Marzo, 22 2025.

**MANUAL TECNICO PADILLAROUTE.**

v 1.0.0

TABLA DE CONTENIDO

[**INTRODUCCION** 4](#_Toc193796873)

[**CAPITULO 1: ARQUITECTURA.** 5](#_Toc193796874)

[**A) PATRON ARQUITECTONICO.** 5](#_Toc193796875)

[**CAPITULO 2: BASE DE DATOS.** 7](#_Toc193796876)

[**B) REALTIME DATABASE.** 7](#_Toc193796877)

[**C) OBJECTBOX STORE.** 7](#_Toc193796878)

[**CAPITULO 3: BACKEND.** 7](#_Toc193796879)

[**A) APIS.** 7](#_Toc193796880)

[**i.** **MAPS.** 7](#_Toc193796881)

[**ii.** **GEOLOCATOR.** 7](#_Toc193796882)

[**B) MODELOS.** 7](#_Toc193796883)

[**i.** **USUARIOS.** 7](#_Toc193796884)

[**ii.** **VEHICULOS.** 7](#_Toc193796885)

[**iii.** **PARADAS.** 7](#_Toc193796886)

[**iv.** **RUTAS.** 7](#_Toc193796887)

[**v.** **VIAJES.** 7](#_Toc193796888)

[**vi.** **INCIDENTES.** 7](#_Toc193796889)

[**vii.** **LOGS.** 7](#_Toc193796890)

[**C) SERVICIOS.** 7](#_Toc193796891)

[**i.** **CONECTOR REALTIME DATABASE.** 7](#_Toc193796892)

[**ii.** **CONECTOR OBJECTBOX STORE.** 7](#_Toc193796893)

[**iii.** **AUTENTICACIÓN.** 7](#_Toc193796894)

[**iv.** **NOTFICACIÓNES.** 7](#_Toc193796895)

[**v.** **GESTION DE USUARIOS.** 7](#_Toc193796896)

[**vi.** **GESTION DE VEHICULOS.** 7](#_Toc193796897)

[**vii.** **GESTION DE PARADAS.** 7](#_Toc193796898)

[**viii.** **GESTION DE RUTAS.** 7](#_Toc193796899)

[**ix.** **GESTION DE VIAJES.** 8](#_Toc193796900)

[**x.** **GESTION DE INCIDENTES.** 8](#_Toc193796901)

[**xi.** **LOGS DEL SISTEMA.** 8](#_Toc193796902)

[**CAPITULO 4: INTERFAZ (FRONTEND).** 8](#_Toc193796903)

[**A) PANTALLAS.** 8](#_Toc193796904)

[**B) WIDGETS PERSONALIZADOS.** 8](#_Toc193796905)

[**CAPITULO 5: SEGURIDAD.** 8](#_Toc193796906)

[**A) AUTENTICACIÓN DE IDENTIDAD DE USUARIOS.** 8](#_Toc193796907)

[**B) BUENAS PRACTICAS.** 8](#_Toc193796908)

[**i.** **ENCRIPTACION DE CONTRACSEÑAS.** 8](#_Toc193796909)

[**C) PERMISOS**. 8](#_Toc193796910)

# **INTRODUCCION**

Este manual técnico describe el diseño, arquitectura y funcionamiento de [nombre de la aplicación], una aplicación desarrollada en **Flutter** con almacenamiento local en **ObjectBox** y sincronización con Firebase Realtime **Database**.

El objetivo de esta aplicación es [breve descripción del propósito de la app]. Para lograrlo, se ha diseñado con una arquitectura modular y escalable, permitiendo una fácil expansión y mantenimiento.

A lo largo de este documento, se detallarán los componentes clave**, la estructura de la base de datos, las interacciones con Firebase y las mejores prácticas utilizadas en su desarrollo**. Este manual está dirigido a desarrolladores, ingenieros de software y cualquier persona interesada en comprender el funcionamiento interno de la aplicación.

***Nota:******se presupone que quien esté leyendo esto tiene, de antemano, conocimientos en el stack mencionado anteriormente y usado para este proyecto. Por lo cual, no nos centraremos en explicar cosas como la estructura de un proyecto de Flutter, qué es Firebase, patrones de diseño, etc.***

# **ARQUITECTURA.**

## **PATRON ARQUITECTONICO.**

Este proyecto utiliza una versión personalizada del patrón de diseño arquitectónico **Modelo-Vista-Controlador**; dentro de lo poco que se personalizó se encuentran los nombres de los componentes: **Modelo-Pantalla-Servicio**.

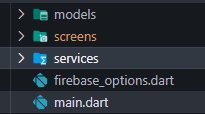


Imagen 1.1 Patrón arquitectónico de la aplicación.

* **Models** contiene los modelos de datos necesarios para trabajar con las bases de datos local y remota (Realtime Database y ObjectBox Store respectivamente).
* **Screens**, justo como lo están pensado, contiene las pantallas y widgets utilizados para construir la interfaz gráfica de la aplicación.
* **Services** guarda toda la lógica de negocios; conectores a bases de datos, servicios para realizar operaciones CRUD, servicios de autenticación de identidad, generadores de archivos, entre otros. Mas adelante se explorará a profundidad.

## **FLUJO DE DATOS.**

Revisemos cómo funciona el sistema. Se pueden distinguir tres actores distintos: un encargado de la administración de recursos como usuarios, vehículos, reportes, rutas, etc; otro cuya tarea es llevar el registro de los viajes en tiempo real; finalmente la base de datos remota, que es el principal centro de almacenamiento de datos.

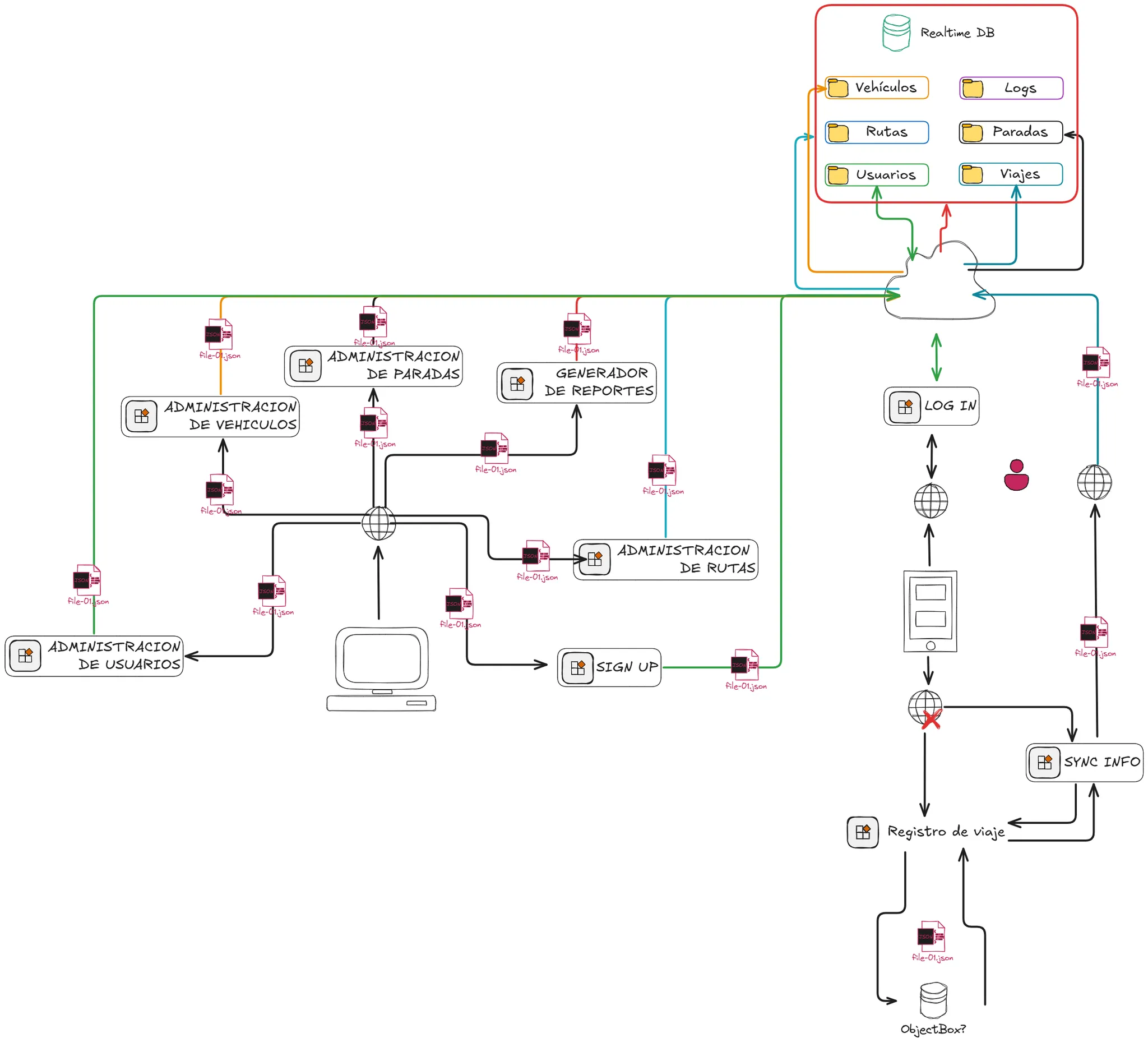


Imagen 1.2 Flujo de datos del sistema.

# **BASE DE DATOS.**

El proyecto utiliza dos bases de datos cuyos fines son distintos; por un lado, Realtime Database, cómo principal medio de almacenamiento; por el otro lado, Objectbox Storage, para almacenamiento local temporal.

## **REALTIME DATABASE.**

Esta es el principal medio de almacenamiento de la aplicación. La base de datos contiene 7 colecciones:

1. Usuarios.
2. Vehículos.
3. Paradas.
4. Rutas.
5. Registro de viajes.
6. Registro de incidentes.
7. Logs.

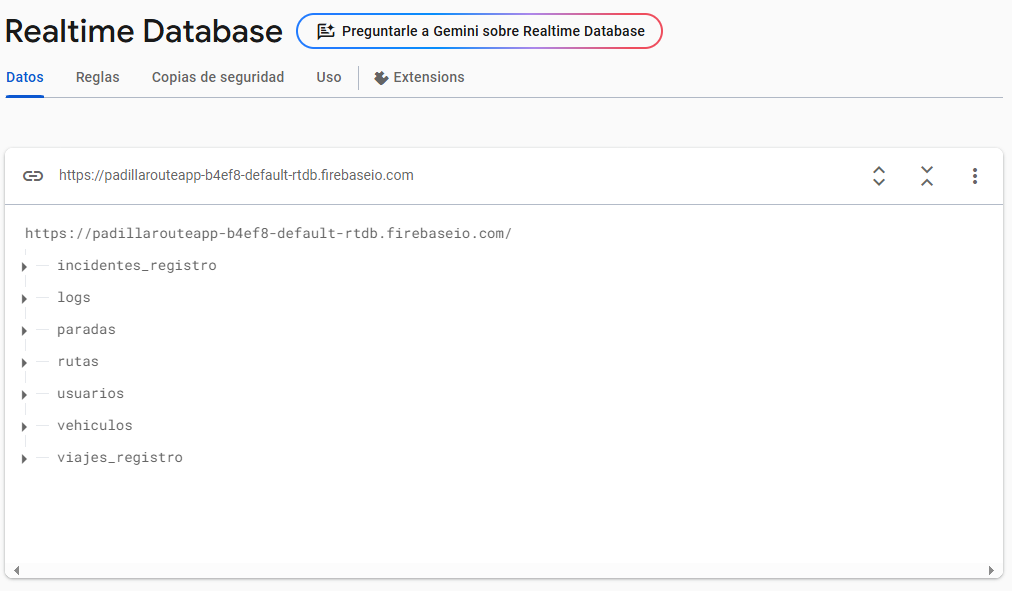


Imagen 2.1 Base de datos de la aplicación.

A continuación, se detallan los modelos de datos de cada colección:

### **MODELOS**.

### **USUARIOS.**

Representa a los empleados de la empresa y los clasifica en tres tipos: choferes, administrativos y gerentes mediante el uso de un enum llamado Rol.

enum Rol {

  @JsonValue('chofer')

  chofer,

  @JsonValue('administrativo')

  administrativo,

  @JsonValue('gerente')

  gerente

}

También se definen los campos necesarios; Id único del usuario, nombre, apellidos, numero de teléfono, correo electrónico, contraseña, rol, estatus, id del vehículo usado, token personal para mensajeria.

final int idUsuario;

final String nombre;

final String apellidos;

final int? Teléfono;

final String correo;

final String contrasena;

final Rol rol;

final bool activo;

final int? idVehiculo;

final String? fcmToken;

### **VEHICULOS.**

Son las unidades que los choferes usan para realizar su labor: transporta personal a sus diferentes destinos. Se define el identificador único, matricula, marca, modelo, capacidad de pasajeros, número de serie, estatus.

  final int idVehiculo;

  final String placa;

  final String marca;

  final String modelo;

  final int capacidad;

  final int numeroSerie;

  final Estatus estatus;

El estatus se refiere a si la unidad esta en uso activo o en mantenimiento.

enum Estatus {

  @JsonValue('activo')

  activo,

  @JsonValue('inactivo')

  inactivo,

  @JsonValue('mantenimiento')

  mantenimiento,

}

### **PARADAS.**

Las paradas son los puntos en los que los pasajeros abordan los vehículos de la empresa. Se define el identificador único, nombre de la parada, hora en la que se debe llegar, hora en la que se debe partir y las coordenadas en las que se localiza.

  final int idParada;

  final String nombre;

  final String horaLlegada;

  final String horaSalida;

  final String coordenadas;

### **RUTAS.**

Son colecciones de paradas que los choferes deben recorrer para llevar a sus pasajeros a sus destinos. Define el identificador único, identificador único del chofer encargado, identificador único del vehículo usado, nombre de la ruta, parada de origen, parada de destino, colección de todas las paradas que la conforman.

  final int idRuta;

  final int idChofer;

  final int idVehiculo;

  final String nombre;

  final String origen;

  final String destino;

  final List<String> paradas;

### **VIAJES.**

Son los registros de cada viaje realizado a manera de bitácora; cada registro tiene definido un identificador único, identificador de la ruta recorrida, identificador del chofer responsable, identificador del vehículo usado, lista de paradas con el numero de pasajeros recogidos, hora de inicio y termino del viaje, duración total del viaje, cantidad de pasajeros transportados, distancia recorrida, velocidad promedio del vehículo.

  final int idRegistro;

  final int idRuta;

  final int idVehiculo;

  final int idUsuario;

  final Map<String, dynamic> paradasRegistro;

  final String horaInicio;

  final String horaFinal;

  final int tiempoTotal;

  final int totalPasajeros;

  final int distanciaRecorrida;

  final int velocidadPromedio;

### **INCIDENTES.**

Los incidentes son situaciones excepcionales, previstas o no, que obstaculizan el cumplimiento de la labor de los choferes. Pueden ser retrasos o desvíos debido a tráfico, averías, etc. No son estrictamente accidentes con víctimas mortales.

Define un identificador único, identificador del usuario autor, identificador del vehículo usado, descripción del incidente y fecha en que se registró.

  final int idRegistro;

  final int idUsuario;

  final int idVehiculo;

  final String descripcion;

  final String fecha;

## **OBJECTBOX STORAGE.**

Objectbox es la base de datos local utilizada únicamente para almacenar los registros de los viajes y sincronizarlos con Realtime Database cuando es necesario o posible hacerlo.

Define el identificador único del registro, de la ruta seguida, del chofer y el vehículo; la lista de paradas y cantidad de pasajeros que abordaron en cada una; hora de inicio y termino, duración total, total de pasajeros, distancia recorrida y velocidad promedio del vehículo.

  int id = 0;

  int idRuta;

  int idVehiculo;

  int idChofer;

  String paradasRegistro; // Store paradasRegistro as JSON

  String horaInicio;

  String horaFinal;

  int tiempoTotal;

  int totalPasajeros;

  double distanciaRecorrida;

  double velocidadPromedio;

# **BACKEND.**

## **APIS.**

### **MAPS.**

### **GEOLOCATOR.**

## **SERVICIOS.**

### **CONECTOR REALTIME DATABASE.**

### **CONECTOR OBJECTBOX STORE.**

### **AUTENTICACIÓN.**

### **NOTFICACIÓNES.**

### **GESTION DE USUARIOS.**

### **GESTION DE VEHICULOS.**

### **GESTION DE PARADAS.**

### **GESTION DE RUTAS.**

### **GESTION DE VIAJES.**

### **GESTION DE INCIDENTES.**

### **LOGS DEL SISTEMA.**

# **INTERFAZ (FRONTEND).**

## **PANTALLAS.**

## **WIDGETS PERSONALIZADOS.**

# **SEGURIDAD.**

## **AUTENTICACIÓN DE IDENTIDAD DE USUARIOS.**

## **BUENAS PRACTICAS.**

### **ENCRIPTACION DE CONTRACSEÑAS.**

## **PERMISOS**.

TABLA DE IMAGENES.

[Imagen 1.1 Patrón arquitectónico de la aplicación. 5](#_Toc194841085)

[Imagen 1.2 Flujo de datos del sistema. 6](#_Toc194841086)

[Imagen 2.1 Base de datos de la aplicación. 7](#_Toc194841087)