

**Informe avances Proyecto**

**SCAV**

**CAPSTONE\_006V**

**Integrantes**

Nibaldo Quezada

Cristian Arroyo

Nicolas Peña

**Docente**

Fabián Alcántara

**Contenido**

Contenido

[Introducción 3](#_Toc179819424)

[Descripción del proyecto APT 4](#_Toc179819425)

[Requerimientos de Alto Nivel: 4](#_Toc179819426)

[Diagrama de arquitectura 6](#_Toc179819427)

[Planificación 7](#_Toc179819428)

[Avances del proyecto 8](#_Toc179819429)

[Ajustes del proyecto 9](#_Toc179819430)

[Metodología: Corrección hacia Scrum 9](#_Toc179819431)

[Cronograma del Proyecto: Corrección de Sprints 10](#_Toc179819432)

[Desarrollo de Aplicaciones: División entre Aplicación Administrativa y de Residentes 11](#_Toc179819433)

# Introducción

El proyecto TechApps - SCAV tiene como objetivo desarrollar un sistema integral de control de acceso vehicular para el condominio Vista Parque 1 y 2. Esta iniciativa surge a partir de las dificultades que enfrenta el condominio en la gestión manual del acceso vehicular, lo que ocasiona congestión en los puntos de entrada y salida, además de vulnerabilidades de seguridad ante posibles robos. El sistema propuesto permitirá autorizar, monitorear y restringir el acceso de vehículos de manera automática, empleando tecnologías como reconocimiento de patentes, bases de datos, y aplicaciones móviles. Este proyecto permitirá a los estudiantes aplicar competencias clave de su perfil profesional, como la gestión de proyectos, el desarrollo de software y la implementación de soluciones informáticas para problemas reales, asegurando una experiencia práctica y acorde a los estándares de la industria.

# Descripción del proyecto APT

El objetivo del proyecto es desarrollar un sistema que permita mejorar la gestión y seguridad del acceso vehicular en el condominio. El sistema deberá registrar, autorizar y monitorear el ingreso y salida de vehículos, asegurando un control eficiente y seguro.

## Requerimientos de Alto Nivel:

Uso Administradores y Personal de Seguridad:

* Manejo de Registro de Residentes:
  + Registro y actualización de la información vehicular y personal de los residentes.
  + Administración de permisos de acceso para los vehículos registrados de cada residente.
* Manejo de Registro de Visitas:
  + Registro anticipado de visitas por parte de los residentes o por personal administrativo.
  + Autorización o denegación de acceso a las visitas, basada en políticas definidas por el recinto.
  + Notificación automática a los residentes sobre la llegada de una visita.
* Autorización de Vehículos Comerciales:
* Registro de solicitudes de acceso para vehículos comerciales (ej. entregas de encomiendas, servicios de comida, servicios de transporte).
* Revisión, aprobación o rechazo de dichas solicitudes por parte del personal autorizado.
* Ingreso de Vehículos Emergencia:
  + El conserje autoriza la entrada de forma manual en caso de vehículos que correspondan a Bomberos, ambulancia, policiales.
* Monitoreo y Reportes de Acceso:
  + Monitoreo en tiempo real de entradas y salidas de vehículos, mediante bitácora y reportería.
  + Generación de reportes de acceso diarios, semanales o mensuales, que incluyan datos sobre todos los ingresos, con filtros por tipo de vehículo, residente, visita, etc.
* Integración con API Externa para Validación Vehicular:
  + Validación automática de los datos vehiculares (marca, modelo, propietario) mediante una API externa, asegurando que solo vehículos registrados puedan acceder.
* Aplicación automática de multas:
  + El sistema deberá ser capaz de identificar cuando una visita permanece por más de las horas autorizadas (Definido por el condominio), y aplicar la multa correspondiente.

Uso Residentes:

* Solicitud de Acceso para Visitas:
  + App móvil donde los residentes pueden registrar visitas anticipadamente, proporcionando la información del vehículo.
  + Recepción de notificaciones y actualizaciones sobre el estado de la visita (aprobada, rechazada).
* Solicitud de Acceso para Servicios Comerciales:
  + Validación por parte del residente receptor del servicio, si se requiere.
* Recepción de Notificaciones y Alertas:
  + Envío de notificaciones automáticas a los residentes cuando una visita registrada llega al recinto.
  + Actualizaciones periódicas sobre políticas de acceso y seguridad del recinto.
* Consulta de Historial de Accesos:
  + Acceso a un historial personal de entradas y salidas, tanto de los vehículos del residente como de las visitas autorizadas, y servicios externos, disponible a través de la app.

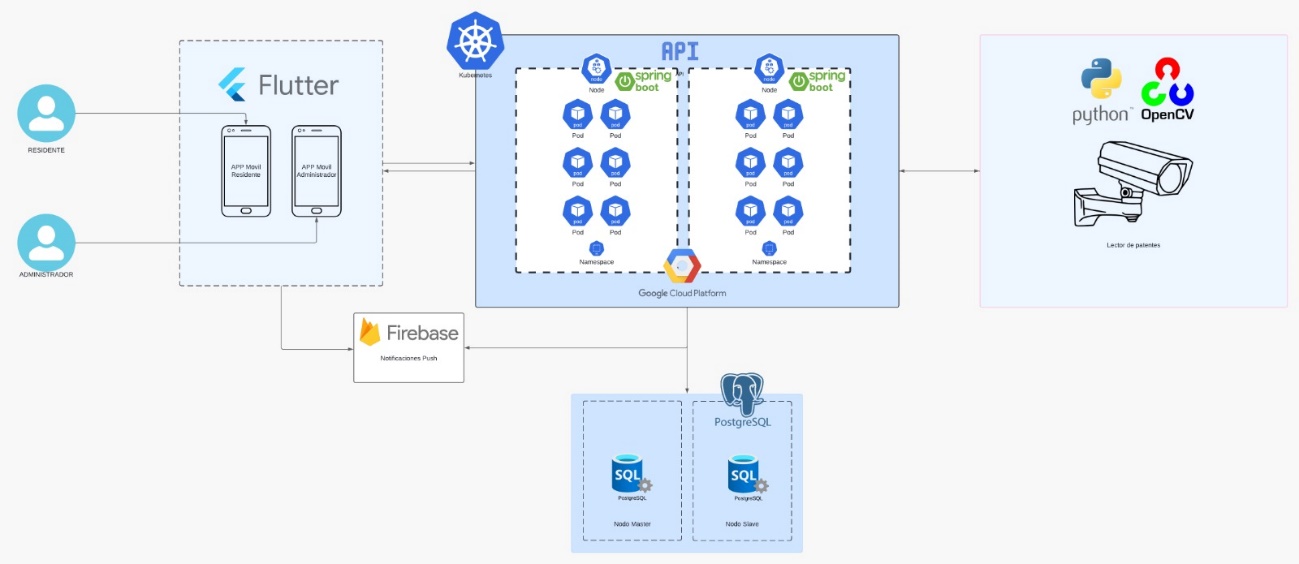
Arquitectura del Sistema:

* Lector de patente: Python OpenCV
* Base de Datos: PostgreSQL
* Api Rest: Spring Boot
* App Móvil: Flutter
* Api Externa: Auth ApiKey
* Nube: GCP (Google Cloud Platform)

Este sistema permitirá una gestión eficaz del acceso vehicular, mejorando la seguridad y el control en recintos residenciales, y garantizando una experiencia fluida tanto para los residentes como para los visitantes y proveedores.

# Diagrama de arquitectura

El sistema se compone de tres componentes principales, el primero corresponde a una API desarrollada en Springboot con Java17, el segundo un sistema de lectores de patentes realizado en Python 3 con OpenCV para el procesamiento de imágenes, y el tercero, dos aplicaciones desarrolladas en Flutter, una para los usuarios del condominio y otro para los administrativos. Por otra parte, el sistema contará con una base de datos relacional implementada en PostgreSQL. Finalmente se realizará las notificaciones del sistema con Firebase mediante su servicio de notificaciones-push.



# Planificación

El proyecto cuenta con un total de 18 semanas, las cuales se dividieron en 9 sprints de dos semanas cada uno, repartiendo las tareas como se muestra a continuación, actualmente nos encontramos en el sprint numero 6 correspondiente al desarrollo de la aplicación Flutter.

Gráfico

Descripción generada automáticamente

# Avances del proyecto

Actualmente nos encontramos en la fase de desarrollo de las aplicaciones móviles en flutter, a continuación, se adjunta un cuadro resumen con el estado de cada tarea realizada durante el proyecto.

Actividades y Estado de Avance Materiales:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Actividad | Estado | Dificultades | Materiales/Recursos Físicos | Lenguajes/Entorno |
| Lector de Patentes | Completado | Dificultades para hacer pruebas en vivo con la cámara en la calle debido a condiciones climáticas y tráfico | Cámara de video, PC | Python, OpenCV |
| Diseño de Base de Datos | Completado | N/A | PC, Google Cloud SQL | PostgreSQL |
| API REST | En curso | N/A | PC, GCP (Google Cloud Platform) API server | Java, Spring Boot |
| Aplicación Móvil en Flutter | En curso | N/A | PC, smartphone para pruebas | Flutter, Dart |
| Pruebas de Integración | No iniciado | Coordinación para pruebas con cámara en tiempo real en un entorno controlado | Cámara de video, PC, red local para la integración | Python, Java, SQL, Flutter |

# Ajustes del proyecto

## Metodología: Corrección hacia Scrum

Desde el inicio del proyecto TechApps - SCAV, la metodología Scrum fue la propuesta y establecida como la guía para gestionar el desarrollo. Sin embargo, durante las primeras fases de implementación, por un error del equipo, las actividades comenzaron a gestionarse bajo un enfoque más tradicional. Este desvío se debió a la organización de tareas como bloques independientes, con fechas de entrega fijas y poca interacción entre los módulos en desarrollo (lector de patentes, API REST, base de datos y aplicación móvil).

Al identificar este error, se realizó un ajuste inmediato para volver al enfoque ágil Scrum, lo cual permitió:

1. Recuperar la planificación iterativa e incremental: Se organizaron los trabajos en los sprints, lo que garantiza mayor flexibilidad para incorporar cambios según las necesidades.
2. Mejor coordinación y colaboración: Las reuniones diarias (daily meetings) se reintrodujeron para alinear al equipo y resolver bloqueos rápidamente.
3. Entregas parciales frecuentes: Se retomaron las entregas incrementales en cada sprint, lo que permitió presentar avances funcionales, como el lector de patentes y la integración de la API con la base de datos.
4. Retroalimentación continua: Se fomenta la interacción constante con los involucrados para garantizar que cada módulo se desarrollará conforme a las expectativas y requerimientos del proyecto.

Este ajuste ha sido crucial para recuperar la dinámica ágil que se había perdido. El equipo se alineó nuevamente con la metodología Scrum, permitiendo una mejor sincronización entre los módulos, optimizando los tiempos de desarrollo y asegurando que el proyecto avance según lo planeado.

## Cronograma del Proyecto: Corrección de Sprints

Inicialmente, el proyecto TechApps - SCAV se organizó en un número diferente de sprints, lo cual no permitía una planificación adecuada para la implementación de todos los módulos. Tras una revisión del cronograma, se realizó una corrección en la cantidad y duración de los sprints, con el fin de optimizar el desarrollo y cumplir con los objetivos planteados.

Nuevo Cronograma:

* Total de Sprints: 9 Sprints.
* Duración de cada Sprint: 2 semanas cada uno.
* Distribución por Módulos:

1. Gestión de Proyecto y Backlog: Sprint 1 (S1) y (S2), Sprint 2 (S3) y (S4).
2. Desarrollo del Sistema de Reconocimiento de Patentes: Sprint 3 (S5) y (S6).
3. Desarrollo del Modelo de Base de Datos: Sprint 4 (S7) y (S7).
4. Implementación de API REST: Sprint 5 (S9) y (S10).
5. Aplicación Móvil en Flutter: Se ajustó a dos sprints completos: Sprint 6 (S11) y (S11), Sprint 7 (S13) y (S14).
6. Pruebas de Calidad e Integración: Sprint 8 (S15) y (S16).
7. Marcha Blanca y Capacitación: Sprint 9 (S17) y (S18).

Impacto del Ajuste:

* Redistribución del Tiempo: Las fechas se ajustaron para que los módulos más críticos tengan más tiempo asignado, como el desarrollo de la aplicación móvil en Flutter, que se desarrollará a lo largo de 2 sprints completos.
* Optimización del Flujo: El resto de los módulos se desarrollarán en sprints únicos para evitar la fragmentación del trabajo.

## Desarrollo de Aplicaciones: División entre Aplicación Administrativa y de Residentes

Unificación en una Única App con Roles Administrativos y Residentes

Durante la fase inicial de planificación del proyecto TechApps - SCAV, se propuso desarrollar dos aplicaciones independientes: una para administradores y otra para residentes. Sin embargo, a medida que avanzamos en el desarrollo y tras una revisión del flujo y la usabilidad, identificamos que la división generaba redundancia y aumentaba la carga de mantenimiento del sistema.

Decisión de Cambio: Unificación en una Única Aplicación con Roles Dinámicos

En lugar de gestionar dos aplicaciones separadas, decidimos integrar todas las funcionalidades en una única aplicación móvil. La nueva estrategia simplifica el desarrollo, facilita el mantenimiento y mejora la experiencia del usuario al unificar las operaciones en una sola plataforma.

Cómo funciona la nueva estructura:

* Al iniciar sesión, la aplicación detecta el tipo de cuenta (Administrador o Residente) basado en las credenciales del usuario.
* Dependiendo del rol del usuario, se derivan a diferentes pantallas y funcionalidades:
  + Administradores:
    - Gestión de residentes, visitas y vehículos comerciales.
    - Acceso a reportes y monitoreo en tiempo real de accesos vehiculares.
    - Administración de permisos de entrada y salida.
    - Automatización y gestión de multas.
  + Residentes:
    - Registro anticipado de visitas.
    - Notificaciones en tiempo real sobre la llegada de visitas y entregas.
    - Visualización del historial de accesos personales.
    - Solicitud y autorización de acceso para servicios comerciales.

Ventajas del Enfoque Unificado

1. Reducción de Complejidad: Al unificar ambas aplicaciones en una sola, se eliminan redundancias y se simplifica la administración del sistema.
2. Flexibilidad en el Uso de Roles: La aplicación ahora se adapta dinámicamente al rol del usuario, proporcionando un flujo intuitivo y centralizado.
3. Mantenimiento Simplificado: Facilita la gestión de actualizaciones, ya que cualquier mejora se refleja en una sola app.
4. Experiencia del Usuario Mejorada: Los usuarios (administradores y residentes) pueden acceder a todas sus funcionalidades desde una única plataforma sin tener que cambiar de aplicación.

## Implementación de Servicios en la Nube y Migración de la Base de Datos:

Contexto del Cambio: Durante las fases iniciales del proyecto TechApps - SCAV, el sistema estaba configurado para operar en un entorno local, lo que generaba ciertas limitaciones en cuanto al acceso y escalabilidad. Para mejorar el rendimiento y garantizar un acceso más robusto y seguro, se decidió migrar la aplicación y la base de datos a un entorno en la nube utilizando Google Cloud Platform (GCP).

Detalles de la Implementación:

1. Migración de la Base de Datos:
   * La base de datos originalmente desarrollada en SQL Server fue migrada a un servicio en la nube utilizando Google Cloud SQL.
   * Esta migración permite un acceso más rápido y confiable a la información de residentes, visitas y vehículos comerciales desde cualquier lugar.
   * Ventajas:
     + Elimina la dependencia de un entorno localhost, mejorando la disponibilidad y accesibilidad.
     + Aumenta la seguridad al aprovechar las políticas de seguridad y encriptación de GCP.
     + Facilita el escalado de la base de datos según la demanda del sistema.
2. Despliegue del Servicio de Autenticación (Microservicio msautenticar):
   * El primer microservicio (msautenticar) ha sido desplegado en la nube usando Google Kubernetes Engine (GKE).
   * Esto permite un despliegue automatizado y escalable del sistema de autenticación, que es crucial para gestionar los accesos tanto de administradores como de residentes.
   * Ventajas:
     + Mejora la escalabilidad y disponibilidad del servicio de autenticación.
     + Facilita la gestión de cargas en función del número de usuarios concurrentes.
     + Reduce el tiempo de inactividad al permitir actualizaciones en caliente sin afectar a los usuarios.

Impacto de estos Cambios en el Proyecto

1. Eliminación de Dependencias Locales:

Con la migración a la nube, se elimina la necesidad de ejecutar servicios en localhost, lo que simplifica la integración y despliegue de la aplicación en entornos de producción

## Uso de PostgreSQL para la Gestión de la Base de Datos

**Contexto del Cambio**: Para optimizar el almacenamiento, la consulta y la escalabilidad de los datos del sistema, se tomó la decisión de implementar **PostgreSQL** como el motor de base de datos principal, utilizando **Google Cloud SQL**. Esto ha permitido un enfoque más moderno y eficiente para la gestión de datos del proyecto **TechApps - SCAV**.

**Detalles de la Implementación:**

1. **Migración a PostgreSQL en Google Cloud SQL**:
   * La base de datos ha sido migrada a **PostgreSQL**, aprovechando las ventajas de **Google Cloud SQL** para asegurar un acceso rápido y seguro.
   * **PostgreSQL** se seleccionó por su capacidad para gestionar grandes volúmenes de datos y su compatibilidad con **consultas avanzadas** y **transacciones ACID**.
2. **Ventajas del Uso de PostgreSQL**:
   * **Escalabilidad Automática**: PostgreSQL en Google Cloud ajusta automáticamente la capacidad de la base de datos en función de la demanda.
   * **Alta Disponibilidad**: La configuración en la nube garantiza la **disponibilidad continua** del servicio, incluso en caso de fallos en el hardware.
   * **Despliegue Continuo y Backup Automático**: Google Cloud SQL permite **copias de seguridad automatizadas**, lo que facilita la recuperación ante desastres y actualizaciones sin interrupciones.
   * **Seguridad Mejorada**: Soporta **encriptación de datos** en tránsito y en reposo, garantizando la confidencialidad de la información.

**Impacto de estos Cambios en el Proyecto**

1. **Eliminación de Dependencias Locales**:
   * Con la migración a PostgreSQL en Google Cloud, se elimina la necesidad de ejecutar bases de datos en entornos locales, simplificando el despliegue y la integración en producción.
2. **Mejora en la Escalabilidad y Disponibilidad**:
   * El sistema ahora puede **escalar dinámicamente** según la carga, lo que es crucial para un proyecto que optimiza el acceso vehicular en un condominio de gran tamaño como **Vista Parque**.
3. **Facilita la Colaboración y el Desarrollo**:
   * Al centralizar la base de datos en la nube, todos los miembros del equipo pueden acceder al entorno de desarrollo y producción **de manera remota**, lo que agiliza las pruebas y el desarrollo colaborativo.

# Evidencias de avance

A continuación, se adjuntarán algunos prints de pantalla correspondientes a los avances de los distintos módulos desarrollados durante el proyecto.

## Módulo Lector de patentes:

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 1 - Implementación lector de patentes

A continuación, describiremos el funcionamiento del sistema de control de acceso vehicular desarrollado para optimizar la seguridad en recintos privados, tales como condominios, oficinas o estacionamientos. El sistema permite el acceso automatizado de vehículos autorizados mediante la detección de patentes (placas de vehículos), y registra tanto la entrada como la salida en una bitácora a través de una API. Adicionalmente, se integra un servomotor controlado por un Arduino para manejar una barrera que se eleva automáticamente al detectar una patente válida.

**Descripción General del Sistema**

El sistema consta de varios componentes que interactúan entre sí:

1. **Cámara**: Captura en tiempo real las imágenes del vehículo al acercarse al punto de control.
2. **Lector de Patentes (Python)**: Detecta la patente del vehículo a partir del flujo de video y realiza consultas a una API para verificar si el vehículo está registrado.
3. **API (Backend)**: Se encarga de manejar la información sobre los vehículos autorizados y de registrar los accesos en una base de datos.
4. **Control de Barrera (Arduino)**: Utiliza un servomotor para controlar la apertura y cierre de una barrera física, activada por comandos enviados desde el sistema Python.

**Funcionamiento del Sistema**

**1. Flujo de Entrada de Vehículos**

El proceso de entrada al recinto sigue estos pasos:

1. **Captura de Imagen**: La cámara captura continuamente imágenes en tiempo real.
2. **Detección de la Patente**:
   * Utilizando la biblioteca OpenCV en Python, se detecta la presencia de una patente en el flujo de video.
   * El número de la patente se extrae utilizando técnicas de reconocimiento óptico de caracteres (OCR).
3. **Consulta a la API**:
   * El número de la patente detectada se envía a un endpoint específico (/api/v2/vehiculo/patente/{patente}) para verificar si el vehículo está registrado.
   * La API devuelve información sobre el vehículo, incluyendo su estado (residente, visita, etc.).
4. **Registro en la Bitácora**:
   * Si el vehículo está registrado, se crea un nuevo registro de entrada en la bitácora mediante una solicitud POST a la API (/api/v2/bitacora).
   * La bitácora almacena la hora de entrada y deja el campo de salida como null hasta que el vehículo salga.
5. **Activación de la Barrera**:
   * Si la patente es válida, se envía un comando al Arduino a través de la comunicación serial para que el servomotor eleve la barrera.

**2. Flujo de Salida de Vehículos**

El proceso de salida sigue una lógica similar, con algunas diferencias:

1. **Captura y Detección**:
   * La cámara detecta la patente del vehículo cuando intenta salir del recinto.
2. **Verificación y Actualización en la Bitácora**:
   * El número de la patente se consulta en la API para obtener el ID del registro de entrada correspondiente.
   * Se envía una solicitud PUT a la API (/api/v2/bitacora/salida/{vehiculoId}) para actualizar el campo fechaout con la hora de salida.
3. **Cálculo del Tiempo de Permanencia**:
   * Una vez registrado el horario de salida, se calcula la duración total del tiempo de permanencia del vehículo en el recinto.
4. **Apertura de la Barrera para Salida**:
   * Al confirmar que el registro ha sido actualizado correctamente, se envía un comando al Arduino para abrir la barrera y permitir la salida del vehículo.

**Conexión con el Arduino para Control de Barrera**

El sistema utiliza un Arduino conectado a un servomotor que controla la barrera. El Arduino se comunica con el sistema Python mediante un puerto serial (COM6, en este caso). La integración se realiza de la siguiente forma:

1. **Inicialización**:
   * El Arduino se configura para escuchar comandos desde el puerto serial.
2. **Comando de Apertura**:
   * Cuando se detecta una patente válida, se envía el comando subir al Arduino, el cual activa el servomotor para elevar la barrera.
3. **Cierre Automático**:
   * Después de un breve intervalo (5 segundos), el Arduino baja automáticamente la barrera.

## Módulo API Rest:

Imagen que contiene Aplicación

Descripción generada automáticamente

Ilustración 2 - Implementación API Rest

## Módulo Flutter:Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto Descripción generada automáticamenteInterfaz de usuario gráfica, Texto Descripción generada automáticamenteInterfaz de usuario gráfica Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Modelo Entidad Relación:

Diagrama

Descripción generada automáticamente

Ilustración 4 - MER