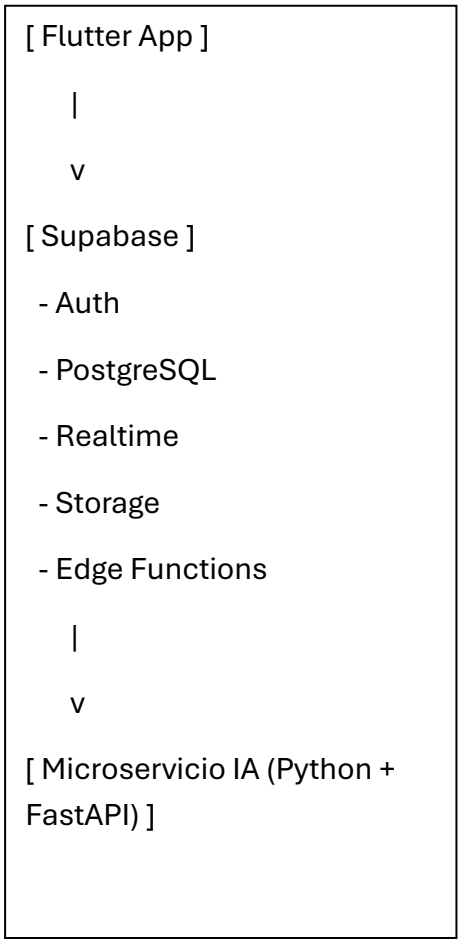
1. ARQUITECTURA TÉCNICA DETALLADA – CONVIVE

1.1 Visión general de la arquitectura

ConVive se basa en una **arquitectura modular y desacoplada**, donde el frontend consume servicios backend gestionados por Supabase y un microservicio de IA independiente.



1.2 Frontend – Flutter

Responsabilidades

- Registro y autenticación de usuarios
- Gestión de perfil y hábitos
- Visualización de publicaciones (lista + mapa)
- Swipe tipo Tinder
- Chat en tiempo real
- Notificaciones
- Gestión de suscripción (free / premium)

Comunicación

- Supabase SDK (Auth, DB, Realtime)
- HTTP REST hacia microservicio de IA
- OneSignal SDK para push notifications

1.3 Backend – Supabase

1.3.1 Autenticación (Supabase Auth)

- Email/password
- OAuth (Google opcional)
- Rol de usuario:
    - student
    - non_student
    - admin

Se controla mediante **claims + RLS**.

1.3.2 Base de datos (PostgreSQL)

- Modelo relacional normalizado
- Integridad referencial
- Índices para búsqueda geográfica
- Políticas de seguridad (Row Level Security)

1.3.3 Realtime

- Chat 1 a 1 entre matches
- Actualización de estado de match
- Notificaciones internas

1.3.4 Storage

- Fotos de perfil

- Fotos de departamentos
- Reglas:
    - Foto de perfil obligatoria
    - Publicaciones solo con imágenes válidas

1.3.5 Edge Functions

Funciones ligeras:

- Validar si el usuario puede chatear (free / premium)
- Llamar al microservicio de IA
- Moderación básica de contenido
- Control de boosts y visibilidad

1.4 Microservicio de IA – Python + FastAPI

Responsabilidades

- Calcular score de compatibilidad
- Validar imágenes (perfil / departamentos)
- Reentrenar modelo con feedback

Endpoints principales

POST /compatibility-score

POST /validate-profile-image

POST /validate-property-image

Características

- Modelos open-source
- Ligero
- Stateless
- Escalable

1.5 Notificaciones – OneSignal

Eventos

- Nuevo match
- Nuevo mensaje
- Publicación destacada
- Recordatorios de perfil incompleto

2. DIAGRAMA ER COMPLETO – CONVIVE

2.1 users

Campo	Tipo
id	UUID (PK)
email	varchar
role	enum (student, non_student, admin)
subscription_type	enum (free, premium)
created_at	timestamp

Relaciones:

- 1 → 1 profile
- 1 → 1 habits
- 1 → N properties
- 1 → N swipes
- 1 → N matches
- 1 → N messages

2.2 profiles

Campo	Tipo
id	PK
user_id	FK → users.id

full_name	varchar
birth_date	date
gender	enum
bio	text
profile_image_url	text
verified	boolean
created_at	timestamp

2.3 habits (IA – compatibilidad)

Campo	**Tipo**
id	PK
user_id	FK → users.id
sleep_start	int
sleep_end	int
cleanliness_level	int
noise_tolerance	int
party_frequency	int
guests_tolerance	int
pets	boolean
pet_tolerance	int
alcohol_frequency	int
work_mode	enum
time_at_home	int
communication_style	int
conflict_management	int
responsibility_level	int
created_at	timestamp

2.4 properties (departamentos / habitaciones)

Campo	Tipo
id	PK
owner_id	FK → users.id
title	varchar
description	text
price	decimal
latitude	decimal
longitude	decimal
address	text
available_from	date
is_active	boolean
created_at	timestamp

2.5 property_images

Campo	Tipo
id	PK
property_id	FK → properties.id
image_url	text
validated	boolean
created_at	timestamp

2.6 swipes

Campo	Tipo
id	PK
swiper_id	FK → users.id

target_user_id	FK → users.id
direction	enum (like, dislike)
created_at	timestamp

2.7 matches

Campo	**Tipo**
id	PK
user_a	FK → users.id
user_b	FK → users.id
compatibility_score	decimal
created_at	timestamp

2.8 chats

Campo	**Tipo**
id	PK
match_id	FK → matches.id
created_at	timestamp

2.9 messages

Campo	**Tipo**
id	PK
chat_id	FK → chats.id
sender_id	FK → users.id
content	text
created_at	timestamp

2.10 subscriptions

Campo	Tipo
id	PK
user_id	FK → users.id
price	decimal
is_student	boolean
start_date	date
end_date	date
status	enum

JUSTIFICACIÓN ACADÉMICA

La arquitectura propuesta desacopla la lógica de presentación, negocio e inteligencia artificial, permitiendo escalabilidad, mantenibilidad y evolución del sistema sin afectar la experiencia del usuario.