GOYO

Documentación Trabajo Final de Grado

Jesús Muñoz García

Victoria FP

1. Introducción

1.1 Descripción del Proyecto

La aplicación **Goyo** es una plataforma web innovadora diseñada para revolucionar la gestión de citas veterinarias. Esta solución digital permite a los propietarios de mascotas programar, gestionar y hacer seguimiento de las citas médicas de sus animales de compañía de manera sencilla e intuitiva.

Goyo conecta eficientemente a los dueños de mascotas con personal veterinario, facilitando la reserva de citas, el acceso al historial médico de las mascotas y la comunicación directa con profesionales veterinarios. La aplicación ha sido desarrollada siguiendo las mejores prácticas de desarrollo de software, implementando una arquitectura escalable y una interfaz de usuario intuitiva que garantiza una experiencia óptima tanto para propietarios como para personal veterinario.

1.2 Contexto y Motivación

En la actualidad, muchas clínicas veterinarias aún dependen de sistemas de reserva tradicionales basados en llamadas telefónicas, lo que genera inconvenientes tanto para los propietarios de mascotas como para el personal veterinario. La motivación principal para el desarrollo de Goyo surge de la necesidad de digitalizar y optimizar este proceso, ofreciendo una solución moderna que se adapte a las necesidades del siglo XXI.

1.3 Alcance del proyecto

El proyecto Goyo abarca el desarrollo completo de una aplicación web para la gestión integral de citas veterinarias. El alcance incluye:

- **Sistema de gestión de usuarios**: Registro y perfiles para propietarios y clínicas veterinarias
- **Gestión de mascotas**: Perfiles completos con información médica y fotográfica
- **Sistema de reserva de citas**: Calendario interactivo con disponibilidad en tiempo real
- **Historial médico digital**: Registro completo de consultas, tratamientos y vacunas
- **Sistema de notificaciones**: Recordatorios automáticos y confirmaciones de citas
- **Panel de administración**: Herramientas para clínicas veterinarias para gestionar su agenda
- **Sistema de comunicación**: Mensajería entre propietarios y veterinarios
- **Reportes y estadísticas**: Análisis de datos para clínicas veterinarias

1.4 Valor Añadido

Goyo se diferencia de soluciones existentes por:

- **Enfoque específico veterinario**: Diseñado exclusivamente para las necesidades del sector veterinario
- **Facilidad de uso**: Interfaz intuitiva que pueden usar personas de todas las edades
- **Acceso 24/7**: Posibilidad de gestionar citas en cualquier momento del día
- **Historial centralizado**: Toda la información médica de las mascotas en un solo lugar
- **Recordatorios inteligentes**: Sistema automatizado de notificaciones para vacunas y revisiones
- **Diseño responsive**: Funciona perfectamente en móviles, tablets y computadoras

1.5 Beneficios para los Usuarios

Para Propietarios de Mascotas:

- Reserva de citas sin restricciones horarias
- Acceso completo al historial médico de sus mascotas
- Recordatorios automáticos de vacunas y revisiones
- Comunicación directa con el veterinario
- Gestión de múltiples mascotas en una sola cuenta

Para Clínicas Veterinarias:

- Reducción de llamadas telefónicas para reservas
- Gestión automatizada de la agenda
- Mejor organización del historial de pacientes
- Comunicación eficiente con los propietarios
- Análisis de datos para mejorar el servicio

2. Diseño de la base de datos

En esta app utiliza **Supabase** (PostgreSQL) como base de datos principal, implementando un diseño relacional optimizado para la gestión de citas veterinarias. La base de datos está estructurada en 4 tablas principales con relaciones bien definidas y políticas de seguridad.

2.1 Descripción de las tablas

Tabla `animales`

Almacena la información de las mascotas registradas en el sistema.

Campo	Tipo	Descripción	Restricciones
`id`	UUID	Identificador único	PRIMARY KEY, auto-generado
`nombre`	VARCHAR(100)	Nombre de la mascota	NOT NULL
`correo`	VARCHAR(255)	Email del propietario	UNIQUE, NOT NULL
`foto_url`	TEXT	URL de la foto de la mascota	Opcional
`contraseña`	TEXT	Contraseña encriptada	NOT NULL, bcrypt
`ubicacion`	VARCHAR(255)	Ciudad/ubicación	NOT NULL
`tipo`	VARCHAR(50)	Tipo de animal	CHECK constraint con 8 opciones
`raza`	VARCHAR(100)	Raza específica	NOT NULL
`edad`	VARCHAR(50)	Edad en formato libre	NOT NULL ("2 años", "6 meses")
`altura`	VARCHAR(50)	Altura en formato libre	NOT NULL ("30 cm", "1.2 metros")
`created_at`	TIMESTAMPTZ	Fecha de creación	DEFAULT NOW()
`updated_at`	TIMESTAMPTZ	Última actualización	Auto-actualizada

Índices: `correo`, `tipo`, `ubicacion` **Validaciones**: Tipos permitidos: Perro, Gato, Pájaro, Caballo, Conejo, Hamster, Pez, Reptil

Tabla `veterinarios`

Contiene los datos de los profesionales veterinarios.

Campo	Tipo	Descripción	Restricciones
`id`	UUID	Identificador único	PRIMARY KEY, auto-generado
`nombre`	VARCHAR(100)	Nombre completo	NOT NULL
`correo`	VARCHAR(255)	Email profesional	UNIQUE, NOT NULL
`foto_url`	TEXT	URL foto de perfil	Opcional
`contraseña`	TEXT	Contraseña encriptada	NOT NULL, bcrypt
`ubicacion`	VARCHAR(255)	Ciudad donde ejerce	NOT NULL
`especialidad`	VARCHAR(50)	Especialización	CHECK constraint
`numero_colegiado`	VARCHAR(50)	Número colegio profesional	Opcional
`años_experiencia`	INTEGER	Años de experiencia	DEFAULT 0
`telefono`	VARCHAR(20)	Teléfono de contacto	Opcional
`created_at`	TIMESTAMPTZ	Fecha de registro	DEFAULT NOW()
`updated_at`	TIMESTAMPTZ	Última actualización	Auto-actualizada

Índices: `correo`, `especialidad`, `ubicacion` **Validaciones**: Especialidades incluyen tipos de animales + "General"

Tabla `citas`

Gestiona todas las citas entre animales y veterinarios.

Campo	Tipo	Descripción	Restricciones
`id`	UUID	Identificador único	PRIMARY KEY, auto-generado
`animal_id`	UUID	Referencia al animal	FK to animales(id), CASCADE
`veterinario_id`	UUID	Referencia al veterinario	FK to veterinarios(id), CASCADE
`fecha`	DATE	Fecha de la cita	NOT NULL, no fines de semana
`hora_inicio`	TIME	Hora de inicio	NOT NULL
`hora_fin`	TIME	Hora de finalización	NOT NULL
`motivo`	TEXT	Motivo de la consulta	Opcional
`notas_veterinario`	TEXT	Notas del veterinario	Opcional
`estado`	VARCHAR(20)	Estado actual	CHECK constraint, DEFAULT 'programada'
`precio`	DECIMAL(10,2)	Costo de la consulta	Opcional
`created_at`	TIMESTAMPTZ	Fecha de creación	DEFAULT NOW()
`updated_at`	TIMESTAMPTZ	Última actualización	Auto-actualizada

Restricciones especiales:

- Horarios válidos: 09:00-13:30 ó 17:00-20:30
- Solo días laborables (lunes a viernes)
- No duplicar citas en mismo horario por veterinario
- Estados: programada, confirmada, en_curso, completada, cancelada, no_asistio

Índices: `animal_id`, `veterinario_id`, `fecha`, `estado`

Tabla horarios_veterinario

Define la disponibilidad horaria de cada veterinario.

Campo	Tipo	Descripción	Restricciones
id	UUID	Identificador único	PRIMARY KEY, auto-generado
veterinario_id	UUID	Referencia al veterinario	FK to veterinarios(id), CASCADE
dia_semana	INTEGER	Día (1=Lunes, 5=Viernes)	CHECK 1-5, NOT NULL
hora_inicio_mañana	TIME	Inicio turno mañana	DEFAULT '09:00:00'
hora_fin_mañana	TIME	Fin turno mañana	DEFAULT '13:30:00'
hora_inicio_tarde	TIME	Inicio turno tarde	DEFAULT '17:00:00'
hora_fin_tarde	TIME	Fin turno tarde	DEFAULT '20:30:00'
disponible_mañana	BOOLEAN	Disponibilidad mañana	DEFAULT true
disponible_tarde	BOOLEAN	Disponibilidad tarde	DEFAULT true
created_at	TIMESTAMPTZ	Fecha de creación	DEFAULT NOW()

Restricción: Un horario único por veterinario y día

2.2 Funciones y Triggers Implementados

Seguridad de Contraseñas

```
-- Encriptación automática con bcrypt
CREATE OR REPLACE FUNCTION encrypt_password()
RETURNS TRIGGER AS $$
BEGIN
    NEW.contraseña = crypt(NEW.contraseña, gen_salt('bf'));
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Actualización Automática de Timestamps

```
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = NOW();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Creación Automática de Horarios

Al registrar un veterinario, se crean automáticamente sus horarios de lunes a viernes.

2.3 Políticas de Seguridad (RLS)

El sistema implementa **Row Level Security** para proteger los datos:

- **Animales**: Solo pueden acceder a sus propios datos
- **Veterinarios**: Solo pueden acceder a sus propios datos
- **Citas**: Visibles solo para el animal y veterinario involucrados
- **Storage**: Políticas para subida y acceso a imágenes

2.4 Storage de Archivos

Buckets configurados:

- `animal-photos`: Fotos de mascotas (público)
- `veterinario-photos`: Fotos de perfil de veterinarios (público)

2.5 Diseño de la Interfaz

La aplicación Goyo está desarrollada con **Flutter**, implementando un diseño moderno, responsive y centrado en la usabilidad. La interfaz sigue los principios de **Material Design** adaptados al contexto veterinario.

3. Tecnologías Utilizadas

3.1 Frontend - Aplicación Móvil

3.1.1 Framework Principal

- **Flutter 3.6.0**: Framework de desarrollo multiplataforma de Google
    - Permite crear aplicaciones nativas para iOS, Android, Web y Desktop
    - Renderizado de alto rendimiento con engine gráfico Skia
    - Desarrollo con un solo código base (write once, run anywhere)
    - Hot reload para desarrollo ágil

3.1.2 Lenguaje de Programación

- **Dart 3.6.0**: Lenguaje optimizado para desarrollo de UI
    - Compilación AOT (Ahead-of-Time) para rendimiento nativo
    - Sistema de tipos robusto con null safety
    - Programación orientada a objetos con herencia, mixins e interfaces
    - Garbage collection automático

3.1.3 Paquetes y Dependencias Principales

Autenticación y Backend

```
supabase_flutter: ^2.5.6
```

- Cliente oficial de Supabase para Flutter
- Autenticación integrada (email/password, OAuth)
- Conexión en tiempo real con PostgreSQL
- Storage para archivos e imágenes

Interfaz de Usuario

```
cupertino_icons: ^1.0.8
```

- Iconos nativos de iOS para diseño multiplataforma
- Consistencia visual entre plataformas

Funcionalidades Específicas

```
image_picker: ^1.0.4        # Selección de fotos desde galería/cámara
table_calendar: ^3.0.9      # Calendario interactivo para citas
intl: ^0.19.0               # Internacionalización y formateo de fechas
shared_preferences: ^2.2.2   # Almacenamiento local (Remember me)
```

Herramientas de Desarrollo

```yaml flutter_lints: ^5.0.0 # Análisis estático de código flutter_launcher_icons: ^0.13.1 # Generación de iconos personalizados.
```

3.1.4 Arquitectura de UI

- **Material Design 3**: Sistema de diseño moderno y adaptativo
- **Cupertino Design**: Elementos nativos de iOS cuando es necesario
- **Responsive Design**: Adaptable a diferentes tamaños de pantalla
- **Widgets personalizados**: Componentes reutilizables específicos para veterinaria

3.1.5 Plataformas Soportadas

- **Android**: API 21+ (Android 5.0 Lollipop)
- **iOS**: iOS 12.0+
- **Web**: Navegadores modernos (Chrome, Firefox, Safari, Edge)
- **Windows**: Windows 10+
- **macOS**: macOS 10.14+
- **Linux**: Ubuntu 18.04+

3.2 Backend - Supabase (PostgreSQL)

3.2.1 Base de Datos

- **PostgreSQL 15**: Base de datos relacional robusta y escalable
    - ACID compliance para transacciones seguras
    - Extensiones avanzadas (uuid-ossp, pgcrypto)
    - Índices optimizados para consultas rápidas
    - Triggers y funciones almacenadas

3.2.2 Características de Supabase

- **API REST automática**: Generada automáticamente desde el schema
- **Realtime subscriptions**: Actualizaciones en tiempo real via WebSockets
- **Row Level Security (RLS)**: Seguridad a nivel de fila
- **Storage integrado**: Para archivos multimedia (fotos de mascotas/veterinarios)
- **Edge Functions**: Funciones serverless para lógica personalizada

3.2.3 Autenticación y Autorización

- **Autenticación JWT**: Tokens seguros para sesiones
- **Bcrypt encryption**: Encriptación de contraseñas con salt
- **Políticas de seguridad**: Control granular de acceso a datos
- **Session management**: Gestión automática de sesiones

3.2.4 Estructura de Datos

```
-- Tablas principales
animales           -- Registro de mascotas
veterinarios       -- Profesionales veterinarios
citas             -- Sistema de citas
horarios_veterinario -- Disponibilidad de veterinarios

-- Funciones personalizadas
encrypt_password()           -- Encriptación automática
verify_password()           -- Verificación de contraseñas
update_updated_at_column()  -- Timestamps automáticos
crear_horarios_default()   -- Horarios por defecto
```

3.3 Herramientas de Desarrollo

3.3.1 Control de Versiones

- **Git**: Sistema de control de versiones distribuido
- **GitHub**: Hosting del repositorio con integración CI/CD
- **Conventional Commits**: Formato estandardizado de commits

3.3.2 Entorno de Desarrollo

- **Visual Studio Code**: IDE principal recomendado
    - Extensión oficial de Flutter/Dart
    - IntelliSense avanzado para Dart
    - Debug integrado para todas las plataformas
    - Terminal integrado
- **Android Studio**: IDE alternativo con herramientas Android nativas
- **Xcode**: Para desarrollo y testing en iOS/macOS

3.3.3 Herramientas de Análisis

- **Flutter Doctor**: Diagnóstico del entorno de desarrollo
- **Dart Analyzer**: Análisis estático de código
- **Flutter Inspector**: Debug visual de widgets
- **Dart DevTools**: Profiling y análisis de rendimiento

3.3.4 Testing

```
flutter_test: sdk    # Framework de testing integrado
```

- **Unit Tests**: Pruebas de lógica de negocio
- **Widget Tests**: Pruebas de componentes UI
- **Integration Tests**: Pruebas end-to-end
- **Golden Tests**: Pruebas visuales de regresión

3.3.5 Build y Deployment

Configuración Multi-plataforma

```
flutter_launcher_icons: ^0.13.1
```

- **Generación automática de iconos** para todas las plataformas
- **Configuración específica** por plataforma (Android, iOS, Web, Desktop)
- **Build variants**: Debug, Profile, Release

Plataformas de Deployment

- **Google Play Store**: Distribución Android
- **Apple App Store**: Distribución iOS
- **Web Hosting**: Netlify, Vercel, Firebase Hosting
- **Microsoft Store**: Distribución Windows
- **Snapcraft**: Distribución Linux

3.4 Servicios y APIs Externas

3.4.1 Servicios de Supabase

- **Database API**: CRUD operations via REST
- **Auth API**: Autenticación y gestión de usuarios
- **Storage API**: Subida y gestión de archivos
- **Realtime API**: Suscripciones en tiempo real

3.4.2 Características de Seguridad

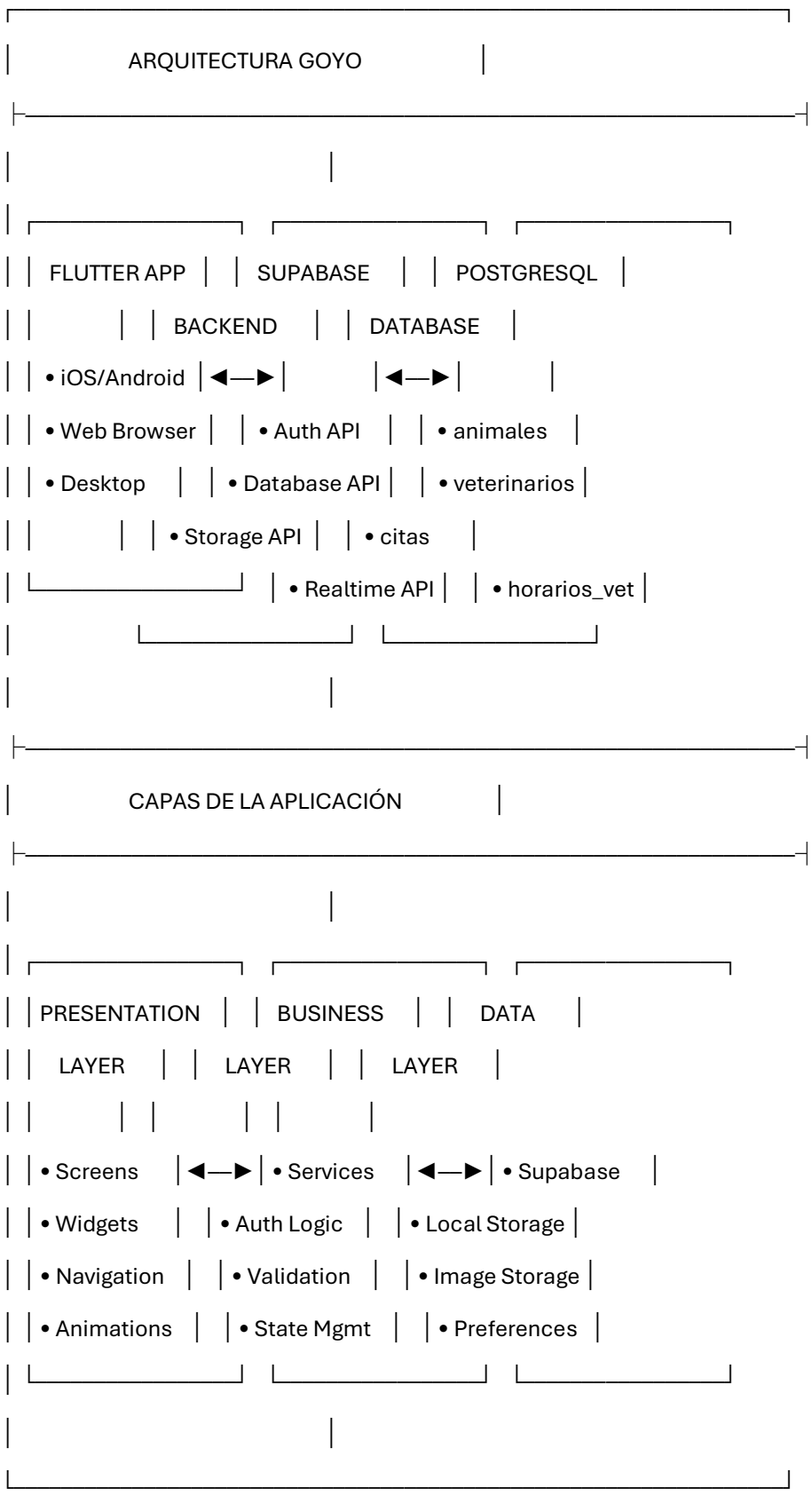
- **HTTPS/SSL**: Comunicación encriptada
- **JWT Tokens**: Autenticación segura
- **Rate Limiting**: Protección contra ataques
- **CORS Configuration**: Configuración de origen cruzado

4. Arquitectura del sistema

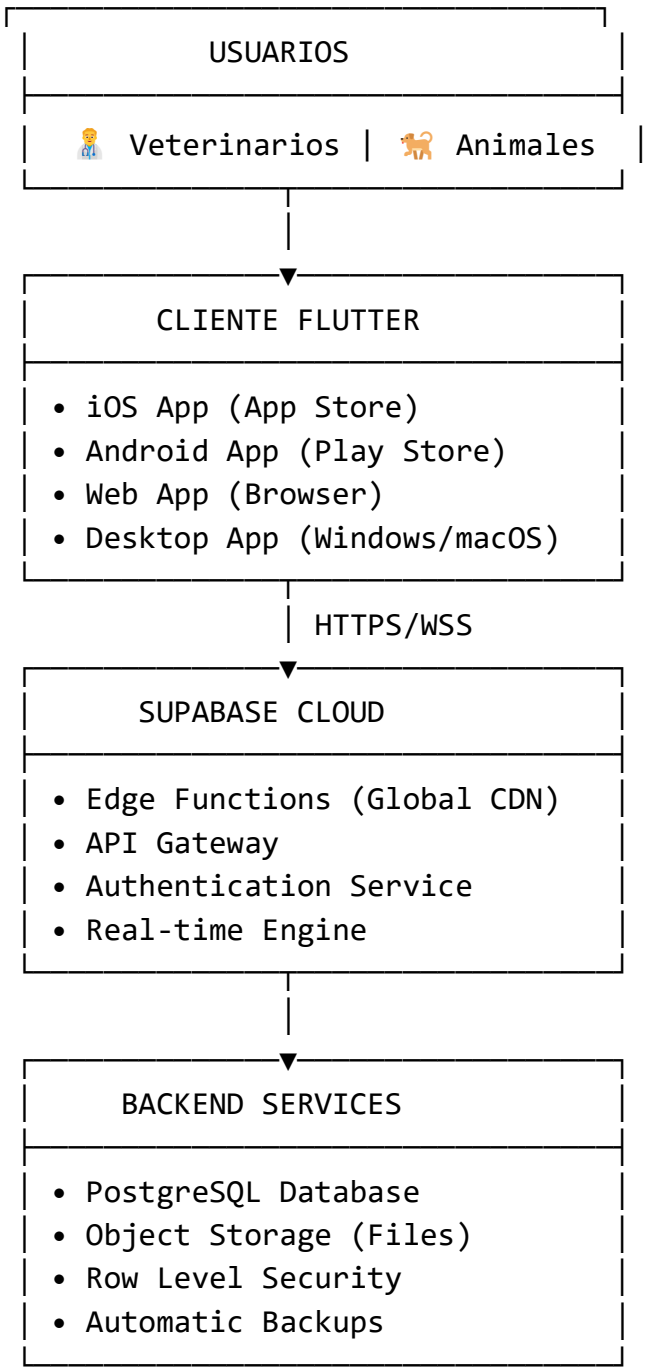
4. 1 Arquitectura general

La aplicación Goyo implementa una **arquitectura multiplataforma moderna** basada en **Flutter + Supabase**, siguiendo los principios de **separación de responsabilidades** y **escalabilidad horizontal**. El sistema está diseñado para operar de manera eficiente en móvil, web y desktop.

4.1.1 Diagrama de Arquitectura Principal

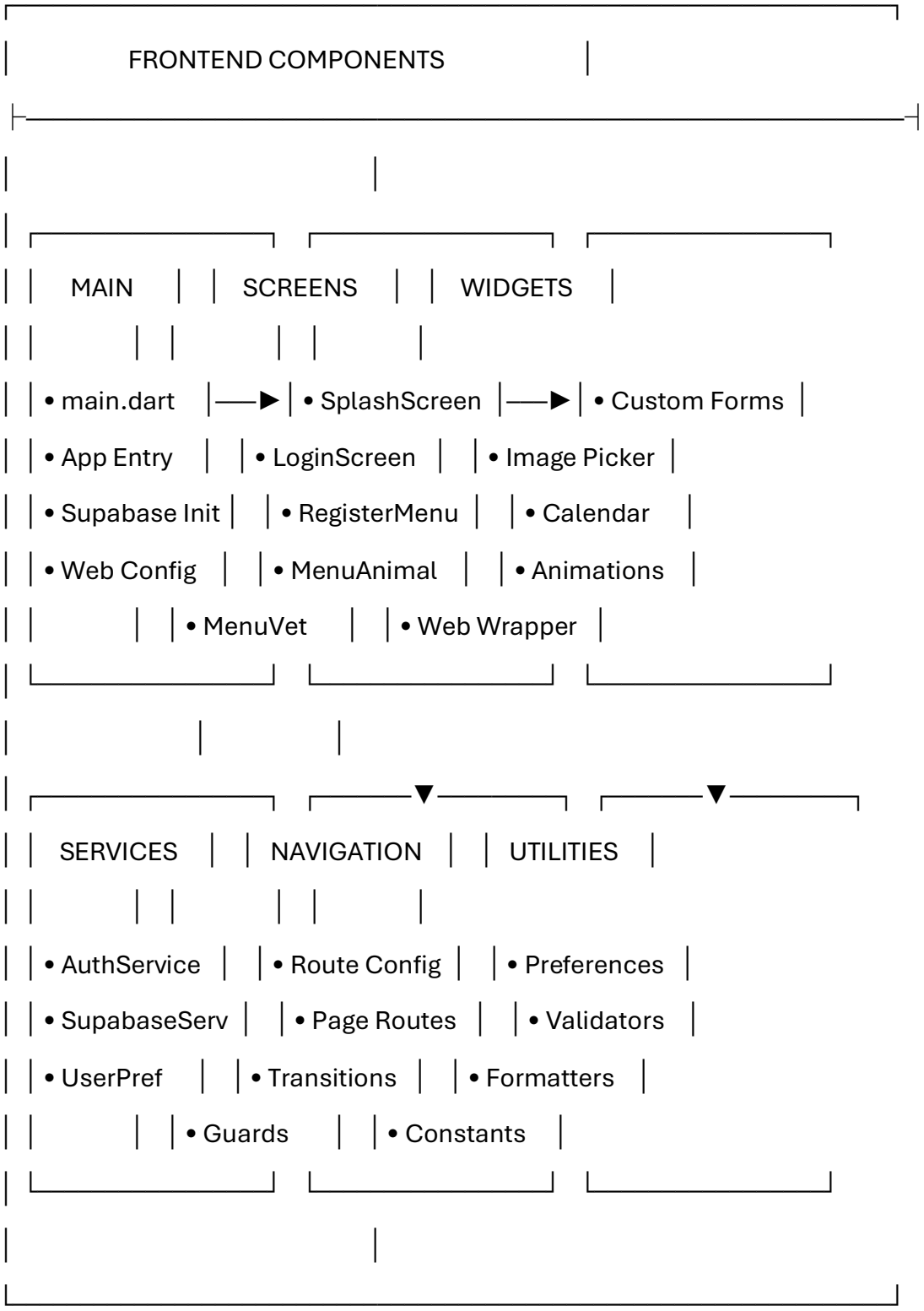


4.1.2 Arquitectura de Despliegue



4.2 Diagrama de Componentes

4.2.1 Componentes Frontend (Flutter)



4.2.3 Componentes Backend (Supabase)