

Instituto politécnico Nacional

Escuela Superior de Computo

Tarea 5

**Propuesta de proyecto final: Desarrollo de
aplicación movil nativa**

Alumno: Costilla Caballero Salvador

Profesor: Hurtado Avilés Gabriel

Fecha de entrega: 10.04.2025

1. Información General

Título de la aplicación

SleepCycle Alarm: Despertador Inteligente

Descripción conceptual

SleepCycle Alarm es una aplicación de despertador inteligente diseñada para despertar gradualmente a los usuarios respetando sus ciclos de sueño. En lugar de utilizar una única alarma abrupta, el sistema implementa una serie de alarmas progresivas que aumentan en intensidad, permitiendo una transición más natural del sueño a la vigilia. La aplicación ofrece por defecto tres alarmas previas a la hora objetivo, separadas por intervalos de 5 minutos, con intensidad incrementalmente mayor: comenzando con una vibración suave, seguida por vibración moderada con sonido tenue, y finalizando con una alarma completa. Los usuarios pueden personalizar todos los parámetros, incluyendo cantidad de alarmas previas, intervalos entre ellas e intensidad de cada una. Además, la versión nativa para Android cuenta con integración opcional con dispositivos de iluminación inteligente para simular un amanecer natural, aumentando gradualmente la intensidad de la luz sincronizada con las alarmas.

Público objetivo y casos de uso principales

Público objetivo:

- Personas con horarios de sueño irregulares que desean optimizar su despertar
- Usuarios que experimentan dificultad para despertarse con una única alarma
- Personas interesadas en mejorar su calidad de sueño y despertar
- Profesionales con horarios estrictos que necesitan asegurar su despertar a tiempo

Casos de uso principales:

- Despertar gradual y menos estresante para el usuario
- Configuración de alarmas personalizadas según preferencias individuales
- Utilización de vibración y sonidos progresivos para personas con sueño profundo
- Integración con entornos domésticos inteligentes para crear rutinas matutinas completas
- Adaptación a diferentes necesidades de sueño en distintos días de la semana

2. Aspectos básicos del desarrollo

Descripción del ecosistema móvil seleccionado

Se ha seleccionado el ecosistema Android para desarrollo nativo, aprovechando el amplio acceso a APIs de bajo nivel y funcionalidades específicas del sistema. Android proporciona APIs robustas para:

- Gestión de alarmas en segundo plano (AlarmManager)
- Control de vibración con patrones personalizables
- Reproducción de audio con control preciso
- Interfaces con hardware del dispositivo
- Conectividad con dispositivos externos (Bluetooth, WiFi)
- Justificación de desarrollo nativo sobre otras alternativas

La elección de desarrollo nativo Android sobre alternativas como Flutter se justifica por:

- Acceso a APIs de sistema de bajo nivel: Las alarmas requieren permisos especiales y funcionalidad en segundo plano que es más fiable y eficiente con código nativo.
- Integración con hardware: El control preciso de la vibración, sonido y conectividad con dispositivos de iluminación externos requiere acceso directo a las APIs del sistema.

- Rendimiento optimizado: Al tratarse de una aplicación que debe funcionar de manera confiable incluso cuando el dispositivo está en modo de ahorro de energía.
- Funcionalidades avanzadas: La integración con dispositivos de iluminación inteligente requiere APIs específicas de Android para Bluetooth/WiFi que son más sólidas en implementación nativa.
- Experiencia de usuario fluida: La interfaz de usuario puede aprovechar las animaciones y transiciones nativas de Android, ofreciendo una experiencia más integrada con el sistema.

Especificación de niveles de API objetivo

- API mínima: Android 8.0 (API level 26, Oreo)
- API objetivo: Android 14 (API level 34)

Esta selección permite cubrir aproximadamente el 94% de los dispositivos Android activos mientras se aprovechan las características modernas para una mejor experiencia de usuario.

Emuladores y entornos de desarrollo a utilizar

IDE principal: Android Studio (última versión estable)

Emuladores:

- Pixel 6 con Android 12 (API 31)
- Pixel 4 con Android 11 (API 30)
- Samsung Galaxy S10 con Android 10 (API 29)
- Dispositivo de gama baja con Android 8 (API 26)

Entorno de pruebas físicas: Dispositivos reales representativos de diferentes fabricantes (Samsung, Xiaomi, Google)

Herramientas adicionales:

- Firebase Test Lab para pruebas automatizadas
- Android Debug Bridge (ADB) para depuración avanzada

3. Especificaciones Técnicas

Versión mínima y objetivo de Android

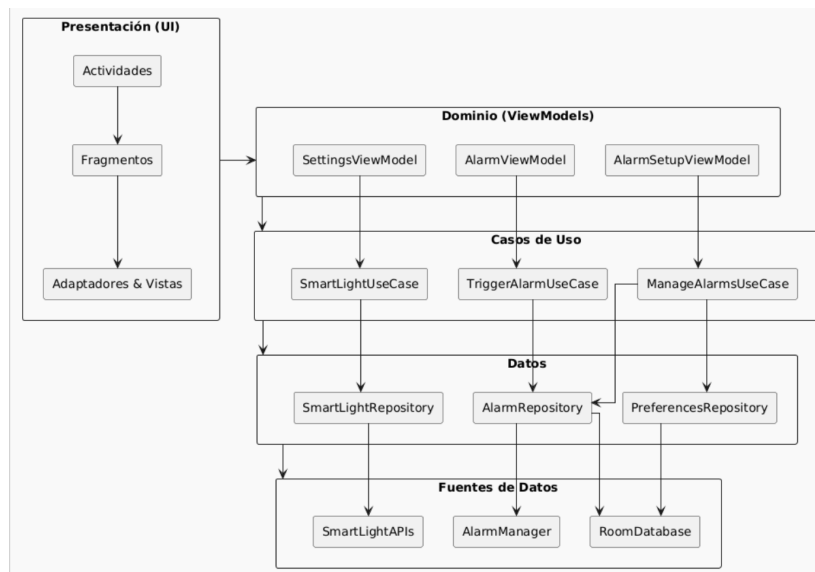
- Versión mínima: Android 8.0 (API level 26, Oreo)
- Versión objetivo: Android 14 (API level 34)

Arquitectura propuesta

Se implementará la arquitectura Clean Architecture con el patrón MVVM (Model-View-ViewModel) para garantizar:

- Separación de responsabilidades: Clara división entre lógica de presentación, lógica de negocio y acceso a datos
- Facilidad de pruebas: Cada componente puede ser probado de manera aislada
- Escalabilidad: Estructura que permite añadir nuevas características sin modificar el código existente
- Mantenibilidad: Organización del código en capas bien definidas

Diagrama de componentes principales



Tecnologías y bibliotecas a utilizar

Componentes de UI

- Jetpack Compose: Para la construcción de interfaces modernas y declarativas
- Material Components: Para seguir las directrices de Material Design
- ConstraintLayout: Para layouts complejos en pantallas específicas
- Navigation Component: Para la gestión de navegación entre pantallas
- View Binding: Para interacción con vistas en componentes XML

Almacenamiento de datos

- Room Database: Para persistencia estructurada de alarmas y configuraciones
- DataStore: Para preferencias de usuario y configuraciones simples
- WorkManager: Para programar y gestionar tareas en segundo plano

APIs y servicios externos

- AlarmManager API: Para programar alarmas precisas
- Notification API: Para mostrar notificaciones de alarmas
- Bluetooth/WiFi APIs: Para la comunicación con dispositivos de iluminación
- Firebase Analytics: Para seguimiento de uso y comportamiento
- Firebase Crashlytics: Para monitoreo de errores

Características específicas de hardware

- Vibrator Service: Para control personalizado de patrones de vibración
- MediaPlayer: Para reproducción de sonidos de alarma
- Bluetooth Low Energy: Para conectar con dispositivos de iluminación inteligente
- Sensores de luz: Para detectar nivel de luz ambiental (opcional)
- Sensores de movimiento: Para detección de movimiento al despertar (opcional)

4. Estructura y componentes de interfaz de usuario

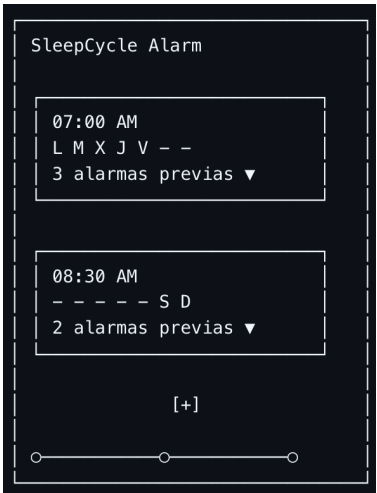
Descripción de la estructura de la aplicación

La aplicación se organizará en cuatro secciones principales:

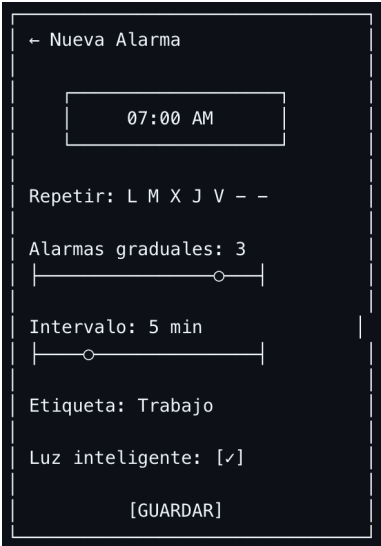
- 1. Pantalla principal: Muestra las alarmas configuradas y permite acceso rápido para crear nuevas.
- 2. Configuración de alarma: Interfaz para crear y editar alarmas con todas las opciones disponibles.
- 3. Configuración de gradualidad: Permite personalizar los parámetros de las alarmas progresivas.
- 4. Ajustes generales: Configuraciones globales de la aplicación y preferencias del usuario.

Wireframes de las pantallas principales

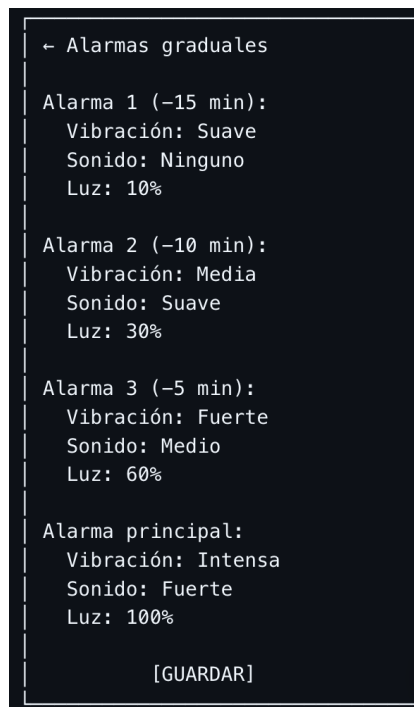
Pantalla principal:



Configuración de alarma:



Configuración de gradualidad



```
← Alarmas graduales

Alarma 1 (-15 min):
  Vibración: Suave
  Sonido: Ninguno
  Luz: 10%

Alarma 2 (-10 min):
  Vibración: Media
  Sonido: Suave
  Luz: 30%

Alarma 3 (-5 min):
  Vibración: Fuerte
  Sonido: Medio
  Luz: 60%

Alarma principal:
  Vibración: Intensa
  Sonido: Fuerte
  Luz: 100%

[GUARDAR]
```

Especificación de layouts y componentes visuales a utilizar

Componentes principales:

- TopAppBar: Para la navegación y título de sección
- BottomNavigationBar: Para acceso rápido a secciones principales
- FloatingActionButton: Para añadir nuevas alarmas
- TimePicker: Componente personalizado para selección de hora
- Sliders: Para ajustar cantidad e intervalos de alarmas
- Switches: Para activar/desactivar funciones
- Cards: Para mostrar alarmas configuradas
- ChipGroup: Para selección de días de la semana

Layouts principales:

- ConstraintLayout: Para pantallas con elementos complejos
- LazyColumn: Para listas de alarmas configuradas
- Column/Row: Para organización básica de elementos

Intents principales:

1. Intent para AlarmReceiver: Broadcast para activar alarmas
2. Intent para ActiveAlarmActivity: Intent explícito que muestra la pantalla de alarma activa
3. Intent para pantalla de ajustes del sistema: Para gestión de permisos
4. Intent para selección de archivos: Para selección de tonos personalizados

Implementación de material design y consideraciones de UX

- Tema dinámico: Adaptación a Material You en dispositivos compatibles
- Modo oscuro: Soporte completo para temas claro y oscuro
- Animaciones y transiciones: Animaciones fluidas entre pantallas y estados
- Accesibilidad: Soporte para TalkBack, tamaños de texto ajustables y etiquetas de contenido
- Retroalimentación táctil: Respuesta háptica sutil en interacciones importantes
- Espaciado generoso: Áreas de toque amplias para facilidad de uso
- Gestos intuitivos: Deslizamiento para descartar alarmas o realizar acciones comunes

5. Almacenamiento y manejo de datos

Descripción de la estrategia de persistencia de datos

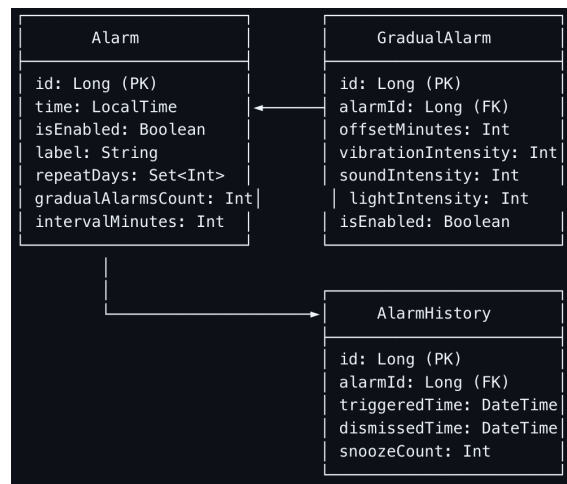
La aplicación utilizará un enfoque de persistencia en capas:

1. Room Database: Almacenamiento estructurado para:
 - Configuración de alarmas (hora, días, gradualidad)
 - Historial de alarmas disparadas
 - Configuraciones de sonidos personalizados
 - DataStore: Para preferencias del usuario:
2. Configuraciones globales de la aplicación
 - Preferencias de tema y comportamiento
 - Últimos dispositivos conectados
3. Almacenamiento en caché: Para recursos frecuentemente utilizados:
 - Sonidos de alarma personalizados
 - Datos de rendimiento y uso

Especificación de permisos requeridos

- WAKE_LOCK: Para activar la pantalla durante las alarmas
- RECEIVE_BOOT_COMPLETED: Para restaurar alarmas después de reinicio
- VIBRATE: Para control de vibración
- INTERNET: Para conectividad con servicios (opcional)
- BLUETOOTH y BLUETOOTH_ADMIN: Para conectar con dispositivos de luz
- SCHEDULE_EXACT_ALARM: Para programación precisa de alarmas
- POST_NOTIFICATIONS: Para mostrar notificaciones de alarma
- FOREGROUND_SERVICE: Para mantener el servicio de alarma activo

Diseño de la base de datos



Implementación de preferencias compartidas, archivos y/o bases de datos

1. DataStore para preferencias:

```
data class AppPreferences(  
    val isDarkMode: Boolean = false,  
    val isVibrationEnabled: Boolean = true,  
    val defaultAlarmSound: String = "default",  
    val defaultGradualCount: Int = 3,  
    val defaultIntervalMinutes: Int = 5,  
    val isSmartLightEnabled: Boolean = false,  
    val lastConnectedDeviceId: String? = null  
)
```

2. Room para datos estructurados:

```
@Entity(tableName = "alarms")  
data class AlarmEntity(  
    @PrimaryKey(autoGenerate = true) val id: Long = 0,  
    val timeInMinutes: Int,  
    val isEnabled: Boolean,  
    val label: String,  
    val repeatDays: String, // Stored as JSON array  
    val gradualAlarmsCount: Int,  
    val intervalMinutes: Int,  
    val useSmartLight: Boolean  
)
```

3. Almacenamiento de archivos:

- Carpeta privada para sonidos de alarma personalizados
- Caché para recursos temporales

Uso de proveedores de contenido

- AlarmClock Provider: Para integración con aplicación de reloj de sistema
- ContentProvider personalizado: Para posible expansión futura con widgets
- FileProvider: Para compartir configuraciones de alarma entre dispositivos

6. Servicios y comunicaciones

Descripción de los sensores del dispositivo a utilizar

1. Sensores de luz ambiental:
 - Detección del nivel de luz en el ambiente
 - Ajuste adaptativo del brillo de la pantalla durante las alarmas
2. Sensores de movimiento (acelerómetro):
 - Detección de movimiento para función "voltear para silenciar"
 - Análisis básico de movimiento para detectar si el usuario se levantó
3. Sensores de proximidad:
 - Detección de gestos para silenciar o posponer alarmas
 - Optimización de comportamiento cuando el dispositivo está en bolsillo/mesa

Especificación de servicios web a consumir

1. API de clima (opcional):
 - Integración con servicio meteorológico para ajustar alarmas según condiciones
 - Sugerencia de hora de despertar basada en condiciones meteorológicas
2. APIs de dispositivos inteligentes:
 - Phillips Hue API para control de iluminación
 - LIFX API para dispositivos de iluminación alternativa
 - API genérica para dispositivos compatibles con IFTTT

Integración con servicios en la nube

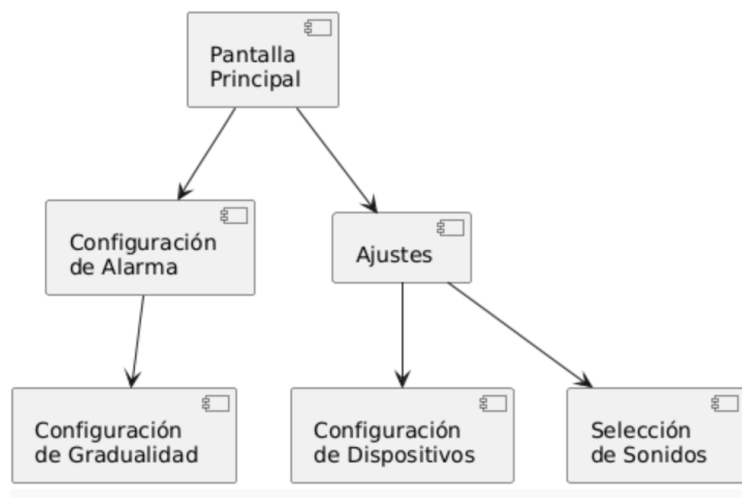
1. Firebase:
 - Firebase Analytics: Para análisis de uso y comportamiento
 - Firebase Crashlytics: Para reporte y análisis de errores
 - Firebase Remote Config: Para configuración remota de parámetros
2. Google Fit (opcional):
 - Integración para sugerir mejores tiempos de despertar basados en patrones de sueño
 - Almacenamiento de métricas de calidad de sueño y tiempo de despertar

Implementación de notificaciones push

1. Notificaciones locales:
 - Notificaciones persistentes durante alarmas activas
 - Recordatorios de alarmas programadas
 - Categorización por canales de notificación según Android guidelines
2. Notificaciones expandibles:
 - Controles de alarma directamente en la notificación
 - Opción para posponer o desactivar desde la notificación
 - Información contextual (clima, próximas actividades)

7. Diseño de la Aplicación

Flujo de navegación entre pantallas



Paleta de colores y elementos de identidad visual

Paleta de colores principal:

- Primario: #3F51B5 (Indigo)
- Secundario: #FF9800 (Orange)
- Fondo (claro): #F5F5F5 (Off-white)
- Fondo (oscuro): #121212 (Dark grey)
- Acentos funcionales:
 - Alarma activa: #4CAF50 (Green)
 - Alarma inactiva: #9E9E9E (Grey)
 - Alerta/Importante: #F44336 (Red)

Elementos de identidad visual:

- Iconografía: Líneas simples y redondeadas que sugieren movimiento fluido
- Tipografía: Familia Roboto para consistencia con Material Design
- Ilustraciones: Estilo minimalista con animaciones sutiles
- Logo: Interpretación estilizada de un ciclo de sueño con transición gradual de color

Consideraciones de UX/UI y accesibilidad

1. Accesibilidad:
 - Contraste de color que cumple WCAG AAA
 - Soporte completo para lectores de pantalla
 - Diseño que mantiene funcionalidad con fuentes grandes
 - Etiquetas de contenido descriptivas
 - Alternativas táctiles para interacciones sonoras
2. Usabilidad:
 - Operaciones críticas (como desactivar alarmas) con área de toque amplia
 - Retroalimentación clara para todas las acciones
 - Confirmación para acciones destructivas
 - Flujos de configuración simplificados con opciones avanzadas ocultas
 - Estado del sistema siempre visible (próxima alarma, estado de conexión)
3. Experiencia de usuario:

- Transiciones fluidas entre estados
- Animaciones sutiles que indican cambios de estado
- Micro-interacciones que proporcionan satisfacción al usuario
- Opciones de personalización para adaptarse a preferencias individuale

8. Funcionalidades

Lista detallada de las funcionalidades principales

1. Configuración de alarma principal:
 - Selección de hora de despertar objetivo
 - Configuración de días de repetición
 - Etiquetas personalizables para alarmas
2. Sistema de alarmas graduales:
 - Configuración de 1-10 alarmas previas a la principal
 - Intervalos configurables de 1-30 minutos entre alarmas
 - Aumento progresivo automático de intensidad (vibración, sonido)
3. Integración con dispositivos de iluminación inteligente:
 - Descubrimiento y emparejamiento de dispositivos compatibles
 - Control gradual de intensidad de luz sincronizado con alarmas
 - Perfil de color e intensidad personalizable
4. Modos de silenciar y posponer inteligentes:
 - Detección de movimiento para confirmar despertar
 - Opciones de posponer con duración ajustable
 - Modos "voltear para silenciar" y "agitar para posponer"
5. Análisis básico de patrones de sueño:
 - Registro de tiempo de respuesta a alarmas
 - Historial de posponer y desactivar
 - Recomendaciones básicas para optimizar rutinas

Lista de funcionalidades secundarias o para futuras versiones

1. Integración con servicios de salud y bienestar:
 - Conexión con Google Fit y Samsung Health
 - Análisis avanzado de ciclos de sueño con wearables
2. Modos contextuales inteligentes:
 - Ajuste automático basado en calendario y eventos
 - Alarmas adaptativas según tráfico o clima
3. Personalización avanzada de sonidos:
 - Biblioteca de sonidos naturales gradualmente intensificados
 - Integración con servicios de streaming de música
 - Creación de mezclas personalizadas de sonido
4. Rutinas matutinas completas:
 - Secuencias programables post-despertar
 - Integración con asistentes de voz para briefing matutino
 - Control de dispositivos adicionales (cafetera, termostato)
5. Función social y gamificación:
 - Desafíos para mantener horarios regulares
 - Competición amistosa con amigos
 - Recompensas por consistencia

Descripción de casos de uso principales

Caso de uso 1: Configuración de alarma básica

1. Usuario abre la aplicación
2. Presiona el botón "+" para crear nueva alarma
3. Selecciona hora objetivo (7:00 AM)
4. Selecciona días (L-V)
5. Sistema configura automáticamente 3 alarmas graduales (-15, -10, -5 min)
6. Usuario guarda la configuración
7. Sistema programa las alarmas en el sistema

Caso de uso 2: Experiencia de despertar gradual

1. Sistema activa primera alarma (vibración suave) a las 6:45 AM
2. Si no hay respuesta, sistema activa segunda alarma (vibración media + sonido suave) a las 6:50 AM
3. Si no hay respuesta, sistema activa tercera alarma (vibración fuerte + sonido medio) a las 6:55 AM
4. Sistema activa alarma principal (vibración intensa + sonido fuerte) a las 7:00 AM
5. Usuario interactúa para silenciar o posponer
6. Sistema registra datos de respuesta

Caso de uso 3: Configuración de integración con luces inteligentes

1. Usuario accede a configuración de alarma
2. Activa opción "Usar luz inteligente"
3. Sistema muestra dispositivos disponibles para emparejar
4. Usuario selecciona dispositivo(s)
5. Usuario configura intensidad de luz para cada nivel de alarma
6. Sistema prueba conexión y confirma configuración
7. Las alarmas futuras incluirán control de iluminación

Caso de uso 4: Personalización avanzada de gradualidad

1. Usuario accede a una alarma existente
2. Selecciona "Configuración avanzada"
3. Ajusta número de alarmas graduales a 5
4. Configura intervalos personalizados (20, 15, 10, 5 min)
5. Personaliza intensidad de cada nivel
6. Guarda configuración personalizada
7. Sistema aplica el nuevo patrón a futuras activaciones
8. Requisitos funcionales y no funcionales

Requisitos Funcionales:

1. El sistema deberá permitir configurar alarmas con hora, días de repetición y etiqueta.
2. El sistema deberá soportar entre 1 y 10 alarmas graduales previas a la alarma principal.
3. El sistema deberá permitir configurar intervalos entre alarmas graduales de 1 a 30 minutos.
4. El sistema deberá incrementar automáticamente la intensidad de vibración y sonido en alarmas sucesivas.
5. El sistema deberá comunicarse con dispositivos de iluminación compatibles cuando esté configurado.
6. El sistema deberá soportar modos de silenciar y posponer mediante gestos físicos.
7. El sistema deberá mantener un registro histórico de interacciones con alarmas.
8. El sistema deberá restaurar las alarmas configuradas tras el reinicio del dispositivo.
9. El sistema deberá mostrar notificaciones persistentes durante alarmas activas.
10. El sistema deberá soportar personalización de tonos de alarma.

Requisitos No Funcionales:

1. Confiabilidad: El sistema debe garantizar activación de alarmas con precisión de ± 1 segundo.
2. Rendimiento: La aplicación debe iniciar en menos de 2 segundos desde el toque del icono.
3. Eficiencia energética: El consumo de batería en segundo plano no debe exceder 1% por hora.
4. Usabilidad: Configurar una nueva alarma no debe requerir más de 3 toques en el flujo básico.
5. Escalabilidad: El sistema debe soportar hasta 50 alarmas configuradas simultáneamente.
6. Compatibilidad: La aplicación debe funcionar correctamente en el 94% de dispositivos Android activos.
7. Mantenibilidad: El código debe seguir principios SOLID y tener cobertura de pruebas $>80\%$.
8. Seguridad: Los datos del usuario deben almacenarse localmente y de forma encriptada.
9. Accesibilidad: La aplicación debe cumplir con las guidelines WCAG 2.1 nivel AA.
10. Internacionalización: Soporte para español, inglés, francés, alemán y japonés.

9. Plan de implementación

(Por definir)