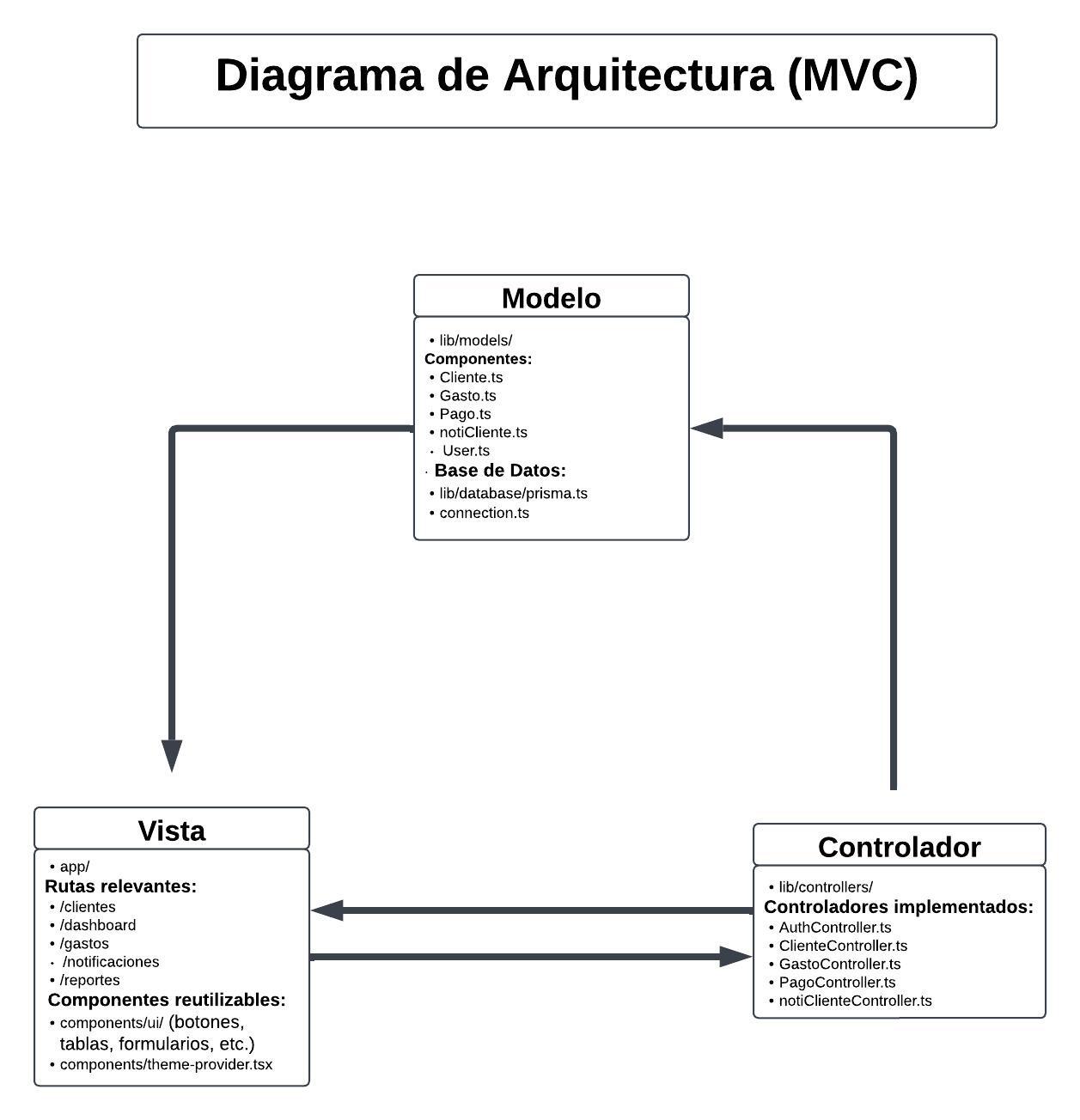
## ****Documentación del Proyecto: Diagrama de Arquitectura****

### **1. TELTEC Diagrama**



### **2. Propósito del Diagrama**

El presente diagrama de arquitectura tiene como objetivo describir la organización lógica del sistema basado en la arquitectura MVC (Modelo-Vista-Controlador).

Este enfoque facilita la separación de responsabilidades, mantenimiento y escalabilidad del sistema.

### **3. Estilo de Arquitectura**

**Modelo-Vista-Controlador (MVC)**  
Permite separar:

* La lógica de negocio (Modelo)
* La interfaz de usuario (Vista)
* El procesamiento de eventos o lógica de control (Controlador).

### **4. Descripción de Componentes**

#### **4.1. Modelo**

Encargado de la lógica de negocio, manipulación de datos y comunicación con la base de datos.

* **Ubicación:** lib/models/
* **Componentes:**
* Cliente.ts
* Gasto.ts
* Pago.ts
* notiCliente.ts
* User.ts
* **Base de Datos:**
  + Definida en lib/database/prisma.ts y connection.ts

#### **4.2. Vista**

Representa la interfaz gráfica y páginas que el usuario final visualiza e interactúa.

* **Ubicación:** app/
* **Rutas relevantes:**
* /clientes
* /dashboard
* /gastos
* /notificaciones
* /reportes
* **Componentes reutilizables:**
  + components/ui/ (botones, tablas, formularios, etc.)
  + components/theme-provider.tsx

#### **4.3. Controlador**

Gestiona la lógica que responde a los eventos del usuario, interactúa con el modelo y actualiza la vista.

* **Ubicación:** lib/controllers/
* **Controladores implementados:**
* AuthController.ts
* ClienteController.ts
* GastoController.ts
* PagoController.ts
* notiClienteController.ts

### **5. Flujo de Comunicación**

1. El **usuario interactúa** con la Vista ( desde /clientes).
2. El evento es **capturado por el Controlador** correspondiente (ClienteController.ts).
3. El Controlador **solicita o modifica información en el Modelo** (Cliente.ts).
4. El Modelo realiza las operaciones necesarias y retorna los datos.
5. El Controlador **envía los datos actualizados a la Vista**, que se renderiza.

### **6. Herramientas y Lenguajes Utilizados**

* **Frontend:** Next.js (React + TypeScript)
* **Backend:** Node.js + Prisma ORM
* **Base de Datos:** PostgreSQL
* **Estilos:** TailwindCSS
* **Gestión de dependencias:** PNPM