Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemático

Programación de Sistemas Móviles

Segunda entrega

Integrantes del equipo:

Juan Rodríguez Lara

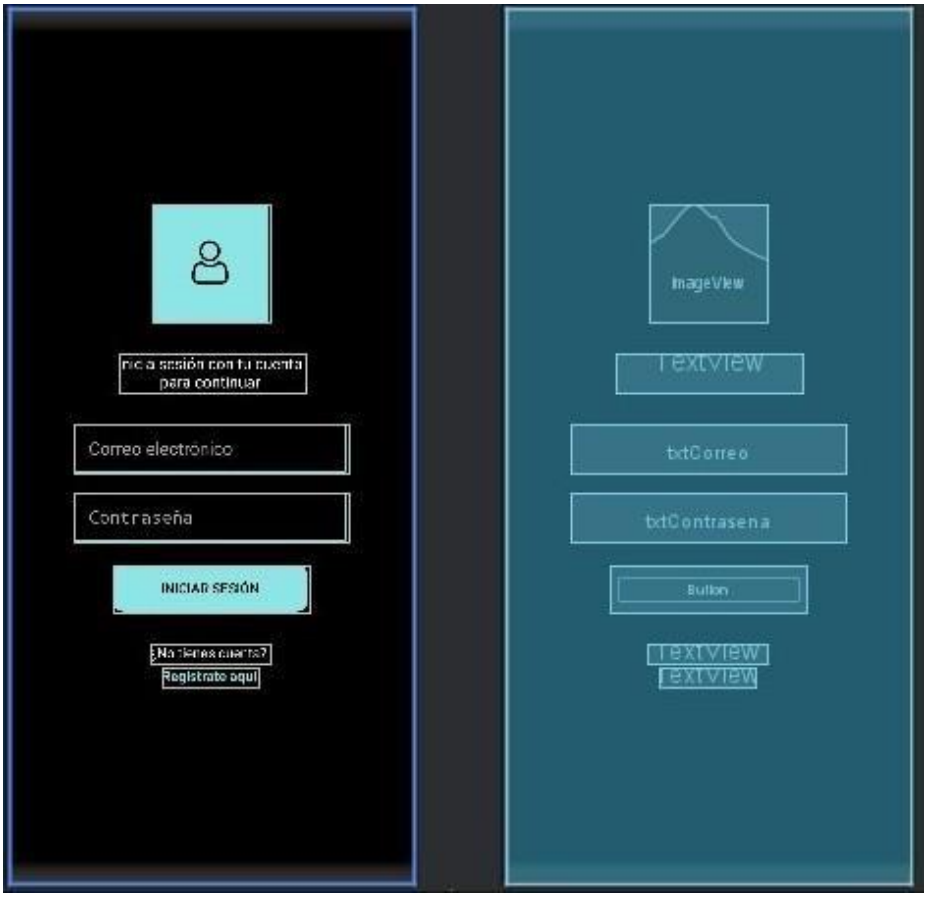
Andrés Medina Costilla

Profesor:

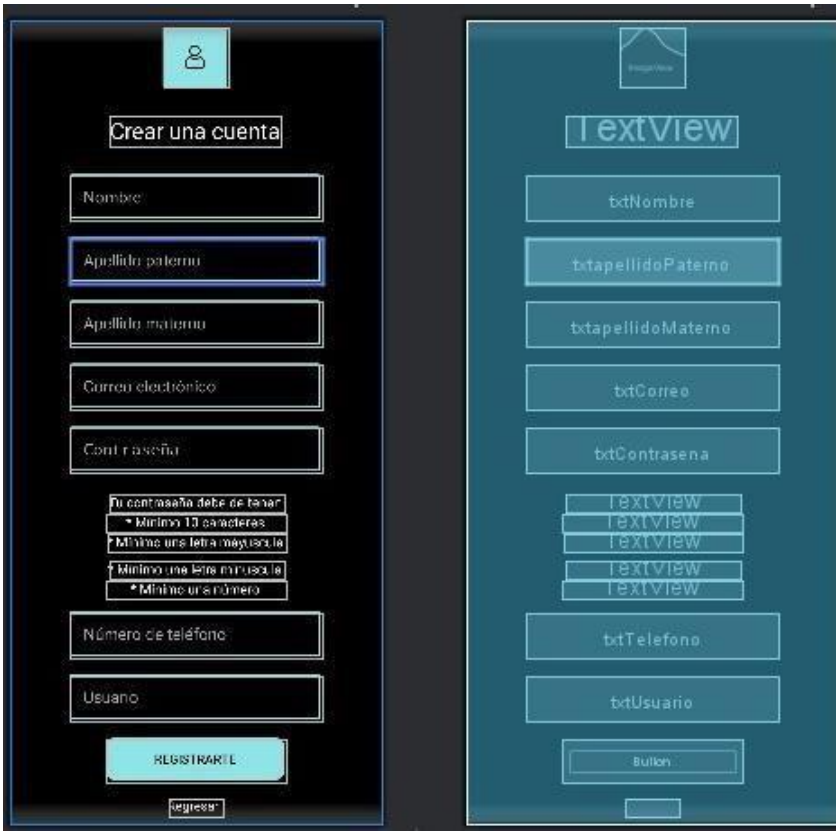
Juan Alejandro Villareal Mojica

XML de las pantallas

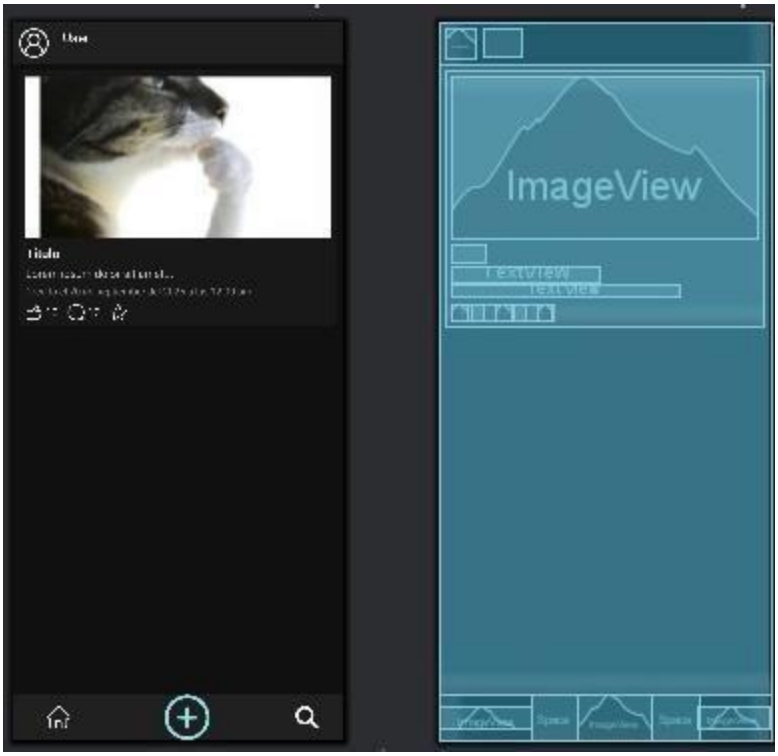
Login



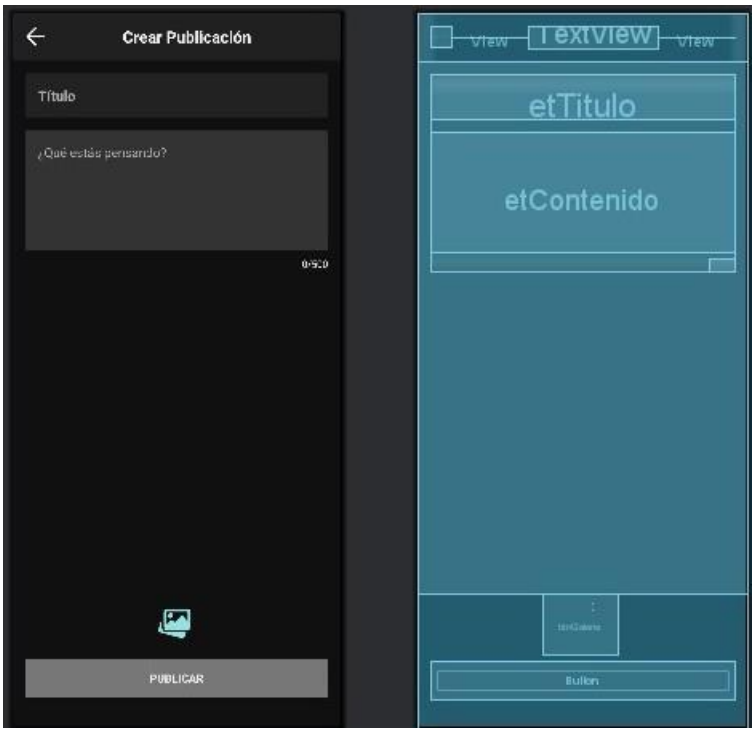
Registro



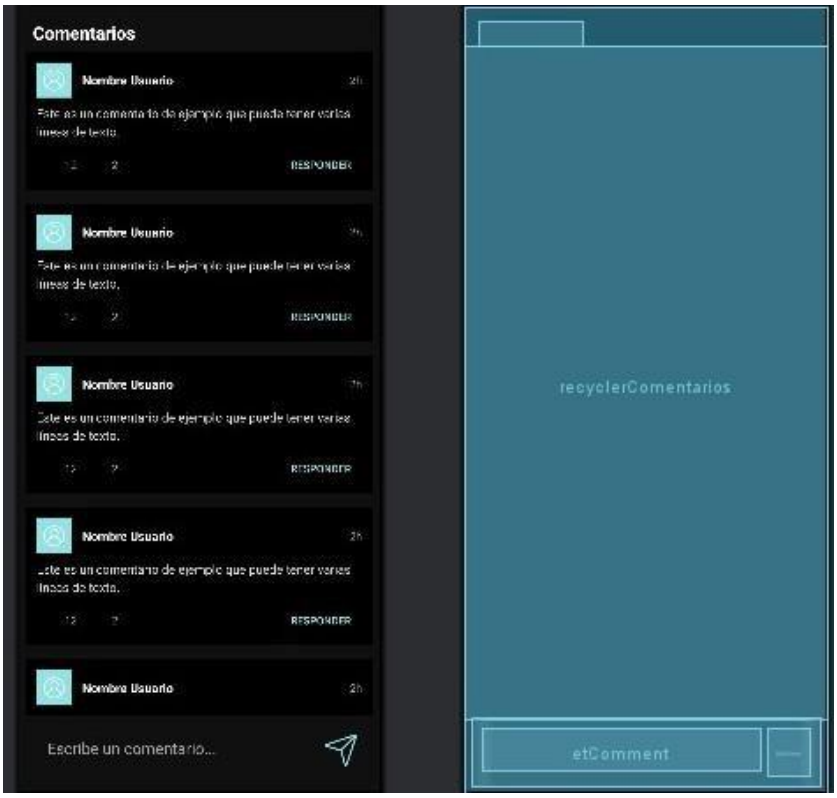
Pantalla principal



Crear publicación

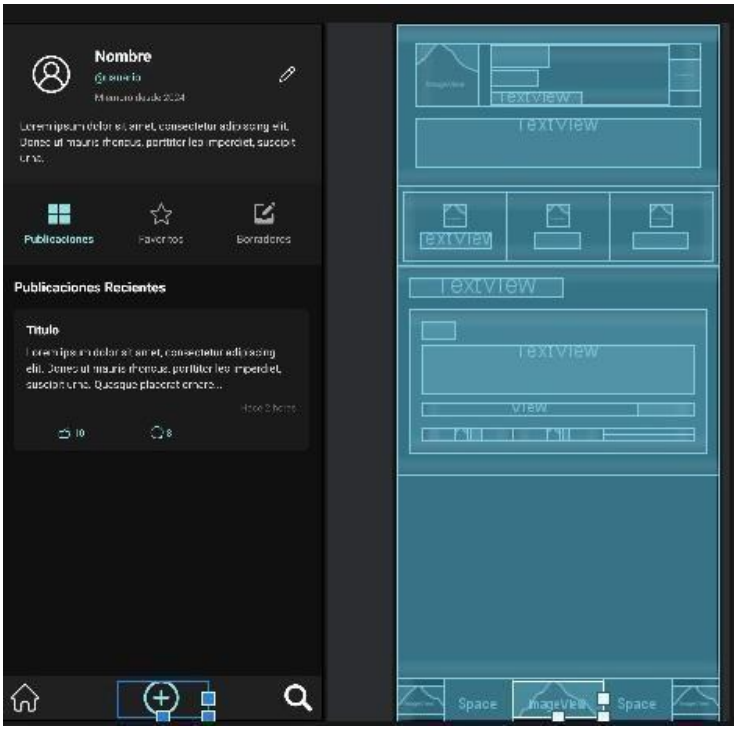


Comentarios



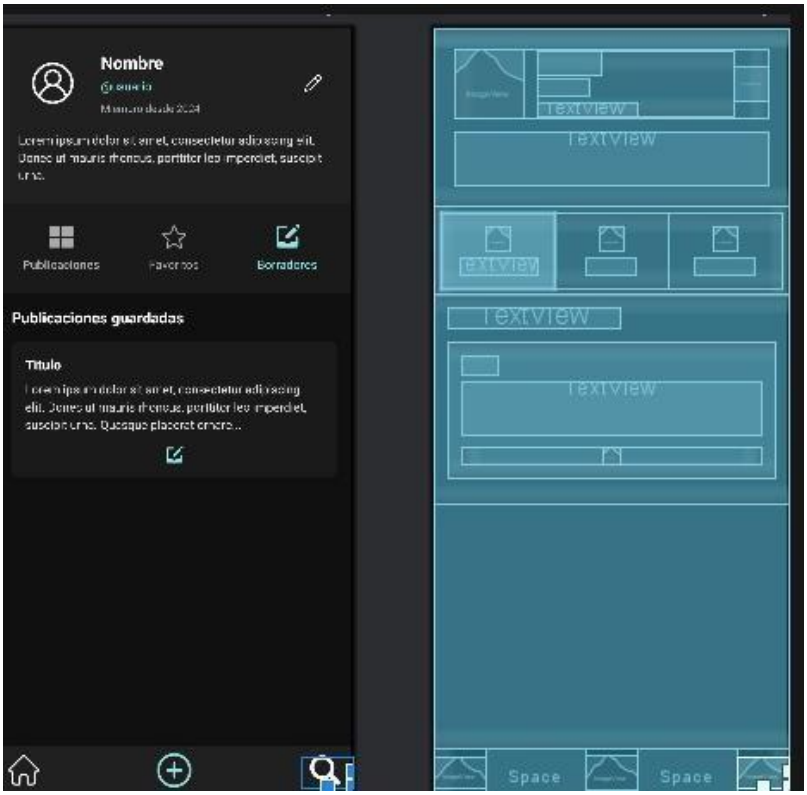
Búsqueda

Perfil (Publicaciones creadas)

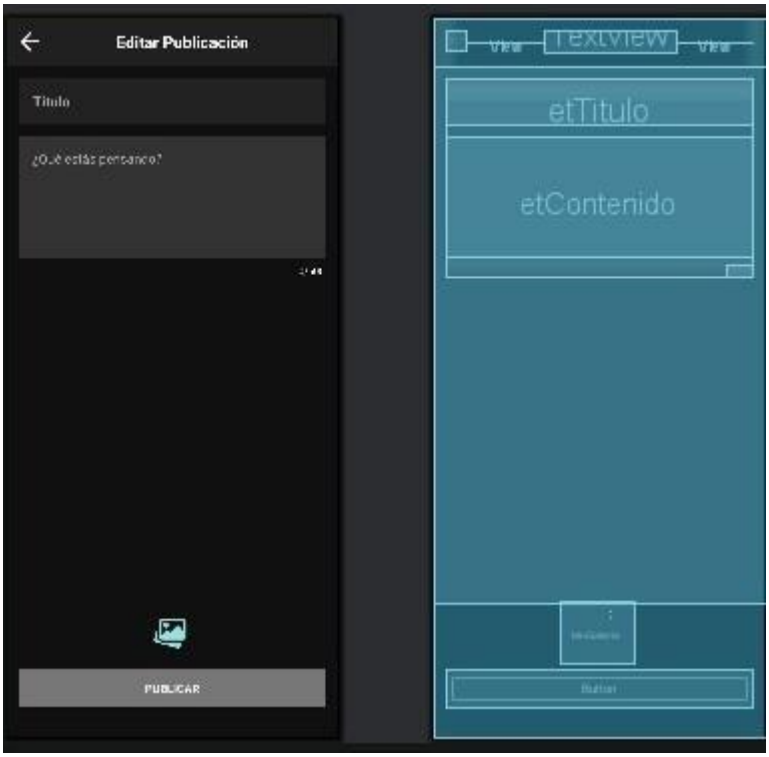


Perfil (Publicaciones guardadas)

Perfil (Borrador de publicaciones)



Edición de publicaciones del borrador



Editar perfil

Modelos y patrones de diseño

El modelo de base de datos propuesto está diseñado para una aplicación de tipo red social, donde los usuarios pueden crear publicaciones, añadir imágenes, realizar comentarios, reaccionar con "me gusta" o "no me gusta", y guardar publicaciones como favoritas.
Para garantizar un diseño estructurado, eficiente y flexible, se implementaron diversos **patrones de diseño de bases de datos** que mejoran la integridad, escalabilidad y rendimiento del sistema.

Diseño de la base de datos

Patrón	Tablas Aplicadas	Propósito
Tipo enumerado	`likes_comentarios`	Permitir diferentes tipos de reacciones (`like`, `dislike`) dentro de una misma tabla mediante el uso de un campo `ENUM`.
Uno a muchos	`imagenes_publicacion`	Permitir que una publicación pueda contener múltiples imágenes asociadas.
Autorreferencia	`comentarios`	Facilitar la creación de comentarios anidados o respuestas a otros comentarios dentro de la misma publicación.
Tabla intermedia	`favoritos`, `likes_publicaciones`	Gestionar relaciones de tipo muchos-a-muchos entre usuarios y publicaciones (usuarios que dan "me gusta" o guardan publicaciones).
Borrado lógico / suave	`publicaciones.estado`	Implementar borrado lógico para conservar registros sin eliminarlos físicamente, permitiendo mantener historial y consistencia.
Contador agregado	`publicaciones (cantidad_likes, cantidad_comentarios, cantidad_favoritos)`	Optimizar el rendimiento manteniendo contadores actualizados para evitar cálculos frecuentes de agregación.
Llave compuesta / Restricción única compuesta	Claves `UNIQUE` en `favoritos` y `likes_publicaciones`	Evitar duplicidad en las relaciones entre usuarios y publicaciones.

Este conjunto de patrones proporciona una **estructura coherente y escalable**, permitiendo manejar correctamente las relaciones entre entidades y optimizar las operaciones más comunes de la aplicación.

Además, el diseño mantiene la **integridad referencial** mediante claves foráneas y restricciones únicas, y considera aspectos de **rendimiento y mantenimiento** gracias al uso de contadores y estados lógicos.

MODELOS

Usuario

```
data class Usuario(
    val idUsuario: Int,
    val nombre: String,
    val apellidoPaterno: String,
    val apellidoMaterno: String?,
    val usuario: String,
    val correoElectronico: String,
    val contraseña: String,
    val fotoPerfil: String?,
    val telefono: String?,
    val fechaRegistro: String)
```

Publicación

```
data class Publicacion(
    val idPublicacion: Int,
    val idUsuario: Int,
    val titulo: String,
    val descripcion: String?,
    val fechaPublicacion: String,
    val cantidadLikes: Int,
    val cantidadComentarios: Int,
    val cantidadFavoritos: Int,
    val estado: String)
```

ImagenPublicacion

```
data class ImagenPublicacion(
    val idImagen: Int,
    val idPublicacion: Int,
    val urlImagen: String,
    val orden: Int
)
```

Comentario

```
data class Comentario(
    val idComentario: Int,
    val idPublicacion: Int,
    val idUsuario: Int,
    val idComentarioPadre: Int?,
    val descripcion: String,
    val cantidadLikes: Int,
    val cantidadDislikes: Int,
    val fechaComentario: String
)
```

Favorito

```
data class Favorito(
    val idFavorito: Int,
    val idUsuario: Int,
    val idPublicacion: Int,
    val fechaGuardado: String
)
```

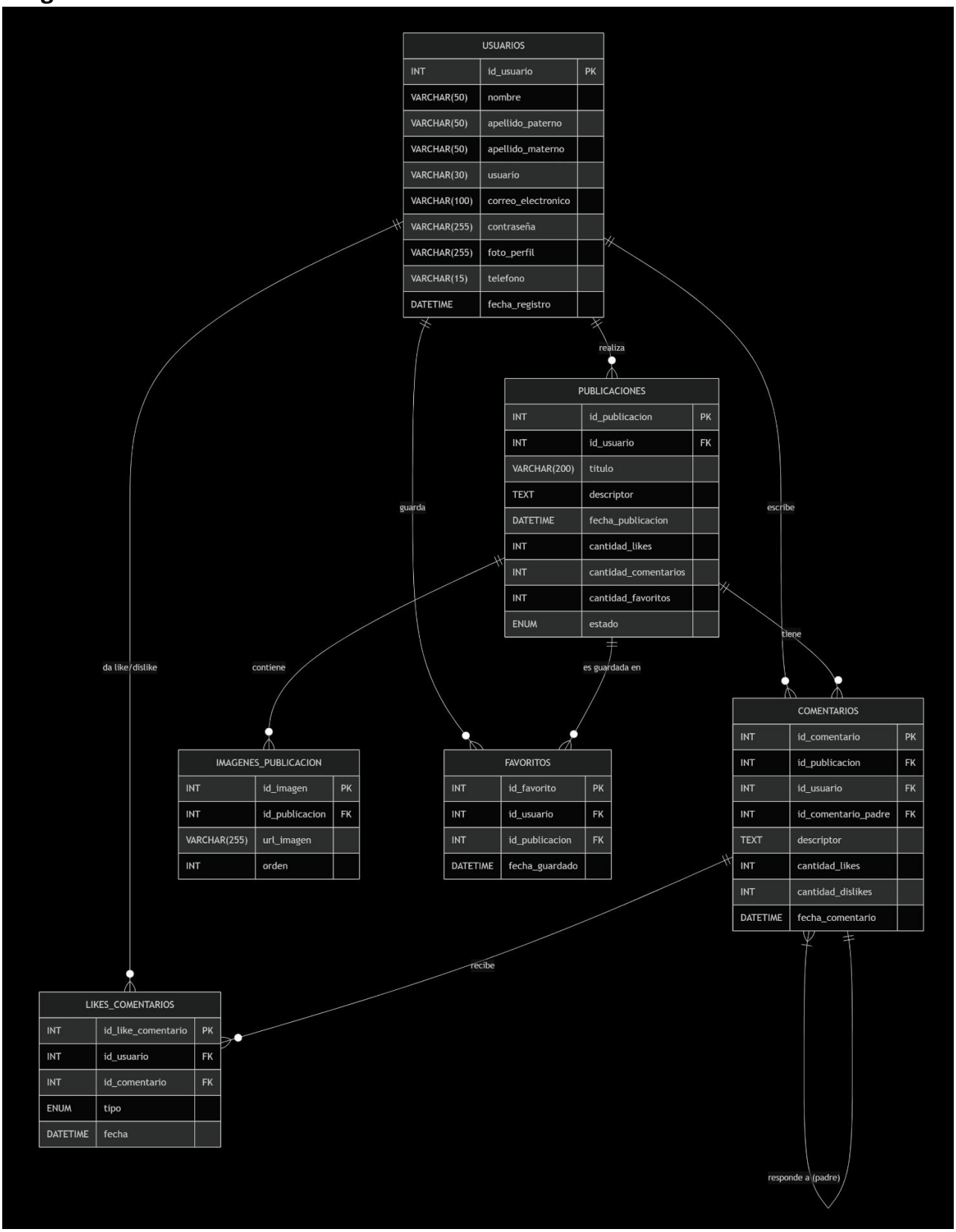
LikePublicacion

```
data class LikePublicacion(
    val idLike: Int,
    val idUsuario: Int,
    val idPublicacion: Int,
    val fechaLike: String
)
```

LikeComentario

```
data  class  LikeComentario(
    val  idLikeComentario: Int,
    val idUsuario: Int,
    val idComentario: Int,
    val tipo: String, // "like" o "dislike"
    val fecha: String
)
```

Diagrama de la base de datos



Modelo de datos

Usuarios

```
CREATE TABLE usuarios (
   id_usuario INTEGER PRIMARY KEY AUTOINCREMENT,
   nombre TEXT NOT NULL,
   apellido_paterno TEXT NOT NULL,
   apellido_materno TEXT,
   usuario TEXT UNIQUE NOT NULL,
   correo_electronico TEXT UNIQUE NOT NULL,
   contraseña TEXT NOT NULL,
   foto_perfil TEXT,
   telefono TEXT,
   fecha_registro DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Publicaciones

```
CREATE TABLE publicaciones (
   id_publicacion INTEGER PRIMARY KEY AUTOINCREMENT,
   id_usuario INTEGER NOT NULL,
   titulo TEXT NOT NULL,
   descriptor TEXT,
   fecha_publicacion DATETIME DEFAULT CURRENT_TIMESTAMP,
   cantidad_likes INTEGER DEFAULT 0,
   cantidad_comentarios INTEGER DEFAULT 0,
   cantidad_favoritos INTEGER DEFAULT 0,
   estado TEXT CHECK(estado IN ('activo', 'inactivo', 'eliminado')) DEFAULT 'activo',
   FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario) ON DELETE CASCADE
);
```

Comentarios

```
CREATE TABLE comentarios (
   id_comentario INTEGER PRIMARY KEY AUTOINCREMENT,
   id_publicacion INTEGER NOT NULL,
   id_usuario INTEGER NOT NULL,
   id_comentario_padre INTEGER,
   descriptor TEXT NOT NULL,
```

```
  cantidad_likes INTEGER DEFAULT 0,
  cantidad_dislikes INTEGER DEFAULT 0,
  fecha_comentario DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (id_publicacion) REFERENCES publicaciones(id_publicacion) ON
DELETE CASCADE,
  FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario) ON DELETE CASCADE,
  FOREIGN KEY (id_comentario_padre) REFERENCES comentarios(id_comentario) ON
DELETE CASCADE
);
```

Likes / Comentarios

```
CREATE TABLE likes_comentarios (
  id_like_comentario INTEGER PRIMARY KEY AUTOINCREMENT,
  id_usuario INTEGER NOT NULL,
  id_comentario INTEGER NOT NULL,
  tipo TEXT CHECK(tipo IN ('like', 'dislike')) NOT NULL,
  fecha DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario) ON DELETE CASCADE,
  FOREIGN KEY (id_comentario) REFERENCES comentarios(id_comentario) ON DELETE
CASCADE,
  UNIQUE(id_usuario, id_comentario)
);
```

Imágenes publicación

```
CREATE TABLE imagenes_publicacion (
  id_imagen INTEGER PRIMARY KEY AUTOINCREMENT,
  id_publicacion INTEGER NOT NULL,
  url_imagen TEXT NOT NULL,
  orden INTEGER DEFAULT 0,
  FOREIGN KEY (id_publicacion) REFERENCES publicaciones(id_publicacion) ON
DELETE CASCADE
);
```

Favoritos

```
CREATE TABLE favoritos (
   id_favorito INTEGER PRIMARY KEY AUTOINCREMENT,
   id_usuario INTEGER NOT NULL,
   id_publicacion INTEGER NOT NULL,
   fecha_guardado DATETIME DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario) ON DELETE CASCADE,
   FOREIGN KEY (id_publicacion) REFERENCES publicaciones(id_publicacion) ON
DELETE CASCADE,
   UNIQUE(id_usuario, id_publicacion)
);
```