

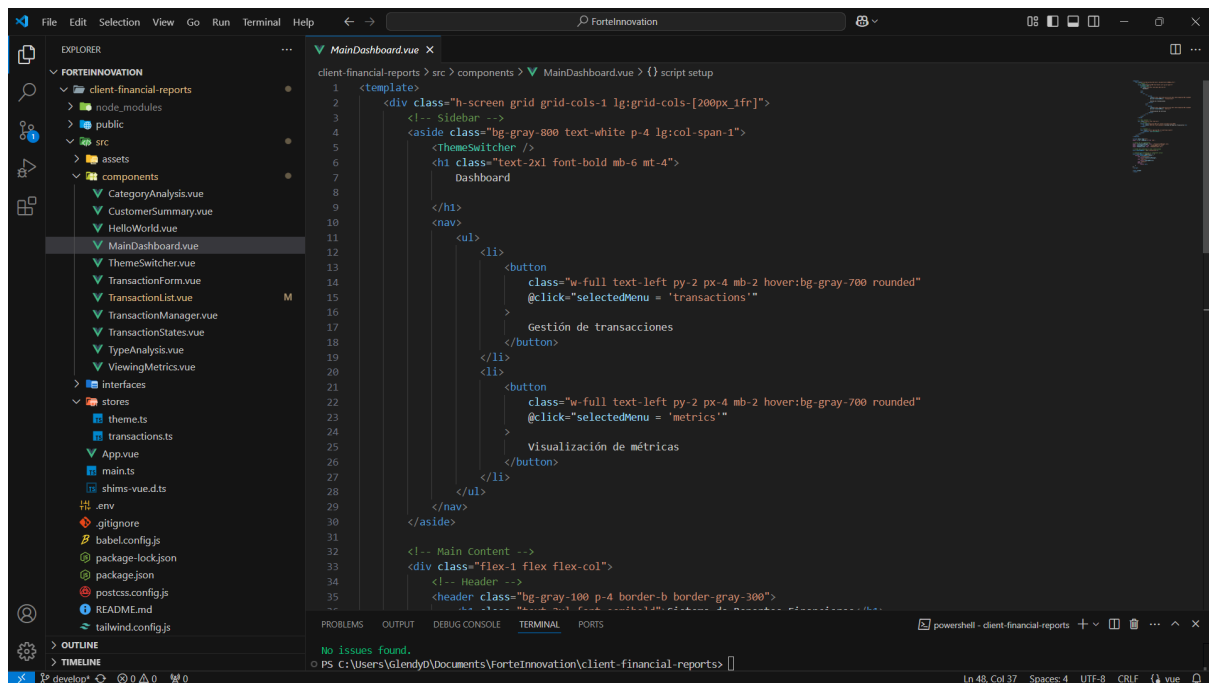
# Documentación:

## Frontend

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

## Componentes creados

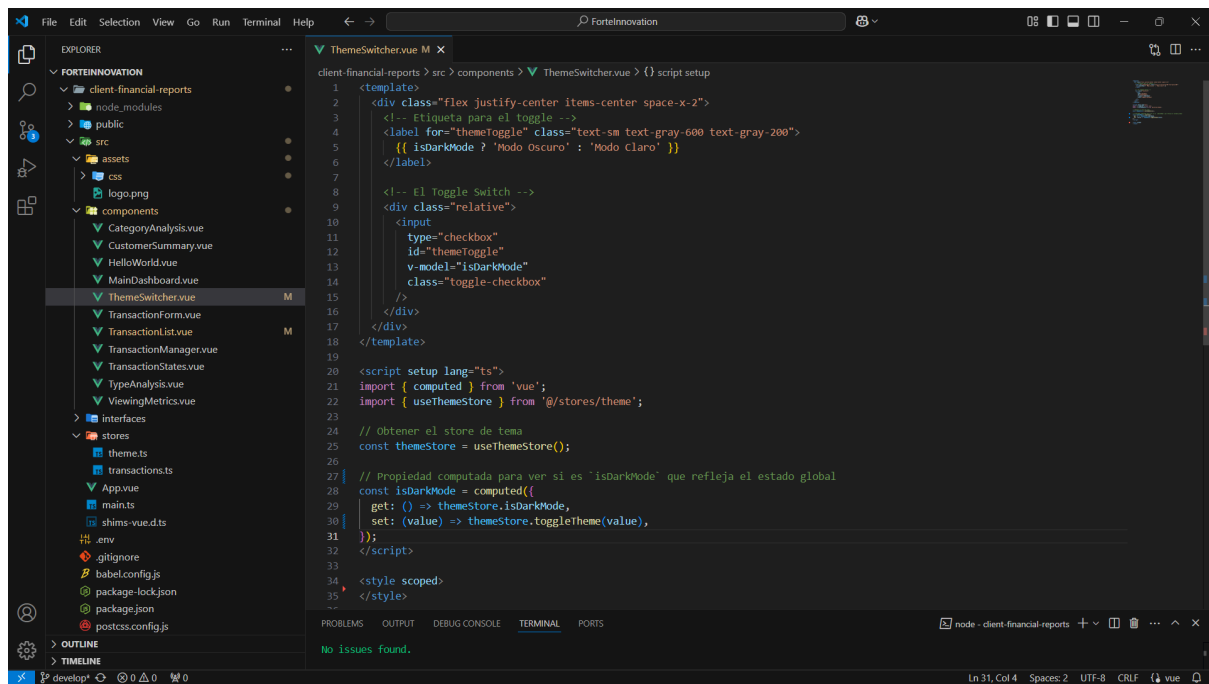
MainDashboard.vue => Componente principal para montar la plantilla del proyecto creado. Donde incluye el Menú lateral izquierdo y el contenido para mostrar el apartado correspondiente seleccionado.



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

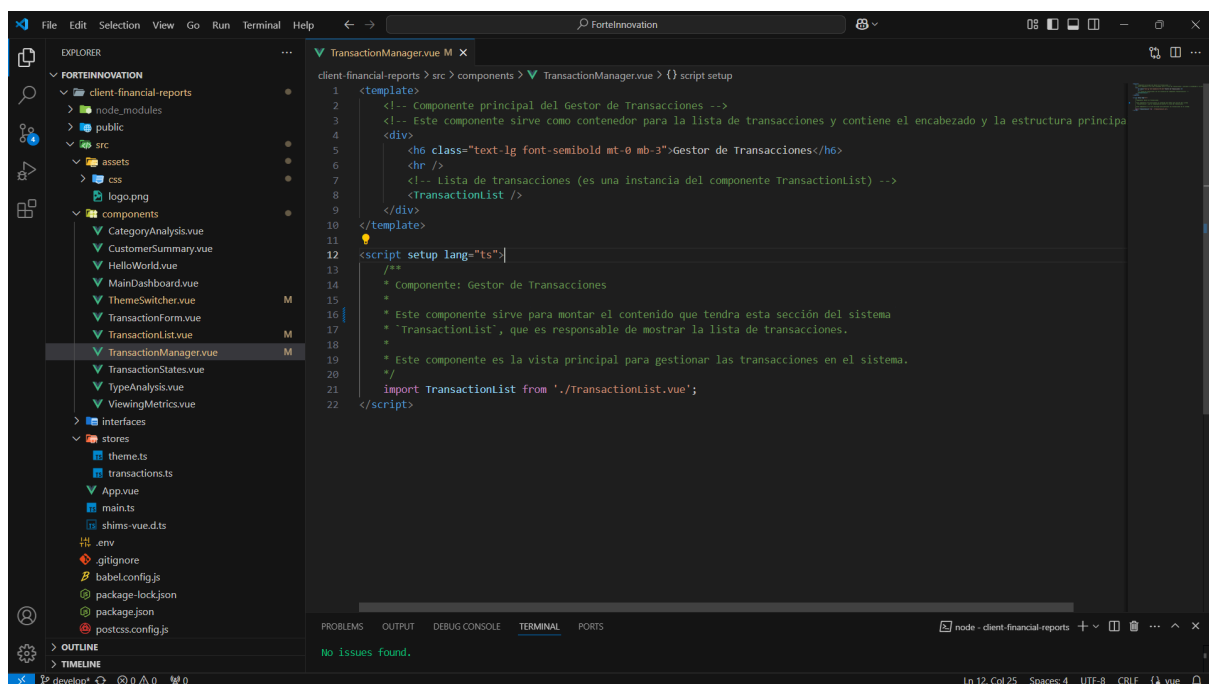
ThemeSwitcher.vue => Se encarga de customizar el tema oscuro y claro.

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



```
client-financial-reports > src > components > ThemeSwitcher.vue > {} script setup
1 <template>
2 <div class="flex justify-center items-center space-x-2">
3 <!-- Etiqueta para el toggle -->
4 <label for="themeToggle" class="text-sm text-gray-600 text-gray-200">
5 <!-- isDarkMode ? 'Modo Oscuro' : 'Modo Claro' -->
6 </label>
7
8 <!-- El Toggle Switch -->
9 <div class="relative">
10 <input
11 type="checkbox"
12 id="themeToggle"
13 v-model="isDarkMode"
14 class="toggle-checkbox"
15 </div>
16 </div>
17 </template>
18
19
20 <script setup lang="ts">
21 import { computed } from 'vue';
22 import { useThemeStore } from '@stores/theme';
23
24 // Obtener el store de tema
25 const themeStore = useThemeStore();
26
27 // Propiedad computada para ver si es 'isDarkMode' que refleja el estado global
28 const isDarkMode = computed({
29 get: () => themeStore.isDarkMode,
30 set: (value) => themeStore.toggleTheme(value),
31 });
32 </script>
33
34 <style scoped>
35 </style>
36
```

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.



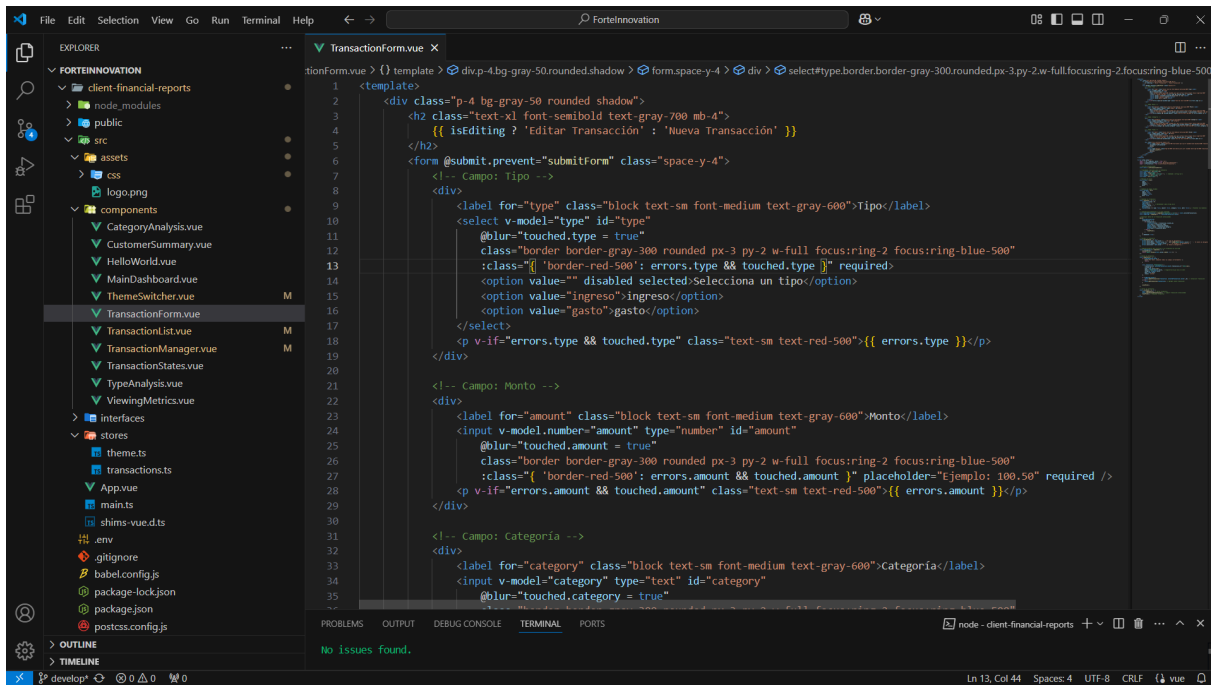
```
client-financial-reports > src > components > TransactionManager.vue > {} script setup
1 <template>
2 <!-- Componente principal del Gestor de Transacciones -->
3 <!-- Este componente sirve como contenedor para la lista de transacciones y contiene el encabezado y la estructura principal -->
4 <div>
5 <h6 class="text-lg font-semibold mt-0 mb-3">Gestor de Transacciones</h6>
6 <hr />
7 <!-- Lista de transacciones (es una instancia del componente TransactionList) -->
8 <TransactionList />
9 </div>
10 </template>
11
12 <script setup lang="ts">
13 /**
14  * componente: Gestor de Transacciones
15  *
16  * Este componente sirve para montar el contenido que tendrá esta sección del sistema
17  * 'TransactionList', que es responsable de mostrar la lista de transacciones.
18  *
19  * Este componente es la vista principal para gestionar las transacciones en el sistema.
20  */
21 import TransactionList from './TransactionList.vue';
22 </script>
```

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.

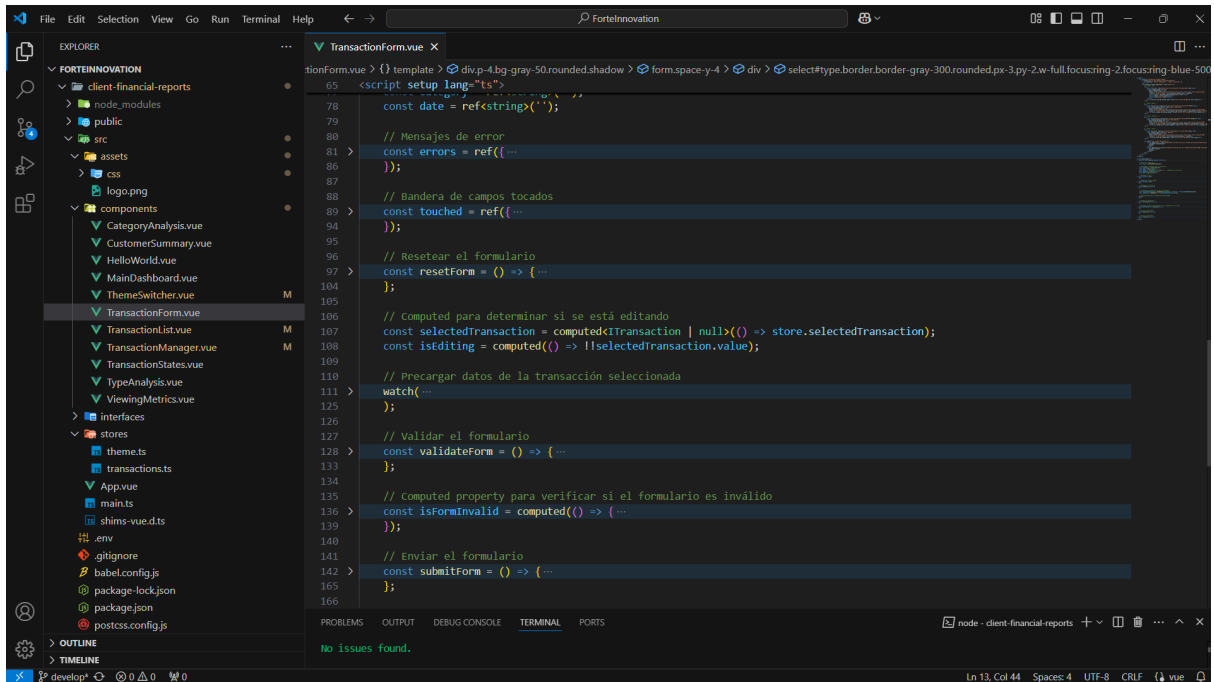
```
1 <template>
2   <!-- Componente que maneja el listado de transacciones y permite agregar, editar, eliminar transacciones -->
3   <div class="space-y-4 mt-3">
4     <!-- Formulario para agregar nueva transacción -->
5     <div v-if="showForm" class="p-4 bg-gray-100 rounded shadow">
6       <TransactionForm />
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      <!-- Tabla de transacciones -->
38      <div class="mt-4">
39        <table class="w-full border-collapse">
40          <thead>
41            <tr>
42              <th>ID</th>
43              <th>Cliente</th>
44              <th>Categoría</th>
45              <th>Monto</th>
46              <th>Fecha</th>
47              <th>Acciones</th>
48            </tr>
49          </thead>
50          <tbody>
51            <tr v-for="transaction in filteredTransactions" :key="transaction.id">
52              <td>{{ transaction.id }}</td>
53              <td>{{ transaction.clientName }}</td>
54              <td>{{ transaction.category }}</td>
55              <td>{{ transaction.amount }}</td>
56              <td>{{ transaction.date }}</td>
57              <td>
58                <button @click="editTransaction(transaction.id)">Editar</button>
59                <button @click="deleteTransaction(transaction.id)">Eliminar</button>
60              </td>
61            </tr>
62          </tbody>
63        </table>
64      </div>
65    </div>
66  </div>
67</template>
```

```
147 <script setup lang="ts">
175
176   onMounted(() => { ...
177   });
178
179   // Computed para filtrar y ordenar las transacciones
180   const filteredTransactions = computed(() => { ...
181   });
182
183   // Función para resetear filtros
184   const resetFilters = () => { ...
185   };
186
187   // Funciones para manejar transacciones
188   const addTransaction = () => { ...
189   };
190
191   const editTransaction = (transaction: ITransaction) => { ...
192   };
193
194   const inactiveTransaction = (id: number, transaction: ITransaction) => { ...
195   };
196
197   // Función para exportar las transacciones a XLSX
198   const downloadXLSX = () => { ...
199   };
200
201   // Estado de ordenación
202   const sortConfig = ref({ ...
203   });
204
205   // Función para ordenar las transacciones
206   const sortTransactions = (a: ITransaction, b: ITransaction) => { ...
207   };
208
209   // ...
210</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          class="border border-gray-300 rounded px-3 py-2 w-full focus:ring-2 focus:ring-blue-500"
35          :class="{ 'border-red-500': errors.category && touched.category }" placeholder="Ejemplo: Alimentos" required />
36        <p v-if="errors.category && touched.category" class="text-sm text-red-500">{{ errors.category }}</p>
37      </div>
38    </div>
39  </template>
```



```
65 <script setup lang="ts">
66   // Definición de tipos
67   const type = ref<string>("");
68   const amount = ref<number>(0);
69   const category = ref<string>("");
70   const date = ref<string>("");
71   const errors = ref<Record<string, string>>({});
72   const touched = ref<Record<string, boolean>>({});
73   const resetForm = () => {
74     type.value = "";
75     amount.value = 0;
76     category.value = "";
77     date.value = "";
78     errors.value = {};
79     touched.value = {};
80   };
81   // Computed para determinar si se está editando
82   const selectedTransaction = computed<Transaction | null>(() => store.selectedTransaction);
83   const isEditing = computed(() => !!selectedTransaction.value);
84   // Precargar datos de la transacción seleccionada
85   watch(selectedTransaction, () => {
86     if (selectedTransaction.value) {
87       type.value = selectedTransaction.value.type;
88       amount.value = selectedTransaction.value.amount;
89       category.value = selectedTransaction.value.category;
90       date.value = selectedTransaction.value.date;
91     }
92   });
93   // Validar el formulario
94   const validateForm = () => {
95     const newErrors: Record<string, string> = {};
96     if (!type.value) newErrors.type = "El tipo es obligatorio";
97     if (!amount.value || amount.value < 0) newErrors.amount = "El monto debe ser un número positivo";
98     if (!category.value) newErrors.category = "La categoría es obligatoria";
99     if (!date.value) newErrors.date = "La fecha es obligatoria";
100    return newErrors;
101  };
102  // Computed property para verificar si el formulario es inválido
103  const isFormInvalid = computed(() => Object.keys(validateForm()).length > 0);
104  // Enviar el formulario
105  const submitForm = () => {
106    const newErrors = validateForm();
107    if (Object.keys(newErrors).length > 0) return;
108    const newTransaction: Transaction = {
109      type: type.value,
110      amount: amount.value,
111      category: category.value,
112      date: date.value,
113    };
114    store.addTransaction(newTransaction);
115    resetForm();
116  };
117 </script>
```

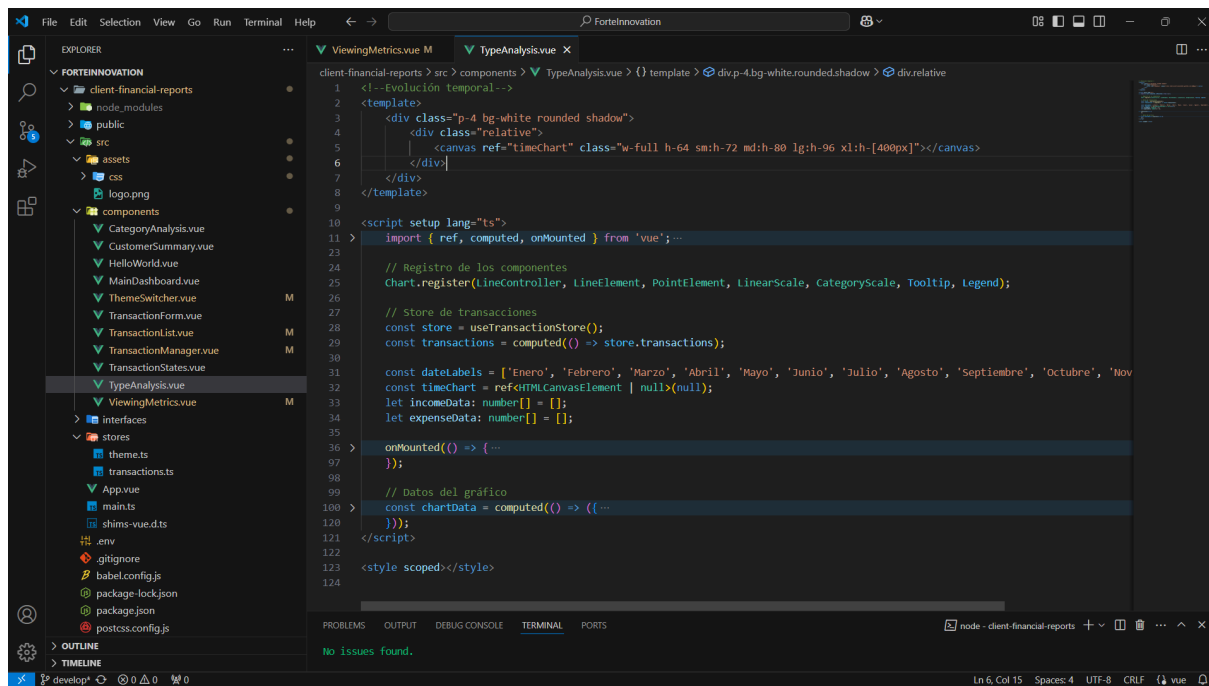
ViewingMetrics.vue => Es el componente principal encargado de la sección de Visualización de métricas donde se monta qué contenido se quiere ver en este módulo. Aquí se refleja todo el análisis de las gráficas.

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

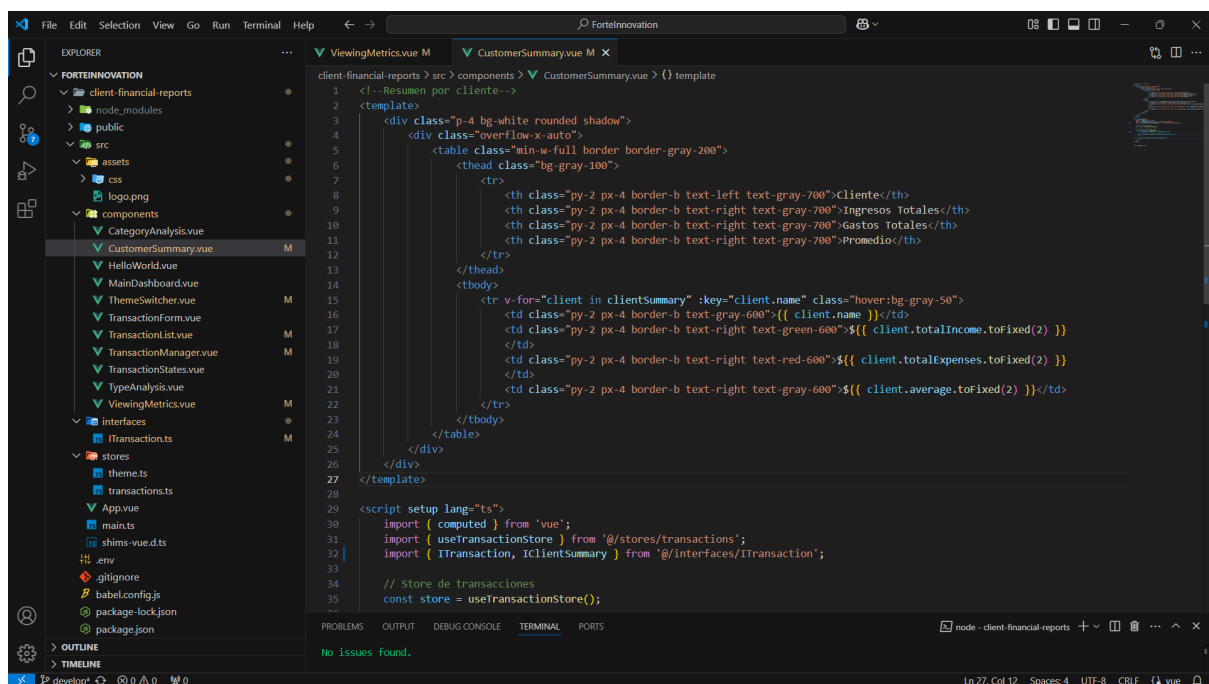
CategoryAnalysis.vue => Este componente se encarga de cargar y mostrar la gráfica de Análisis por categoría. Como también aquí se encuentra toda su funcionalidad de cómo se arma esa gráfica con los datos obtenidos.

```
client-financial-reports > src > components > CategoryAnalysis.vue {} template > div.p-4.bg-gray-50.rounded.shadow
1 <!-- Analisis por categoria -->
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 categorias -->
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 categoria
33   let categoryData = {};
34
35   // Funcion para generar colores aleatorios podria colocarse en otro lado del proyecto para reutilizar como un utilities
36 </script>
```

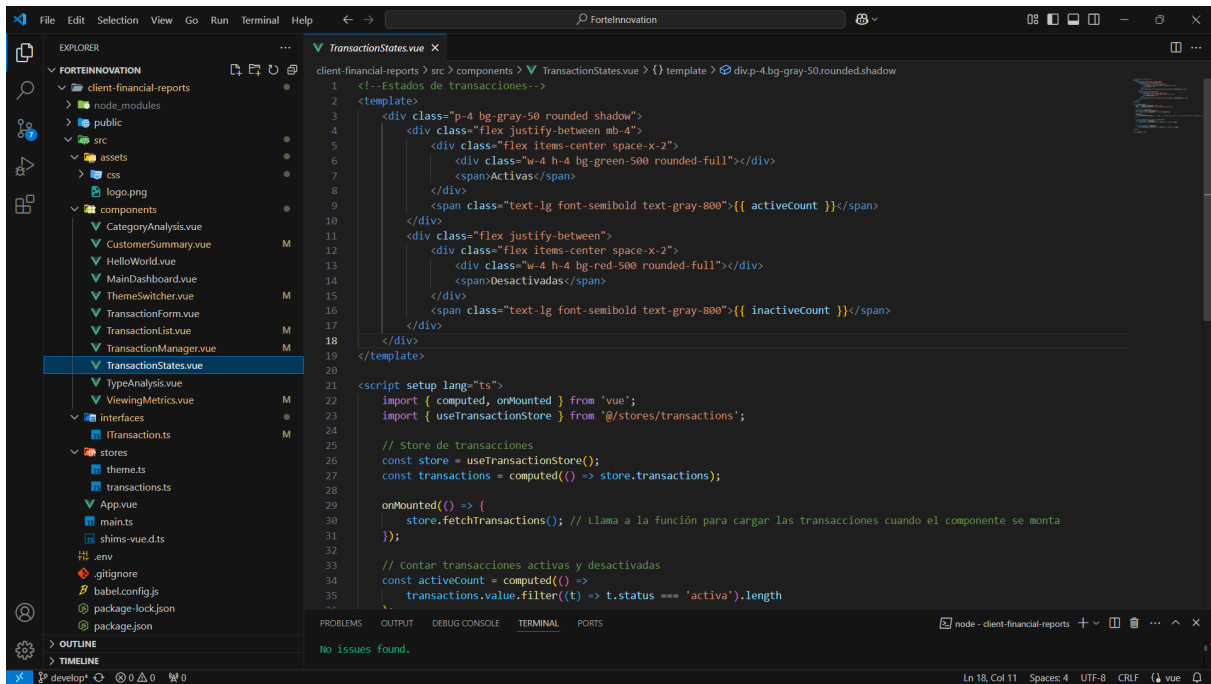
TypeAnalysis.vue => Este componente se encarga de cargar y mostrar la gráfica de Evolución temporal. Como también aquí se encuentra toda su funcionalidad de cómo se arma esa gráfica con los datos obtenidos.



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.

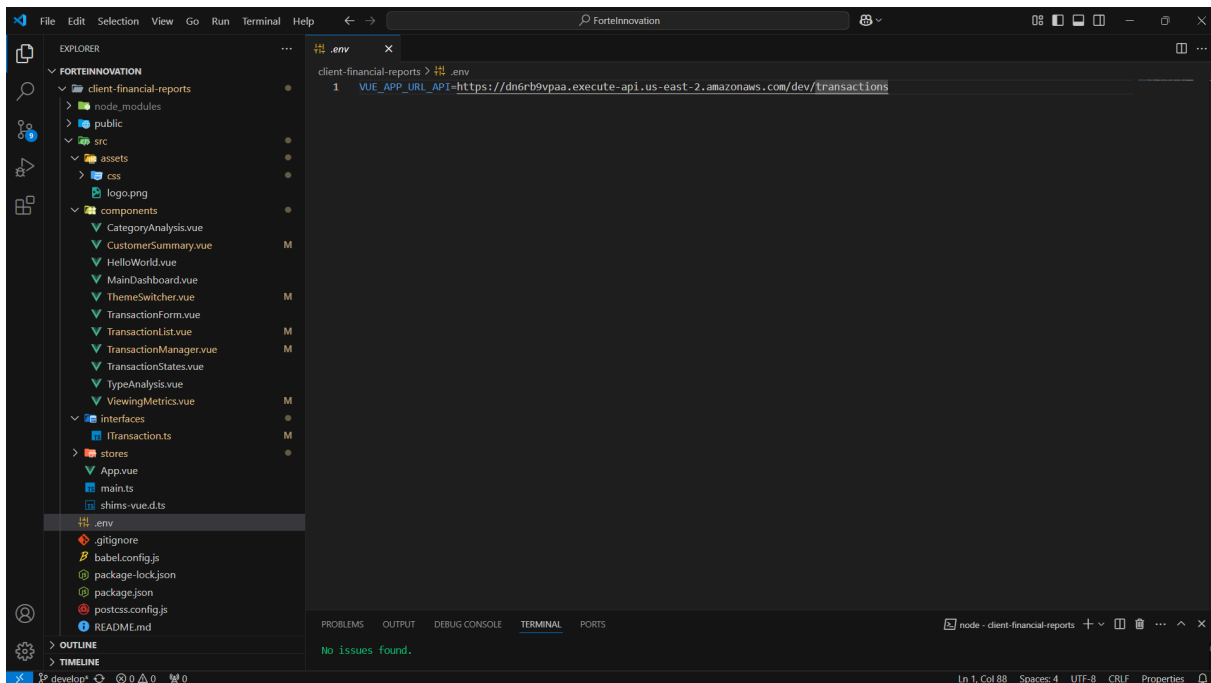


TransactionStates.vue => Este componente se encarga de cargar y mostrar el contenido de Estados de transacciones. Como también aquí se encuentra toda su funcionalidad de como se calcula el contenido.

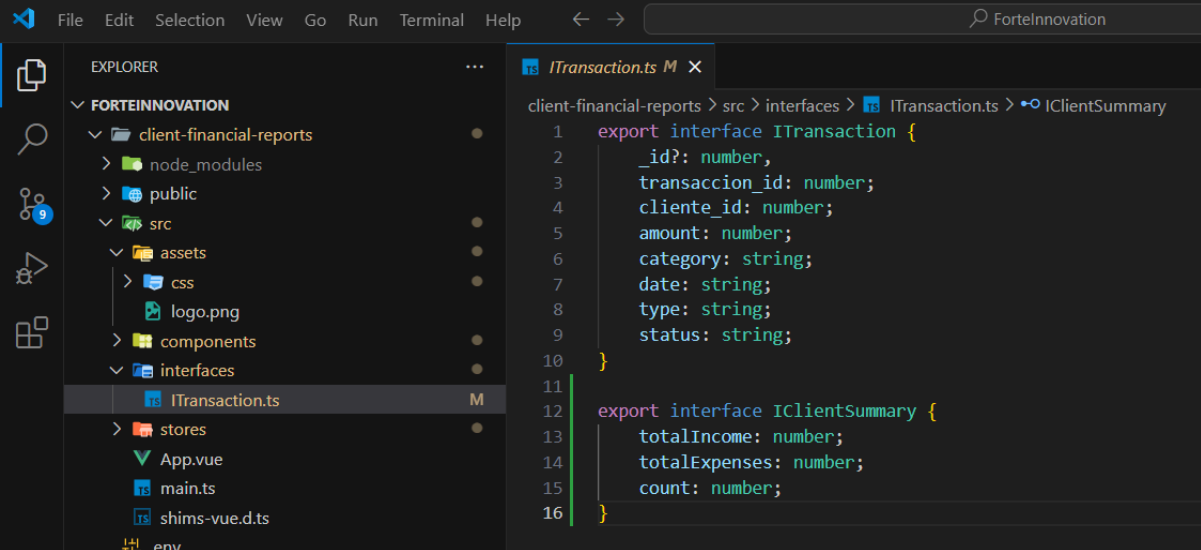


## Flujo de datos

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



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.



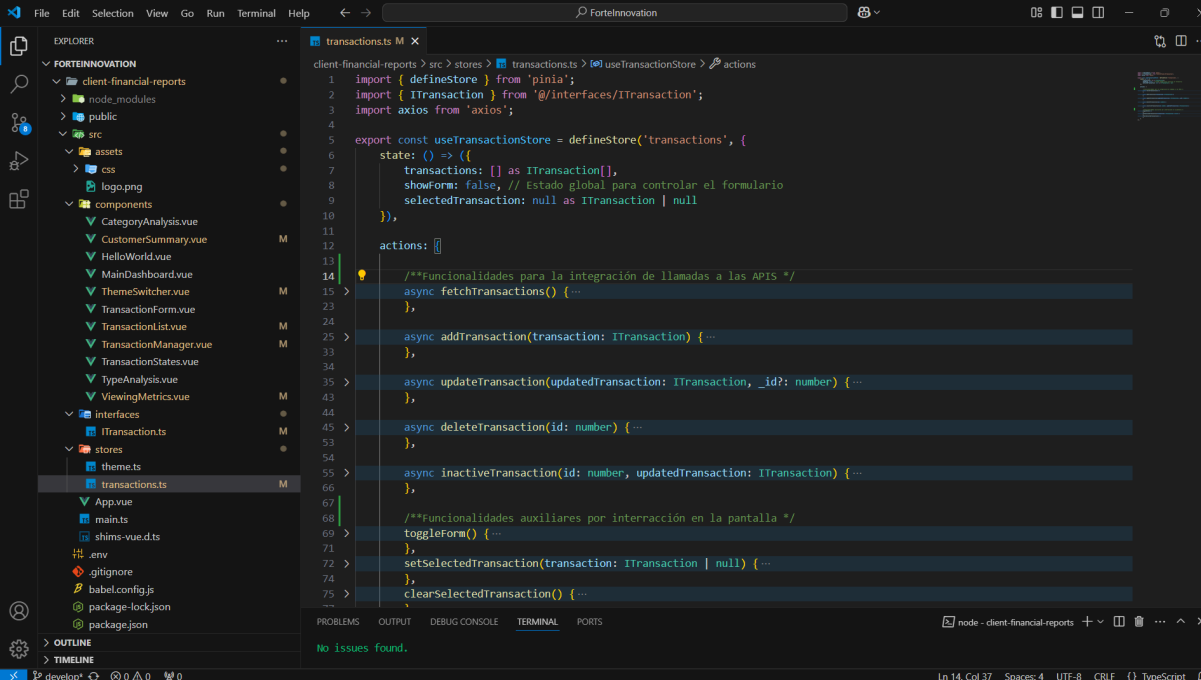
```
1 export interface ITransaction {
2   _id?: number;
3   transaccion_id: number;
4   cliente_id: number;
5   amount: number;
6   category: string;
7   date: string;
8   type: string;
9   status: string;
10 }
11
12 export interface IClientSummary {
13   totalIncome: number;
14   totalExpenses: number;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

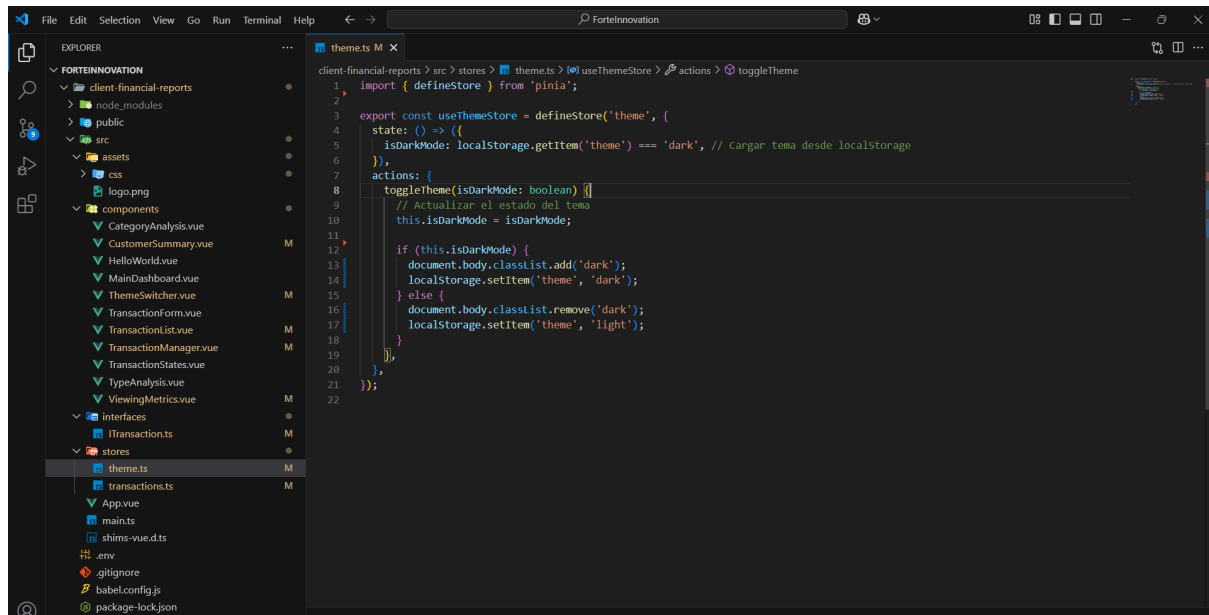
`actions` => Integración de mis llamadas a las APIS



```
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   actions: {
12     /**Funcionalidades para la integración de llamadas a las APIS */
13     async fetchTransactions() { ... },
14     async addTransaction(transaction: ITransaction) { ... },
15     async updateTransaction(updatedTransaction: ITransaction, _id?: number) { ... },
16     async deleteTransaction(id: number) { ... },
17     async inactiveTransaction(id: number, updatedTransaction: ITransaction) { ... },
18     /**Funcionalidades auxiliares por interacción en la pantalla */
19     toggleForm() { ... },
20     setSelectedTransaction(transaction: ITransaction | null) { ... },
21     clearSelectedTransaction() { ... }
22   }
23 })
```



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
```

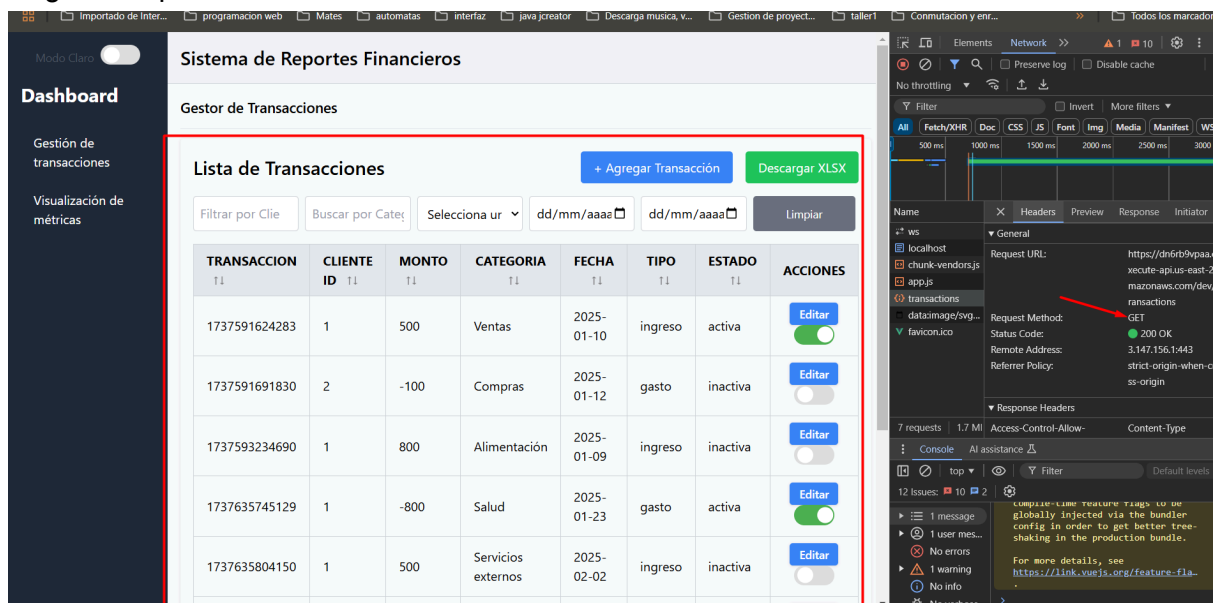
## Pruebas unitarias e integración con el backend

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

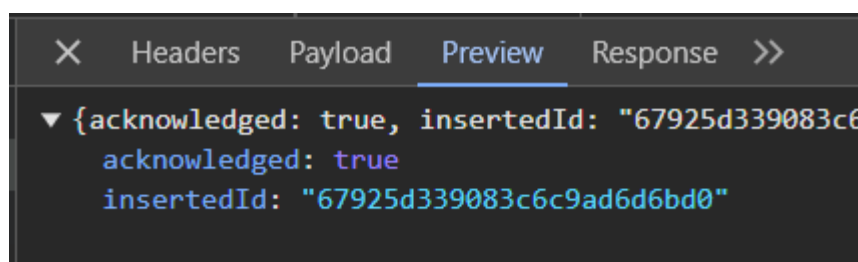
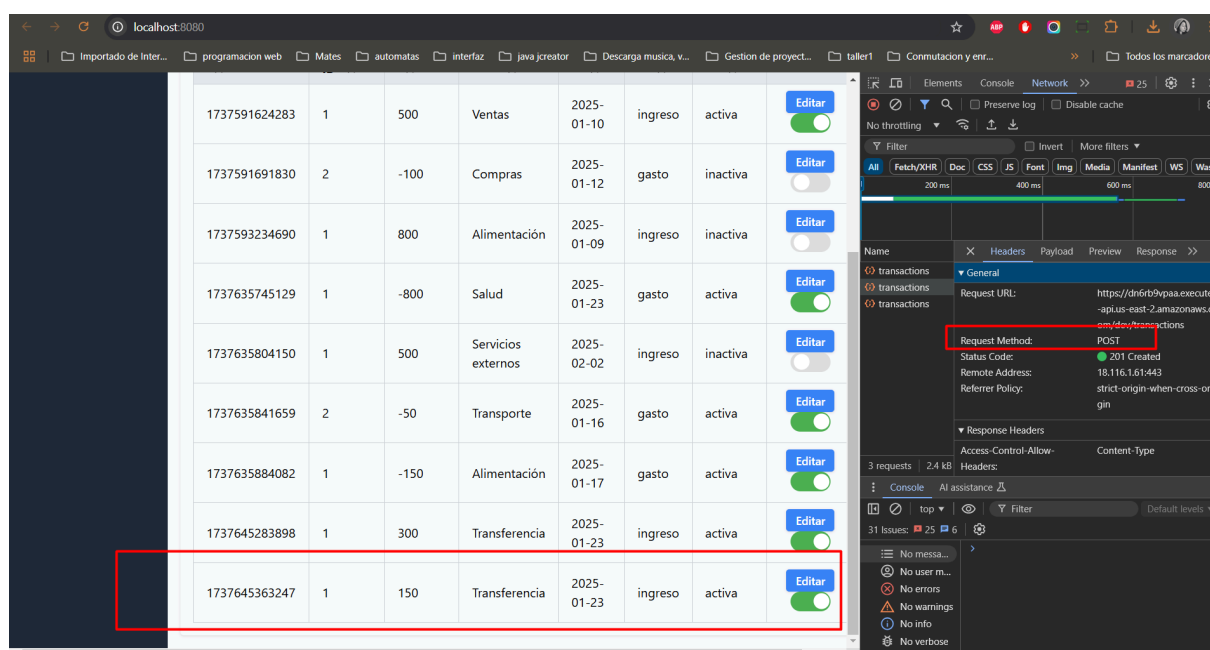
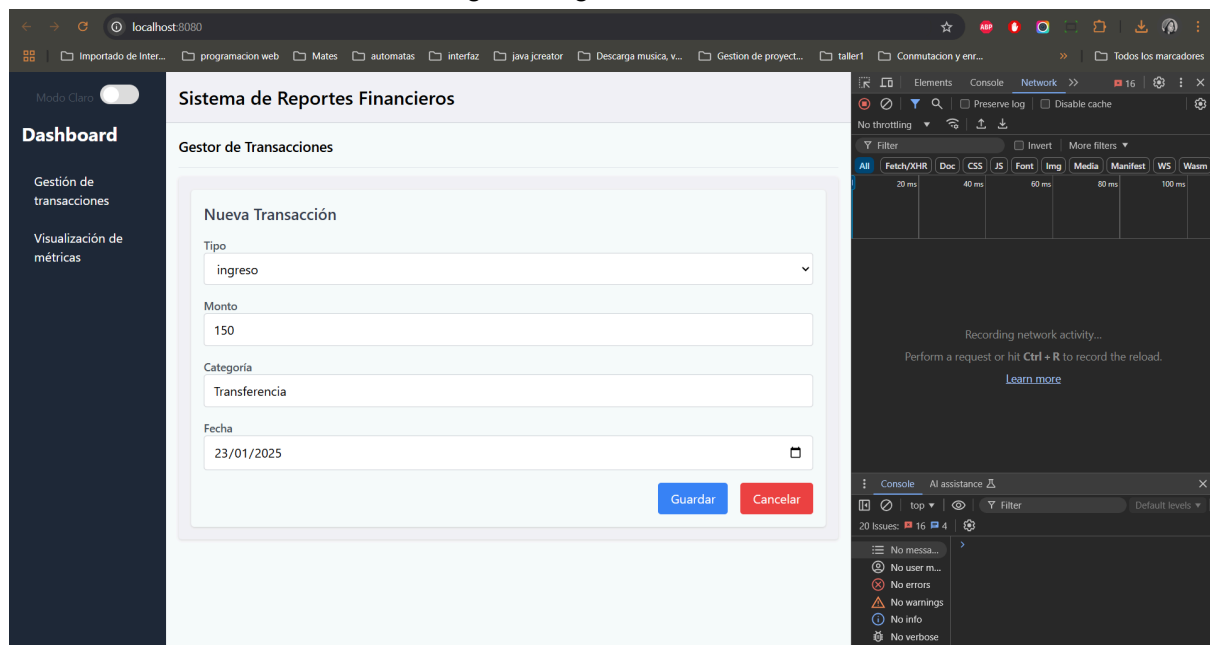
TRANSACCION	CLIENTE	MONTO	CATEGORIA	FECHA	TIPO	ESTADO	ACCIONES
T1	ID T1	T1	T1	T1	T1	T1	
1737591624283	1	500	Ventas	2025-01-10	ingreso	activa	<span>Editar</span>
1737591691830	2	-100	Compras	2025-01-12	gasto	inactiva	<span>Editar</span>
1737593234690	1	800	Alimentación	2025-01-09	ingreso	inactiva	<span>Editar</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 Panel:**

- Request URL: <https://dr6r6t9ypas.xecute-api-us-east-2.amazonaws.com/dev/transactions>
- 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	Editar
1737591691830	2	-100	Compras	2025-01-12	gasto	inactiva	Editar
1737593234690	1	800	Alimentación	2025-01-09	ingreso	inactiva	Editar
1737635745129	1	-800	Salud	2025-01-23	gasto	activa	Editar
1737635804150	1	500	Servicios externos	2025-02-02	ingreso	inactiva	Editar
1737635841659	2	-50	Transporte	2025-01-16	gasto	activa	Editar
1737635884082	1	-150	Alimentación	2025-01-17	gasto	activa	Editar
1737645283898	1	300	Transferencia	2025-01-23	ingreso	activa	Editar
1737645363247	1	200	Transferencia	2025-01-23	ingreso	activa	Editar

```
{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

Modo Claro

Dashboard

Gestión de transacciones

Visualización de métricas

Sistema de Reportes Financieros

Gestor de Transacciones

Lista de Transacciones

Filtrar por Cliente ID...

Buscar por Categoría...

Selecciona un tipo

01/01/2025

31/01/2025

Limpiar

+ Agregar Transacción

Descargar XLSX

TRANSACCION	CLIENTE ID	MONTO	CATEGORIA	FECHA	TIPO	ESTADO	ACCIONES
1737591624283	1	500	Ventas	2025-01-10	ingreso	activa	<div>Editar</div> <div></div>
1737591691830	2	-100	Compras	2025-01-12	gasto	inactiva	<div>Editar</div> <div></div>
1737593234690	1	800	Alimentación	2025-01-09	ingreso	inactiva	<div>Editar</div> <div></div>
1737635745129	1	-800	Salud	2025-01-23	gasto	activa	<div>Editar</div> <div></div>
1737635884082	1	-150	Alimentación	2025-01-17	gasto	activa	<div>Editar</div> <div></div>
1737645283898	1	300	Transferencia	2025-01-23	ingreso	activa	<div>Editar</div> <div></div>

ID	ClientelD	Monto	Categoria	Fecha	Tipo	Estado
1.7376E+12	1	500	Ventas	2025-01-10	ingreso	activa
1.7376E+12	2	-100	Compras	2025-01-12	gasto	inactiva
1.7376E+12	1	800	Alimentación	2025-01-09	ingreso	inactiva
1.7376E+12	1	-800	Salud	2025-01-23	gasto	activa
1.7376E+12	1	-150	Alimentación	2025-01-17	gasto	activa
1.7376E+12	1	300	Transferencia	2025-01-23	ingreso	activa

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> <div></div>
1737635745129	1	-800	Salud	2025-01-23	gasto	activa	<div>Editar</div> <div></div>
1737635841659	1	-50	Transporte	2025-02-07	gasto	activa	<div>Editar</div> <div></div>
1737635884082	1	-150	Alimentación	2025-01-17	gasto	activa	<div>Editar</div> <div></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?

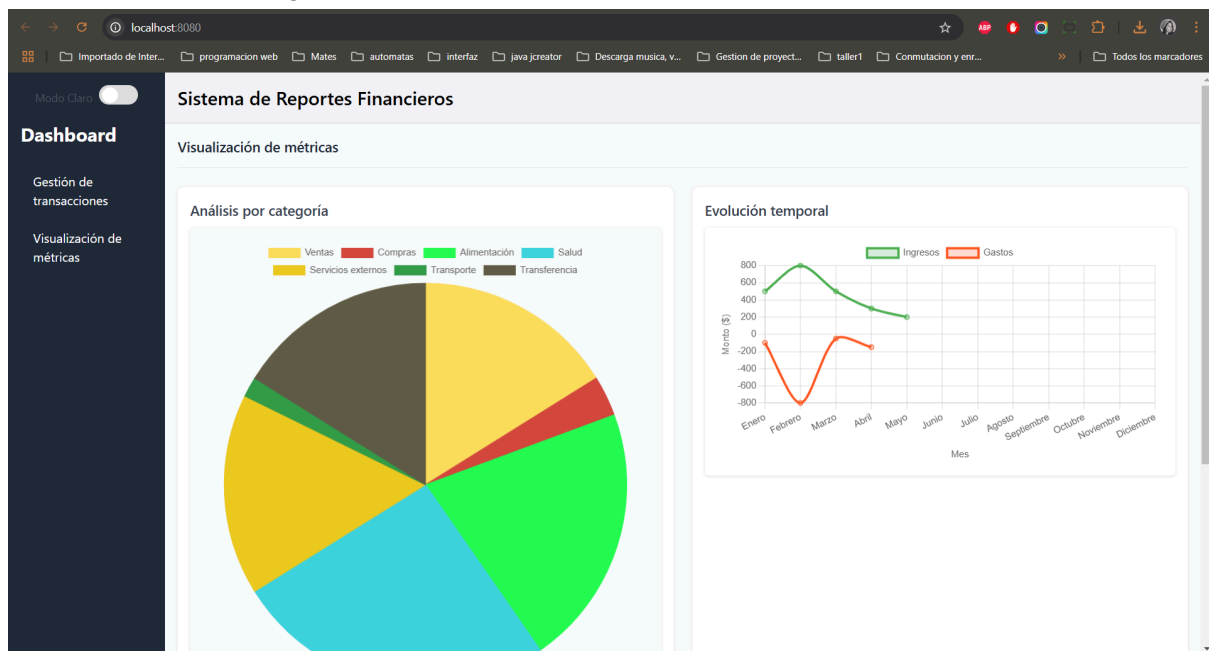
HABILITAR EDICIÓN Para editar archivos en esta versión gratuita de Excel, deben guardarse en OneDrive. Introducción Más información

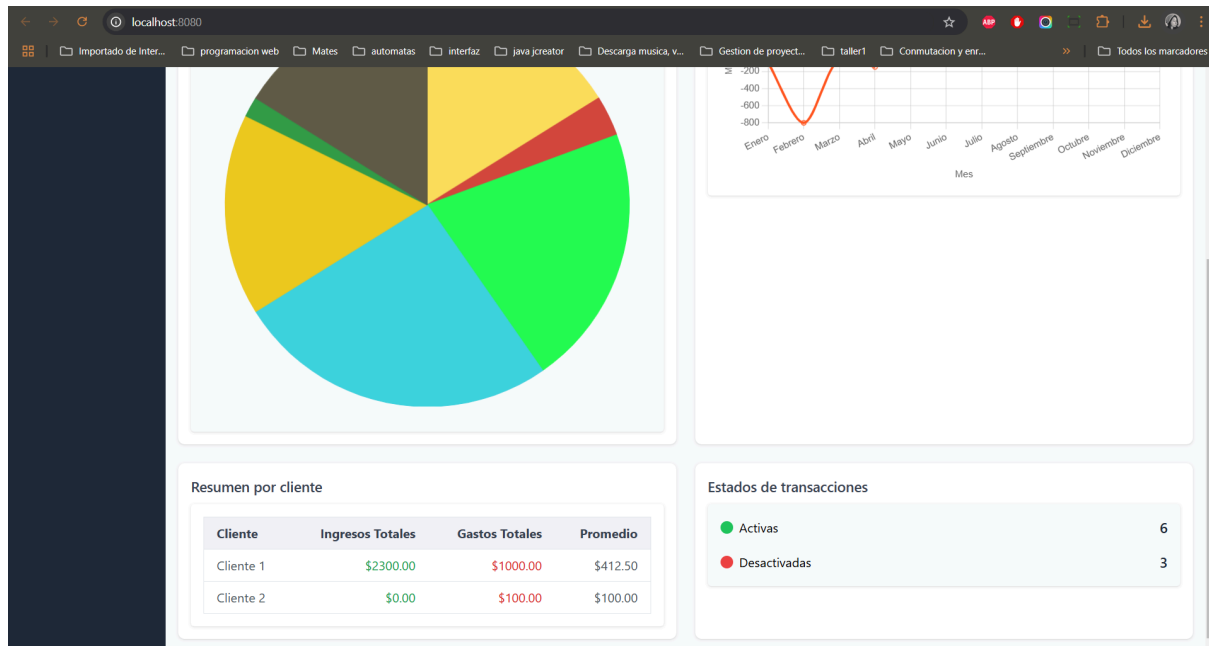
	A	B	C	D	E	F	G	H
1	ID	ClientelID	Monto	Categoría	Fecha	Tipo	Estado	
2	1.7376E+12	2	-100	Compras	2025-01-12	gasto	inactiva	
3								

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

Versiones de node y npm

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\GlendyD\Documents\ForteInnovation\client-financial-reports> node --version
v22.11.0
PS C:\Users\GlendyD\Documents\ForteInnovation\client-financial-reports> npm --version
10.9.0
```

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)