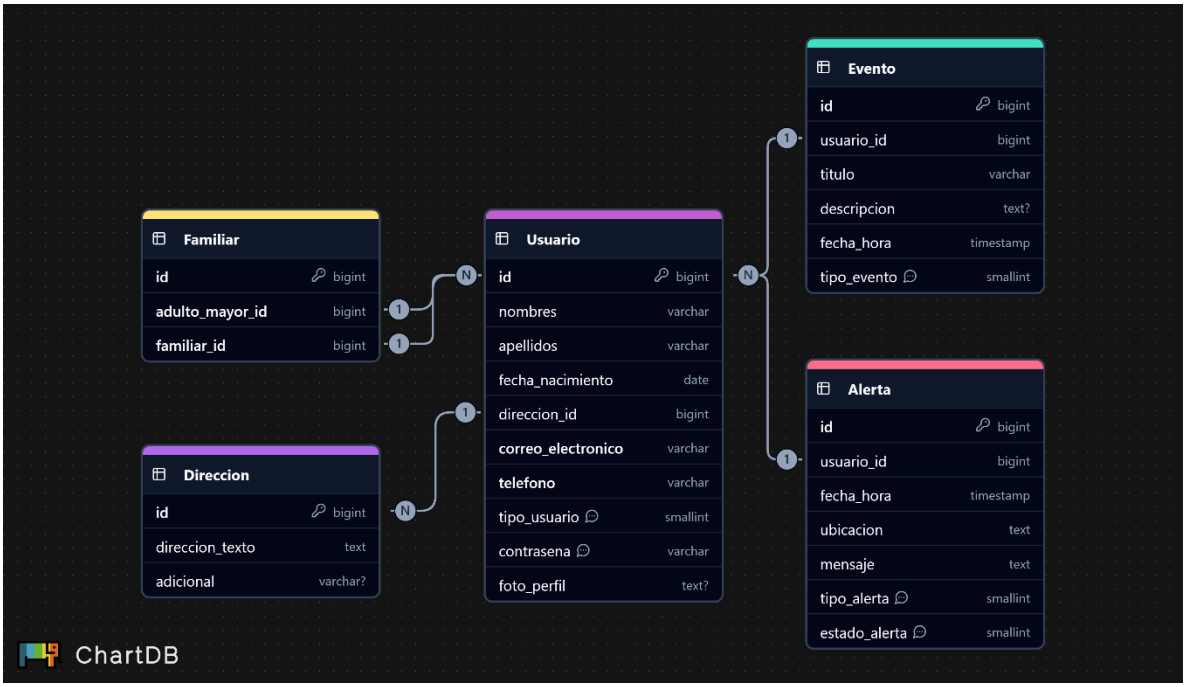
1. Diagrama de Modelo Entidad-Relación (ER)

El modelo relacional de TatasApp está compuesto por las siguientes entidades:

- Usuario: Almacena información personal (nombres, apellidos, correo, etc.) y se relaciona con Direccion mediante direccion_id. También tiene un campo tipo_usuario para roles.
- Familiar: Establece relaciones entre usuarios (ejemplo: adulto mayor y su familiar) mediante claves foráneas (adulto_mayor_id, familiar_id).
- Evento: Registra actividades creadas por usuarios (usuario_id), con detalles como título, fecha/hora y tipo.
- Alerta: Gestiona alertas generadas por usuarios (usuario_id), incluyendo ubicación, mensaje y estado.
- Dirección: Almacena direcciones físicas con un campo opcional (adicional).

Relaciones:

- Usuario-Direccion: 1 a 1 (un usuario tiene una dirección).
- Usuario-Evento/Alerta: 1 a N (un usuario puede crear múltiples eventos/alertas).
- Familiar-Usuario: N a N (implementada como tabla intermedia para relaciones familiares).



2. Diagrama de Componentes (Cliente-Servidor)

Cliente (App móvil):

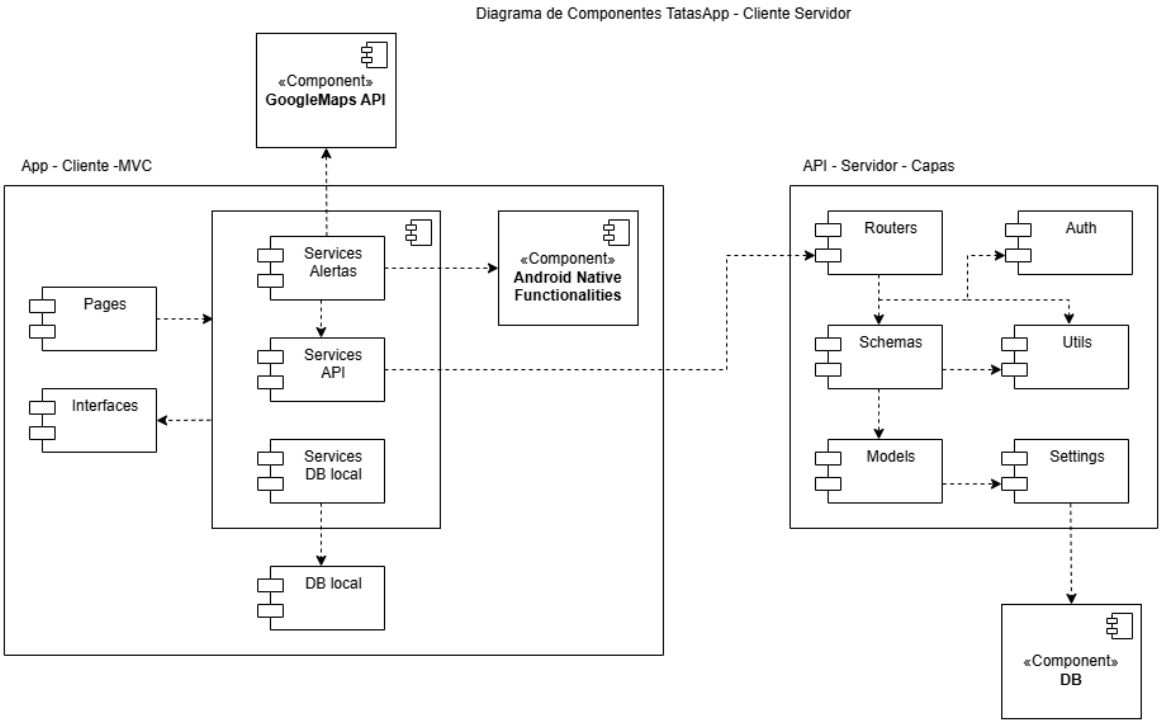
Arquitectura MVC

- Modelo: Gestiona datos locales (DB local) y estructura de datos.
- Vista: Representado en "Pages" e "Interfaces" (UI).
- Controlador: Coordina lógica mediante "Services" (Alertas, API, DB local).
- Integraciones: Usa Google Maps API y funcionalidades nativas de Android (ejemplo: GPS, notificaciones).

Servidor (API):

Arquitectura en Capas

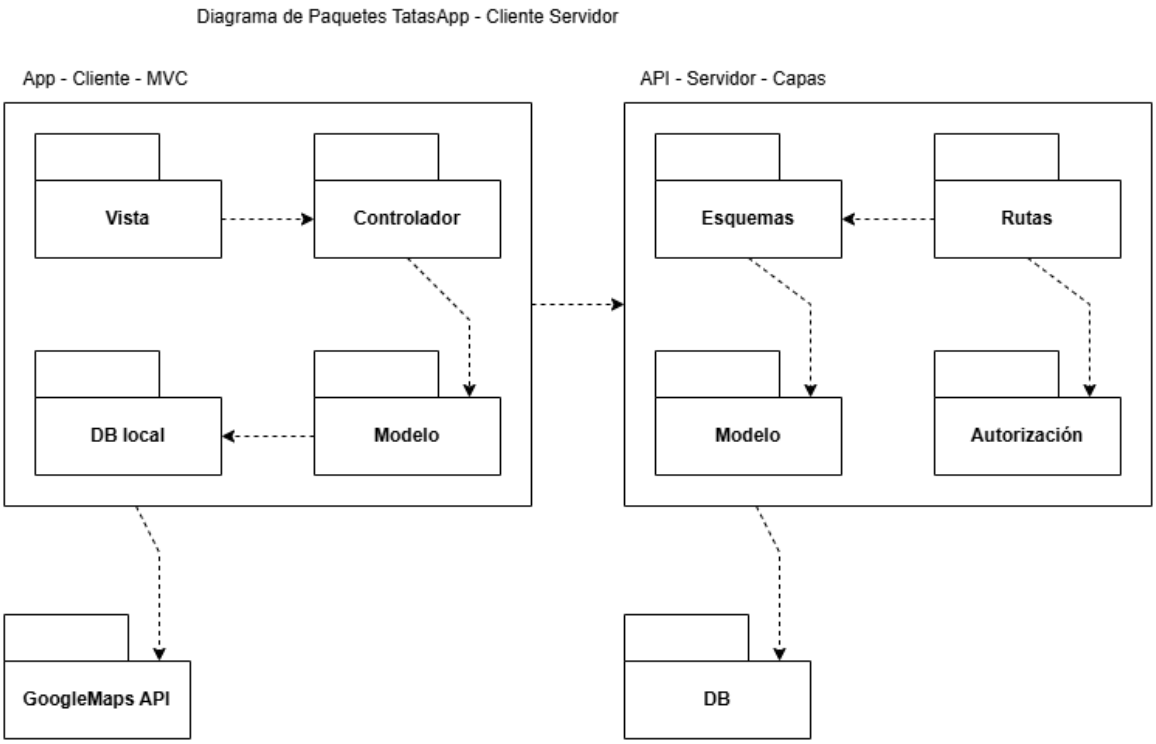
- Routers: Maneja endpoints y redirección de solicitudes.
- Auth: Gestiona autenticación/autorización.
- Models/Schemas: Define estructuras de datos y validaciones.
- DB: Capa de persistencia (base de datos central).



3. Diagrama de Paquetes

Refuerza la estructura cliente-servidor:

- Cliente (MVC): Paquetes como Vista (interfaz), Controlador (lógica), y Modelo (datos locales).
- Servidor (Capas): Paquetes separados para Rutas, Esquemas, Modelo, y Autorización.



Justificación de la Arquitectura Cliente-Servidor

Escalabilidad: El servidor centralizado (API) permite manejar múltiples clientes (móviles, web futuros) sin modificar la lógica de negocio. La base de datos local en el cliente (SQLite) optimiza el acceso offline a datos críticos (ejemplo: alertas).

Mantenibilidad: MVC en el cliente: Separa responsabilidades (UI vs. lógica), facilitando actualizaciones. Capas en el servidor: Aísla preocupaciones (ejemplo: auth, DB), simplificando pruebas y extensión.

Rendimiento: El cliente procesa UI y datos locales rápidamente, mientras el servidor maneja operaciones pesadas (ejemplo: geolocalización con Google Maps API).

Seguridad: La capa Auth en el servidor centraliza la gestión de accesos, protegiendo datos sensibles (ejemplo: información de usuarios).

Flexibilidad: APIs claras permiten integrar nuevos servicios (ejemplo: notificaciones push) sin afectar al cliente.

Conclusión

La arquitectura cliente-servidor con MVC en el cliente (para modularidad) y capas en el servidor (para escalabilidad) es ideal para TatasApp. El modelo relacional soporta relaciones complejas (ejemplo: familiares) mientras mantiene consistencia. Esta estructura equilibra rendimiento, seguridad y capacidad de crecimiento.