- Modelo Funcional: Diagrama de Casos de Uso.

El modelo funcional de la aplicación se puede representar utilizando un Diagrama de Casos de Uso. Este diagrama ilustra las interacciones entre los usuarios (actores) y el sistema, mostrando las principales funcionalidades que el sistema proporcionará. El sistema, en este caso, es el Sistema de Alerta de Telegram con Acceso Facial.

- Explicación de los Casos de Uso:

    - Monitoreo y Detección de Intrusión:
        - Gestionar Sensores [RF1.1]: Permite a los usuarios agregar, configurar y eliminar diferentes tipos de sensores.
        - Activar/Desactivar Sistema [RF1.2]: Permite al usuario activar (armar) y desactivar (desarmar) la alarma de forma remota a través de la aplicación móvil o comandos de Telegram.
        - Detectar Eventos [RF1.3]: El sistema debe ser capaz de detectar activaciones de sensores y clasificarlas como eventos de seguridad.
        - Configurar Zonas de Monitoreo [RF1.4]: La aplicación debe permitir al usuario definir y asignar sensores a diferentes zonas de monitoreo.
        - Ajustar Sensibilidad [RF1.5]: La aplicación debe permitir al usuario ajustar la sensibilidad de los sensores para minimizar falsas alarmas.
        - Ver Estado del Sistema en Tiempo Real [RF1.6]: La aplicación debe mostrar el estado actual del sistema (armado/desarmado, estado de los sensores) en tiempo real.

    - Alertas y Notificaciones en Tiempo Real:
        - Enviar Notificaciones a Telegram [RF2.1]: El sistema debe enviar notificaciones instantáneas y detalladas a un chat o grupo de Telegram preconfigurado cuando se detecta un evento.
        - Personalizar Mensajes de Alerta [RF2.2]: La aplicación debe permitir al usuario personalizar el contenido de los mensajes de alerta enviados a Telegram.
        - Enviar Medios (Opcional) [RF2.3]: Si se integra con cámaras, el sistema debe ser capaz de adjuntar imágenes o clips de video cortos a las notificaciones de Telegram.
        - Enviar Comandos Remotos por Telegram [RF2.4]: El sistema debe ser capaz de recibir y ejecutar comandos simples enviados a través de Telegram.

- Gestionar Destinatarios de Alertas [RF2.5]: La aplicación debe permitir al usuario añadir o eliminar destinatarios de Telegram que recibirán las alertas.
- Ver Historial de Notificaciones Enviadas [RF2.6]: La aplicación debe mantener un registro de todas las notificaciones enviadas a Telegram, con su estado de entrega.

- Acceso con Reconocimiento Facial:
  - Registrar Rostros de Usuarios [RF3.1]: La aplicación debe permitir a los administradores registrar y almacenar los patrones faciales de usuarios autorizados.
  - Verificar Identidad Facial [RF3.2]: El sistema debe ser capaz de capturar una imagen del rostro en el punto de acceso y compararla con la base de datos de rostros autorizados para verificar la identidad.
  - Conceder Acceso Automático [RF3.3]: Si se verifica la identidad del usuario, el sistema debe desencadenar la apertura de un dispositivo de acceso o desactivar la alarma en la zona correspondiente.
  - Registrar Intentos de Acceso [RF3.4]: El sistema debe registrar todos los intentos de acceso, incluyendo la fecha, hora, si fue exitoso o fallido, y, si es posible, una imagen del rostro capturado.
  - Enviar Notificaciones de Acceso [RF3.5]: El sistema debe enviar notificaciones a Telegram sobre los intentos de acceso, tanto exitosos como fallidos.
  - Gestionar Perfiles Faciales [RF3.6]: La aplicación debe permitir a los administradores actualizar o eliminar los patrones faciales de los usuarios.
  - Detectar Vidas (Anti-spoofing) [RF3.7]: (Opcional) El sistema debe implementar mecanismos para detectar intentos de suplantación de identidad (ej., uso de fotos o videos).

- Gestión de Usuarios y Permisos:
  - Crear y Editar Perfiles de Usuario [RF4.1]: La aplicación debe permitir a un administrador crear, editar y eliminar perfiles de usuario con información relevante.
  - Asignar Roles y Permisos [RF4.2]: La aplicación debe permitir asignar roles a los usuarios con permisos específicos sobre las funcionalidades del sistema.
  - Programar Acceso por Usuario [RF4.3]: La aplicación debe permitir establecer horarios o días específicos en los que un

usuario tiene permitido el acceso mediante reconocimiento facial.
        - Revocar Acceso [RF4.4]: La aplicación debe permitir revocar de inmediato el acceso de un usuario al sistema o a través del reconocimiento facial.

    - Historial y Registro de Eventos:
        - Registrar Detalladamente Eventos [RF5.1]: El sistema debe registrar de forma persistente todos los eventos significativos, incluyendo activaciones de sensores, cambios de estado del sistema, intentos de acceso (exitosos y fallidos), y notificaciones enviadas.
        - Consultar Historial de Eventos [RF5.2]: La aplicación debe permitir al usuario consultar el historial completo de eventos del sistema.
        - Filtrar y Buscar Eventos [RF5.3]: La aplicación debe proporcionar funcionalidades de filtrado y búsqueda para eventos específicos.
        - Exportar Registros [RF5.4]: La aplicación debe permitir la exportación del historial de eventos en un formato legible (ej., CSV, PDF) para fines de auditoría o respaldo.

- Los actores que interactúan con el sistema son:

    - Propietarios de Pequeñas y Medianas Empresas: Empresarios que necesitan un sistema de seguridad eficiente y fácil de usar para sus locales comerciales u oficinas.
    - Administradores de Espacios Restringidos: Individuos o equipos a cargo de la seguridad y el control de acceso en áreas que requieren protección especial.
    - Personas con Movilidad Reducida: Aquellos que se beneficiarían de un sistema de alarma y acceso remoto que no requiera su presencia física constante.
    - Sistema Externo (API de Telegram, Sensores, Cámaras, Dispositivos de Acceso).

El actor "Usuario" representa una generalización de los tipos de usuario específicos (Propietario de Vivienda, Propietario de PYME, Administrador de Espacios Restringidos, Persona con Movilidad Reducida), ya que comparten muchas interacciones comunes con el sistema. El actor "Administrador" es un rol especializado de un Usuario, que tiene permisos adicionales para gestionar el sistema.

- Escenarios para Casos de Uso Clave:

Escenario para RF1.2: Activar/Desactivar Sistema

Nombre del Caso de Uso: Activar/Desactivar Sistema

Identificador: RF1.2

Prioridad: Alta

Actores: Usuario (Propietario de Pequeña y Mediana Empresa, Administrador), Sistema Externo (API de Telegram).

- Escenario 1: Activación Remota del Sistema de Alarma a través de Telegram
  - Precondiciones:
    - La cuenta de Telegram del usuario está vinculada al sistema.
    - El sistema se encuentra actualmente en estado "Desarmado".
    - Los sensores están configurados y operativos.
    - El usuario tiene conectividad a internet.
  - Disparador: El usuario envía el comando /armar al chat de Telegram designado vinculado al sistema de seguridad.
  - Flujo de Eventos:
    - La API de Telegram recibe el comando y lo reenvía al sistema de seguridad.
    - El sistema verifica la autorización del usuario para armar el sistema.
    - Tras la autorización exitosa, el sistema inicia la secuencia de armado.
    - El sistema cambia su estado de "Desarmado" a "Armado".
    - El sistema envía un mensaje de confirmación al chat de Telegram del usuario: "Sistema de alarma armado con éxito".
    - sistema registra este evento: "Sistema armado por [Nombre de Usuario] vía Telegram".
  - Postcondiciones:
    - El sistema está en estado "Armado".
    - Los sensores están monitoreando activamente.
    - El usuario recibe un mensaje de confirmación en Telegram en <5s.
    - El evento se registra en el historial del sistema.

    - Flujos Alternativos:
        - Comando Inválido: Si el usuario envía un comando no reconocido, el sistema responde con "Comando no reconocido. Por favor, intente con /armar o /desarmar."
        - Usuario No Autorizado: Si un usuario no autorizado envía el comando, el sistema responde con "Acceso denegado. No tiene permisos para realizar esta acción." y registra el intento fallido.
        - Problema de Conectividad: Si el sistema no puede conectarse a Telegram, intenta restablecer la conexión y registra la falla.

- Escenario 2: Desactivación Remota del Sistema de Alarma a través de la Aplicación Móvil
    - Precondiciones:
        - El usuario ha iniciado sesión en la aplicación móvil.
        - El sistema se encuentra actualmente en estado "Armado".
        - El usuario tiene conectividad a internet.
    - Disparador: El usuario pulsa el botón "Desarmar" en la aplicación móvil.
    - Flujo de Eventos:
        - La aplicación móvil envía una solicitud al sistema para desarmar.
        - El sistema verifica la autorización del usuario (basada en la sesión iniciada).
        - Tras la autorización exitosa, el sistema inicia la secuencia de desarmado.
        - El sistema cambia su estado de "Armado" a "Desarmado".
        - La aplicación móvil se actualiza inmediatamente para reflejar el estado "Desarmado".
        - El sistema envía un mensaje de confirmación al chat de Telegram vinculado del usuario: "Sistema de alarma desarmado." (Opcional, pero buena práctica).
        - El sistema registra este evento: "Sistema desarmado por [Nombre de Usuario] vía aplicación móvil".
    - Postcondiciones:
        - El sistema está en estado "Desarmado".
        - La aplicación móvil refleja el estado actualizado en <2s.
        - El evento se registra en el historial del sistema.
    - Flujos Alternativos:
        - Fallo de Autenticación: Si la sesión del usuario ha caducado o es inválida, la aplicación solicita volver a iniciar sesión.

        - Error del Sistema: Si hay un error interno del sistema que impide el desarmado, la aplicación muestra "Error al desarmar el sistema. Por favor, intente de nuevo." y registra el error.

- Escenario para RF2.1: Envío de Notificaciones a Telegram

Nombre del Caso de Uso: Envío de Notificaciones a Telegram

Identificador: RF2.1

Prioridad: Alta

Actores: Sistema, Usuario (Destinatario de Alertas), Sistema Externo (API de Telegram), Sensor.

- Escenario: Intrusión Detectada y Notificación Enviada
    - Precondiciones:
        - El sistema está en estado "Armado".
        - Un sensor (ej., sensor de movimiento PIR) está configurado y activo en una zona específica ("Sala de Estar").
        - El chat de Telegram del usuario o un grupo designado está configurado como destinatario de alertas.
        - El sistema tiene conexión a internet activa y acceso a la API de Telegram.
        - El sistema no ha enviado recientemente una alerta similar (para evitar saturación por detección continua).
    - Disparador: El sensor de movimiento PIR en la "Sala de Estar" detecta movimiento.
    - Flujo de Eventos:
        - El sensor envía una señal al sistema de seguridad.
        - El sistema procesa la señal y la identifica como un evento de "Intrusión Detectada".
        - El sistema genera un mensaje de alerta: "Intrusión detectada en Sala de Estar - 18:30".
        - El sistema envía este mensaje a través de la API de Telegram al chat o grupo de usuario preconfigurado.
        - El cliente de Telegram del usuario recibe y muestra la notificación.
        - El sistema registra este evento, incluyendo el tipo de evento, la hora, la ubicación y que se envió una notificación.
    - Postcondiciones:
        - El usuario recibe una notificación instantánea y detallada en Telegram en <3s de la detección del evento.

        - La notificación incluye el tipo de evento, la hora y el sensor/zona específica.
        - El evento se registra en el historial del sistema.
    - Flujos Alternativos:
        - API de Telegram Caída: Si la API de Telegram no responde, el sistema intenta reenviar la notificación varias veces y luego registra un error crítico indicando el fallo en la entrega.
        - Fallo de Red: Si el sistema pierde la conectividad a internet, pone la notificación en cola para enviarla una vez que se restablezca la conexión y registra el problema de conectividad.
        - Múltiples Activaciones: Si el mismo sensor se activa repetidamente en un corto período, el sistema puede consolidar las notificaciones o enviar una sola actualización para evitar abrumar al usuario (ej., "Actividad continua en Sala de Estar").

- Escenario para RF3.2: Verificación de Identidad Facial

Nombre del Caso de Uso: Verificación de Identidad Facial

Identificador: RF3.2

Prioridad: Alta

Actores: Individuo (Usuario Autorizado o No Autorizado), Sistema, Sistema Externo (Cámara, Base de Datos de Rostros).

- Escenario 1: Verificación Exitosa de Identidad Facial y Acceso Concedido
    - Precondiciones:
        - El sistema de reconocimiento facial está activo en un punto de acceso (ej., puerta de entrada).
        - El sistema tiene una base de datos de patrones faciales de usuarios autorizados (ej., el rostro de Juan Pérez está registrado).
        - El dispositivo de acceso (ej., cerradura eléctrica) está conectado y operativo.
        - Las cámaras del sistema están funcionando y proporcionando imágenes claras.
        - Disparador: Juan Pérez se para frente a la cámara de reconocimiento facial en el punto de acceso.
    - Flujo de Eventos:
        - La cámara captura una imagen del rostro de Juan Pérez.
        - El sistema procesa la imagen para extraer patrones faciales.

        - El sistema compara estos patrones con la base de datos de rostros autorizados.
        - El sistema encuentra una coincidencia con Juan Pérez con un nivel de confianza por encima del umbral predefinido (>95%).
        - El sistema determina que Juan Pérez es un usuario autorizado.
        - El sistema envía una orden a la cerradura eléctrica para desbloquear la puerta (o desactivar la alarma en esa zona).
        - El sistema registra el intento de acceso exitoso: "Acceso concedido a Juan Pérez - Puerta Principal".
        - El sistema envía una notificación a Telegram: "Acceso concedido a Juan Pérez - Puerta Principal".
    - Postcondiciones:
        - El dispositivo de acceso se desbloquea en <2 segundos.
        - Juan Pérez obtiene acceso al área restringida.
        - Se almacena un registro detallado del intento de acceso, incluyendo fecha, hora y (si es posible) una imagen del rostro capturado.
        - Se envía una notificación de acceso exitoso a Telegram.
    - Flujos Alternativos:
        - Falso Positivo (Poco probable): Si una persona no autorizada es identificada incorrectamente como autorizada, el sistema registra el incidente para su revisión y envía una alerta sobre una posible violación de seguridad.
        - Fallo de Cámara: Si la cámara no logra capturar una imagen clara, el sistema muestra "Error de cámara. Por favor, reintente."

- Escenario 2: Verificación Fallida de Identidad Facial (Intento No Autorizado)
    - Precondiciones:
        - El sistema de reconocimiento facial está activo en un punto de acceso.
        - El individuo que intenta acceder NO está registrado en la base de datos de usuarios autorizados.
        - Las cámaras del sistema están funcionando y proporcionando imágenes claras.
    - Disparador: Un individuo desconocido se para frente a la cámara de reconocimiento facial en el punto de acceso.
    - Flujo de Eventos:
        - La cámara captura una imagen del rostro del individuo.
        - El sistema procesa la imagen para extraer patrones faciales.
        - El sistema compara estos patrones con la base de datos de rostros autorizados.

        - El sistema NO encuentra una coincidencia o el nivel de confianza está por debajo del umbral predefinido.
        - El sistema determina que el individuo no está autorizado.
        - El sistema envía una orden para mantener la cerradura eléctrica segura (o activar una alarma).
        - El sistema registra el intento de acceso fallido: "Intento de acceso no autorizado detectado".
        - El sistema envía una notificación a Telegram: "Intento de acceso no autorizado detectado - Puerta Principal (10:15)".
    - Postcondiciones:
        - Se deniega el acceso.
        - Se almacena un registro detallado del intento de acceso fallido, incluyendo fecha, hora y (si es posible) una imagen del rostro capturado.
        - Se envía una notificación del intento de acceso fallido a Telegram.
    - Flujos Alternativos:
        - Falso Negativo (Usuario legítimo no reconocido): Si un usuario legítimo no es reconocido (ej., debido a la iluminación, gafas nuevas), el sistema puede pedirle que lo intente de nuevo u ofrecer un método de acceso alternativo (ej., PIN). También registra el intento fallido para su revisión.
        - Anti-spoofing Activado (RF3.7, si se implementa): Si el sistema detecta un intento de suplantación (ej., una foto), deniega el acceso y envía una alerta específica como "Intento de suplantación de identidad detectado".

- Escenario para RF5.1: Registro Detallado de Eventos

Nombre del Caso de Uso: Registro Detallado de Eventos

Identificador: RF5.1

Prioridad: Alta

Actores: Sistema.

- Escenario: Registro de Cambio de Estado del Sistema e Intento de Acceso
    - Precondiciones:
        - El mecanismo de registro del sistema está activo y configurado para almacenar eventos de forma persistente.
        - Hay suficiente espacio de almacenamiento disponible.

- Disparador 1: Un usuario (ej., Andrea Algarín) desarma con éxito el sistema a través de la aplicación móvil.
- Flujo de Eventos (para Desarmado del Sistema):
  - El sistema procesa el comando "Desarmar".
  - El sistema registra una nueva entrada en su log de eventos.
  - Esta entrada incluye:
    - Marca de Tiempo: Fecha y hora actual (ej., 2025-07-20 16:50:30).
    - Tipo de Evento: "Cambio de Estado del Sistema".
    - Detalles: "Sistema desarmado por Andrea Algarín vía aplicación móvil".
    - Origen: "Aplicación Móvil - Usuario: Andrea Algarín."
  - El sistema asegura que el registro sea inmutable.
- Disparador 2: Un individuo no autorizado intenta el acceso facial en la puerta principal.
- Flujo de Eventos (para Intento de Acceso Fallido):
  - El módulo de reconocimiento facial detecta un rostro no reconocido.
  - El sistema registra otra nueva entrada en su log de eventos.
  - Esta entrada incluye:
    - Marca de Tiempo: Fecha y hora actual (ej., 2025-07-20 16:51:15).
    - Tipo de Evento: "Intento de Acceso Fallido".
    - Detalles: "Intento de acceso no autorizado detectado en Puerta Principal".
    - Origen: "Reconocimiento Facial - Ubicación: Puerta Principal."
    - Imagen (Opcional): Una pequeña miniatura o referencia a la imagen capturada.
  - El sistema asegura que el registro sea inmutable.
- Postcondiciones:
  - Todos los eventos significativos (cambios de estado del sistema, intentos de acceso exitosos y fallidos) se registran automáticamente, de forma persistente y con alta precisión.
  - Cada entrada de log contiene detalles relevantes para auditoría y análisis.
  - Se mantiene la integridad de los logs, impidiendo alteraciones no autorizadas.
  - Los logs se conservan durante al menos 30 días.

- Modelo Estructural: Diagrama de Clases.

El Diagrama de Clases representa la estructura estática de la aplicación, mostrando las clases del sistema, sus atributos, métodos y relaciones. Este modelo se centra en los objetos de negocio y sus interacciones dentro del dominio.

- Explicación de Clases y Relaciones:

1. SistemaSeguridad (SecuritySystem): La clase central que gestiona las operaciones generales de seguridad. Orquesta las interacciones entre los diferentes componentes.

    - Atributos: **estado: EstadoSistema** (Armado/Desarmado), **nombreSistema: String.**

    - Métodos: **armar(), desarmar(), procesarEvento(evento: Evento), enviarNotificacion(notificacion: Notificacion).**

    - Relaciones:

        - Compone **ZonaMonitoreo** (puede tener múltiples zonas).
        - Agrega **Usuario** (los usuarios interactúan con el sistema).
        - Se asocia con **HistorialEventos** (gestiona el registro de eventos).
        - Utiliza **ModuloNotificaciones** para enviar alertas.
        - Utiliza **ModuloReconocimientoFacial** para el control de acceso.

2. Usuario (User): Representa a cualquier usuario del sistema, incluidos propietarios, empleados, etc.

    - Atributos: **idUsuario: String, nombre: String, email: String, telefonoTelegram: String.**

    - Métodos: **autenticar(), cambiarContrasena().**

    - Relaciones:
        - Asociado con **SistemaSeguridad**.
        - Asociado con **PerfilFacial** (si aplica).
        - Tiene un **RolUsuario**.

3. Administrador (Administrator): Un tipo especializado de Usuario con permisos elevados.

    - Atributos: (Hereda de **Usuario**).

    - Métodos: **crearUsuario(), editarUsuario(), eliminarUsuario(), asignarRol(), revocarAcceso().**

    - Relaciones: Hereda de **Usuario**.

4. RolUsuario (UserRole): Define los permisos y funcionalidades disponibles para un usuario (ej., Administrador, Usuario Regular).

    - Atributos: **nombreRol: String, permisos: List<String>.**

    - Relaciones: Asociado con **Usuario** (un usuario tiene un rol).

5. ZonaMonitoreo (MonitoringZone): Representa un área definida dentro del espacio físico que es monitoreada por sensores.

    - Atributos: **idZona: String, nombreZona: String, descripcion: String.**

    - Relaciones:
        - Compone **Sensor** (una zona contiene múltiples sensores).
        - Asociado con **SistemaSeguridad**.

6. Sensor: Una clase abstracta que representa cualquier tipo de sensor (movimiento, contacto de puerta/ventana, vibración).

    - Atributos: **idSensor: String, tipo: String, ubicacion: String, sensibilidad: int, estado: Boolean** (activo/inactivo).

    - Métodos: **detectar().**

    - Relaciones:
        - Agregado por **ZonaMonitoreo**.
        - Heredado por tipos de sensores concretos.

7. Sensor de Movimiento (MotionSensor), SensorPuertaVentana (DoorWindowSensor): Implementaciones concretas de la clase **Sensor**.
    - Atributos: (Atributos específicos relacionados con su tipo, si los hay)
    - Métodos: (Lógica de detección específica).
    - Relaciones: Hereda de **Sensor**.

8. ModuloNotificaciones (NotificationModule): Gestiona la generación y el envío de notificaciones, principalmente a Telegram.
    - Atributos: **configuracionTelegram: String** (clave API, ID de chat).
    - Métodos: **enviarAlertaTelegram(mensaje: String), adjuntarMedio(medio: Media).**
    - Relaciones: Utiliza **Notificacion**.

9. Notificacion (Notification): Representa un mensaje enviado a los usuarios.
    - Atributos: **idNotificacion: String, tipo: String, mensaje: String, fechaHora: DateTime, destinatarios: List<String>, estadoEnvio: String.**
    - Relaciones: Generado por **ModuloNotificaciones.**

10. ModuloReconocimientoFacial (FacialRecognitionModule): Gestiona el registro de patrones faciales y la verificación de identidad.
    - Atributos: **umbralConfianza: double.**
    - Métodos: **registrarRostro(imagen: Imagen, idUsuario: String), verificarIdentidad(imagen: Imagen): Usuario, detectarVidas().**
    - Relaciones:
        - Agrega **PerfilFacial**.
        - Interactúa con **DispositivoAcceso**.

11. PerfilFacial (FacialProfile): Almacena los datos biométricos (patrones faciales) de un usuario autorizado.
    - Atributos: **idPerfil: String, patronFacial: Blob** (datos binarios).
    - Relaciones: Asociado con Usuario.

12. DispositivoAcceso (AccessDevice): Representa los dispositivos físicos controlados por el sistema para el acceso (ej., cerradura eléctrica).
    - Atributos: **idDispositivo: String, tipo: String, estado: String.**
    - Métodos: **abrir(), cerrar().**
    - Relaciones: Utilizado por **ModuloReconocimientoFacial**.

13. HistorialEventos (EventHistory): Gestiona el almacenamiento persistente y la recuperación de todos los eventos del sistema.
    - Atributos: **rutaAlmacenamiento: String, tiempoRetencion: int** (días).
    - Métodos**: registrarEvento(evento: Evento), consultarEventos(filtros: Map), exportarEventos(formato: String).**
    - Relaciones: Agrega **Evento**.

14. Evento (Event): Una clase abstracta que representa cualquier ocurrencia significativa en el sistema.
    - Atributos: **idEvento: String, tipoEvento: String, fechaHora: DateTime, descripcion: String.**
    - Relaciones: Heredado por tipos de eventos concretos.

15. EventoIntrusion (IntrusionEvent), EventoAcceso (AccessEvent), EventoSistema (SystemEvent): Implementaciones concretas de la clase Evento, que representan tipos específicos de ocurrencias.

- Atributos: (Atributos específicos como **zonaAfectada**, **usuarioInvolucrado**, **exitoso**).
- Relaciones: Hereda de Evento.