Universidad Nacional Experimental de las Telecomunicaciones e Informática.

Trayecto: 2

U.C: Proyecto Socio – Tecnológico

Ingeniería de Requisitos

Integrantes:

Ángel Mujica

Andrea Algarín.

Luis Fernández

Héctor Pacheco. Yuli Delgado.

Celenia León

Profesor

Yuli Delgado.

Definir la aplicación estableciendo para ello: su alcance, sus usuarios directos, sus aspectos claves (servicios que la aplicación debe proveer a los usuarios o la comunidad) y su ámbito de operación.

Alcance

Esta aplicación está diseñada para mejorar la seguridad y el monitoreo de espacios físicos, como hogares, oficinas o almacenes. Su objetivo es proporcionar a los usuarios notificaciones en tiempo real sobre eventos inusuales o intrusiones, y ofrecer un método de acceso seguro y sin contacto mediante reconocimiento facial.

Usuarios Directos

Los usuarios principales de este sistema serían:

- Propietarios de Viviendas: Personas que desean proteger su hogar y sus bienes, recibiendo alertas instantáneas sobre posibles intrusiones o actividades sospechosas.
- 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, como laboratorios o depósitos.
- Personas con Movilidad Reducida: Aquellos que se beneficiarían de un sistema de alarma y acceso remoto que no requiera su presencia física constante

Aspectos Clave (Servicios)

Los servicios principales que esta aplicación ofrecería a sus usuarios o a la comunidad son:

- Monitoreo y Detección de Intrusiones:
    - Alertas Inteligentes: El sistema se conectaría con sensores (movimiento, apertura de puertas/ventanas) para detectar actividades sospechosas y enviaría alertas inmediatas.
    - Control Remoto: Los usuarios podrían armar o desarmar la alarma a distancia desde su teléfono.
    - Zonas Personalizables: Sería posible configurar diferentes áreas de monitoreo para una seguridad más detallada.
- Alertas y Notificaciones en Tiempo Real a Través de Telegram:
    - Mensajes Instantáneos: Todas las alertas (ej. sensor activado, intento de acceso no autorizado) se enviarían directamente al chat de Telegram del usuario o de un grupo designado.
    - Información Detallada: Las notificaciones incluirían el tipo de evento, la hora y la ubicación afectada.
    - Integración con Cámaras (Opcional): Si se conecta con cámaras, podría enviar imágenes o videos cortos del evento.

    - Comandos por Telegram: Los usuarios podrían interactuar con la alarma enviando comandos sencillos por Telegram (ej. "armar", "desarmar").
- Acceso con Reconocimiento Facial:
    - Registro de Usuarios: Se registrarían los perfiles de los usuarios autorizados, asociando sus rostros al sistema. ○ Verificación Automática: La aplicación utilizaría el reconocimiento facial para verificar la identidad de quienes intentan acceder. ○ Control de Acceso Integrado: Al reconocer a un usuario autorizado, el sistema podría abrir puertas automáticamente o desactivar la alarma.
    - Registro de Accesos: Mantendría un historial detallado de todos los intentos de acceso, incluyendo la fecha, hora y una imagen del rostro si es posible.
    - Notificaciones de Acceso: Se enviarían alertas a Telegram cada vez que alguien acceda o intente un acceso no autorizado.
- Gestión de Usuarios y Permisos:
    - Perfiles Personalizables: Se podrían definir distintos niveles de acceso para diferentes usuarios (administrador, usuario regular, etc.). ○ Programación de Acceso: Permitiría establecer horarios o días específicos en los que ciertos usuarios tienen permiso para entrar.
- Historial y Registro de Eventos:
    - Log de Actividad: Un registro completo de todos los eventos del sistema (activaciones, accesos, notificaciones) estaría disponible para consulta.
    - Búsqueda y Filtro: Funcionalidades para buscar y filtrar eventos específicos en el historial.

¿Ámbito de Operación

El sistema podría operar en diversos entornos, incluyendo:

- Residencial: Apartamentos, casas y condominios.
- Comercial: Oficinas, pequeñas y medianas empresas, locales comerciales.
- Almacenes y Depósitos: Espacios de almacenamiento de bienes o equipos.
- Puntos de Acceso: Cualquier entrada que requiera un control de acceso seguro y monitoreado.

Identificar, para cada proceso de negocio, los requisitos (requerimientos) funcionales; esto es, qué funciones deberá realizar la aplicación para que los actores puedan ejecutar el proceso.

1. Proceso de Negocio: Monitoreo y Detección de Intrusión

Este proceso asegura la vigilancia constante del área protegida y la identificación de actividades sospechosas.

- RF1.1. Gestión de Sensores: La aplicación debe permitir al usuario agregar, configurar y eliminar diferentes tipos de sensores (ej., movimiento PIR, contacto de puerta/ventana, vibración).
- RF1.2. Activación/Desactivación del Sistema: El sistema debe permitir al usuario activar (armar) y desactivar (desarmar) la alarma de forma remota a través de la aplicación móvil o comandos de Telegram.
- RF1.3. Detección de Eventos: El sistema debe ser capaz de detectar activaciones de sensores y clasificarlas como eventos de seguridad (ej., intrusión detectada, puerta abierta).
- RF1.4. Configuración de Zonas de Monitoreo: La aplicación debe permitir al usuario definir y asignar sensores a diferentes zonas de monitoreo (ej., "Sala de Estar", "Dormitorio Principal", "Acceso Principal").
- RF1.5. Ajuste de Sensibilidad: La aplicación debe permitir al usuario ajustar la sensibilidad de los sensores para minimizar falsas alarmas (ej., nivel de detección de movimiento).
- RF1.6. Estado del Sistema en Tiempo Real: La aplicación debe mostrar el estado actual del sistema (armado/desarmado, estado de los sensores) en tiempo real.

2. Proceso de Negocio: Alertas y Notificaciones en Tiempo Real

Este proceso garantiza que los usuarios sean informados inmediatamente sobre cualquier evento relevante detectado por el sistema.

- RF2.1. Envío de Notificaciones a Telegram: El sistema debe enviar notificaciones instantáneas y detalladas a un chat o grupo de Telegram preconfigurado cuando se detecta un evento (ej., "Intrusión detectada en Sala de Estar - 18:30").
- RF2.2. Personalización de Mensajes de Alerta: La aplicación debe permitir al usuario personalizar el contenido de los mensajes de alerta enviados a Telegram (ej., incluir nombres de zonas, tipos de sensores).
- RF2.3. Envío de Medios (Opcional): Si se integra con cámaras, el sistema debe ser capaz de adjuntar imágenes o clips de video cortos a las notificaciones de Telegram para verificar el evento.
- RF2.4. Comandos Remotos por Telegram: El sistema debe ser capaz de recibir y ejecutar comandos simples enviados a través de Telegram (ej., "desarmar alarma", "solicitar estado").

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

3. Proceso de Negocio: Acceso con Reconocimiento Facial

Este proceso gestiona el acceso seguro a través de la identificación biométrica.

- RF3.1. Registro de Rostros de Usuarios: La aplicación debe permitir a los administradores registrar y almacenar los patrones faciales de usuarios autorizados, asociándolos a un perfil de usuario único.
- RF3.2. Verificación de Identidad Facial: 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.
- RF3.3. Concesión de Acceso Automático: Si se verifica la identidad del usuario, el sistema debe desencadenar la apertura de un dispositivo de acceso (ej., cerradura eléctrica, torniquete) o desactivar la alarma en la zona correspondiente.
- RF3.4. Registro de Intentos de Acceso: 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.
- RF3.5. Notificaciones de Acceso (Exitoso/Fallido): El sistema debe enviar notificaciones a Telegram sobre los intentos de acceso, tanto exitosos (ej., "Acceso concedido a Juan Pérez - Puerta Principal") como fallidos (ej., "Intento de acceso no autorizado detectado - 10:15").
- RF3.6. Gestión de Perfiles Faciales: La aplicación debe permitir a los administradores actualizar o eliminar los patrones faciales de los usuarios.
- RF3.7. Detección de Vidas (Anti-spoofing): (Opcional, pero recomendado para alta seguridad) El sistema debe implementar mecanismos para detectar intentos de suplantación de identidad (ej., uso de fotos o videos).

4. Proceso de Negocio: Gestión de Usuarios y Permisos

Este proceso permite controlar quién tiene acceso al sistema y qué acciones puede realizar.

- RF4.1. Creación y Edición de Perfiles de Usuario: La aplicación debe permitir a un administrador crear, editar y eliminar perfiles de usuario con información relevante (nombre, rol).
- RF4.2. Asignación de Roles y Permisos: La aplicación debe permitir asignar roles a los usuarios (ej., "Administrador", "Usuario Regular") con permisos específicos sobre las funcionalidades del sistema (ej., armar/desarmar, ver historial, añadir usuarios).
- RF4.3. Programación de Acceso por Usuario: La aplicación debe permitir establecer horarios o días específicos en los que un usuario tiene permitido el acceso mediante reconocimiento facial.
- RF4.4. Revocación de Acceso: La aplicación debe permitir revocar de inmediato el acceso de un usuario al sistema o a través del reconocimiento facial.

5. Proceso de Negocio: Historial y Registro de Eventos

Este proceso permite la auditoría y el análisis retrospectivo de la actividad del sistema.

- RF5.1. Registro Detallado de Eventos: 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.
- RF5.2. Consulta del Historial de Eventos: La aplicación debe permitir al usuario consultar el historial completo de eventos del sistema.
- RF5.3. Filtrado y Búsqueda de Eventos: La aplicación debe proporcionar funcionalidades de filtrado y búsqueda para eventos específicos (ej., por fecha, tipo de evento, zona, usuario).
- RF5.4. Exportación de Registros: La aplicación debe permitir la exportación del historial de eventos en un formato legible (ej., CSV, PDF) para fines de auditoría o respaldo.

Organizar los requisitos: identificarlos, clasificarlos y asignarles prioridades

Clasificación de Prioridades:

- Alta (Obligatorio para MVP): Funcionalidades esenciales que el sistema debe tener para ser operativo y cumplir su propósito principal. Sin ellas, el sistema no es viable.
- Media (Importante para MVP, Mejora Significativa): Funcionalidades que añaden valor considerable y mejoran la usabilidad o seguridad. Son deseables para el MVP, pero el sistema podría operar con limitaciones sin ellas inicialmente.
- Baja (Deseable, Futuras Mejoras): Funcionalidades que añaden comodidad, optimización o características avanzadas. Pueden ser implementadas en fases posteriores del desarrollo.

1. Proceso de Negocio: Monitoreo y Detección de Intrusión

Este es el corazón del sistema de alarma.

- RF1.1. Gestión de Sensores: Permite agregar y configurar sensores.
  - Clasificación: Funcional.
  - Prioridad: Alta.
- RF1.2. Activación/Desactivación del Sistema: Permite armar y desarmar la alarma.
  - Clasificación: Funcional.
  - Prioridad: Alta.
- RF1.3. Detección de Eventos: El sistema debe detectar las activaciones de los sensores.
  - Clasificación: Funcional.
  - Prioridad: Alta.
- RF1.4. Configuración de Zonas de Monitoreo: Permite definir áreas específicas para los sensores.
  - Clasificación: Funcional.
  - Prioridad: Media.
- RF1.5. Ajuste de Sensibilidad: Permite configurar la sensibilidad de los sensores.
  - Clasificación: Funcional.
  - Prioridad: Media.
- RF1.6. Estado del Sistema en Tiempo Real: Muestra el estado actual de la alarma y los sensores. ○ Clasificación: Funcional. ○ Prioridad: Alta. 2. Proceso de Negocio: Alertas y Notificaciones en Tiempo Real

La comunicación es clave para un sistema de alarma.

- RF2.1. Envío de Notificaciones a Telegram: Notificaciones instantáneas y detalladas a Telegram. ○ Clasificación: Funcional. ○ Prioridad: Alta.

- RF2.2. Personalización de Mensajes de Alerta: Permite adaptar el contenido de las alertas.
    - Clasificación: Funcional.
    - Prioridad: Media.
- RF2.3. Envío de Medios (Opcional): Adjunta imágenes/videos a las notificaciones.
    - Clasificación: Funcional.
    - Prioridad: Baja (Requiere integración de cámara, que no es un requisito base para la alarma).
- RF2.4. Comandos Remotos por Telegram: Permite controlar la alarma desde Telegram.
    - Clasificación: Funcional.
    - Prioridad: Alta.
- RF2.5. Gestión de Destinatarios de Alertas: Permite añadir/eliminar quién recibe las alertas.
    - Clasificación: Funcional.
    - Prioridad: Alta.
- RF2.6. Historial de Notificaciones Enviadas: Mantiene un registro de las alertas enviadas.
    - Clasificación: Funcional. ○ Prioridad: Media.

3. Proceso de Negocio: Acceso con Reconocimiento Facial

El componente de seguridad y acceso avanzado del sistema.

- RF3.1. Registro de Rostros de Usuarios: Permite registrar los patrones faciales de usuarios autorizados. ○ Clasificación: Funcional.
    - Prioridad: Alta.
- RF3.2. Verificación de Identidad Facial: El sistema debe verificar la identidad del rostro capturado. ○ Clasificación: Funcional.
    - Prioridad: Alta.
- RF3.3. Concesión de Acceso Automático: Abre el dispositivo de acceso o desarma la alarma si la verificación es exitosa.
    - Clasificación: Funcional.
    - Prioridad: Alta.
- RF3.4. Registro de Intentos de Acceso: Guarda un log de todos los intentos de acceso.
    - Clasificación: Funcional.
    - Prioridad: Media.
- RF3.5. Notificaciones de Acceso (Exitoso/Fallido): Envía alertas a Telegram sobre los intentos de acceso. ○ Clasificación: Funcional.
    - Prioridad: Alta.
- RF3.6. Gestión de Perfiles Faciales: Permite actualizar o eliminar los patrones faciales.
    - Clasificación: Funcional.
    - Prioridad: Media.
- RF3.7. Detección de Vidas (Anti-spoofing): Evita la suplantación de identidad con fotos/videos.
    - Clasificación: Funcional.
    - Prioridad: Baja (Importante para seguridad avanzada, pero puede ser una mejora futura para el MVP).

4. Proceso de Negocio: Gestión de Usuarios y Permisos

Fundamental para la administración y control del sistema.

- RF4.1. Creación y Edición de Perfiles de Usuario: Permite gestionar los usuarios del sistema. ○ Clasificación: Funcional.
  - Prioridad: Alta.
- RF4.2. Asignación de Roles y Permisos: Define lo que cada tipo de usuario puede hacer.
  - Clasificación: Funcional.
  - Prioridad: Alta.
- RF4.3. Programación de Acceso por Usuario: Permite establecer horarios de acceso para usuarios. ○ Clasificación: Funcional.
  - Prioridad: Media.
- RF4.4. Revocación de Acceso: Permite eliminar rápidamente el acceso de un usuario.
  - Clasificación: Funcional. ○ Prioridad: Alta. 5. Proceso de Negocio:

Historial y Registro de Eventos

Necesario para auditoría y seguimiento.

- RF5.1. Registro Detallado de Eventos: Guarda un registro de todas las actividades importantes. ○ Clasificación: Funcional.
  - Prioridad: Alta.
- RF5.2. Consulta del Historial de Eventos: Permite a los usuarios ver el log de actividades.
  - Clasificación: Funcional.
  - Prioridad: Alta.
- RF5.3. Filtrado y Búsqueda de Eventos: Permite encontrar eventos específicos en el historial.
  - Clasificación: Funcional.
  - Prioridad: Media.
- RF5.4. Exportación de Registros: Permite descargar el historial para auditorías.
  - Clasificación: Funcional. ○ Prioridad: Baja.

Utilizar la plantilla VOLERE o cualquier otra plantilla para documentar los requisitos más importantes.

Caso de

dentificador del Requisito:	Tipo de Requisito:	Uso/Evento:
RF1.2	Funcional	Armar Sistema, Desarmar Sistema, Control Remoto

Descripción:
El sistema debe permitir a los usuarios armar (activar) y desarmar (desactivar) la alarma de seguridad de forma remota. Esto se realizará a través de la aplicación móvil dedicada y también mediante el envío de comandos específicos a un chat de Telegram asociado. Tras la ejecución, el sistema debe enviar una confirmación del cambio de estado al usuario.

Justificación del requisito:
Es fundamental para el control y la gestión remota de la seguridad del espacio. Permite al usuario interactuar con el sistema sin estar físicamente presente, lo cual es crucial para la conveniencia y la respuesta rápida.

Fuente (que interesado lo propone):	Unidad en la que se origina:
Usuarios Finales, Propietarios de Viviendas, Propietarios de Negocios, Administradores.	Seguridad del Hogar/Negocio

Criterios de validación:
- El usuario puede armar el sistema enviando el comando /armar a Telegram y recibe confirmación en <5s.
- El usuario puede desarmar el sistema desde la app móvil y el estado se actualiza en <2s. - El sistema rechaza y notifica intentos de desarmado por usuarios no autorizados.

Grado de satisfacción del interesado:	Grado de insatisfacción del interesado:
Muy Alta (control total y remoto).	Baja (si la funcionalidad no es fiable o lenta).
Dependencias (qué requisitos depende de este):	Conflictos (qué requisitos son incompatibles o inconsistentes con este):

- RF1.6 (Estado del Sistema en Tiempo Real) - RF2.4 (Comandos Remotos por Telegram)	N/A	
dentificador del Requisito:	Tipo de Requisito:	Caso de Uso/Evento:
Documentos de soporte:	Histórico de cambios:	
Especificación de Interfaz de Telegram API, Diseño de Interfaz de Usuario de Aplicación Móvil.	1.0 (2025-07-02): Creación inicial.	
Proyecto:	Analista:	
Sistema de Alerta de Telegram con Acceso Facial	[Tu Nombre]	

Exportar a Hojas de cálculo

---

2. Requisito Clave: RF2.1 - Envío de Notificaciones a Telegram

Identificador del Requisito:	Tipo de Requisito:	Caso de Uso/Evento:
RF2.1	Funcional	Alerta de Intrusion, Notificación de Evento de Seguridad

Descripción:

El sistema debe enviar notificaciones instantáneas, claras y detalladas a un chat o grupo de Telegram preconfigurado cuando se detecte un evento de seguridad. Las notificaciones deben incluir la hora del evento, el tipo de evento y la ubicación o sensor específico que lo activó.

Justificación del requisito:

Es el pilar de la funcionalidad de alerta del sistema. Proporciona a los usuarios información crucial en tiempo real,

permitiéndoles tomar decisiones informadas y rápidas ante cualquier incidencia de seguridad.

Fuente (que interesado lo propone):	Unidad en la que se origina:
Usuarios Directos (Propietarios), Expertos en Seguridad.	Usuarios Finales, Operaciones de Seguridad

Criterios de validación:

- Una vez detectado el evento, la notificación aparece en Telegram en <3s.
- La notificación incluye tipo de evento, hora y sensor/zona. - Las notificaciones se envían solo a los destinatarios configurados.

	Caso de Uso/Evento:
Tipo de Requisito:	
Grado de satisfacción del interesado:	Grado de insatisfacción del interesado:
Muy Alta (información en tiempo real).	Alta (si las notificaciones se retrasan o no llegan).
Dependencias (qué requisitos depende de este): - RF1.3 (Detección de Eventos) - RF3.5 (Notificaciones de Acceso)	Conflictos (qué requisitos son incompatibles o inconsistentes con este): N/A
Documentos de soporte:	Histórico de cambios:
Especificación de Interfaz de Telegram API, Diagrama de Flujo de Eventos de Alarma.	1.0 (2025-07-02): Creación inicial.
Proyecto: Sistema de Alerta de Telegram con Acceso Facial	Analista: [Tu Nombre]

Exportar a Hojas de cálculo

---

3. Requisito Clave: RF3.2 - Verificación de Identidad Facial

Identificador del Requisito:	Tipo de Requisito:	Caso de Uso/Evento:
RF3.2	Funcional	Acceso a Zona Restringida, Verificación de Ingreso

Descripción:
En el punto de acceso, el sistema debe capturar una imagen del rostro del individuo y compararla en tiempo real con la base de datos de patrones faciales de usuarios autorizados. El sistema debe determinar una coincidencia con un umbral de confianza predefinido para verificar la identidad.

Justificación del requisito:

Es la columna vertebral del sistema de control de acceso biométrico. Proporciona una capa de seguridad robusta y un método de acceso conveniente y sin contacto, mejorando la eficiencia y reduciendo la necesidad de llaves físicas.

Caso de Uso/Evento:

Tipo de Requisito:

Unidad en la que se origina:

Seguridad, I+D (Investigación y Desarrollo)

Fuente (que interesado lo propone):

Gerencia de Seguridad, Propietarios (conveniencia y seguridad).

Criterios de validación:

- Identifica correctamente a usuarios autorizados con una tasa de éxito del >95% en condiciones normales.
- Tiempo de verificación facial no excede 2 segundos.
- No concede acceso a individuos no registrados.

Grado de satisfacción del interesado:

Muy Alta (seguridad y comodidad).

Grado de insatisfacción del interesado: Alta (si hay falsos negativos/positivos, o es lento). Conflictos (qué requisitos son incompatibles o inconsistentes con este):

Dependencias (qué requisitos depende de este):

- RF3.1 (Registro de Rostros de Usuarios)
- RF3.3 (Concesión de Acceso Automático)
- RNF-PRECISION (Precisión de Reconocimiento Facial)

N/A

Documentos de soporte:

Especificaciones de Módulo de Reconocimiento Facial, Políticas de Datos Biométricos.

Histórico de cambios:

1.0 (2025-07-02): Creación inicial. Privacidad

Proyecto:

Sistema de Alerta de Telegram con Acceso

Analista:

[Tu Nombre]

Facial

Exportar a Hojas de cálculo

---

4. Requisito Clave: RF5.1 - Registro Detallado de Eventos

Identificador del Requisito:	Tipo de Requisito:	Caso de Uso/Evento:
RF5.1	Funcional	Auditoría de Seguridad, Registro de Actividad del Sistema

Descripción:

El sistema debe registrar de forma persistente, inmutable y detallada todos los eventos significativos (activaciones de sensores, cambios de estado, intentos de acceso exitosos/fallidos-, notificaciones enviadas). Cada registro debe incluir una marca de tiempo precisa y detalles relevantes.

Justificación del requisito:

Es esencial para la auditoría de seguridad, la resolución de problemas y el análisis forense post-incidente. Proporciona un rastro de auditoría completo y mantiene la trazabilidad de la seguridad.

Fuente (que interesado lo propone):	Unidad en la que se origina:
del Sistema, Expertos en Seguridad, Requisitos de Auditoría.	Operaciones de Seguridad, Administradores

Cumplimiento Normativo

Criterios de validación:

- Todos los eventos críticos se registran automáticamente y con precisión.
- Cada registro incluye fecha, hora, tipo de evento y detalles.
- El historial de eventos es accesible y se mantiene por al menos 30 días. - La integridad de los registros es inalterable por usuarios no autorizados.

Grado de satisfacción del interesado:

Muy Alta (trazabilidad completa).

Grado de insatisfacción del interesado:

Alta (si los registros se pierden o son incompletos).

Dependencias (qué requisitos depende de este):

- RF1.3 (Detección de Eventos)
- RF3.4 (Registro de Intentos de Acceso)
- RF5.2 (Consulta del Historial de Eventos)

Conflictos (qué requisitos son incompatibles o inconsistentes con este):

N/A

Documentos de soporte:

Políticas de Retención de Datos, Normas de Auditoría de Seguridad.

Histórico de cambios:

1.0 (2025-07-02):

Creación inicial.

Proyecto:

Sistema de Alerta de Telegram con Acceso Facial

Analista:

[Tu Nombre]

Elaborar el modelo funcional de la aplicación usando diagramas de casos de usos que integre todos los requisitos funcionales de cada uno de los procesos

- 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:

o 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.

o 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.
    - o 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:

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

Elaborar al menos un escenario de cada uno de los casos de usos más importantes

- 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 o 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". o 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 o

  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". o 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 o 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). o Disparador: El sensor de movimiento PIR en la "Sala de Estar" detecta movimiento.

  o 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. o 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. o

  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 o 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. o 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) o 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. o 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)". o 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. o 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 o 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.

Elaborar el modelo estructural de la aplicación usando un diagrama de clases. (Este modelo representa los objetos de negocio del dominio y sus relaciones)

- 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.

Establecer y definir los requisitos no funcionales que deberá cumplir la aplicación.

Requisitos No Funcionales:

Estos requisitos no funcionales son esenciales para garantizar que el Sistema de Alerta de Telegram con Acceso por Reconocimiento Facial cumpla con las expectativas de los usuarios y los estándares de calidad necesarios para su operación efectiva y segura.

1. Rendimiento
1.1 Tiempo de Respuesta: La aplicación debe procesar el reconocimiento facial y enviar alertas a Telegram en un tiempo máximo de 2 segundos desde la detección del rostro.
1.2 Escalabilidad: La aplicación debe ser capaz de manejar al menos 1000 usuarios concurrentes sin degradar el rendimiento.

2. Seguridad
2.1 Autenticación y Autorización: La aplicación debe implementar un sistema de autenticación basado en reconocimiento facial, asegurando que solo los usuarios autorizados puedan acceder.

2.2 Protección de Datos
Todos los datos biométricos y de usuario deben ser almacenados de forma encriptada utilizando estándares de encriptación robustos (por ejemplo, AES256).
2.3 Seguridad en la Comunicación: Las comunicaciones entre la aplicación y Telegram, así como las interacciones con la base de datos, deben estar protegidas mediante HTTPS y protocolos seguros.

3. Usabilidad
3.1 Interfaz de Usuario: La interfaz debe ser intuitiva y fácil de usar, permitiendo a los usuarios interactuar con la aplicación sin necesidad de capacitación extensa.
3.2 Accesibilidad: La aplicación debe cumplir con las pautas de accesibilidad WCAG 2.1 para garantizar que sea utilizable por personas con discapacidades.

4. Mantenibilidad
4.1 Documentación: La aplicación debe contar con documentación técnica completa que incluya guías de instalación, configuración y mantenimiento.
4.2 Modularidad: El código debe ser modular y seguir principios de diseño que faciliten su mantenimiento y actualización.

5. Disponibilidad
5.1 Tiempo de Actividad: La aplicación debe garantizar un tiempo de actividad del 99.9%, lo que significa que no debe estar inactiva más de 43.2 minutos al mes.
5.2 Recuperación ante Desastres: Debe existir un plan de recuperación ante desastres que permita restaurar el servicio en un tiempo máximo de 1 hora tras una falla crítica.

6. Compatibilidad
6.1 Dispositivos: La aplicación debe ser compatible con dispositivos móviles (iOS y Android) y navegadores web modernos (Chrome, Firefox, Safari).
6.2 Integración con Telegram: La aplicación debe integrarse sin problemas con la API de Telegram, asegurando que las alertas se envíen de manera efectiva y eficiente.

7. Portabilidad

7.1 Plataforma: La aplicación debe ser capaz de ejecutarse en diferentes sistemas operativos (Windows, Linux, macOS) con el mismo nivel de funcionalidad.

8. Monitoreo y Auditoría

8.1 Registro de Actividades: La aplicación debe registrar todas las actividades relevantes, incluidas las alertas enviadas y los intentos de acceso, para su posterior auditoría.

8.2 Monitoreo de Rendimiento: Debe implementarse un sistema de monitoreo que permita evaluar el rendimiento y la disponibilidad de la aplicación en tiempo real.

Definir una estructura apropiada para el Documento de Requisitos de la aplicación.

Para definir una estructura apropiada para el Documento de Requisitos de una aplicación, es importante incluir secciones que aborden tanto los requisitos funcionales como los no funcionales, así como otros aspectos relevantes del proyecto# Estructura del Documento de Requisitos.

Esta estructura proporciona un marco claro y organizado para documentar los requisitos de la aplicación, asegurando que todos los aspectos relevantes sean considerados y comunicados efectivamente.

1. Introducción

1.1 Propósito: Describir el objetivo del documento y su importancia.
1.2 Alcance: Definir el alcance de la aplicación y las limitaciones.
1.3 Definiciones, Acrónimos y Abreviaturas: Explicar términos técnicos y acrónimos utilizados en el documento.
1.4 Referencias: Listar documentos y fuentes consultadas.

2. Descripción General

2.1 Perspectiva del Producto: Describir cómo se integra la aplicación en un sistema mayor.
2.2 Funciones del Producto: Resumen de las funciones principales que ofrecerá la aplicación
2.3 Usuarios Previstas: Identificar los diferentes tipos de usuarios que interactuarán con la aplicación
2.4 Restricciones: Detallar limitaciones técnicas y operativas.
3. Requisitos Funcionales
3.1 Requisitos de Usuario: Listar los requisitos desde la perspectiva del usuario.
3.2 Requisitos del Sistema: Detallar los requisitos técnicos y operativos que debe cumplir el sistema.
3.3 Casos de Uso: Incluir diagramas y descripciones de casos de uso relevantes.

Requisitos No Funcionales

4.1 Rendimiento: Especificar requisitos de velocidad, capacidad y eficiencia.
4.2 Seguridad: Detallar requisitos de seguridad y protección de datos.
4.3 Usabilidad: Describir los requisitos relacionados con la experiencia del usuario.
4.4 Mantenibilidad: Especificar requisitos para la facilidad de mantenimiento y actualización.

5. Requisitos de Interfaz

5.1 Interfaz de Usuario: Describir cómo será la interacción del usuario con la aplicación.
5.2 Interfaz del Sistema: Detallar la comunicación entre la aplicación y otros sistemas.
6. Requisitos de Implementación
6.1 Entorno de Desarrollo: Especificar el entorno y las herramientas que se utilizarán.
6.2 Lenguajes de Programación: Listar los lenguajes y tecnologías a utilizar.

7.Anexos

7.1 Diagramas: Incluir diagramas relevantes (UML, flujos de trabajo, etc.).
7.2 Documentación Adicional: Listar cualquier documento adicional que complemente el documento de requisitos.

8. Aprobaciones

8.1 Firmas: Espacio para las firmas de aprobación de las partes interesadas.

Definir las estrategias y técnicas que el grupo o estudiante deberá emplear para: Validar los requisitos, Hacerles seguimiento a los requisitos, Controlar los cambios.

Validar los requisitos.

•Hacerles seguimiento a los requisitos.
•Controlar los cambios.

La implementación de estas estrategias y técnicas permitirá un manejo efectivo de los requisitos en el desarrollo del sistema de alerta de Telegram con reconocimiento facial, asegurando que se cumplan las expectativas de los usuarios y se minimicen los riesgos asociados con cambios no controlados.

	1. Estrategias	2. Técnicas
1. Validación de Requisitos	- Reuniones de Requisitos: Organizar sesiones con los interesados (stakeholders) para discutir y aclarar los requisitos del sistema.	- Revisión de Requisitos: Realizar revisiones formales de los documentos de requisitos con el equipo y los interesados.
	- Prototipos: Crear prototipos del sistema que permitan a los usuarios visualizar y validar los requisitos antes de la implementación.	- Técnicas de Modelado: Usar diagramas de casos de uso o diagramas de flujo para representar visualmente los requisitos y facilitar su validación.
	- Historias de Usuario: Utilizar historias de usuario para capturar requisitos desde la perspectiva del usuario final, facilitando la comprensión y validación.	- Pruebas de Aceptación: Definir criterios de aceptación claros que se utilizarán para validar que los requisitos se han cumplido en el producto final.
2. Seguimiento de Requisitos	Gestión de Requisitos: Implementar herramientas de gestión de requisitos para rastrear el estado de cada requisito a lo largo del ciclo de vida del desarrollo.	Matriz de Trazabilidad: Crear una matriz de trazabilidad que vincule los requisitos a sus respectivas pruebas, asegurando que todos los requisitos sean cubiertos y verificados.

	Documentación Continua: Mantener una documentación actualizada que refleje el estado de los requisitos y cualquier cambio realizado.	Revisiones Periódicas: Establecer reuniones periódicas para revisar el progreso de los requisitos y ajustar el enfoque según sea necesario.
3. Control de Cambios	Proceso de Control de Cambios: Establecer un proceso formal para solicitar, evaluar y aprobar cambios en los requisitos.	Registro de Cambios: Mantener un registro detallado de todos los cambios realizados en los requisitos, incluyendo la razón del cambio y su impacto en el proyecto.
	Comunicación Clara: Asegurar que todos los interesados estén informados sobre los cambios en los requisitos y sus implicaciones	Evaluación de Impacto: Realizar análisis de impacto para cada cambio propuesto, evaluando cómo afectará el alcance, el tiempo y el costo del proyecto.

Introducción: descripción general del proyecto y su propósito, justificación y alcance.

En el contexto actual de alta conectividad, garantizar comunicaciones seguras y rápidas se ha convertido en un factor clave. Este proyecto plantea la creación de un Sistema de Alertas vía Telegram con Autenticación mediante Reconocimiento Facial, una solución que integra la inmediatez de la mensajería de Telegram con la seguridad que ofrece la biometría facial.

El objetivo principal es ofrecer una herramienta confiable para el envío de alertas relevantes únicamente a usuarios previamente validados. La propuesta surge como respuesta a las limitaciones de los sistemas de notificación convencionales, que suelen carecer de mecanismos sólidos de autenticación y pueden ser susceptibles a accesos no autorizados.

La incorporación del reconocimiento facial refuerza la seguridad al garantizar que solo personas autorizadas accedan a información sensible o ejecuten acciones críticas. El desarrollo contempla una plataforma web que permitirá gestionar usuarios, almacenar imágenes faciales, procesar la autenticación y coordinar el envío de alertas a través de la API de Telegram.

Asimismo, se incluirá una interfaz administrativa para la configuración del sistema. El diseño se enfocará en asegurar la escalabilidad, permitiendo adaptar la solución a diferentes contextos, desde el ámbito doméstico hasta aplicaciones corporativas.

Objetivos: los objetivos del proyecto y lo que se espera lograr con la aplicación web.

Los principales objetivos de esta iniciativa son:

Diseñar un sistema de autenticación confiable: Implementar reconocimiento facial con alta precisión y baja latencia para validar el acceso de los usuarios de forma segura.

Automatizar el envío de alertas: Desarrollar un mecanismo eficiente para emitir notificaciones a usuarios o grupos de Telegram, ya sea de forma manual o en respuesta a eventos configurados.

Proteger la información sensible: Aplicar estándares avanzados de seguridad en el manejo de datos e imágenes, incluyendo cifrado y control estricto de accesos.
Ofrecer una interfaz de administración clara e intuitiva: Permitir una experiencia de uso sencilla para la gestión de usuarios, alertas y parámetros del sistema.
Integrar eficazmente con Telegram: Asegurar una comunicación fluida mediante el uso dela API de Telegram para el envío de mensajes y administración del bot.
Preparar una arquitectura escalable:

Diseñar el sistema para soportar futuros crecimientos en usuarios, reglas de alerta y funcionalidades adicionales.

Requisitos funcionales: una lista de las funciones que la aplicación web debe realizar, como la autenticación de usuarios, la gestión de contenido y la generación de informes.

El sistema deberá contar con las siguientes capacidades:

Gestión de Usuarios:

Alta de usuarios: Registrar nuevos perfiles con datos personales e imágenes faciales para su posterior identificación.

Edición de perfiles: Permitir la actualización de información o el reemplazo de imágenes faciales.

Eliminación de usuarios: Posibilidad de eliminar registros completos del sistema.
Control de roles y permisos: Definir distintos niveles de acceso (administrador, operador, usuario) según funciones específicas.

Autenticación y Control de Acceso:

Reconocimiento facial: Validar la identidad mediante comparación de imagen facial con la base de datos.

Acceso alternativo por credenciales: Activar ingreso por usuario y contraseña para administradores como mecanismo complementario.

Recuperación de acceso: Proveer una opción para restablecer contraseñas cuando aplique.

Manejo de Alertas

Creación de alertas: Configurar tipos de alerta, prioridad, contenido y destinatarios.
Envío manual: Emitir notificaciones en tiempo real a contactos específicos.

Automatización de alertas: Definir eventos disparadores para envío automático (como detección de movimiento o lectura de sensores).

Historial de alertas: Registrar cada alerta enviada, incluyendo fecha, remitente y estado.

Integración con Telegram

Conexión segura con API: Establecer comunicación estable y protegida con la plataforma de Telegram.

Envío de contenidos: Permitir el envío de mensajes de texto, multimedia y documentos.

Recepción de comandos: Procesar respuestas de los usuarios al bot (por ejemplo,

confirmaciones o solicitudes).

Gestión de Contenido

Plantillas de mensaje: Crear estructuras predefinidas para agilizar el proceso de notificación.

Personalización dinámica: Insertar datos variables en los mensajes, como nombres o detalles específicos del evento.

Reportes y Auditoría

Reportes de actividad: Generar informes de uso del sistema y comportamiento del usuario.

Bitácora de acciones: Registrar todas las operaciones realizadas para auditoría y seguimiento.

Estadísticas: Mostrar métricas generales como número de alertas enviadas, usuarios activos, entre otros.

Configuración General

Parámetros del sistema: Ajustar opciones generales como umbrales de reconocimiento, horarios y zonas.

Gestión de notificaciones internas: Determinar qué avisos administrativos deben generarse ante cambios o incidencias del sistema.

Usuario Github:

Celenia-Leo (Puntos trabajo 8-10)
Hectuneti (Puntos trabajo 2da parte entregable 1-3)
Mujicaa459 (Puntos trabajo 5-7)
Revan2404 (Puntos trabajo 2da parte entregable 4-6)
Dakota18 (Puntos trabajo 1-4)

Link Github:
https://github.com/dakota18/RequisitoSistemas.git