**“Año de la recuperación y consolidación de la economía peruana”**

**FACULTAD DE INGENIERÍA**

**Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media**

**TEMA:**

**TP2.E5 Diseño de arquitectura**

**Curso:**

**Taller de Proyectos 2**

**NRC: 16810**

**Docente:**

**Americo Estrada Sanchez**

**Alumno:**

**● CHRISTIAN CASTRO USTO**

**CUSCO - PERÚ**

**2025**

Diseño de Arquitectura del Sistema - TaskManager

**Fecha:** 15/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

Este documento describe de manera detallada la arquitectura del sistema para la aplicación móvil **SLEEPAPP**, cuyo propósito es ayudar a los usuarios a establecer y mantener una rutina de sueño saludable en función de sus actividades diarias. Se define la estructura general del sistema, los componentes que lo conforman, sus interacciones y las tecnologías empleadas.

# 2. Visón General de la Arquitectura

## 2.1. Estilo Arquitectónico

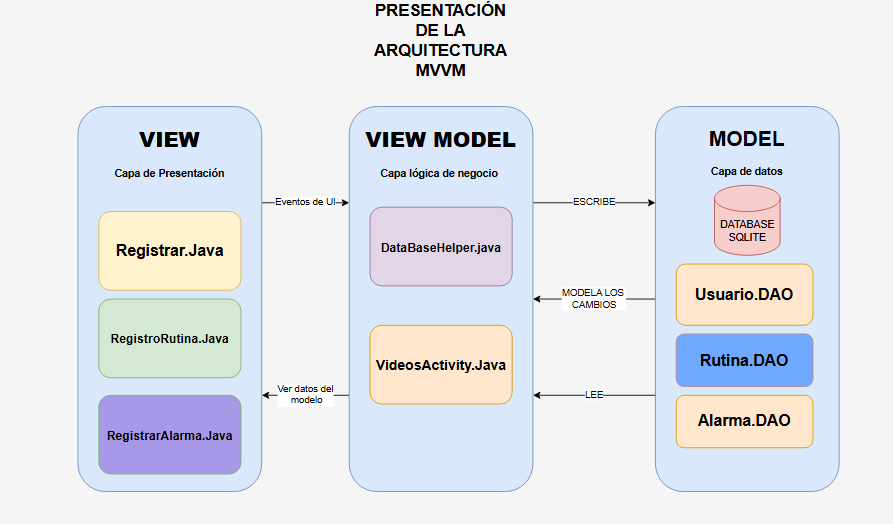
El sistema adopta una arquitectura en capas, basada en el patrón cliente-servidor, y se organiza en tres capas principales:

* **Capa de presentación (Frontend móvil)**  
  Interfaz de usuario desarrollada en **Android (Java)**, que permite el registro de actividades, visualización de patrones de sueño, configuración de alarmas personalizadas y reproducción de videos de relajación.
* **Capa de lógica de negocio**  
  Se encarga del procesamiento de datos ingresados por el usuario, cálculos de horas de descanso necesarias y generación de recomendaciones personalizadas. Esta capa está integrada dentro del cliente móvil y sincronizada con la base de datos.
* **Capa de datos (Backend)**  
  Implementada mediante **SQLITE**, que permite el almacenamiento de datos en local, autenticación de usuarios, sincronización en tiempo real y persistencia de la información.

## 2.2. Diagrama General de Arquitectura:

Diagrama

El contenido generado por IA puede ser incorrecto.



# 3. Descripción de Componentes Principales

## 3.1. VIEW (Presentación)

La capa View es la encargada de interactuar directamente con el usuario. Se encarga de renderizar pantallas visuales, recibir entradas del usuario y mostrar respuestas o resultados. No gestiona lógica de negocio ni accede a datos directamente; su responsabilidad es meramente de presentación y captura de eventos.

• **Framework / Tecnología:** Java (Android Studio)  
• **Estilo:** XML (Material Design)  
• **Responsabilidades:**

* Renderizar una interfaz gráfica intuitiva y amigable para el usuario
* Capturar entradas del usuario a través de formularios, botones y controles personalizados
* Mostrar mensajes de éxito, advertencias o errores relacionados con las actividades del usuario
* Navegar entre pantallas utilizando Intents
* Presentar datos almacenados (por ejemplo, hábitos, horas de sueño, rutinas) de forma visual y comprensible mediante gráficos o listas

REGISTRAR.JAVA

Responsable de gestionar el login y el registro de usuarios de forma local.

* Permite ingresar un correo y una contraseña.
* Valida los datos con el modelo de usuarios a través del UsuarioDAO.
* Muestra mensajes si el registro fue exitoso o si hubo errores en el inicio de sesión.

REGISTRORUTINA.JAVA

Encargada de la creación y modificación de la rutina diaria del usuario.

* Captura datos sobre las actividades diarias del usuario (horas de estudio, ejercicio, etc.).
* Permite guardar, actualizar o eliminar rutinas.
* Muestra al usuario la rutina registrada mediante listas o formularios prellenados.

REGISTRARALARMA.JAVA

Permite al usuario configurar alarmas personalizadas según su rutina de sueño.

* Ofrece controles para seleccionar la hora, días activos y nombre de la alarma.
* Guarda y actualiza alarmas usando el AlarmaDAO.
* Permite activar/desactivar alarmas y muestra mensajes relacionados a su estado.

## 3.2. VIEW MODEL (Lógica de negocio)

## La capa ViewModel actúa como un puente entre la Vista (View) y el Modelo (Model). En la arquitectura MVVM, esta capa tiene como objetivo manejar la lógica de presentación y procesamiento de datos sin necesidad de que la Vista acceda directamente al modelo de datos (SQLite).

• **Lenguaje:** Java (Android)  
• **Responsabilidades:**

* Controlar y coordinar el flujo de datos entre la interfaz (View) y la base de datos (Model)
* Validar entradas del usuario antes de guardar o procesar datos
* Aplicar reglas de negocio, como calcular las horas de sueño necesarias según la rutina del usuario
* Determinar cuándo mostrar alertas, recordatorios o recomendaciones personalizadas
* Gestionar la lógica de navegación entre pantallas (Activities o Fragments)
* Controlar el estado de los datos de la sesión local del usuario

DATABASEHELPER.JAVA

Clase central para la **gestión de la base de datos SQLite**, funciona como un gestor de acceso a los DAOs (UsuarioDAO, RutinaDAO, AlarmaDAO).

* Proporciona los métodos necesarios para insertar, actualizar, eliminar y consultar datos de usuarios, rutinas y alarmas.
* Sirve como intermediario para que las actividades no interactúen directamente con la base de datos.
* Encapsula la lógica SQL y se comunica con los DAOs, asegurando la separación de responsabilidades.

VIDEOSACTIVITY.JAVA

Se encuentra en el paquete MV, esta clase actúa como parte de la lógica de presentación al encargarse de cargar y controlar la reproducción de videos de relajación.

* Su rol es ofrecer contenido multimedia útil para mejorar el descanso del usuario.
* Accede a recursos locales (almacenados en la app) o a URLs externas para reproducir videos.
* No interactúa con la base de datos directamente, pero se encarga de la lógica necesaria para adaptar este contenido al usuario.

## 3.3 MODEL (Capa de datos)

La capa de datos es el núcleo de la aplicación, ya que se encarga del almacenamiento, recuperación y manipulación de la información. En la arquitectura MVVM, esta capa se encuentra completamente separada de la interfaz de usuario, garantizando una mejor organización, mantenibilidad y escalabilidad del sistema.

* Define la estructura de las entidades o modelos de datos (usuarios, rutinas, alarmas).
* Implementa la lógica para acceder y modificar la base de datos.
* Permite que el ViewModel recupere o actualice información sin conocer los detalles de la base de datos subyacente.

**DAOs (Data Access Object)**

Los DAOs son interfaces o clases que permiten separar el acceso a datos del resto de la lógica. Cada DAO representa una entidad y contiene las operaciones específicas que se pueden realizar sobre ella.

**🔹 UsuarioDAO**

* Se encarga de gestionar los datos del usuario: nombre, correo, contraseña, etc.
* Operaciones principales: insertar, validar inicio de sesión, actualizar datos del perfil.

**🔹 RutinaDAO**

* Administra la información de la rutina diaria del usuario: horas de estudio, ejercicio, sueño, actividades recreativas.
* Operaciones principales: crear rutina, actualizar rutina, consultar datos para análisis o gráficas.

**🔹 AlarmaDAO**

* Controla el registro y la modificación de alarmas personalizadas configuradas por el usuario.
* Operaciones principales: insertar nueva alarma, actualizar hora o estado (activo/inactivo), eliminar o consultar alarmas existentes.

## 3.3. Base de Datos (Persistencia)

* **Sistema Gestor**: SQLite (almacenamiento local en dispositivos Android)
* Esquema de Datos:
  + **Tabla tusuarios**:  
    dni, usuario, contraseña, nombre, apellido, correo, fechanacimiento
  + **Tabla trutina**:  
    edad, horasTrabajo, ejercicioMinutos, recreacionMinutos, cafeConsumo, tiempoPantalla, estrés
  + **Tabla talarma**:  
    id, horaDespertar, tonoAlarma, dniUsuario (clave foránea hacia tusuarios)
  + **Tabla tvideos**:  
    id\_video, titulo, video\_id

# 4. Integraciones Externas (Opcionales)

* **Servicios de Notificación**: Integración futura con Firebase Cloud Messaging (FCM) para enviar notificaciones push sobre alarmas o recomendaciones de descanso.
* **Autenticación Social**: Opcionalmente, se podrá integrar login con Google mediante Firebase Authentication.

# 5. Seguridad

* **Verificación de datos**: Validación manual de los datos insertados en formularios para evitar errores o datos malformados.Tokens JWT con expiración
* **Restricciones en SQLite**: Uso de claves primarias y foráneas para mantener integridad referencial.
* **Control de acceso local**: Solo usuarios con credenciales válidas pueden iniciar sesión en la app.
* **Sin almacenamiento en la nube**: Los datos están guardados localmente, lo que reduce riesgos de exposición remota.

# 6. Escalabilidad y Despliegue

Dado que la aplicación se ejecuta de forma local en dispositivos Android y utiliza SQLite como base de datos integrada, su arquitectura no contempla un entorno de despliegue en la nube.

* **Aplicación Android:** Se distribuye mediante la Play Store o instalación manual (APK)
* **Base de datos local (SQLite):** Funciona eficientemente en dispositivos individuales, pero no es escalable para múltiples usuarios concurrentes o sincronización en la nube

# 7. Conclusiones

La arquitectura MVVM propuesta para la SleepApp busca lograr un equilibrio entre simplicidad, robustez y escalabilidad. Esta estructura permite una clara separación de responsabilidades entre la interfaz de usuario, la lógica de negocio y los datos, lo que facilita el mantenimiento y la evolución del sistema. Además, se alinea con buenas prácticas del desarrollo de aplicaciones móviles modernas y permite una extensión progresiva del sistema conforme crezcan las necesidades del usuario.