**“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.E2 Especificación de Requisitos del Software**

**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.

Diagrama

Descripción generada automáticamente

# 3. Descripción de Componentes Principales

## 3.1. VIEW (Presentación)

• **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

## 3.2. VIEW MODEL (Lógica de negocio)

• **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

### 3.2.1. Módulos del Backend

## Registrar.java: Manejo del login y registro de usuarios (de forma local)

## RegistroRutina.java: Operaciones CRUD para registrar y actualizar la rutina diaria del usuario

## RegistrarAlarma.java: Control de alarmas personalizadas y notificaciones locales

## DatabaseHelper.java: Acceso y gestión de la base de datos SQLite (inserciones, actualizaciones, consultas)

## VideosActivity.java: Carga y reproducción de videos de relajación desde recursos locales o en línea

## 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.