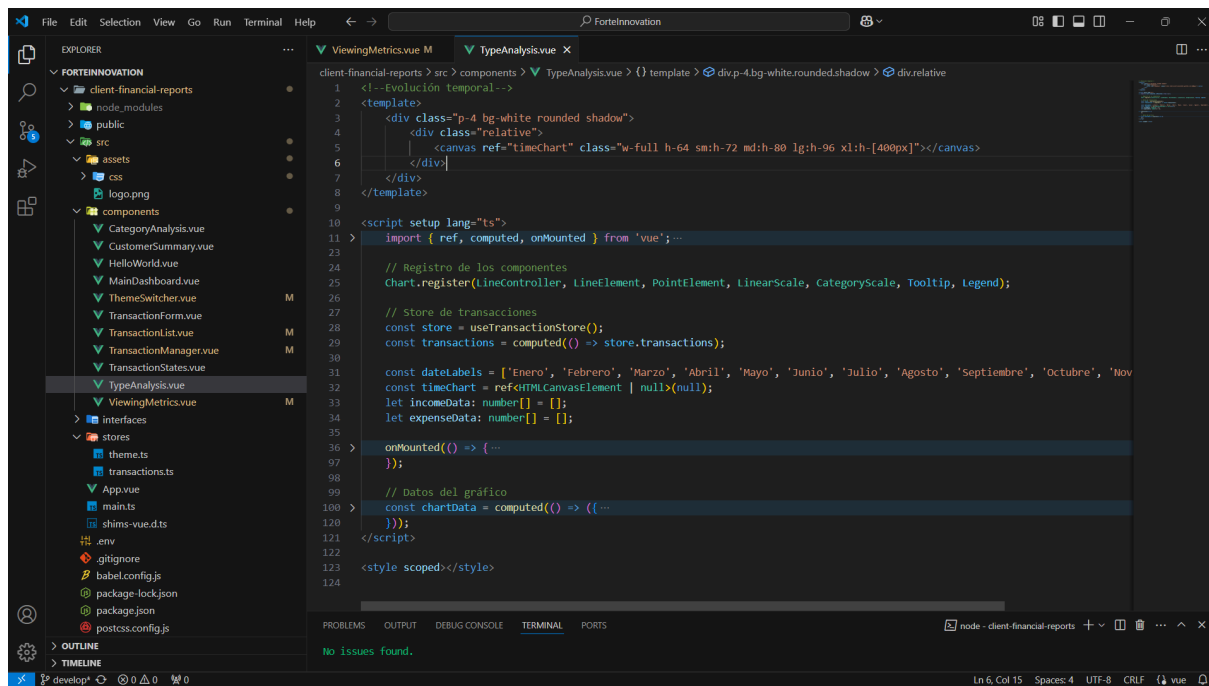
ViewingMetrics.vue => Es el componente principal encargado de la sección de Visualización de métricas donde se monta qué contenido se quiere ver en este módulo. Aquí se refleja todo el análisis de las gráficas.

```
1 <template>
2   <div class="space-y-6">
3     <!-- Titulo principal -->
4     <div>
5       <h6 class="text-lg font-semibold mt-0 mb-3 text-gray-800">Visualización de métricas</h6>
6     </div>
7
8     <!-- Cards contenedores -->
9     <div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-2 gap-6">
10      <div class="bg-white shadow rounded-lg p-4">
11        <h2 class="text-lg font-medium text-gray-700 mb-2">Análisis por categoría</h2>
12        <CategoryAnalysis />
13      </div>
14
15      <div class="bg-white shadow rounded-lg p-4">
16        <h2 class="text-lg font-medium text-gray-700 mb-2">Evolución temporal</h2>
17        <TypeAnalysis />
18      </div>
19
20      <div class="bg-white shadow rounded-lg p-4">
21        <h2 class="text-lg font-medium text-gray-700 mb-2">Resumen por cliente</h2>
22        <CustomerSummary />
23      </div>
24
25      <div class="bg-white shadow rounded-lg p-4">
26        <h2 class="text-lg font-medium text-gray-700 mb-2">Estados de transacciones</h2>
27        <TransactionStates />
28      </div>
29    </div>
30  </div>
31 </template>
32
33 <script setup lang="ts">
34   import CategoryAnalysis from './CategoryAnalysis.vue';
35   // ...
```

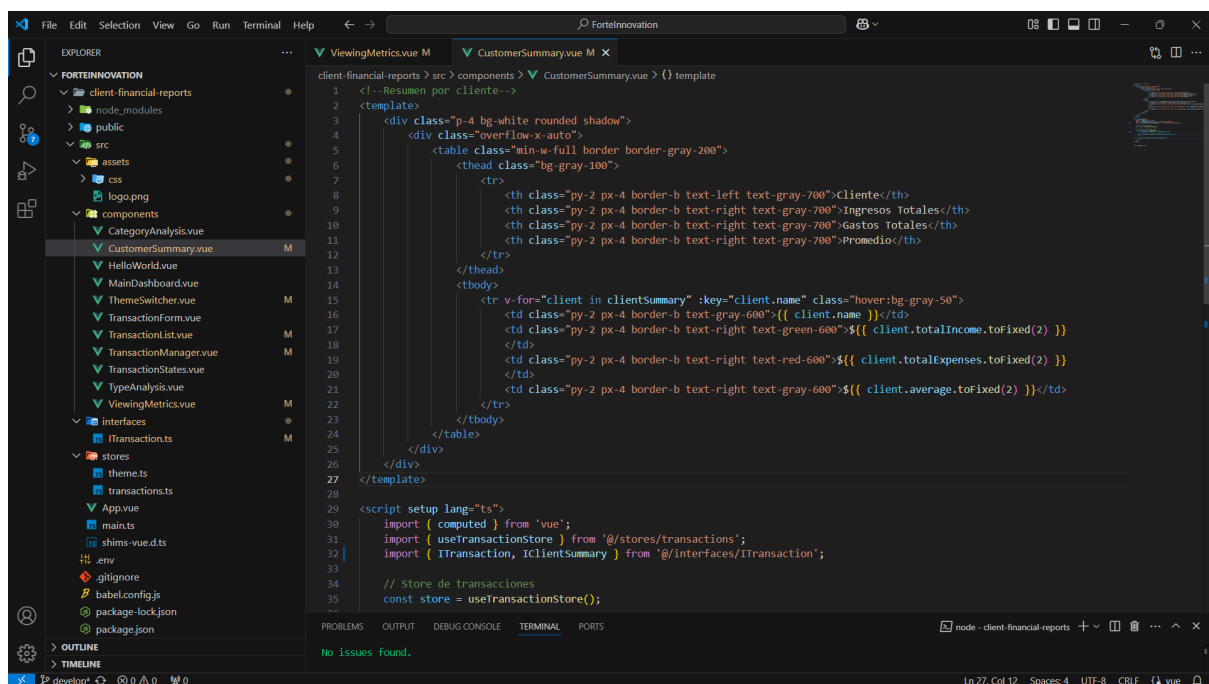
CategoryAnalysis.vue => Este componente se encarga de cargar y mostrar la gráfica de Análisis por categoría. Como también aquí se encuentra toda su funcionalidad de cómo se arma esa gráfica con los datos obtenidos.

```
1 <!-- Analisis por categoría -->
2 <template>
3   <div class="p-4 bg-gray-50 rounded shadow">
4     <!-- contenedor del gráfico -->
5     <div class="relative aspect-square sm:aspect-video">
6       <canvas ref="categoryChart" class="w-full h-full"></canvas>
7     </div>
8
9     <!-- Desglose de categorías -->
10    <div class="mt-4">
11      <div v-for="(value, category) in categoryData" :key="category"
12        class="flex justify-between items-center text-sm md:text-base mb-2">
13        <span class="text-gray-700">{{ category }}</span>
14        <span class="font-semibold text-gray-900">{{ formatAmount(value) }}</span>
15      </div>
16    </div>
17  </div>
18 </template>
19
20 <script setup lang="ts">
21   import { ref, computed, onMounted } from 'vue';
22
23   // Registra los componentes necesarios
24   Chart.register(PieController, ArcElement, Tooltip, Legend);
25
26   // Referencia al canvas para el gráfico
27   const categoryChart = ref<HTMLCanvasElement | null>(null);
28
29   // Store de transacciones
30   const store = useTransactionStore();
31
32   // Calculamos el desglose de ingresos y gastos por categoría
33   let categoryData = {};
34
35   // Funcion para generar colores aleatorios podria colocarse en otro lado del proyecto para reutilizar como un utilities
```

TypeAnalysis.vue => Este componente se encarga de cargar y mostrar la gráfica de Evolución temporal. Como también aquí se encuentra toda su funcionalidad de cómo se arma esa gráfica con los datos obtenidos.



CustomerSummary.vue => Este componente se encarga de cargar y mostrar la tabla de Resumen por cliente. Como también aquí se encuentra toda su funcionalidad de cómo se calcula con los datos obtenidos para la tabla.



TransactionStates.vue => Este componente se encarga de cargar y mostrar el contenido de Estados de transacciones. Como también aquí se encuentra toda su funcionalidad de como se calcula el contenido.

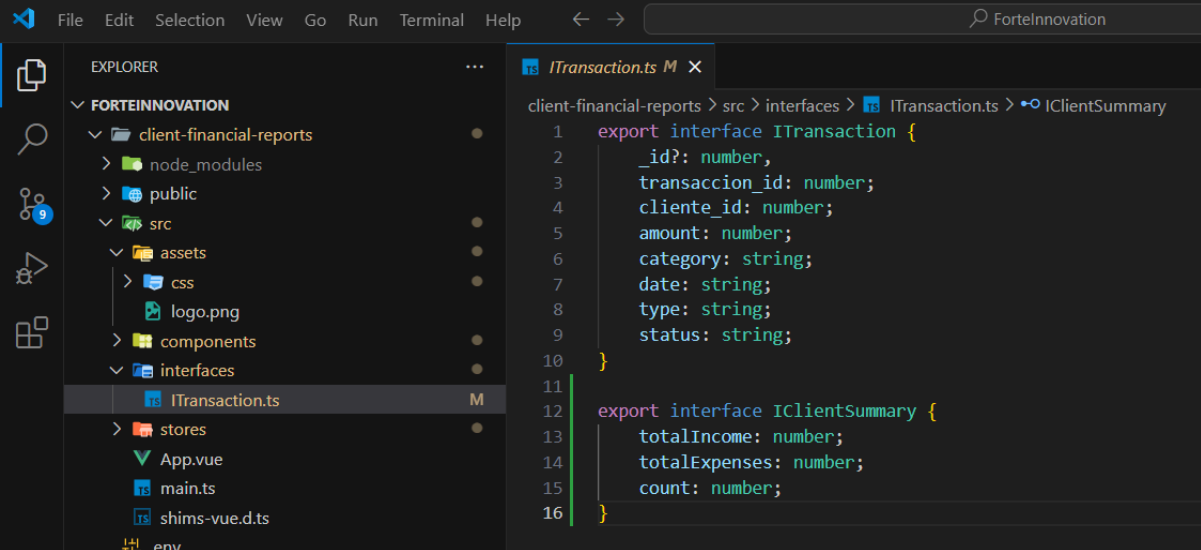
```
1 <!-- Estados de transacciones -->
2 <template>
3   <div class="p-4 bg-gray-50 rounded shadow">
4     <div class="flex justify-between mb-4">
5       <div class="flex items-center space-x-2">
6         <div class="w-4 h-4 bg-green-500 rounded-full"></div>
7         <span>Activas</span>
8       </div>
9       <span class="text-lg font-semibold text-gray-800">{{ activeCount }}</span>
10     </div>
11     <div class="flex justify-between">
12       <div class="flex items-center space-x-2">
13         <div class="w-4 h-4 bg-red-500 rounded-full"></div>
14         <span>Desactivadas</span>
15       </div>
16       <span class="text-lg font-semibold text-gray-800">{{ inactiveCount }}</span>
17     </div>
18   </div>
19 </template>
20
21 <script setup lang="ts">
22   import { computed, onMounted } from 'vue';
23   import { useTransactionStore } from '@/stores/transactions';
24
25   // Store de transacciones
26   const store = useTransactionStore();
27   const transactions = computed(() => store.transactions);
28
29   onMounted(() => {
30     store.fetchTransactions(); // Llama a la función para cargar las transacciones cuando el componente se monta
31   });
32
33   // Contar transacciones activas y desactivadas
34   const activeCount = computed(() =>
35     transactions.value.filter((t) => t.status === 'activa').length
36   );
37   const inactiveCount = computed(() =>
38     transactions.value.filter((t) => t.status === 'inactiva').length
39   );
40 </script>
```

## Flujo de datos

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

```
1 VUE_APP_API_URL=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 clients.



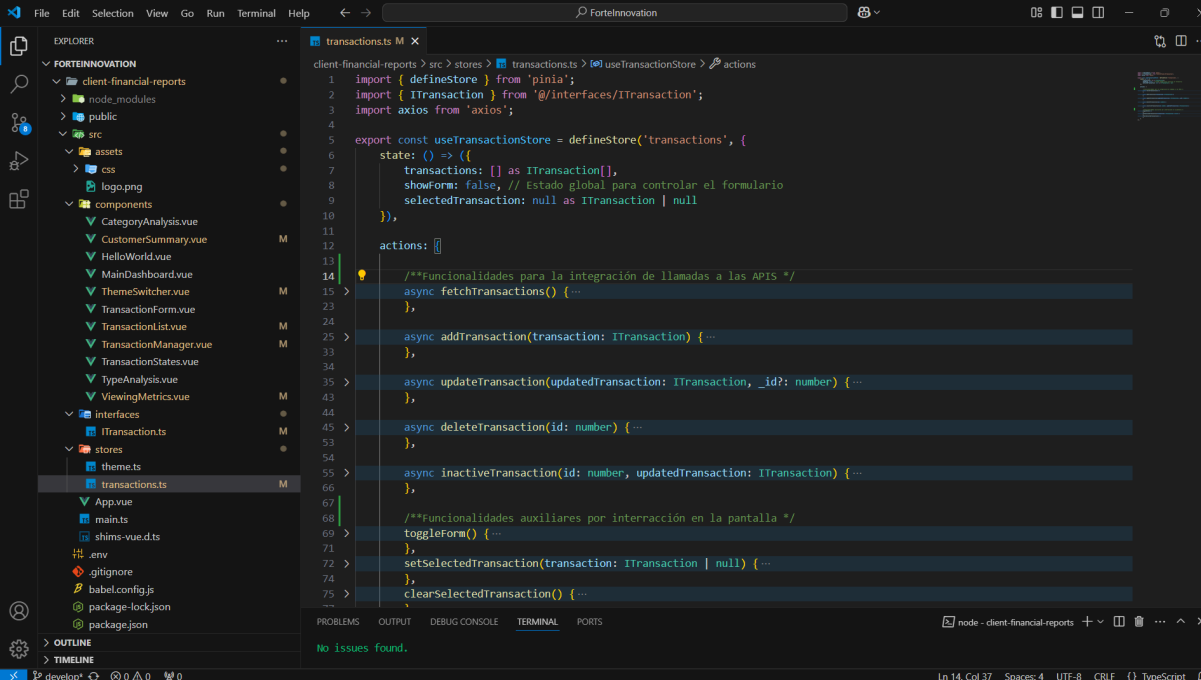
```
client-financial-reports > src > interfaces > ITransaction.ts > IClientSummary
1  export interface ITransaction {
2      _id?: number;
3      transaccion_id: number;
4      cliente_id: number;
5      amount: number;
6      category: string;
7      date: string;
8      type: string;
9      status: 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

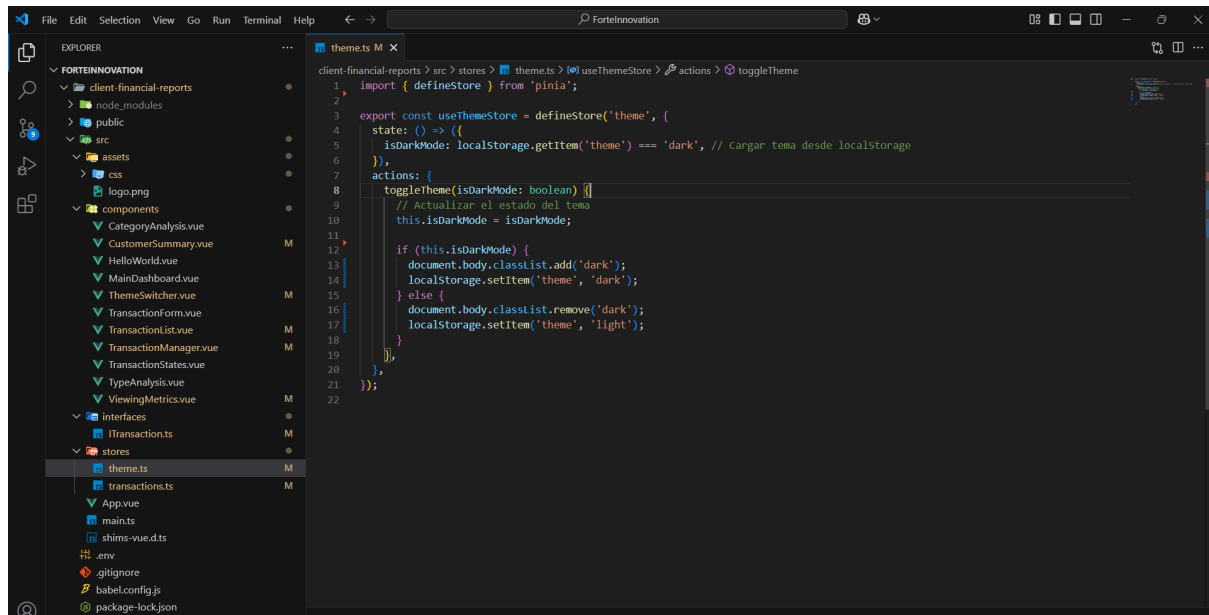
`actions` => Integración de mis llamadas a las APIS



```
client-financial-reports > src > stores > transactions.ts > useTransactionStore > actions
1  import { defineStore } from 'pinia';
2  import { ITransaction } from '@/interfaces/ITransaction';
3  import axios from 'axios';
4
5  export const useTransactionStore = defineStore('transactions', {
6      state: () => ({
7          transactions: [] as ITransaction[],
8          showForm: false, // Estado global para controlar el formulario
9          selectedTransaction: null as ITransaction | null
10      }),
11
12      actions: {
13
14          /**Funcionalidades para la integración de llamadas a las APIS */
15          async fetchTransactions() { ... },
16
17          async addTransaction(transaction: ITransaction) { ... },
18
19          async updateTransaction(updatedTransaction: ITransaction, _id?: number) { ... },
20
21          async deleteTransaction(id: number) { ... },
22
23          async inactiveTransaction(id: number, updatedTransaction: ITransaction) { ... },
24
25          /**Funcionalidades auxiliares por interacción en la pantalla */
26          toggleForm() { ... },
27
28          setSelectedTransaction(transaction: ITransaction | null) { ... },
29
30          clearSelectedTransaction() { ... }
31      }
32 })
```



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



```
1 import { defineStore } from 'pinia';
2
3 export const useThemeStore = defineStore('theme', {
4   state: () => ({
5     isDarkMode: localStorage.getItem('theme') === 'dark', // cargar tema desde localStorage
6   }),
7   actions: {
8     toggleTheme(isDarkMode: boolean) {
9       // Actualizar el estado del tema
10      this.isDarkMode = isDarkMode;
11
12      if (this.isDarkMode) {
13        document.body.classList.add('dark');
14        localStorage.setItem('theme', 'dark');
15      } else {
16        document.body.classList.remove('dark');
17        localStorage.setItem('theme', 'light');
18      }
19    },
20  });
21
22
```

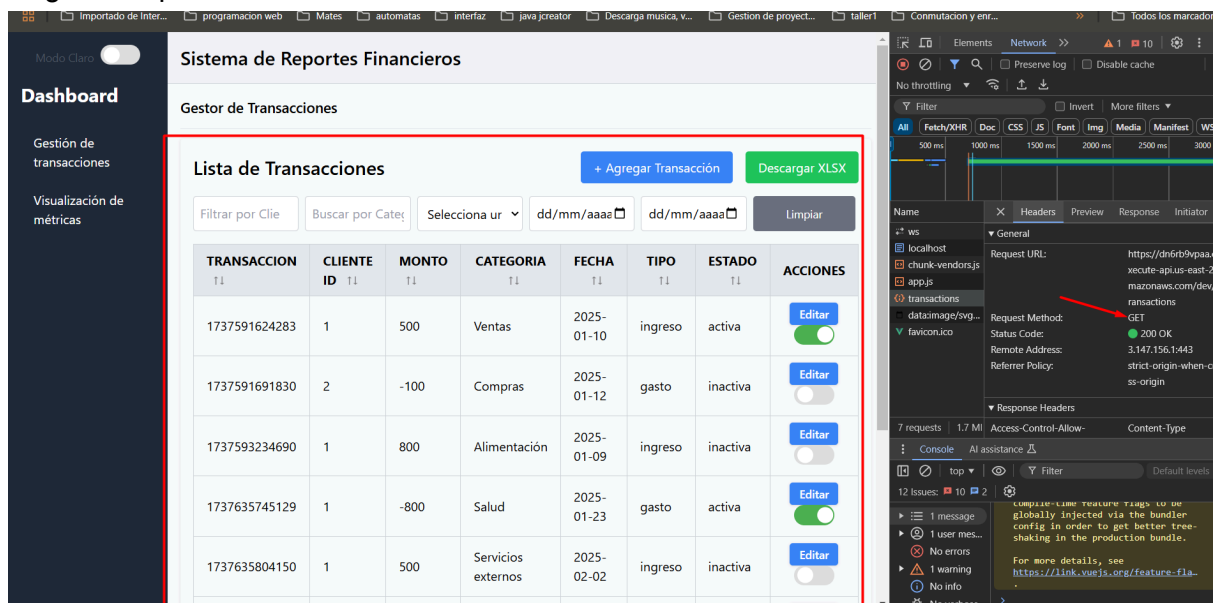
## Pruebas unitarias e integración con el backend

Gestionar transacciones financieras (CRUD)

Consultar

1-El usuario ingresara por la URL

2-El sistema se encarga de montar la información del sitio y consultar a la BD la información cargada en pantalla



**Sistema de Reportes Financieros**

**Gestor de Transacciones**

**Lista de Transacciones** + Agregar Transacción Descargar XLSX

Filtrar por Cliente:  Buscar por Categoría:  Selecciona una categoría:  dd/mm/aaaa  dd/mm/aaaa  Limpiar

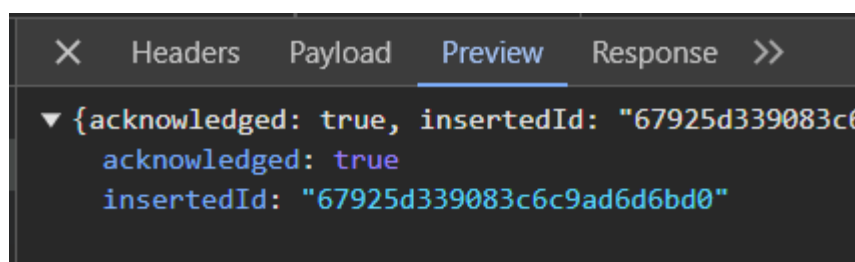
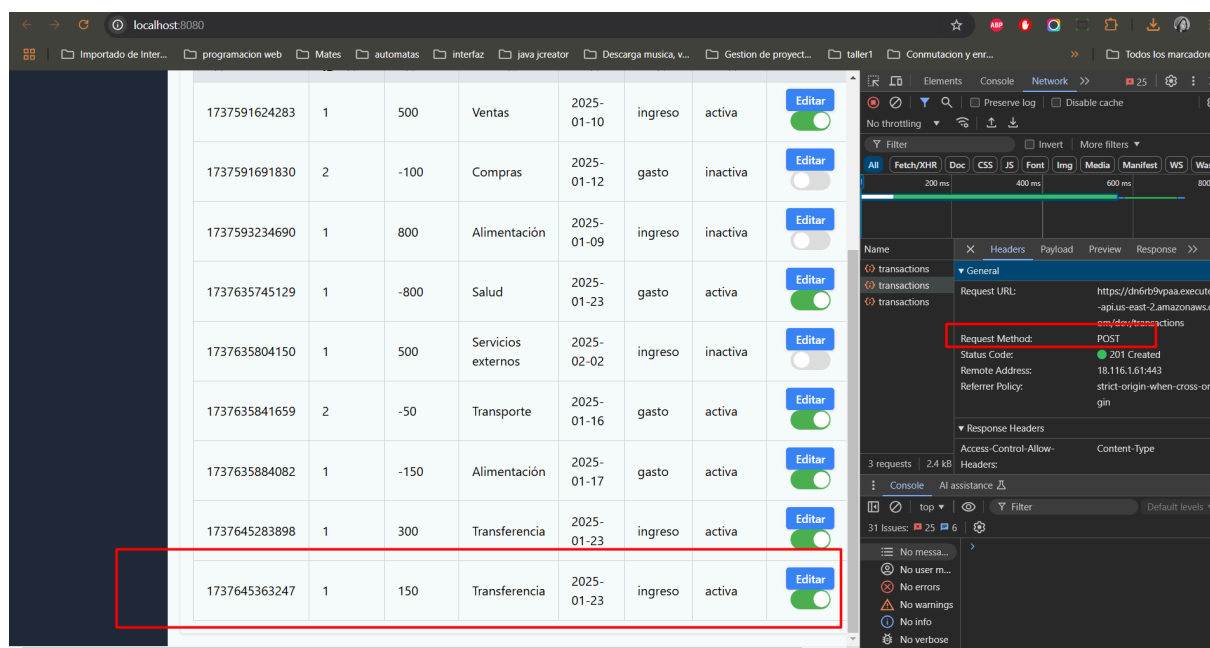
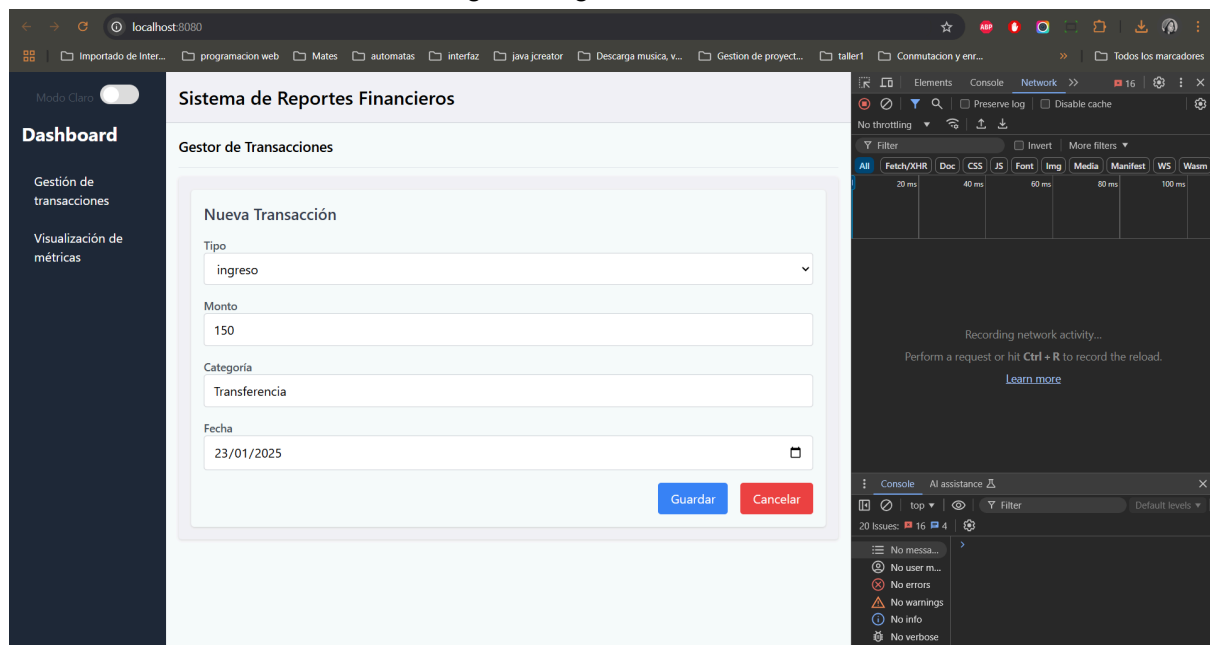
TRANSACCION	CLIENTE	MONTO	CATEGORIA	FECHA	TIPO	ESTADO	ACCIONES
T1	ID T1	T1	T1	T1	T1	T1	
1737591624283	1	500	Ventas	2025-01-10	ingreso	activa	<span>Editar</span>
1737591691830	2	-100	Compras	2025-01-12	gasto	inactiva	<span>Editar</span>
1737593234690	1	800	Alimentación	2025-01-09	ingreso	inactiva	<span>Editar</span>
1737635745129	1	-800	Salud	2025-01-23	gasto	activa	<span>Editar</span>
1737635804150	1	500	Servicios externos	2025-02-02	ingreso	inactiva	<span>Editar</span>

**Network Tab:**

Request Method: GET  
Status Code: 200 OK  
Remote Address: 3.147.156.1443  
Referrer Policy: strict-origin-when-cross-origin

## Creación

- 1-El usuario dará clic al botón de agregar transacción
- 2-El sistema mostrar en pantalla el formulario
- 3-El usuario capturar la información correspondiente
- 4-El usuario dará clic a guardar
- 5-El sistema internamente se encargará de guardar esa información a la BD



## Edición

- 1-El usuario dará clic al botón de editar en alguna de las filas de la tabla
- 2-El sistema mostrar en pantalla el formulario pre cargado con la información correspondiente
- 3-El usuario capturara la información que desee modificar
- 4-El usuario dará clic a guardar
- 5-El sistema internamente se encargará de actualizar esa información a la BD

**Sistema de Reportes Financieros**

**Gestor de Transacciones**

**Editar Transacción**

Tipo: ingreso

Monto: 150

Categoría: Transferencia

Fecha: 23/01/2025

**Guardar** **Cancelar**

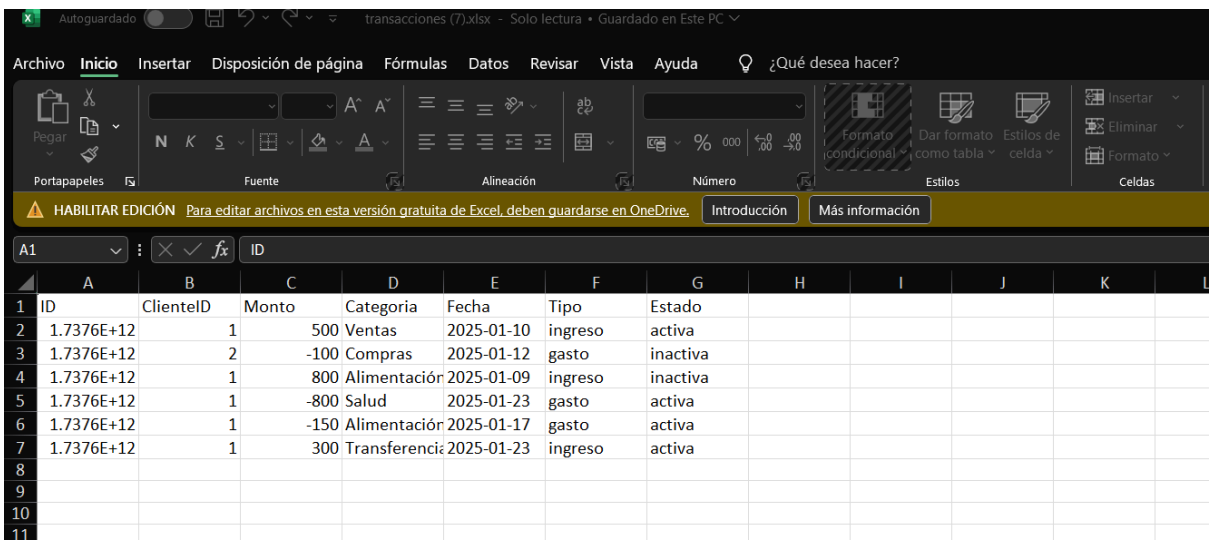
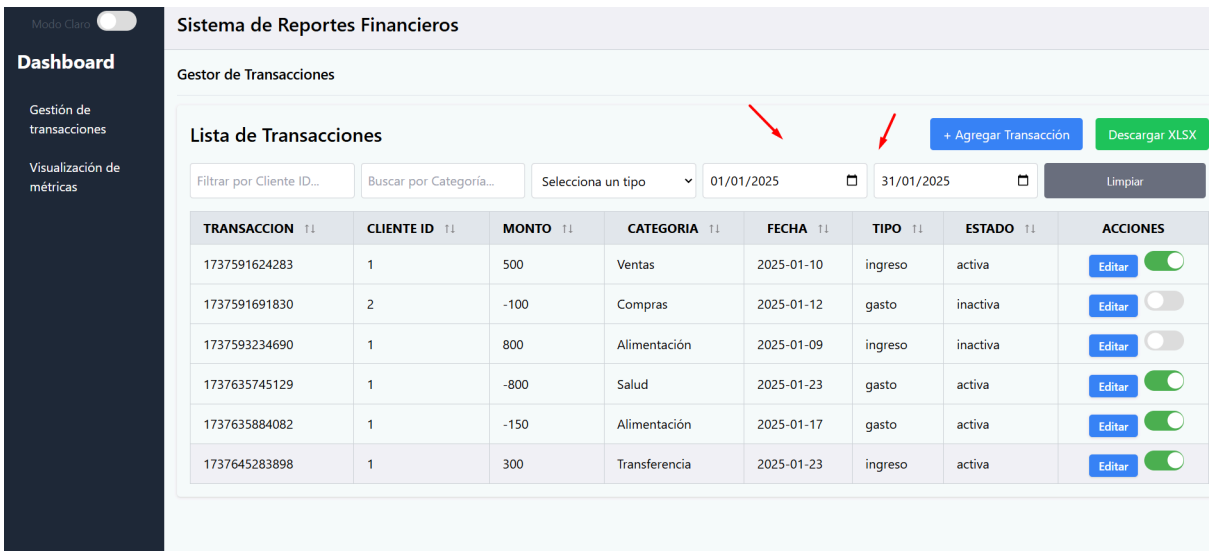
1737591624283	1	500	Ventas	2025-01-10	ingreso	activa	<b>Editar</b>
1737591691830	2	-100	Compras	2025-01-12	gasto	inactiva	<b>Editar</b>
1737593234690	1	800	Alimentación	2025-01-09	ingreso	inactiva	<b>Editar</b>
1737635745129	1	-800	Salud	2025-01-23	gasto	activa	<b>Editar</b>
1737635804150	1	500	Servicios externos	2025-02-02	ingreso	inactiva	<b>Editar</b>
1737635841659	2	-50	Transporte	2025-01-16	gasto	activa	<b>Editar</b>
1737635884082	1	-150	Alimentación	2025-01-17	gasto	activa	<b>Editar</b>
1737645283898	1	300	Transferencia	2025-01-23	ingreso	activa	<b>Editar</b>
1737645363247	1	200	Transferencia	2025-01-23	ingreso	activa	<b>Editar</b>

```
{acknowledged: true, modifiedCount: 1, upsertedCount: 0, upsertedId: null}
```

Configurar parámetros para la generación de reportes (rango de fechas, cliente específico, categoría)

Rango de fechas

- 1-El usuario ingresa las fechas que quiera filtrar
- 2-El sistema se encargará de solo mostrar ese rango de fechas
- 3-El usuario descargar el reporte dando click al botón
- 4-El sistema realizará esa descarga de archivo



Categoría

- 1-El usuario seleccionara la categoría que quiera filtrar
- 2-El sistema se encargará de solo mostrar esa información
- 3-El usuario descargar el reporte dando click al botón
- 4-El sistema realizará esa descarga de archivo

Modo Claro

Dashboard

Gestión de transacciones

Visualización de métricas

Sistema de Reportes Financieros

Gestor de Transacciones

Lista de Transacciones

+ Agregar Transacción

Descargar XLSX

Filtrar por Cliente ID...

Buscar por Categoría...

gasto

dd/mm/aaaa

dd/mm/aaaa

Limpiar

TRANSACCION	CLIENTE ID	MONTO	CATEGORIA	FECHA	TIPO	ESTADO	ACCIONES
1737591691830	2	-100	Compras	2025-01-12	gasto	inactiva	<div>Editar</div>
1737635745129	1	-800	Salud	2025-01-23	gasto	activa	<div>Editar</div>
1737635841659	1	-50	Transporte	2025-02-07	gasto	activa	<div>Editar</div>
1737635884082	1	-150	Alimentación	2025-01-17	gasto	activa	<div>Editar</div>

Autoguardado transacciones (8).xlsx - Solo lectura • Guardado en Este PC

Archivo Inicio Insertar Disposición de página Fórmulas Datos Revisar Vista Ayuda ¿Qué desea hacer?

Pegar

Portapapeles

N

K

S

Fuente

Alineación

Número

Formato condicional

Dar formato como tabla

Estilos de celda

Estilos

Insertar

Eliminar

Formatear

Cel

HABILITAR EDICIÓN Para editar archivos en esta versión gratuita de Excel, deben guardarse en OneDrive. Introducción Más información

A1 ID

	A	B	C	D	E	F	G	H	I	J	K
1	ID	ClienteID	Monto	Categoría	Fecha	Tipo	Estado				
2	1.7376E+12	2	-100	Compras	2025-01-12	gasto	inactiva				
3	1.7376E+12	1	-800	Salud	2025-01-23	gasto	activa				
4	1.7376E+12	1	-50	Transporte	2025-02-07	gasto	activa				
5	1.7376E+12	1	-150	Alimentación	2025-01-17	gasto	activa				
6											
7											
8											
9											
10											
11											

Cliente específico

- 1-El usuario capturar en el filtro cliente específico
- 2-El sistema se encargará de solo mostrar esa información
- 3-El usuario descargar el reporte dando click al botón
- 4-El sistema realizará esa descarga de archivo

Modo Claro

Dashboard

Gestión de transacciones

Visualización de métricas

Sistema de Reportes Financieros

Gestor de Transacciones

Lista de Transacciones

+ Agregar Transacción

Descargar XLSX

2

Buscar por Categoría...

Selecciona un tipo

dd/mm/aaaa

dd/mm/aaaa

Limpiar

TRANSACCION	CLIENTE ID	MONTO	CATEGORIA	FECHA	TIPO	ESTADO	ACCIONES
1737591691830	2	-100	Compras	2025-01-12	gasto	inactiva	<div>Editar</div>

Autoguardado transacciones (9).xlsx - Solo lectura • Guardado en Este PC

Archivo Inicio Insertar Disposición de página Fórmulas Datos Revisar Vista Ayuda ¿Qué desea hacer?

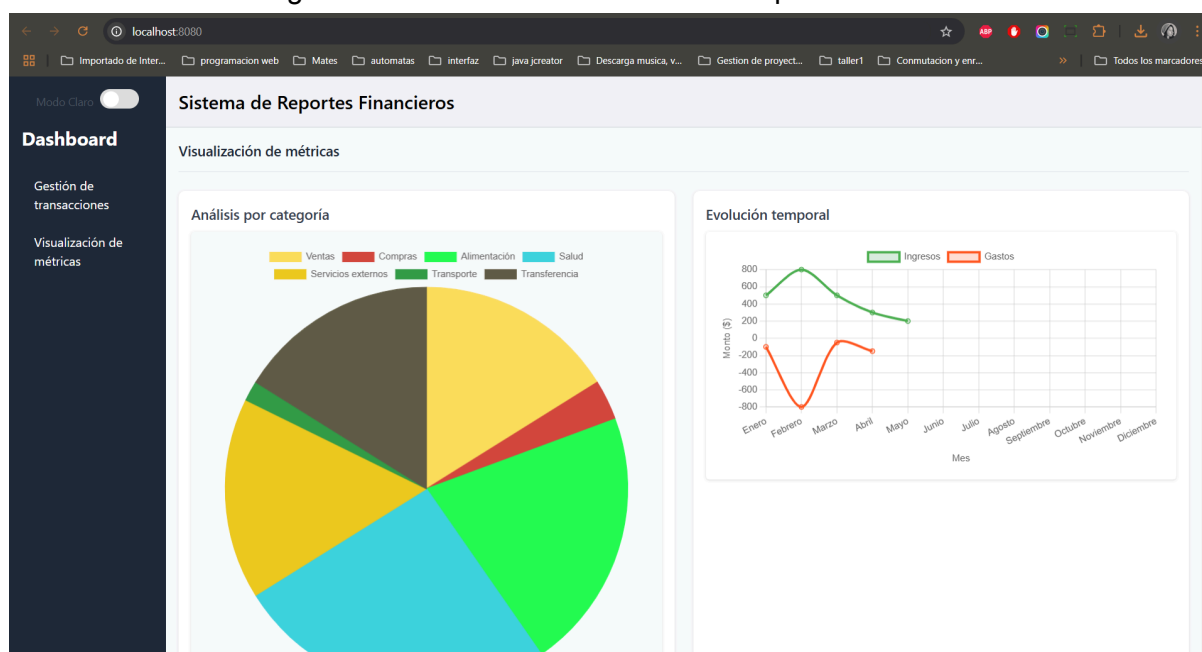
HABILITAR EDICIÓN Para editar archivos en esta versión gratuita de Excel, deben guardarse en OneDrive. Introducción Más información

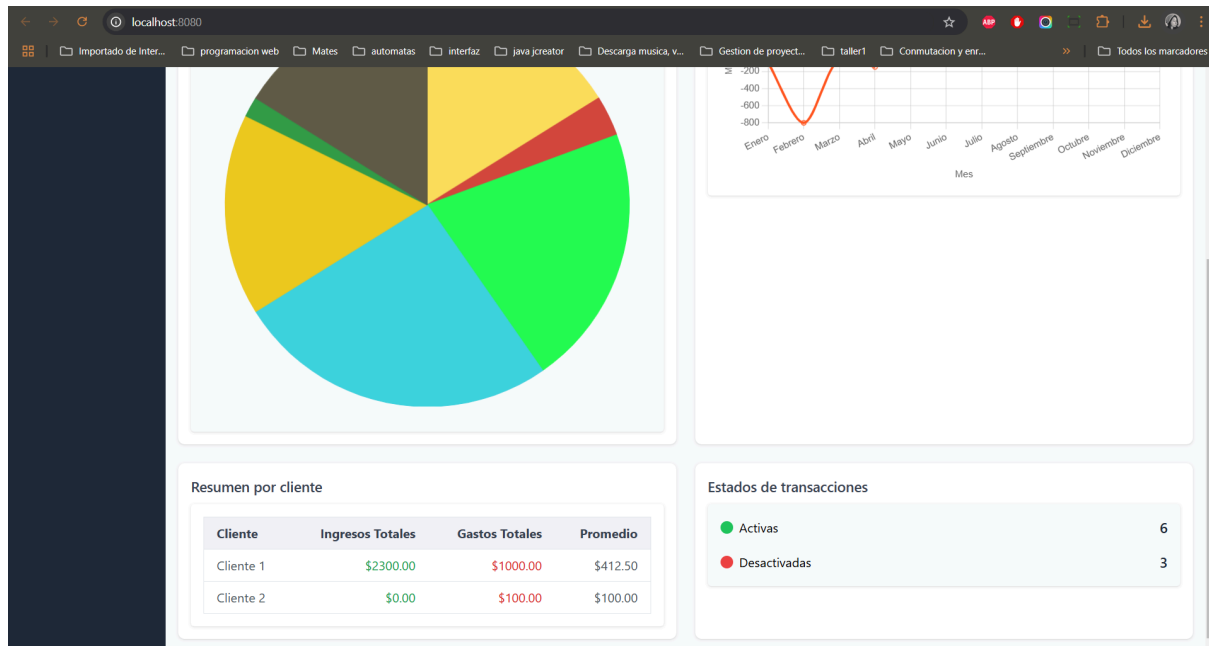
	A	B	C	D	E	F	G	H
1	ID	ClientelID	Monto	Categoría	Fecha	Tipo	Estado	
2	1.7376E+12	2	-100	Compras	2025-01-12	gasto	inactiva	
3								

Visualizar métricas clave (ingresos, gastos y promedios) en tiempo real a través de gráficos interactivos.

1-El usuario dará clic a la sección de Visualización de métricas

2-El sistema se encargará de mostrar la información correspondiente de la sección





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**

Complemento del entregable Información adicional con respecto al documento de ITG\_Evaluación\_Desarrollador Frontend 1.pdf

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