**UNIVERSIDAD CONTINENTAL**

**FACULTAD DE INGENIERÍA **

**ESCUELA ACADÉMICA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**APP MÓVIL PARA MEDIR LAS HORAS DE SUEÑO EN BASE A LA RUTINA DEL USUARIO**

**PRESENTADO POR:**

**CHRISTIAN CASTRO USTO**

**CUSCO-PERÚ**

**2025**

Visión del Proyecto

**Fecha:** 19/03/2025

**Autor: CHRISTIAN CASTRO USTO**

**Versión:** 1.1

# Introducción

El descanso adecuado es un factor fundamental para la salud y el rendimiento diario de las personas. Sin embargo, en la actualidad, muchas personas tienen dificultades para establecer una rutina de sueño óptima debido a sus actividades diarias, lo que puede derivar en fatiga, estrés y problemas de concentración. Para abordar esta problemática, se desarrollará una aplicación móvil que permita calcular las horas de descanso necesarias para cada usuario según su rutina diaria.

La aplicación ofrecerá diversas funcionalidades diseñadas para mejorar la calidad del sueño. Entre ellas, se incluye la selección de una alarma personalizada, un menú donde los usuarios pueden registrar sus actividades diarias utilizando escalas de 1 a 10, un diagrama estadístico que muestra patrones de sueño y videos de relajación que facilitan el descanso. Con estas herramientas, la aplicación buscara proporcionar una guía personalizada para mejorar los hábitos de sueño de sus usuarios.

El problema principal que abordara este proyecto es la falta de conocimiento sobre la cantidad de horas de descanso que realmente necesita una persona en función de su estilo de vida. Si este problema no se resuelve, las personas continuarán enfrentando consecuencias negativas en su rendimiento académico, laboral y en su bienestar general. La aplicación contribuirá a reducir este impacto al ofrecer recomendaciones basadas en los datos ingresados por cada usuario, promoviendo así una mejor calidad de vida.

El propósito de esta aplicación es ayudar a las personas a establecer una rutina de descanso adecuada, facilitando la gestión de sus horarios y actividades diarias de manera equilibrada. Su importancia radica en la necesidad de concienciar sobre los beneficios de un sueño reparador y brindar una herramienta tecnológica que permita optimizar la planificación del descanso, lo que puede repercutir positivamente en la salud y el bienestar de sus usuarios.

Breve descripción del proyecto, su propósito y la importancia de su desarrollo.

# Problema a Resolver

El descanso adecuado es un factor clave para el rendimiento académico y la salud de los estudiantes universitarios. Sin embargo, en la actualidad, muchos de ellos enfrentan dificultades para establecer y mantener una rutina de sueño equilibrada debido a la carga académica, las responsabilidades personales, el estrés y el uso prolongado de dispositivos electrónicos. Estos factores contribuyen a la alteración del ciclo de sueño, lo que genera problemas como fatiga constante, dificultad para concentrarse, disminución del rendimiento en exámenes y trabajos, e incluso efectos negativos en la salud mental, como ansiedad y depresión.

Uno de los principales problemas radica en la falta de conciencia y conocimiento sobre la cantidad de horas de descanso que realmente necesita cada estudiante en función de su estilo de vida y actividades diarias. Muchos universitarios no tienen un control adecuado de sus hábitos de sueño ni herramientas que les permitan medir cómo sus rutinas afectan su descanso. Como resultado, suelen subestimar la importancia de dormir lo suficiente, optando por reducir sus horas de sueño para cumplir con responsabilidades académicas, sociales o personales. Esto provoca un ciclo negativo en el que la falta de descanso adecuado repercute en su desempeño académico, lo que a su vez genera mayor estrés y disminuye su calidad de vida.

Otro problema importante es la ausencia de herramientas personalizadas que permitan a los estudiantes evaluar su descanso de manera práctica y basada en datos reales. Aunque existen aplicaciones que monitorean el sueño, muchas de ellas no consideran la rutina específica de cada usuario, como el tiempo dedicado a estudiar, trabajar, realizar actividades recreativas o utilizar dispositivos electrónicos antes de dormir. Sin un enfoque personalizado, estas soluciones no logran ofrecer recomendaciones precisas ni estrategias efectivas para mejorar la calidad del descanso.

Si este problema no se aborda de manera adecuada, las consecuencias a largo plazo podrían ser graves. La falta de sueño crónica no solo afecta el rendimiento académico, sino que también incrementa el riesgo de problemas de salud como enfermedades cardiovasculares, trastornos metabólicos y deterioro cognitivo. Además, el agotamiento constante puede influir en la toma de decisiones, el estado emocional y la productividad de los estudiantes, limitando su capacidad para alcanzar sus objetivos académicos y personales.

Por lo tanto, es necesario desarrollar una solución tecnológica que ayude a los estudiantes a conocer con precisión cuántas horas necesitan descansar según su rutina diaria. A través de una aplicación móvil con funcionalidades especializadas, se puede brindar un soporte eficiente para que los estudiantes comprendan la importancia del sueño y ajusten sus hábitos de manera informada, promoviendo así un mejor equilibrio entre sus actividades y su bienestar general.

PROBLEMA GENERAL:

Los estudiantes universitarios no tienen conocimiento sobre la cantidad de horas de sueño que duermen ni cuentan con aplicaciones que permitan un seguimiento de sus horas descansadas.

PROBLEMAS ESPECIFICOS

Determinar la cantidad de horas de sueño de los estudiantes universitarios es complicado debido a la falta de conocimiento sobre sus horas de descanso.

Identificar cómo las actividades diarias afectan la calidad y cantidad del sueño es difícil sin aplicaciones adecuadas.

Tener un registro automático y personalizado de las horas de sueño es un desafío, ya que las aplicaciones actuales no ofrecen un seguimiento adaptado a la rutina de cada usuario.

# Objetivos del Proyecto

La aplicación busca ayudar a los estudiantes universitarios a mejorar su descanso, calculando las horas de sueño necesarias según su rutina diaria. Además, permitirá registrar actividades, visualizar su impacto en el descanso y acceder a contenido de relajación, con el objetivo de optimizar el rendimiento académico y el bienestar general.

OBJETIVO GENERAL:

Desarrollar una aplicación móvil que permita a los estudiantes universitarios conocer cuántas horas necesitan descansar según su rutina diaria para tener conocimiento de sus horas de sueño.

OBJETIVOS ESPECIFICOS:

Determinar la cantidad de horas de sueño de los estudiantes universitarios mediante una herramienta que registre y analice su rutina diaria.

Identificar la relación entre las actividades diarias y la calidad del sueño a través de un sistema que evalúe las cantidades de horas de descanso que necesite el usuario.

Desarrollar una aplicación que permita un registro automático y personalizado de las horas de sueño, adaptándose a la rutina de cada usuario para mejorar su seguimiento y control.

# Público Objetivo (Usuarios Finales)

Define quién utilizará el software y qué beneficios obtendrá.

## Usuarios principales:

* La aplicación está dirigida a estudiantes universitarios, brindándoles una herramienta que les permite calcular y gestionar sus horas de sueño de acuerdo con sus actividades diarias. A través de sus diversas funcionalidades, busca facilitar el monitoreo del descanso y fomentar hábitos más saludables en su rutina.

## Beneficios esperados:

* Los estudiantes podrán calcular cuántas horas necesitan dormir según sus actividades diarias.
* La aplicación ayudará a organizar mejor su rutina para evitar la falta de sueño.
* Se proporcionarán videos y recursos para facilitar el descanso y mejorar la calidad del sueño.

# Funcionalidades Principales

Lista de características clave de la primera versión del software.

## Funcionalidades esenciales:

* **Registro de cuenta:** Permite a los usuarios crear y gestionar su perfil en la aplicación.
* **Inserción de rutina:** Los estudiantes pueden registrar sus actividades diarias usando escalas de 1 a 10 para calcular su impacto en el descanso.
* **Registro de alarma:** Registro de alarmas para calcular las horas necesarias a descansar.
* **Registro de alarmas personalizadas:** Configuración de alarmas personalizadas según el tiempo de descanso recomendado.
* **Inserción de videos de relajación:** Acceso a contenido multimedia diseñado para mejorar la calidad del sueño.

## Funcionalidades futuras (Opcionales):

* **Diagrama de horas de sueño:** Visualización de las horas dormidas en la semana para un mejor análisis del descanso.
* **Modo nocturno:** Interfaz con colores oscuros para reducir la fatiga visual antes de dormir.
* **Historial de rutinas:** Permitir a los usuarios revisar sus rutinas pasadas para analizar patrones de descanso.

# Requisitos Técnicos

Define el stack tecnológico del proyecto.

## Lenguajes y Frameworks:

* **Lenguaje de programación:** Java (para el desarrollo de la aplicación en Android Studio).
* **Entorno de desarrollo:** Android Studio.
* **Base de datos:** SQLite (para el almacenamiento local de la información del usuario).
* **Diseño de interfaz:** XML (para la estructura y diseño de la interfaz de usuario).

## Compatibilidad:

* **Sistema operativo:** La aplicación es compatible con dispositivos Android, ya que está desarrollada en Android Studio con Java.
* **Versión de Android:** Se recomienda el uso en dispositivos con Android 6.0 (Marshmallow) o superior para un mejor rendimiento y compatibilidad.
* **Requisitos de hardware:** Funciona en smartphones con al menos 2 GB de RAM y 100 MB de almacenamiento disponible.
* **Modo offline:** La aplicación puede utilizarse sin conexión a Internet, ya que almacena los datos en SQLite.
* **Posible compatibilidad futura:** Se evaluará la integración con dispositivos iOS y la sincronización con la nube mediante Firebase.

# Riesgos y Limitaciones

Posibles problemas que pueden surgir durante el desarrollo.

## Riesgos:

* **Errores en la implementación del cálculo de horas de sueño:** Puede haber dificultades en la precisión del algoritmo que determina el tiempo de descanso ideal.
* **Problemas con la gestión de la base de datos:** SQLite es una base de datos local, por lo que el manejo de grandes volúmenes de datos podría afectar el rendimiento de la app.
* **Compatibilidad con dispositivos antiguos:** Algunos smartphones con versiones antiguas de Android podrían no soportar todas las funcionalidades correctamente.
* **Optimización del rendimiento:** La aplicación debe estar bien optimizada para evitar consumo excesivo de batería y recursos del dispositivo.
* **Posibles fallos en la interfaz de usuario:** Un diseño ineficiente podría afectar la experiencia del usuario y dificultar la navegación.

## Limitaciones:

* **No cuenta con sincronización en la nube:** Actualmente, los datos se almacenan solo en el dispositivo mediante SQLite, lo que impide el acceso desde múltiples dispositivos.
* **Dependencia del usuario para el ingreso de datos:** La precisión de los cálculos dependerá de que el usuario ingrese correctamente su rutina diaria.
* **Falta de análisis avanzado del sueño:** No se incluyen sensores ni integración con dispositivos de monitoreo del sueño.
* **Solo disponible en Android:** No existe una versión para iOS, lo que limita su alcance a usuarios de otros sistemas operativos.

# Alcance del Proyecto

Define qué incluirá y qué quedará fuera en la primera versión.

## Lo que incluirá:

* **Inserción de rutina:** Los usuarios podrán registrar sus actividades diarias y evaluar su impacto en el descanso mediante escalas de 1 a 10.
* **Registro de cuenta:** Creación y gestión de perfiles para personalizar la experiencia en la aplicación.
* **Registro de alarma:** Configuración de alarmas basadas en el cálculo del tiempo de descanso recomendado.

## Lo que NO incluirá (por ahora):

* **Diagrama de horas de sueño:** No se mostrará una representación gráfica del tiempo dormido semanalmente.
* Inserción de videos de relajación: No se integrarán recursos multimedia para mejorar la calidad del sueño en la primera versión.
* **Personalización de tonos de alarma:** Se utilizarán sonidos predeterminados sin opción de personalización.
* **Cuadros estadísticos (barras o pastel):** No se incluirán gráficos avanzados para visualizar patrones de sueño.

Especificación de Requisitos del Software (SRS)

**Fecha:** 26/03/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

## 1.1 Propósito

Este documento define los requisitos para el desarrollo de una aplicación móvil llamada **SleepApp**, diseñada para el seguimiento y mejora de los hábitos de sueño de los estudiantes universitarios. Su propósito es ayudar a los usuarios a registrar, analizar y optimizar sus horas de descanso en función de sus actividades diarias, permitiéndoles adoptar hábitos más saludables.

## 1.2 Alcance

**SleepApp** será una aplicación móvil accesible desde dispositivos Android , permitiendo el registro, análisis y seguimiento de las horas de sueño de los estudiantes universitarios. La aplicación permitirá a los usuarios ingresar sus actividades diarias, establecer alarmas para hacer un seguimiento de horas de sueño, visualizar estadísticas sobre su calidad de sueño.

## 1.3 Definiciones, Acrónimos y Abreviaturas

* **UI (User Interface):** Interfaz de Usuario. Es la parte visual de la aplicación, actualmente implementada con un menú principal y algunas pantallas básicas.
* **SQLite:** Base de datos ligera utilizada en aplicaciones móviles para almacenamiento local. En futuras versiones, podría emplearse para gestionar datos sin conexión en la aplicación.
* **Android Studio:** Entorno de desarrollo integrado (IDE) oficial para crear aplicaciones Android. Es la herramienta utilizada para programar y probar la aplicación móvil.
* **RF:** Requisito funcional. Se centran en las funcionalidades y servicios que la aplicación debe proporcionar para cumplir con su propósito.
* **RNF:** Requisitos no funcionales. Se refieren a aspectos como rendimiento, usabilidad, compatibilidad y seguridad, pero no afectan directamente la funcionalidad del software.

## 1.4 Referencias

* IEEE 29148-2018 - Systems and Software Engineering – Life Cycle Processes – Requirements Engineering ([IEEE SA - IEEE/ISO/IEC 29148-2018](https://standards.ieee.org/ieee/29148/6937/)).
* Documentación oficial de ReactJS y Node.js.
* Documentación oficial de SQLite ([SQLite Documentation](https://www.sqlite.org/docs.html?utm_source=chatgpt.com)).
* Documentación oficial de Android y Java.( [Desarrolla para Android  |  Android Developers](https://developer.android.com/develop?hl=es-419)).
* ¿Cuántas horas de sueño son suficientes? - Mayo Clinic ([¿Cuántas horas de sueño son suficientes? - Mayo Clinic](https://www.mayoclinic.org/es/healthy-lifestyle/adult-health/expert-answers/how-many-hours-of-sleep-are-enough/faq-20057898?utm_source=chatgpt.com)).

## 1.5 Descripción General

El objetivo de este proyecto es desarrollar una aplicación móvil eficiente y fácil de usar para el seguimiento y mejora de la calidad del sueño en estudiantes universitarios. Se busca proporcionar una plataforma intuitiva que ayude a los usuarios a registrar sus horas de rutina diaria y recibir recomendaciones personalizadas para optimizar su descanso.

Con esta aplicación, los usuarios podrán monitorear sus hábitos de sueño en función de sus actividades diarias, establecer alarmas inteligentes y acceder a estadísticas sobre su descanso, permitiéndoles tomar decisiones informadas para mejorar su bienestar.

Este documento está estructurado en diferentes secciones que cubren la descripción general, requisitos funcionales y no funcionales, así como los riesgos y limitaciones del sistema.

# 2. Descripción General del Producto

## 2.1 Perspectiva del Producto

La aplicación de seguimiento del sueño será una aplicación móvil desarrollada en Android Studio, utilizando SQLite como base de datos local para almacenar los registros de sueño y las rutinas de los usuarios. La arquitectura estará basada en principios modernos de desarrollo móvil, garantizando una experiencia fluida y eficiente.

La interfaz será intuitiva y diseñada para facilitar la entrada y visualización de datos, permitiendo a los estudiantes registrar sus horas de descanso, establecer alarmas y acceder a estadísticas personalizadas sobre la calidad de su sueño. Además, se implementarán algoritmos de análisis para proporcionar recomendaciones basadas en los hábitos del usuario.

## 2.2 Funcionalidades Principales

* **Inserción de rutina diaria:** Permite registrar actividades diarias como horas de estudio, recreación, trabajo y ejercicio para analizar su impacto en el sueño.
* **Ingreso de alarma personalizada:** Los usuarios pueden configurar alarmas adaptadas a su rutina de descanso.
* **Menú principal intuitivo:** Facilita la navegación entre las diferentes opciones de la aplicación, permitiendo un acceso rápido a las funcionalidades principales.
* **Seguimiento estadístico del sueño:** Generación de gráficos y reportes sobre las horas dormidas, identificando patrones y tendencias en la calidad del sueño.
* **Visualización de videos de relajación:** Proporciona contenido multimedia para ayudar a los usuarios a mejorar su calidad de sueño mediante técnicas de relajación.

## 2.3 Características de los Usuarios

* **Usuarios principales:** Estudiantes universitarios que llevan una rutina académica demandante y desean mejorar su calidad de sueño mediante el seguimiento de sus hábitos.
* **Beneficios esperados:**
  + **Mejor organización del sueño:** Gracias a la inserción de alarmas y seguimiento estadístico, los usuarios podrán establecer hábitos de sueño más regulares.
  + **Toma de decisiones basada en datos:** A través del análisis de sus patrones de sueño, los estudiantes podrán ajustar su rutina para obtener un mejor rendimiento académico y bienestar general.
  + **Conciencia sobre el descanso:** Los estudiantes podrán conocer la cantidad de horas que duermen y compararlas con las recomendaciones de sueño saludable

## 2.4 Restricciones

* **Compatible con dispositivos Android:** La aplicación estará disponible solo para dispositivos con sistema operativo Android en su primera versión.
* **Almacenamiento local:** La aplicación utilizará SQLite para el almacenamiento de datos, lo que implica que los registros se guardarán únicamente en el dispositivo del usuario.
* **Acceso sin conexión:** Algunas funciones, como la inserción de rutina y la configuración de alarmas, estarán disponibles sin conexión, pero el acceso a contenido en línea (como videos de relajación) requerirá conexión a Internet.
* **Capacidad de almacenamiento limitada:** Debido al uso de almacenamiento local, la cantidad de datos guardados dependerá de la capacidad del dispositivo.
* **Interfaz en español:** La primera versión de la aplicación estará disponible solo en español, sin soporte para otros idiomas.

## 2.5 Suposiciones y Dependencias

* Se asumirá que los usuarios tienen acceso a un dispositivo Android compatible con la aplicación.
* Se considerará que los usuarios ingresarán datos precisos sobre su rutina diaria para obtener mejores recomendaciones.
* Se asumirá que los usuarios activarán los permisos necesarios para el funcionamiento de las alarmas y la visualización de videos.
* Se dará por hecho que los usuarios comprenden el propósito de la aplicación y cómo utilizar sus funciones básicas.
* Se asumirá que los usuarios mantendrán actualizado su sistema operativo para garantizar compatibilidad con futuras versiones de la app

# 3. Requisitos Específicos

## 3.1 Requisitos Funcionales

* **RF1:** Los usuarios podrán registrar, modificar y eliminar su rutina diaria, incluyendo horas de estudio, recreación, trabajo y descanso.
* **RF2:** La aplicación permitirá la configuración y gestión de alarmas personalizadas según la rutina del usuario.
* **RF3:** Se dispondrá de un menú principal para facilitar la navegación entre las distintas funcionalidades.
* **RF4:** El sistema almacenará y mostrará estadísticas sobre las horas de sueño y la distribución del tiempo en actividades diarias.
* **RF5:** La aplicación debe ejecutar operaciones de registro y consulta en la base de datos SQLite en menos de 2 segundos.
* **RF6:** La aplicación deberá funcionar sin necesidad de conexión a Internet, salvo para la visualización de videos de relajación.
* **RF7:** Se garantizará que la aplicación sea compatible con versiones recientes de Android (mínimo Android 8.0 o superior).

## 3.2 Requisitos No Funcionales

* **RNF1:** La interfaz de usuario deberá ser intuitiva y fácil de navegar, con diseño adaptable a distintos tamaños de pantalla en dispositivos Android.
* **RNF2:** La aplicación deberá minimizar el consumo de batería, especialmente en la ejecución de alarmas y recopilación de datos en segundo plano.
* **RNF3:** La aplicación ofrecerá una sección con videos de relajación para mejorar la calidad del sueño.

## 3.3 Requisitos de Interfaz de Usuario

* La interfaz debe ser intuitiva y minimalista, permitiendo una navegación sencilla entre las funcionalidades.
* El menú principal debe incluir accesos rápidos a las funciones principales, como registro de rutina, alarmas, estadísticas y videos de relajación.
* Los botones y controles deben ser lo suficientemente grandes para facilitar su uso en dispositivos móviles.
* La fuente y los elementos visuales deben ser accesibles, con posibilidad de aumentar el tamaño del texto para mejorar la legibilidad.
* Se debe garantizar un diseño responsivo que se adapte a distintos tamaños de pantalla en dispositivos Android.

## 3.4 Requisitos de Hardware y Software

* La aplicación debe ejecutarse en dispositivos Android con Android Studio como entorno de desarrollo.
* Se utilizará SQLite como base de datos local para el almacenamiento de información del usuario.
* La aplicación debe ser compatible con dispositivos con al menos 2 GB de RAM y procesador de gama media o superior.
* El almacenamiento mínimo requerido en el dispositivo debe ser de 50 MB.
* Se recomienda el uso de un emulador o dispositivo físico con Android Studio para pruebas y depuración.

# 4. Riesgos y Limitaciones

## 4.1 Riesgos

* **Dependencia de almacenamiento local:** Al utilizar SQLite, los datos se almacenan en el dispositivo del usuario, lo que puede generar riesgos en caso de pérdida o daño del dispositivo.
* **Acceso a contenido en línea:** La visualización de videos de relajación requiere conexión a Internet, lo que podría afectar la experiencia del usuario en entornos sin acceso a la red.
* **Posibles errores en la inserción de alarmas:** Dado que la funcionalidad de alarmas está en desarrollo, podrían existir fallos en la configuración o ejecución de las mismas.
* **Compatibilidad con versiones de Android:** La aplicación podría no funcionar correctamente en versiones antiguas del sistema operativo, afectando a algunos usuarios

## 4.2 Limitaciones

* **Funcionalidades incompletas:** Actualmente, solo están habilitadas las secciones de menú principal, login, inserción de alarmas y visualización de videos. Funciones clave como el seguimiento estadístico aún no están implementadas.
* **Interfaz básica:** La interfaz de usuario está en una etapa inicial y puede carecer de elementos avanzados de usabilidad y diseño intuitivo.
* **Sincronización de datos:** En esta versión, la aplicación no cuenta con sincronización en la nube, lo que significa que los datos no se pueden recuperar si el usuario cambia de dispositivo

# 5. Alcance del Proyecto

## 5.1 Lo que incluirá

* Configuración y gestión de alarmas personalizadas.
* Menú principal para navegación entre funcionalidades.
* Visualización de videos para la relajación y mejora del sueño.
* Base de datos local en SQLite para almacenamiento de información.
* Aplicación desarrollada para dispositivos Android con Android Studio.

## 5.2 Lo que NO incluirá (por ahora)

* Inserción de estadísticas sobre el sueño.
* Registro de la rutina diaria del usuario (horas de estudio, recreativas, de ejercicio, etc.).

Backlog del Producto - SleepApp

**Fecha:** 02/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

Este documento detalla el backlog del producto para **SLEEPAPP**, una aplicación móvil para la gestión del sueño basada en la rutina del usuario. El backlog está estructurado en **épicas, historias de usuario** y sus respectivos **criterios de aceptación**.

# 2. Diseño de Épicas e Historias de Usuario

Para la correcta estructuración del backlog, cada épica y su respectiva historia de usuario estarán diseñadas bajo los siguientes principios:

* **Épicas**: Representan grandes bloques funcionales del sistema, agrupando historias de usuario relacionadas.
* **Historias de Usuario**: Son descripciones cortas y centradas en el usuario sobre una funcionalidad específica del producto.
* **Criterios de Aceptación**: Reglas que deben cumplirse para que la historia de usuario sea considerada completada.
* **Estimación**: Cada historia de usuario incluye una estimación de esfuerzo en puntos de historia.
* **Definición de "Hecho"**: Se especifica cuándo una historia de usuario se considera completada con éxito.

El diseño y priorización de las épicas y las historias de usuario se revisarán periódicamente para adaptarse a nuevas necesidades del negocio y de los usuarios.

# 3. Épicas e Historias de Usuario

# Este documento detalla el backlog del producto para SLEEPAPP, una aplicación móvil de seguimiento del sueño. El backlog está estructurado en épicas, historias de usuario y sus respectivos criterios de aceptación.

## Épica 1: Registro de usuario y autenticación del usuario

### Historia de Usuario 1.1: Registro de usuario

**Como usuario, quiero registrarme en la aplicación, para almacenar mis datos y preferencias**.

**Criterios de Aceptación:**

* El usuario puede registrarse con su correo electrónico y contraseña.
* El registro funciona sin conexión a Internet.
* Se valida la información ingresada al momento de registrarse.

**Estimación:** 5 puntos

**Definición de "Hecho":**

* La funcionalidad ha sido implementada y probada.
* El usuario puede registrarse sin errores y sus datos se almacenan correctamente.
* Se ha actualizado la documentación.

### Historia de Usuario 1.2: Iniciar sesión

**Como usuario, quiero iniciar sesión en la aplicación, para acceder a mis datos y configuraciones**.

**Criterios de Aceptación:**

* El usuario puede iniciar sesión con su correo electrónico y contraseña.
* Si el usuario ya inició sesión previamente, será dirigido directamente al menú.
* Se muestra un mensaje de error si las credenciales son incorrectas.

**Estimación:** 5 puntos

**Definición de "Hecho":**

* La funcionalidad ha sido implementada y probada.
* La autenticación funciona correctamente sin conexión.
* Compatible con Android 8.0 o superior.

## Épica 2: Registro y Gestión de Rutina

### Historia de Usuario 2.1: Registrar rutina del usuario

**Como usuario, quiero registrar mi rutina diaria incluyendo horas de estudio, recreación, trabajo y descanso, para que la aplicación pueda analizar mis patrones de sueño.**

**Criterios de Aceptación:**

* Se debe permitir ingresar y guardar las actividades diarias en la base de datos SQLite.
* La interfaz debe ser intuitiva y mostrar campos para cada actividad.

**Estimación:** 4 puntos

**Definición de "Hecho":**

* La funcionalidad se ha desarrollado y probado en un entorno controlado.
* El usuario puede registrar su rutina sin errores y los datos se almacenan correctamente en SQLite.
* Se han realizado pruebas de integración con otras funcionalidades.

### Historia de Usuario 2.2: Modificar una rutina

**Como usuario, quiero modificar mi rutina diaria para actualizar cambios en mi planificación.**

**Criterios de Aceptación:**

* Los cambios se guardan correctamente en la base de datos SQLite.
* Se muestra un mensaje de confirmación tras la actualización de una rutina.

**Estimación:** 3 puntos

**Definición de "Hecho":**

* La funcionalidad está implementada y ha pasado las pruebas unitarias y de usuario.
* Las modificaciones se reflejan correctamente en la base de datos.
* Se han actualizado las pruebas y documentación si es necesario.

## Épica 3: Configuración y Gestión de Alarmas

### Historia de Usuario 3.1: Configurar una alarma personalizada

**Como usuario, quiero configurar alarmas personalizadas según mi rutina, para recibir recordatorios en el momento adecuado**.

**Criterios de Aceptación:**

* El usuario puede crear una alarma indicando la hora con total precisión.
* La alarma se guarda en la base de datos y se activa según la configuración del usuario.
* Se muestra una notificación luego de configurar la alarma.

**Estimación:** 5 puntos

**Definición de "Hecho":**

* La funcionalidad ha sido implementada y probada con éxito.
* Las alarmas se activan según lo configurado.
* Se ha actualizado la documentación si es necesario.

## Épica 4: Menú Principal y Navegación

### Historia de Usuario 4.1: Acceder al menú de navegación

**Como usuario, quiero acceder al menú principal de la aplicación, para navegar fácilmente entre sus funcionalidades**.

**Criterios de Aceptación:**

* El menú principal permite acceder a las secciones de rutina, alarmas, estadísticas y videos.
* El diseño es intuitivo y accesible.
* Funciona sin conexión a Internet, excepto para los videos.

**Estimación:** 3 puntos

**Definición de "Hecho":**

* La funcionalidad está implementada y ha pasado las pruebas de usabilidad.
* Se han realizado pruebas de navegación e integración.
* Compatible con Android 8.0 o superior.

# 4. Priorización del Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| Prioridad | Historia de Usuario | Estado | Estimación (Puntos) |
| Alta | Registro de usuario | Pendiente | 5 |
| Alta | Iniciar sesión | Pendiente | 5 |
| Alta | Configurar una alarma personalizada | Pendiente | 5 |
| Media | Registrar rutina del usuario | Pendiente | 4 |
| Baja | Modificar una rutina | Pendiente | 3 |
| Baja | Acceder al menú de navegación | Pendiente | 3 |

Sprint Backlog - SLEEPAPP

**Fecha:** 02/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

Este documento detalla el Sprint Backlog del equipo de desarrollo para el Sprint actual de **SleepApp**. Se desglosan las historias de usuario seleccionadas, sus tareas específicas y la estimación de esfuerzo en horas. El objetivo de este sprint es avanzar en las funcionalidades clave del sistema, asegurando la correcta implementación y pruebas de cada historia de usuario.

# 2. Criterios de Estimación de Esfuerzo

Para estimar el esfuerzo de cada tarea, se utilizó la técnica **Planning Poker**, donde el el desarrollador discutió cada tarea y asignó una cantidad de horas basada en:

* Complejidad técnica
* Dependencias con otras tareas
* Experiencia previa en tareas similares
* Posibles riesgos o bloqueos

Cada tarea se estimó en consenso utilizando la escala de **horas-persona**

# 3. Historias de Usuario Seleccionadas para el Sprint Número 01

## Historia de Usuario 1.1: Registro de usuario

**Descripción:** Como usuario, quiero poder registrarme en la aplicación proporcionando mi correo y contraseña, para poder acceder a mis datos y configuraciones de sueño en cualquier dispositivo.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 1.1.1 | Diseñar la interfaz de formulario de ingreso de datos de usuario | UX/UI Designer | 4 | Pendiente |
| 1.1.2 | Implementar el formulario de ingreso de datos de usuario en la app | Frontend Dev | 6 | Pendiente |
| 1.1.3 | Configurar la conexión con la base de datos para el registro | Backend Dev | 6 | Pendiente |
| 1.1.4 | Implementar validaciones de formulario | Backend Dev | 5 | Pendiente |
| 1.1.5 | Implementar funcionalidad offline para el registro (guardado en local) | Backend Dev | 6 | Pendiente |
| 1.1.6 | Pruebas unitarias y de integración | QA Engineer | 6 | Pendiente |

## Historia de Usuario 1.2: Iniciar sesión

**Descripción:** Como usuario, quiero poder iniciar sesión con mi correo y contraseña, para acceder a mis datos en la aplicación.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 1.2.1 | Diseñar la interfaz de inicio de sesión | UX/UI Designer | 4 | Pendiente |
| 1.2.2 | Implementar el formulario de inicio de sesión en la app | Frontend Dev | 6 | Pendiente |
| 1.2.3 | Configurar Sqlite para consultar los datos del usuario login | Backend Dev | 6 | Pendiente |
| 1.2.4 | |  | | --- | |  |  |  | | --- | | Implementar validaciones (correo válido, contraseña correcta) | | Frontend Engineer | 5 | Pendiente |
| 1.2.5 | Implementar inicio de sesión sin internet | Backend Dev | 6 | Pendiente |

# 3. Total de Esfuerzo Estimado

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 8 |
| Desarrollo Frontend | Frontend Dev | 17 |
| Desarrollo Backend | Backend Dev | 29 |
| Pruebas | QA Engineer | 6 |
| Total General |  | 60 |

Este Sprint Backlog será actualizado diariamente durante las reuniones de seguimiento del equipo reflejará el avance del desarrollo y posibles ajustes en la planificación.

Sprint Backlog - SLEEPAPP

**Fecha:** 02/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

Este documento detalla el Sprint Backlog del equipo de desarrollo para el Sprint actual de **SleepApp**. Se desglosan las historias de usuario seleccionadas, sus tareas específicas y la estimación de esfuerzo en horas. El objetivo de este sprint es avanzar en las funcionalidades clave del sistema, asegurando la correcta implementación y pruebas de cada historia de usuario.

# 2. Criterios de Estimación de Esfuerzo

Para estimar el esfuerzo de cada tarea, se utilizó la técnica **Planning Poker**, donde el el desarrollador discutió cada tarea y asignó una cantidad de horas basada en:

* Complejidad técnica
* Dependencias con otras tareas
* Experiencia previa en tareas similares
* Posibles riesgos o bloqueos

Cada tarea se estimó en consenso utilizando la escala de **horas-persona**

# 3. Historias de Usuario Seleccionadas para el Sprint Número 02

## Historia de Usuario 1.2: Iniciar sesión

**Descripción:** Como usuario, quiero poder iniciar sesión con mi correo y contraseña, para acceder a mis datos en la aplicación.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 1.2.6 | Pruebas de integración | QA Enginner | 5 | Pendiente |

## Historia de Usuario 2.1: Registro y Gestión de Rutina

**Descripción:** Como usuario, quiero registrar y gestionar mi rutina, incluyendo horas de estudio, recreación, trabajo y descanso, para que la aplicación pueda analizar mis patrones de sueño y recomendar mejoras.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.1.1 | Diseñar la interfaz de ingreso y edición de rutina | UX/UI Designer | 5 | Pendiente |
| 2.1.2 | Implementar formulario para ingresar rutina diaria | Frontend Dev | 6 | Pendiente |
| 2.1.3 | Crear validación en los tiempos seleccionados | Backend Dev | 4 | Pendiente |
| 2.1.4 | Implementar almacenamiento local para funcionamiento sin conexión. | Backend Engineer | 6 | Pendiente |
| 2.1.5 | Pruebas unitarias y de integración | QA Enginner | 6 | Pendiente |

## Historia de Usuario 2.2: Modificar una rutina

**Descripción:** **Como usuario, quiero modificar mi rutina diaria para actualizar cambios en mi planificación.**

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.2.1 | Diseñar la interfaz de edición de rutina | UX/UI Designer | 4 | Pendiente |
| 2.2.2 | Agregar validaciones para la modificación de la rutina | Frontend Dev | 4 | Pendiente |
| 2.2.3 | Agregar opción para cancelar cambios antes de guardar | Backend Dev | 4 | Pendiente |
| 2.2.4 | Implementar conexión al almacenamiento local para funcionamiento sin conexión. | Backend Engineer | 6 | Pendiente |
| 2.2.5 | Pruebas unitarias y de integración | QA Enginner | 6 | Pendiente |

## Historia de Usuario 3.1: Configuración y Gestión de alarma.

**Descripción:** **Como usuario, quiero configurar alarmas personalizadas según mi rutina, para recibir en el momento adecuado**.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 1.3.1 | Diseñar la interfaz de configuración de alarma | UX/UI Designer | 5 | Pendiente |

# 3. Total de Esfuerzo Estimado

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 14 |
| Desarrollo Frontend | Frontend Dev | 10 |
| Desarrollo Backend | Backend Dev | 20 |
| Pruebas | QA Engineer | 17 |
| Total General |  | 61 |

Este Sprint Backlog será actualizado diariamente durante las reuniones de seguimiento del equipo reflejará el avance del desarrollo y posibles ajustes en la planificación.

Sprint Backlog - SleepApp

**Fecha:** 02/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

Este documento detalla el Sprint Backlog del equipo de desarrollo para el Sprint actual de **SleepApp**. Se desglosan las historias de usuario seleccionadas, sus tareas específicas y la estimación de esfuerzo en horas. El objetivo de este sprint es avanzar en las funcionalidades clave del sistema, asegurando la correcta implementación y pruebas de cada historia de usuario.

# 2. Criterios de Estimación de Esfuerzo

Para estimar el esfuerzo de cada tarea, se utilizó la técnica **Planning Poker**, donde el el desarrollador discutió cada tarea y asignó una cantidad de horas basada en:

* Complejidad técnica
* Dependencias con otras tareas
* Experiencia previa en tareas similares
* Posibles riesgos o bloqueos

Cada tarea se estimó en consenso utilizando la escala de **horas-persona**

# 3. Historias de Usuario Seleccionadas para el Sprint Número 03

## Historia de Usuario 3.1: Configuración y Gestión de alarma.

**Descripción:** **Como usuario, quiero configurar alarmas personalizadas según mi rutina, para recibir en el momento adecuado**.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 3.1.2 | Implementar selección de hora de alarma | Frontend Dev | 6 | Pendiente |
| 3.1.3 | Permitir personalización de tonos de alarmas | Backend Dev | 6 | Pendiente |
| 3.1.4 | Implementar funcionamiento sin conexión | Backend Engineer | 5 | Pendiente |
| 3.1.5 | Pruebas unitarias y de integración | QA Enginner | 6 | Pendiente |

## Historia de Usuario 4.1: Acceder al menú de Navegación

**Descripción:** Como usuario, quiero acceder fácilmente al menú de navegación de la aplicación para moverme entre las diferentes funcionalidades sin dificultad.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 4.1.1 | Diseñar la interfaz del menú de navegación | UX/UI Designer | 6 | Pendiente |
| 4.1.2 | Agregar animaciones y transiciones para una mejor UX | Frontend Dev | 4 | Pendiente |
| 4.1.3 | Configurar rutas y vinculación de cada sección | Backend Dev | 4 | Pendiente |
| 4.1.4 | |  | | --- | |  |  |  | | --- | | Implementar validación de sesión (el menú solo es visible si el usuario está autenticado) | | Backend Engineer | 6 | Pendiente |
| 2.1.5 | Pruebas de navegación y usabilidad | QA Enginner | 6 | Pendiente |

# 3. Total de Esfuerzo Estimado

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 6 |
| Desarrollo Frontend | Frontend Dev | 10 |
| Desarrollo Backend | Backend Dev | 21 |
| Pruebas | QA Engineer | 12 |
| Total General |  | 49 |

Este Sprint Backlog será actualizado diariamente durante las reuniones de seguimiento del equipo reflejará el avance del desarrollo y posibles ajustes en la planificación.

Diseño de Arquitectura del Sistema - SLEEPAPP

**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**   
  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 modelo**

La **capa de modelo** de la aplicación SleepApp está compuesta por clases que representan las entidades principales del sistema y su correspondencia directa con las tablas de la base de datos SQLite. Cada modelo incluye atributos, constructores y métodos para acceder y actualizar los datos, facilitando la interacción con la base de datos de forma estructurada y coherente.

* **Capa de persistencia**  
  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

Descripción generada automáticamente

# 3. Descripción de Componentes Principales

## 3.1. VIEW (Presentación)

La capa presentación 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 lógica de negocio 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

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 modelo)

La capa de modelo 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.

**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.

**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.

Diseño de Base de Datos - SleepApp

**Fecha:** 23/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

Este documento describe el modelo de base de datos relacional diseñado para la aplicación SleepApp, enfocada en el seguimiento de la rutina diaria del usuario, control de alarmas personalizadas, hábitos de sueño y contenido de relajación. Se detallan las entidades principales, relaciones, restricciones y se presenta el diccionario de datos correspondiente.

# 2. Diagrama Entidad-Relación (ERD)

## 2.1. Entidades y sus Atributos

1. **Usuarios**

* id\_usuario (PK)
* DNI
* usuario
* Contraseña
* Nombre
* Apellidos
* Fecha\_nacimiento
* correo
* fecha\_registro

1. **Rutina**

* id\_rutina (PK)
* Edad
* Hora\_trabajo
* Hora\_ejercicios
* Cantidad\_estimulantes
* Horas\_pantalla
* Cantidad\_estres
* id\_usuario (FK, referencia a Usuarios)

1. **Alarma**

* id\_alarma (PK)
* Hora
* Tono
* Estado
* id\_usuario (FK, referencia a Usuarios)

1. **Relajación**

* id\_video (PK)
* titulo
* url\_video
* duración
* id\_usuario(FK, referencia a Usuarios)

1. **Estadística**

* id\_estadistica(PK)
* Dia
* Horas\_Dormidas
* Promedio
* Id\_rutina(FK\_referencia a Rutina)

## 2.2. Relaciones

* **Un usuario puede tener una única rutina diaria**, lo que refleja una relación **1:1** entre la tabla Usuario y la tabla Rutina. Cada rutina está estrictamente asociada a un solo usuario.
* **Un usuario puede tener una única alarma configurada**, representando una relación **1:1** entre Usuario y Alarma. La aplicación se basa en una alarma principal personalizada por cada usuario.
* **Un usuario puede tener múltiples registros de relajación (videos vistos o recomendados)**. Esto configura una relación **1:N** entre Usuario y Relajacion, ya que un mismo usuario puede acceder a varios recursos de relajación.
* **Cada rutina tiene una única estadística vinculada**, reflejando una relación **1:1** entre Rutina y Estadistica. La tabla de estadísticas resume los datos derivados del comportamiento diario del usuario (por ejemplo, tiempo total de sueño o distribución de actividades).

## 2.3. Reglas y Restricciones

* El correo electrónico del usuario debe ser único, para evitar duplicidad en el registro y garantizar la autenticidad del acceso.
  + Restricción: Clave única sobre el campo correo.
* El DNI también debe ser único, dado que sirve como identificador nacional del usuario.
  + Restricción: Clave única sobre el campo dni
* El usuario puede configurar solo una alarma, eligiendo un único tono y una sola hora. La configuración se confirma mediante un botón de acción.
  + Restricción lógica: 1 alarma por usuario.
* En la sección de relajación, el usuario solo puede reproducir un video a la vez desde los recursos locales o en línea.
  + Restricción de interfaz: Solo se permite un VideoView activo por sesión.
* La rutina diaria del usuario se registra mediante un componente deslizante (probablemente un SeekBar) que permite aumentar o reducir el tiempo dedicado a una actividad específica.
  + Restricción de UI: El valor de rutina se selecciona con SeekBar.
* La sección de estadísticas es únicamente visual, presenta datos en forma de gráficos que resumen el comportamiento del usuario, pero no es editable ni interactiva directamente.
  + Restricción: Vista de solo lectura.

## 2.4. Diagrama de Base de Datos (E/R)

* **Usuarios (1) → (1) Rutinas** 
  + Cada usuario registra una única rutina diaria personalizada.
  + Relación uno a uno: usuario\_id como clave foránea en la tabla rutinas.
* **Usuarios (1) → (1) Alarmas** 
  + Un usuario solo puede tener una alarma activa configurada en la aplicación.
  + Relación uno a uno: usuario\_id como clave foránea en la tabla alarmas.
* **Usuarios (1) → (N) Relajaciones** 
  + Cada usuario puede ver múltiples videos de relajación.
  + Relación uno a muchos: un usuario\_id puede estar asociado a varios registros de relajación.
* **Rutinas (1) → (1) Estadísticas** 
  + Una rutina registrada genera un conjunto de estadísticas asociado (por ejemplo, horas de sueño, actividades realizadas).
  + Relación uno a uno: rutina\_id como clave foránea en la tabla estadisticas.

# 3. Diagrama Relacional

Diagrama, Esquemático

Descripción generada automáticamente

# 4. Diccionario de Datos

**Tabla Usuarios**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de Dato | Descripción | Restricciones |
| id\_usuario | INTEGER | Identificador único del usuario | PRIMARY KEY, AUTOINCREMENT |
| dni | TEXT | Documento de identidad del usuario | UNIQUE, NOT NULL |
| nombre | TEXT | Nombre del usuario | NOT NULL |
| apellidos | TEXT | Apellidos del usuario | NOT NULL |
| correo | TEXT | Correo electrónico del usuario | UNIQUE, NOT NULL |
| contraseña | TEXT | Contraseña de acceso | NOT NULL |
| fecha\_nacimiento | TEXT (DATE) | Fecha de nacimiento del usuario | NOT NULL |
| fecha\_registro | TEXT (DATE) | Fecha en la que se registró el usuario | DEFAULT (fecha actual en SQLite) |

**Tabla Rutina**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de Dato | Descripción | Restricciones |
| id\_rutina | INTEGER | Identificador único de la rutina | PRIMARY KEY, AUTOINCREMENT |
| edad | INTEGER | Edad del usuario en el momento del registro | NOT NULL |
| hora\_trabajo | INTEGER | Tiempo dedicado al trabajo o estudio (en horas) | NOT NULL |
| hora\_ejercicios | INTEGER | Tiempo dedicado a ejercicio físico (en horas) | NOT NULL |
| hora\_actividades\_recreativas | INTEGER | Tiempo en actividades recreativas | NOT NULL |
| cantidad\_estimulantes | INTEGER | Consumo diario de cafeína o estimulantes | NOT NULL |
| horas\_pantalla | INTEGER | Horas frente a pantallas al día | NOT NULL |
| cantidad\_estres | INTEGER | Nivel de estrés percibido (escala 1 a 5) | NOT NULL |
| id\_usuario | INTEGER | Referencia al usuario que registró la rutina | FOREIGN KEY → usuario(id\_usuario), UNIQUE |

**Tabla Alarma**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de Dato | Descripción | Restricciones |
| id\_alarma | INTEGER | Identificador único de la alarma | PRIMARY KEY, AUTOINCREMENT |
| hora | TEXT (TIME) | Hora configurada para la alarma | NOT NULL |
| tono | TEXT | Tono seleccionado para la alarma | NOT NULL |
| estado | INTEGER | Estado de la alarma (1 = activa, 0 = inactiva) | DEFAULT 1 |
| id\_usuario | INTEGER | Usuario que configuró la alarma | FOREIGN KEY → usuario(id\_usuario), UNIQUE |

**Tabla Relajación**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de Dato | Descripción | Restricciones |
| id\_video | INTEGER | Identificador del video de relajación | PRIMARY KEY, AUTOINCREMENT |
| titulo | TEXT | Título del video | NOT NULL |
| url\_video | TEXT | URL o ruta del archivo del video | NOT NULL |
| duracion | INTEGER | Duración del video en segundos o minutos | NOT NULL |
| id\_usuario | INTEGER | Usuario que vio el video | FOREIGN KEY → usuario(id\_usuario) |

**Tabla Estadistica**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de Dato | Descripción | Restricciones |
| id\_estadistica | INTEGER | Identificador de la estadística | PRIMARY KEY, AUTOINCREMENT |
| dia | TEXT (DATE) | Día del registro | NOT NULL |
| horas\_dormidas | INTEGER | Total de horas dormidas ese día | NOT NULL |
| promedio | REAL | Promedio semanal de horas dormidas | NOT NULL |
| id\_rutina | INTEGER | Rutina asociada a esta estadística | FOREIGN KEY → rutina(id\_rutina), UNIQUE |

## 4.1. Relaciones y Restricciones:

**1. Usuarios ↔ Rutina**

* **Relación: 1:1**
* Un usuario tiene una única rutina, y cada rutina pertenece a un solo usuario.
* Representación: campo id\_usuario como clave foránea y única en la tabla rutina.

**2. Usuarios ↔ Alarma**

* **Relación: 1:1**
* Cada usuario configura una única alarma activa (según tu modelo actual).
* Representación: campo id\_usuario como clave foránea y única en la tabla alarma.

**3. Usuarios ↔ Relajación**

* Relación: 1:N
* Un usuario puede reproducir múltiples videos de relajación.
* Representación: campo id\_usuario en la tabla relajacion.

**4. Rutina ↔ Estadística**

* Relación: 1:1
* Cada rutina genera una única estadística semanal (consolidada).
* Representación: campo id\_rutina como clave foránea y única en la tabla estadística.

# 5. Consideraciones de Escalabilidad

* Considerar el particionado de tablas como rutina, relajación o estadística por id\_usuario en sistemas distribuidos o de alto volumen.
* Monitorear el tamaño de las tablas y aplicar técnicas de archivado para datos históricos no utilizados frecuentemente.
* Preparar la base de datos para migraciones automáticas utilizando herramientas de versionado (como Room en Android o scripts SQL).
* Considerar replicación de base de datos en caso de alta demanda o implementación en nube.
* Documentar cada cambio estructural para facilitar el trabajo colaborativo y la trazabilidad del sistema.

# 6. Conclusiones

El modelo relacional propuesto para la SleepApp proporciona una estructura sólida, coherente y escalable que permite gestionar eficazmente usuarios, rutinas, alarmas, videos de relajación y estadísticas personalizadas. Gracias al uso de claves foráneas, relaciones bien definidas y restricciones adecuadas, el sistema asegura la integridad de los datos mientras se adapta al crecimiento progresivo de usuarios y registros. Esta arquitectura facilita futuras ampliaciones, como nuevas funcionalidades o integración con servicios externos, sin comprometer el rendimiento ni la consistencia de la información almacenada.

Wireframes o prototipos. - SLEPPAPP

**Fecha:** 23/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

# Este documento presenta los wireframes y prototipos de alta fidelidad para la aplicación móvil SleepApp, diseñada para ayudar a los usuarios a gestionar su rutina de sueño de manera eficiente. La aplicación permite a los usuarios registrar sus hábitos de sueño, configurar alarmas personalizadas, acceder a videos de relajación y analizar estadísticas sobre su descanso. El objetivo es representar de manera visual la interacción del usuario con la aplicación y la disposición de los elementos clave en cada pantalla.

# 2. Lineamientos de Diseño

 **Diseño centrado en el usuario**: La interfaz está diseñada de manera que sea limpia, accesible y fácil de navegar, con el objetivo de que el usuario pueda gestionar su rutina de sueño sin complicaciones.

 **Responsive**: La aplicación está completamente optimizada para ser compatible con dispositivos móviles, tabletas y escritorios, garantizando una experiencia fluida en diferentes tamaños de pantalla.

 **Consistencia visual**: Se ha utilizado una paleta de colores coherente, tipografías legibles y botones bien definidos para ofrecer una experiencia visual armónica y profesional a lo largo de la aplicación.

 **Elementos reutilizables**: Los componentes de la interfaz de usuario (UI) están diseñados de forma modular, lo que facilita la mantenibilidad y escalabilidad de la aplicación a medida que se agregan nuevas funciones.

# 3. Wireframes por Pantalla

## 3.1. Pantalla de Registro

* Campos: DNI, Usuario, Contraseña, Nombre, Apellidos, Correo, FechaNacimiento
* Botón: Registrarse,
* Botón secundario: Cancelar

Imagen que contiene Carta

Descripción generada automáticamente

## 3.2. Pantalla de Inicio de Sesión

* Campos: DNI,Contraseña
* Botón: Iniciar Sesión, Registrarse, Ver cuentas

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

## 3.3. Panel Principal (Dashboard)

* Sección principal: Mensaje de Bienvenida, Calendar, Hora
* Botónes: Rutina, Alarma, Estadística, Videos, Cerrar Sesión, Datos.

Calendario

Descripción generada automáticamente

## 3.4. Modal: Registro de Actividades

* SeekBar: Edad, Hora de Trabajo/Estudio, TiempoEjercicio, Consumo de Café o Estimulantes, Uso de Pantallas, Nivel de Estrés o ansiedad.
* Botón: Guardar, Cancelar

Diagrama

Descripción generada automáticamente con confianza baja

## 3.5. Pantalla de Alarma

* Reloj: Selección de hora para la alarma
* Botón: Selección de tono de alarma, Confirmar Alarma, Cancelar, ApagarAlarma

Imagen de la pantalla de un celular con un reloj

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

## 3.6 Pantalla de Alarma

* Grafico: Barras, Circular
* Botón: Volver

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## 3.6 Pantalla de Alarma

* Visualizador de Videos
* Botón: Video Anterior, Siguiente Video, Volver



# 4. Prototipo de Navegación

1. Usuario accede a pantalla de inicio de sesión.
2. Si no tiene cuenta, puede registrarse.
3. Tras iniciar sesión, accede al menú principal.
4. Desde el menú principal puede ingresar a las distintas pantallas.
5. Cada pestaña cuenta con un botón para regresar al menú principal.
6. La pestaña de rutina permite ingresar los datos de la rutina del usuario.
7. La pestaña de alarma permite crear una alarma siguiendo los datos de la rutina del usuario.
8. La pestaña de estadística permite ver el progreso de sueño que tiene el usuario.
9. La pestaña de videos permite al usuario realizar ejercicios de relajación.

# 5. Guía de Estilo UI (Diseño Visual)

**Estilo UI para SleepApp**

**Tipografía**

* **Fuente principal:** Inter, sans-serif
* **Títulos:** Estilo bold, tamaños:
  + h1: 24px
  + h2: 20px
  + h3: 18px
  + h4: 16px
* **Texto normal:** Estilo regular, tamaño 14px

**Paleta de colores**

* **Color principal:** #3B82F6 (Azul)
* **Color secundario:** #6366F1 (Índigo)
* **Fondo general:** #F9FAFB (Gris claro)
* **Texto principal:** #111827 (Negro grisáceo)
* **Éxito:** #10B981 (Verde)
* **Advertencia:** #F59E0B (Naranja)
* **Error:** #EF4444 (Rojo)

**Botones**

* Bordes **redondeados**, con **sombra suave**.
* Al pasar el cursor (hover): tono ligeramente **más oscuro**.
* En estado **disabled**: aplicar **opacidad reducida**.

# 7. Componentes XML

## 7.1. Botónes

<Button  
 android:id="@+id/btnSelectTone"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Seleccionar Tono de Alarma"  
 android:layout\_gravity="center"/>

## 7.2. EditText

<EditText  
 android:id="@+id/etDni"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:hint="DNI"  
 android:inputType="textEmailAddress"/>

# 8. Consideraciones Finales

* Es fundamental aplicar pruebas de usabilidad con usuarios de diferentes rangos de edad para asegurar una experiencia intuitiva y accesible.
* Documentar los resultados de las pruebas de usabilidad permitirá ajustes rápidos antes de su lanzamiento o presentación académica.

La interfaz debe adaptarse a diferentes dispositivos móviles para garantizar un uso fluido en diversas resoluciones y sistemas Android.

Sprint 01 - SLEEPAPP

**Fecha:** 29/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

Sprint 01 – SleepApp

# 1. Introducción

Este documento detalla el desarrollo del Sprint 01 del proyecto SleepApp, una aplicación móvil orientada a mejorar la calidad del sueño de los usuarios mediante el registro de rutinas y alarmas personalizadas. En este sprint se trabajó en las funcionalidades fundamentales de acceso: registro de usuario e inicio de sesión, incluyendo interfaces, validaciones y soporte offline. También se documentan los avances técnicos mediante evidencias de código y el seguimiento de tareas a través del burndown chart.

# 3. Lineamientos de Diseño

Durante el Sprint 01, se desarrollará una primera versión funcional de la aplicación móvil SleepApp, enfocada en la gestión personalizada del sueño. Esta versión tendrá como objetivo implementar las funcionalidades básicas que permitirán a los usuarios comenzar a interactuar con el sistema. Las funcionalidades planificadas para este entregable son:

* Registro de usuario
* Inicio de sesión
* Validación offline
* Diseño de interfaces funcionales

Estas funcionalidades servirán como base para los módulos avanzados, como el seguimiento del sueño, alarmas inteligentes y recomendaciones personalizadas, que serán abordados en los siguientes sprints.

# 2. Objetivo del Sprint

Implementar funcionalidades básicas de acceso del usuario: registro e inicio de sesión, incluyendo persistencia de datos local y validaciones.

# 3. Historias de Usuario y Tareas

* Historia de Usuario 1.1: Registro de usuario

Descripción: Como usuario, quiero poder registrarme en la aplicación proporcionando mi correo y contraseña, para poder acceder a mis datos y configuraciones de sueño en cualquier dispositivo.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tarea | Responsable | Horas |
| 1.1.1 | Diseñar la interfaz de formulario de ingreso de datos de usuario | UX/UI Designer | 4 |
| 1.1.2 | Implementar el formulario de ingreso de datos de usuario en la app | Frontend Dev | 6 |
| 1.1.3 | Configurar la conexión con la base de datos para el registro | Backend Dev | 6 |
| 1.1.4 | Implementar validaciones de formulario | Backend Dev | 5 |
| 1.1.5 | Implementar funcionalidad offline para el registro (guardado en local) | Backend Dev | 6 |
| 1.1.6 | Pruebas unitarias y de integración | QA Engineer | 6 |

* Historia de Usuario 1.2: Iniciar sesión

Descripción: Como usuario, quiero poder iniciar sesión con mi correo y contraseña, para acceder a mis datos en la aplicación.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tarea | Responsable | Horas |
| 1.2.1 | Diseñar la interfaz de inicio de sesión | UX/UI Designer | 4 |
| 1.2.2 | Implementar el formulario de inicio de sesión en la app | Frontend Dev | 6 |
| 1.2.3 | Configurar Sqlite para consultar los datos del usuario login | Backend Dev | 6 |
| 1.2.4 | Implementar validaciones (correo válido, contraseña correcta) | Frontend Engineer | 5 |
| 1.2.5 | Implementar inicio de sesión sin internet | Backend Dev | 6 |

# 4. Evidencias de Código

A continuación, se incluirán fragmentos del código implementado para las historias de usuario.

# 4.1 Historia de Usuario 01

* Diseñar la interfaz de formulario de ingreso de datos de usuario

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:background="@drawable/fondo"  
 tools:context=".MainActivity">  
  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="center"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:gravity="center"  
 android:text="CREAR NUEVA CUENTA"  
 android:textColor="#000000"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
 <EditText  
 android:id="@+id/etDni"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:hint="DNI"  
 android:inputType="textEmailAddress"/>  
 <EditText  
 android:id="@+id/etUsuario"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:hint="Usuario"  
 android:inputType="textEmailAddress"/>  
 <EditText  
 android:id="@+id/etContraseña"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:hint="Contraseña"  
 android:inputType="textPassword"/>  
 <EditText  
 android:id="@+id/etNombre"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:hint="Nombre"  
 android:inputType="textPersonName"/>  
 <EditText  
 android:id="@+id/etApellido"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:hint="Apellidos"  
 android:inputType="textPersonName"/>  
 <EditText  
 android:id="@+id/etCorreo"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:hint="Correo"  
 android:inputType="textEmailAddress"/>  
  
 <EditText  
 android:id="@+id/etFechanacimiento"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp"  
 android:layout\_weight="1"  
 android:hint="FechaNacimiento"  
 android:inputType=" "/>  
  
  
 <Button  
 android:id="@+id/btnRegRegistrar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="7dp"  
 android:layout\_weight="1"  
 android:backgroundTint="#12FF00"  
 android:text="Registrarse"  
 android:textColor="#FFFFFF"  
 android:textSize="30dp"  
 android:textStyle="bold" />  
 <Button  
 android:id="@+id/btnRegCancelar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="7dp"  
 android:layout\_weight="1"  
 android:backgroundTint="#F44336"  
 android:text="Cancelar"  
 android:textColor="#ffffff"  
 android:textSize="30dp"  
 android:textStyle="bold"/>  
  
</LinearLayout>

* Implementar el formulario de ingreso de datos de usuario en la app

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* Configurar la conexión con la base de datos para el registro
* // Insertar en la base de datos si las validaciones son correctas  
  try {  
   SQLiteOpenHelper conexion = new SqlLiteHelper(getApplicationContext());  
   SQLiteDatabase baseDeDatos = conexion.getWritableDatabase();  
   ContentValues valores = new ContentValues();  
   valores.put("dni", Integer.*valueOf*(dniStr));  
   valores.put("usuario", usuarioStr);  
   valores.put("contraseña", contraseñaStr);  
   valores.put("nombre", nombreStr);  
   valores.put("apellido", apellidoStr);  
   valores.put("correo", correoStr);  
   valores.put("fechanacimiento", fechanacimientoStr);  
    
   long resultado = baseDeDatos.insert("tusuarios", null, valores);  
   if (resultado != -1) {  
   Toast.*makeText*(getApplicationContext(), "Se guardó correctamente", Toast.*LENGTH\_SHORT*).show();  
   } else {  
   Toast.*makeText*(getApplicationContext(), "Error al guardar el registro", Toast.*LENGTH\_SHORT*).show();  
   }  
    
  } catch (SQLException e) {  
   Toast.*makeText*(getApplicationContext(), "Error: " + e.getMessage(), Toast.*LENGTH\_SHORT*).show();  
  }
* Implementar validaciones de formulario

// Validación de los campos  
if (TextUtils.*isEmpty*(dniStr) || dniStr.length() != 8 || !TextUtils.*isDigitsOnly*(dniStr)) {  
 Toast.*makeText*(getApplicationContext(), "DNI debe ser de 8 dígitos numéricos", Toast.*LENGTH\_SHORT*).show();  
 return;  
}  
  
if (TextUtils.*isEmpty*(contraseñaStr) || contraseñaStr.length() < 6) {  
 Toast.*makeText*(getApplicationContext(), "La contraseña debe tener al menos 6 caracteres", Toast.*LENGTH\_SHORT*).show();  
 return;  
}  
  
if (TextUtils.*isEmpty*(usuarioStr) || TextUtils.*isEmpty*(nombreStr) || TextUtils.*isEmpty*(apellidoStr)  
 || TextUtils.*isEmpty*(correoStr) || TextUtils.*isEmpty*(fechanacimientoStr)) {  
 Toast.*makeText*(getApplicationContext(), "Todos los campos deben estar llenos", Toast.*LENGTH\_SHORT*).show();  
 return;  
}

* Implementar funcionalidad offline para el registro (guardado en local)
* @Override  
  public void onCreate(SQLiteDatabase db) {  
   // Crear tabla tusuarios  
   db.execSQL("CREATE TABLE IF NOT EXISTS " + *TABLA\_USUARIOS* + " (" +  
   *COLUMNA\_DNI* + " INTEGER PRIMARY KEY, " +  
   *COLUMNA\_USUARIO* + " TEXT NOT NULL, " +  
   *COLUMNA\_CONTRASENA* + " TEXT NOT NULL, " +  
   *COLUMNA\_NOMBRE* + " TEXT NOT NULL, " +  
   *COLUMNA\_APELLIDO* + " TEXT NOT NULL, " +  
   *COLUMNA\_CORREO* + " TEXT NOT NULL, " +  
   *COLUMNA\_FECHA\_NACIMIENTO* + " TEXT NOT NULL)");
* Pruebas unitarias y de integración

Texto

Descripción generada automáticamente

# 4.1 Historia de Usuario 1.2: Inicio de Sesión

* Diseñar la interfaz de inicio de sesión
* <?xml version="1.0" encoding="utf-8"?>  
  <LinearLayout  
   xmlns:android="http://schemas.android.com/apk/res/android"  
   xmlns:app="http://schemas.android.com/apk/res-auto"  
   xmlns:tools="http://schemas.android.com/tools"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:orientation="vertical"  
   tools:context=".MainActivity"  
   android:background="@drawable/fondo">  
    
   <ImageView  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:layout\_weight="1"  
   android:src="@drawable/ic\_user" />  
    
   <EditText  
   android:id="@+id/User"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:layout\_margin="20dp"  
   android:layout\_weight="1"  
   android:hint="DNI"  
   android:inputType="textEmailAddress" />  
    
   <EditText  
   android:id="@+id/Pass"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:layout\_margin="20dp"  
   android:layout\_weight="1"  
   android:hint="Contraseña"  
   android:inputType="textPassword"/>  
    
   <Button  
   android:id="@+id/btnInicio"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:layout\_margin="20dp"  
   android:layout\_weight="1"  
   android:backgroundTint="#12FF00"  
   android:text="Iniciar sesión"  
   android:textColor="#FFFFFF"  
   android:textSize="30dp"  
   android:textStyle="bold" />  
   <Button  
   android:id="@+id/btnRegistro"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:layout\_margin="20dp"  
   android:layout\_weight="1"  
   android:backgroundTint="#0000ff"  
   android:text="REGISTRASE"  
   android:textColor="#ffffff"  
   android:textSize="30dp"  
   android:textStyle="bold"/>  
   <Button  
   android:id="@+id/btnVerCuentas"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:layout\_margin="20dp"  
   android:layout\_weight="1"  
   android:backgroundTint="#F44336"  
   android:text="VER CUENTAS"  
   android:textColor="#ffffff"  
   android:textSize="30dp"  
   android:textStyle="bold"/>  
    
  </LinearLayout>
* Implementar el formulario de inicio de sesión en la app

Una captura de pantalla de un celular

Descripción generada automáticamente

* Configurar Sqlite para consultar los datos del usuario login
* String dniIngresado = dni.getText().toString();  
  String contraseñaIngresada = contraseña.getText().toString();
* Implementar validaciones (correo válido, contraseña correcta)

if (cursor.moveToFirst()) {  
 // Inicio de sesión exitoso  
 Toast.*makeText*(getApplicationContext(), "Inicio de sesión exitoso", Toast.*LENGTH\_SHORT*).show();  
  
 // Guardar el DNI del usuario actual para futuras consultas  
 String dniUsuarioActual = cursor.getString(cursor.getColumnIndex("dni"));  
 Intent ventanaPrincipal = new Intent(MainActivity.this, MenuPrincipal.class);  
 ventanaPrincipal.putExtra("dniUsuarioActual", dniUsuarioActual); // Pasar el DNI a la siguiente actividad  
 startActivity(ventanaPrincipal);  
 } else {  
 // Usuario o contraseña incorrectos  
 Toast.*makeText*(getApplicationContext(), "DNI o contraseña incorrectos", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 cursor.close();  
 baseDeDatos.close();  
 }  
});

* Implementar inicio de sesión sin internet
* @Override  
  public void onClick(View v) {  
   SQLiteOpenHelper conexion = new SqlLiteHelper(getApplicationContext());  
   SQLiteDatabase baseDeDatos = conexion.getReadableDatabase();

# 5. Burndown Chart

El siguiente gráfico representa el progreso del sprint, comparando las horas restantes con los días de ejecución.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SPRINT BURNDOWN CHART | | | | | | | | | | | | |
|  |
| Backlog ID | Historia de Usuario | INICIAL ESTIMADO | 1-Abr | 2-Abr | 3-Abr | 4-Abr | 5-Abr | 6-Abr | 7-Abr | 8-Abr | 9-Abr | 10-Abr |  |
| DIA 0 | DIA 1 | DIA 2 | DIA 3 | DIA 4 | DIA 5 | DIA 6 | DIA 7 | DIA 8 | DIA 9 | DIA 10 |  |
| 1.1.1 | Historia de Usuario 01 Registro de usuario | 4 |  |  | 1 | 1 | 1 | 1 |  |  |  |  |  |
| 1.1.2 | 6 |  |  |  |  | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 1.1.3 | 6 |  |  |  |  |  |  | 2 | 2 | 1 |  |  |
| 1.1.4 | 5 |  |  |  |  |  | 1 | 1 | 1 | 1 | 1 |  |
| 1.1.5 | 6 |  | 1 |  |  | 1 |  | 1 | 1 | 1 | 1 |  |
| 1.1.6 | 6 |  |  |  |  |  |  |  | 2 | 2 | 2 |  |
| 1.2.1 | Historia de Usuario 02 Iniciar Sesión | 4 |  | 1 | 1 |  |  | 1 | 1 |  |  |  |  |
| 1.2.2 | 6 |  |  |  |  |  |  | 1 | 1 | 2 | 2 |  |
| 1.2.3 | 6 |  |  |  |  |  |  |  | 2 | 2 | 2 |  |
| 1.2.4 | 5 |  |  |  | 1 | 1 | 1 | 1 | 1 |  |  |  |
| 1.2.5 | 6 |  |  |  |  | 1 | 1 | 1 | 1 | 1 | 2 |  |
| Tiempo Trabajado | | 60 | 60 | 58 | 56 | 54 | 49 | 43 | 34 | 22 | 11 | 0 |  |
| Tiempo Estimado | | 60 | 54 | 48 | 42 | 36 | 30 | 24 | 18 | 12 | 6 | 0 |  |

# 6. Análisis del Sprint

# Durante el Sprint 01, el equipo trabajó en las funcionalidades base de la aplicación: el registro y el inicio de sesión de usuarios, tanto en línea como sin conexión. Se planificaron dos historias principales (1.1 y 1.2), que incluían el diseño de interfaces, implementación de formularios, conexión con la base de datos SQLite, validaciones y soporte offline.

* Se diseñaron interfaces amigables y funcionales para el formulario de registro y login.
* Se logró implementar correctamente las funcionalidades principales de ambas pantallas.
* Se incorporaron validaciones para mejorar la experiencia del usuario y prevenir errores comunes.

# Dificultades Técnicas Encontradas

 **La implementación de SQLite:** Hubo errores en la creación de la base de datos, en especial al momento de realizar consultas para el login o insertar datos desde el formulario de registro.

 **Funcionalidad offline:** Al intentar validar los datos y realizar operaciones sin conexión, surgieron problemas relacionados con la persistencia de datos y sincronización que no estaban completamente previstos.

 **Validaciones complejas:** Algunas validaciones (como el formato del correo y el manejo de errores cuando no se encuentran registros) requirieron más tiempo de prueba y ajuste.

Estas dificultades provocaron que algunas tareas quedaran bloqueadas por varios días, lo que obligó al equipo a **acelerar el ritmo en los últimos días del sprint** para poder cumplir con los entregables. La mayor parte del código funcional fue integrado en la fase final, lo que también redujo el tiempo disponible para pruebas detalladas.

El esfuerzo final se enfocó en:

* Corregir la estructura de la base de datos.
* Ajustar las funciones de inserción y consulta en SQLite.
* Implementar correctamente los flujos de uso sin conexión.
* Realizar pruebas básicas para garantizar que las funciones trabajaran correctamente bajo condiciones normales.

# 7. Conclusiones

El Sprint 01 permitió establecer una base sólida para las funcionalidades de acceso del usuario, cumpliendo con el diseño y parte de la implementación del registro e inicio de sesión. El equipo trabajó de forma colaborativa y organizada, sentando las bases para abordar con mayor eficacia las funcionalidades de monitoreo del sueño en los próximos sprints.

Sprint 02 - SLEEPAPP

**Fecha:** 14/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

Sprint 02 – SleepApp

# 1. Introducción

Durante el **Sprint 02** del proyecto **SleepApp**, se continuará con el desarrollo de funcionalidades clave que permitirán avanzar hacia una versión más completa de la aplicación. En este sprint se dará continuidad a la Historia de Usuario 1.2, añadiendo pruebas de integración para asegurar la correcta funcionalidad del inicio de sesión. Además, se incorporarán nuevas historias de usuario enfocadas en la **gestión de la rutina diaria del usuario**, permitiendo registrar, modificar y validar actividades como estudio, trabajo, recreación y descanso. Esta información será esencial para que la aplicación pueda analizar los hábitos del usuario y posteriormente generar recomendaciones personalizadas sobre el sueño. También se iniciará el desarrollo de la funcionalidad de **configuración de alarmas personalizadas**, que permitirá a los usuarios definir alarmas de acuerdo con su rutina planificada.

# 3. Lineamientos de Diseño

Durante el **Sprint 02**, se desarrollará una versión más completa de la aplicación móvil **SleepApp**, enfocada en incorporar funcionalidades relacionadas con la **rutina diaria del usuario** y la **configuración de alarmas personalizadas**. El objetivo de este sprint será permitir que los usuarios gestionen su planificación diaria (horas de estudio, trabajo, recreación y descanso), y que la aplicación pueda utilizar esta información para futuras recomendaciones sobre el sueño:

* Registro y gestión de rutina diaria del usuario
* Edición y modificación de la rutina registrada
* Almacenamiento local de datos para permitir el uso sin conexión
* Diseño de interfaces para rutina y alarma

Estas funcionalidades permitirán que en los siguientes sprints se integre el módulo de análisis de patrones de sueño, alarmas inteligentes, y recomendaciones personalizadas, basadas en los datos que el usuario haya proporcionado a través de estas rutinas.

# 2. Objetivo del Sprint

Se implementarán las funcionalidades básicas de acceso del usuario, que incluyen el registro y el inicio de sesión, junto con la persistencia de datos de forma local mediante SQLite y la incorporación de validaciones necesarias para garantizar la integridad de los datos ingresados.

# 3. Historias de Usuario y Tareas

* Historia de Usuario 2.1: Registro y gestión de rutinas

Descripción: Como usuario, quiero registrar y gestionar mi rutina, incluyendo horas de estudio, recreación, trabajo y descanso, para que la aplicación pueda analizar mis patrones de sueño y recomendar mejoras.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tarea | Responsable | Horas |
| 2.1.1 | Diseñar la interfaz de ingreso y edición de rutina | UX/UI Designer | 5 |
| 2.1.2 | Implementar formulario para ingresar rutina diaria | Frontend Dev | 6 |
| 2.1.3 | Crear validación en los tiempos seleccionados | Backend Dev | 4 |
| 2.1.4 | Implementar almacenamiento local para funcionamiento sin conexión. | Backend Dev | 4 |
| 2.1.5 | Pruebas unitarias y de integración | Backend Dev | 6 |

* Historia de Usuario 2.2: Modificar una rutina

Descripción: **Como usuario, quiero modificar mi rutina diaria para actualizar cambios en mi planificación**.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tarea | Responsable | Horas |
| 2.2.1 | Diseñar la interfaz de edición de rutina | UX/UI Designer | 4 |
| 2.2.2 | Agregar validaciones para la modificación de la rutina | Frontend Dev | 4 |
| 2.2.3 | Agregar opción para cancelar cambios antes de guardar | Backend Dev | 4 |
| 2.2.4 | Implementar conexión al almacenamiento local para funcionamiento sin conexión. | Frontend Engineer | 6 |
| 2.2.5 | Pruebas unitarias y de integración | Backend Dev | 6 |

# 4. Evidencias de Código

A continuación, se incluirán fragmentos del código implementado para las historias de usuario.

# 4.1 Historia de Usuario 01

* Pruebas de integración

Captura de pantalla de computadora

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# 4.1 Historia de Usuario 02

* Diseñar la interfaz de ingreso y edición de rutina usuario
* <?xml version="1.0" encoding="utf-8"?>  
  <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:padding="16dp"  
   android:background="@drawable/fondo"  
   >  
    
   <LinearLayout  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:orientation="vertical"  
   android:background="@drawable/fondo">  
    
   <TextView  
   android:id="@+id/textViewTitle"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Registro de Actividades"  
   android:textSize="24sp"  
   android:textStyle="bold"  
   android:gravity="center"  
   android:layout\_gravity="center"/>  
    
   <!-- Edad con un Spinner para seleccionar un rango de edades -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Edad"/>  
    
   <Spinner  
   android:id="@+id/spinnerAge"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content" />  
    
   <!-- Horas de trabajo con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Horas de trabajo/estudio (h)" />  
    
   <SeekBar  
   android:id="@+id/seekBarWorkHours"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="16"/>  
    
   <TextView  
   android:id="@+id/textViewWorkHoursValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0 h"/>  
    
   <!-- Tiempo de ejercicio con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Tiempo de ejercicio (min)" />  
    
   <SeekBar  
   android:id="@+id/seekBarExercise"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="120" />  
    
   <TextView  
   android:id="@+id/textViewExerciseValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0 min"/>  
    
   <!-- Actividades recreativas con SeekBar (en horas) -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:visibility="gone"  
   android:text="Tiempo de actividades recreativas (h)" />  
    
   <SeekBar  
   android:id="@+id/seekBarRecreationalActivities"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:visibility="gone"  
   android:max="10" />  
    
   <TextView  
   android:id="@+id/textViewRecreationalActivitiesValue"  
   android:layout\_width="wrap\_content"  
   android:visibility="gone"  
   android:layout\_height="wrap\_content"  
   android:text="0 h"/>  
    
   <!-- Consumo de café o estimulantes con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Consumo de café o estimulantes (tazas)" />  
    
   <SeekBar  
   android:id="@+id/seekBarCaffeine"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="10" />  
    
   <TextView  
   android:id="@+id/textViewCaffeineValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0 tazas"/>  
    
   <!-- Uso de pantallas con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Uso de pantallas (h)" />  
    
   <SeekBar  
   android:id="@+id/seekBarScreenTime"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="12" />  
    
   <TextView  
   android:id="@+id/textViewScreenTimeValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0 h"/>  
    
   <!-- Niveles de estrés o ansiedad con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Niveles de estrés o ansiedad (1-10)" />  
    
   <SeekBar  
   android:id="@+id/seekBarStress"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="10" />  
    
   <TextView  
   android:id="@+id/textViewStressValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0"/>  
    
   <!-- Botón de guardar -->  
   <Button  
   android:id="@+id/btnSubmit"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:text="Guardar" />  
    
   <Button  
   android:id="@+id/btnCancelar"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:text="Cancelar"  
   android:backgroundTint="#F44336"/>  
    
    
   </LinearLayout>  
  </ScrollView>
* Implementar formulario para ingresar rutina diaria

Pantalla de celular con imagen de la pantalla de un computador

Descripción generada automáticamente con confianza media

* Configurar la conexión con la base de datos para el registro
* // Crear tabla trutina  
  db.execSQL("CREATE TABLE IF NOT EXISTS " + *TABLA\_RUTINA* + " (" +  
   *COLUMNA\_DNI* + " INTEGER, " +  
   *COLUMNA\_EDAD* + " TEXT NOT NULL, " +  
   *COLUMNA\_HORAS\_TRABAJO* + " INTEGER NOT NULL, " +  
   *COLUMNA\_EJERCICIO\_MINUTOS* + " INTEGER NOT NULL, " +  
   *COLUMNA\_RECREACION\_MINUTOS* + " INTEGER NOT NULL, " +  
   *COLUMNA\_CAFE\_CONSUMO* + " INTEGER NOT NULL, " + // Café  
   *COLUMNA\_TIEMPO\_PANTALLA* + " INTEGER NOT NULL, " + // Tiempo frente a pantallas  
   *COLUMNA\_ESTRES* + " INTEGER NOT NULL, " + // Estrés  
   "FOREIGN KEY (" + *COLUMNA\_DNI* + ") REFERENCES " + *TABLA\_USUARIOS* + "(" + *COLUMNA\_DNI* + ") ON DELETE CASCADE)");
* Crear validación en los tiempos seleccionados

// Conectar con los elementos del XML  
spinnerAge = findViewById(R.id.*spinnerAge*);  
seekBarWorkHours = findViewById(R.id.*seekBarWorkHours*);  
seekBarExercise = findViewById(R.id.*seekBarExercise*);  
seekBarRecreationalActivities = findViewById(R.id.*seekBarRecreationalActivities*);  
seekBarCaffeine = findViewById(R.id.*seekBarCaffeine*);  
seekBarScreenTime = findViewById(R.id.*seekBarScreenTime*);  
seekBarStress = findViewById(R.id.*seekBarStress*);  
textViewWorkHoursValue = findViewById(R.id.*textViewWorkHoursValue*);  
textViewExerciseValue = findViewById(R.id.*textViewExerciseValue*);  
textViewRecreationalActivitiesValue = findViewById(R.id.*textViewRecreationalActivitiesValue*);  
textViewCaffeineValue = findViewById(R.id.*textViewCaffeineValue*);  
textViewScreenTimeValue = findViewById(R.id.*textViewScreenTimeValue*);  
textViewStressValue = findViewById(R.id.*textViewStressValue*);  
btnSubmit = findViewById(R.id.*btnSubmit*);  
btnCancel = findViewById(R.id.*btnCancelar*);

* Implementar funcionalidad offline para el registro (guardado en local)
* // Manejar el botón de guardar  
  btnSubmit.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   // Obtener los valores de las SeekBars  
   int workHours = seekBarWorkHours.getProgress();  
   int exerciseTime = seekBarExercise.getProgress();  
   int recreationalActivitiesTime = seekBarRecreationalActivities.getProgress();  
   int caffeineConsumption = seekBarCaffeine.getProgress();  
   int screenTime = seekBarScreenTime.getProgress();  
   int stressLevel = seekBarStress.getProgress();  
    
     
    
   // Guardar en la base de datos  
   sqlLiteHelper.insertarRutina(  
   dni,  
   spinnerAge.getSelectedItem().toString(),  
   workHours,  
   exerciseTime,  
   recreationalActivitiesTime,  
   caffeineConsumption,  
   screenTime,  
   stressLevel  
   );  
    
   Toast.*makeText*(registrorutina.this, "Datos guardados correctamente", Toast.*LENGTH\_SHORT*).show();  
   finish(); // Cerrar la actividad  
   }  
  });
* Pruebas unitarias y de integración

Captura de pantalla de computadora

Descripción generada automáticamente

# 4.1 Historia de Usuario 2.2: Modificar una rutina

* Diseñar la interfaz de edición de rutina
* <?xml version="1.0" encoding="utf-8"?>  
  <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:padding="16dp"  
   android:background="@drawable/fondo"  
   >  
    
   <LinearLayout  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:orientation="vertical"  
   android:background="@drawable/fondo">  
    
   <TextView  
   android:id="@+id/textViewTitle"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Registro de Actividades"  
   android:textSize="24sp"  
   android:textStyle="bold"  
   android:gravity="center"  
   android:layout\_gravity="center"/>  
    
   <!-- Edad con un Spinner para seleccionar un rango de edades -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Edad"/>  
    
   <Spinner  
   android:id="@+id/spinnerAge"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content" />  
    
   <!-- Horas de trabajo con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Horas de trabajo/estudio (h)" />  
    
   <SeekBar  
   android:id="@+id/seekBarWorkHours"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="16"/>  
    
   <TextView  
   android:id="@+id/textViewWorkHoursValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0 h"/>  
    
   <!-- Tiempo de ejercicio con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Tiempo de ejercicio (min)" />  
    
   <SeekBar  
   android:id="@+id/seekBarExercise"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="120" />  
    
   <TextView  
   android:id="@+id/textViewExerciseValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0 min"/>  
    
   <!-- Actividades recreativas con SeekBar (en horas) -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:visibility="gone"  
   android:text="Tiempo de actividades recreativas (h)" />  
    
   <SeekBar  
   android:id="@+id/seekBarRecreationalActivities"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:visibility="gone"  
   android:max="10" />  
    
   <TextView  
   android:id="@+id/textViewRecreationalActivitiesValue"  
   android:layout\_width="wrap\_content"  
   android:visibility="gone"  
   android:layout\_height="wrap\_content"  
   android:text="0 h"/>  
    
   <!-- Consumo de café o estimulantes con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Consumo de café o estimulantes (tazas)" />  
    
   <SeekBar  
   android:id="@+id/seekBarCaffeine"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="10" />  
    
   <TextView  
   android:id="@+id/textViewCaffeineValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0 tazas"/>  
    
   <!-- Uso de pantallas con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Uso de pantallas (h)" />  
    
   <SeekBar  
   android:id="@+id/seekBarScreenTime"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="12" />  
    
   <TextView  
   android:id="@+id/textViewScreenTimeValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0 h"/>  
    
   <!-- Niveles de estrés o ansiedad con SeekBar -->  
   <TextView  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Niveles de estrés o ansiedad (1-10)" />  
    
   <SeekBar  
   android:id="@+id/seekBarStress"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:max="10" />  
    
   <TextView  
   android:id="@+id/textViewStressValue"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="0"/>  
    
   <!-- Botón de guardar -->  
   <Button  
   android:id="@+id/btnSubmit"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:text="Guardar" />  
    
   <Button  
   android:id="@+id/btnCancelar"  
   android:layout\_width="match\_parent"  
   android:layout\_height="wrap\_content"  
   android:text="Cancelar"  
   android:backgroundTint="#F44336"/>  
    
    
   </LinearLayout>  
  </ScrollView>
* Agregar validaciones para la modificación de la rutina
* // Conectar con los elementos del XML  
  spinnerAge = findViewById(R.id.*spinnerAge*);  
  seekBarWorkHours = findViewById(R.id.*seekBarWorkHours*);  
  seekBarExercise = findViewById(R.id.*seekBarExercise*);  
  seekBarRecreationalActivities = findViewById(R.id.*seekBarRecreationalActivities*);  
  seekBarCaffeine = findViewById(R.id.*seekBarCaffeine*);  
  seekBarScreenTime = findViewById(R.id.*seekBarScreenTime*);  
  seekBarStress = findViewById(R.id.*seekBarStress*);  
  textViewWorkHoursValue = findViewById(R.id.*textViewWorkHoursValue*);  
  textViewExerciseValue = findViewById(R.id.*textViewExerciseValue*);  
  textViewRecreationalActivitiesValue = findViewById(R.id.*textViewRecreationalActivitiesValue*);  
  textViewCaffeineValue = findViewById(R.id.*textViewCaffeineValue*);  
  textViewScreenTimeValue = findViewById(R.id.*textViewScreenTimeValue*);  
  textViewStressValue = findViewById(R.id.*textViewStressValue*);  
  btnSubmit = findViewById(R.id.*btnSubmit*);  
  btnCancel = findViewById(R.id.*btnCancelar*);
* Agregar opción para cancelar cambios antes de guardar
* btnCancel.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   //Volver a la pantalla principal  
   Intent Volver = new Intent(registrorutina.this,MenuPrincipal.class);  
   startActivity(Volver);  
   }  
  });
* Implementar conexión al almacenamiento local para funcionamiento sin conexión.

// Manejar el botón de guardar  
btnSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // Obtener los valores de las SeekBars  
 int workHours = seekBarWorkHours.getProgress();  
 int exerciseTime = seekBarExercise.getProgress();  
 int recreationalActivitiesTime = seekBarRecreationalActivities.getProgress();  
 int caffeineConsumption = seekBarCaffeine.getProgress();  
 int screenTime = seekBarScreenTime.getProgress();  
 int stressLevel = seekBarStress.getProgress();

cursor.close();  
 baseDeDatos.close();  
 }  
});

// Guardar en la base de datos  
 sqlLiteHelper.insertarRutina(  
 dni,  
 spinnerAge.getSelectedItem().toString(),  
 workHours,  
 exerciseTime,  
 recreationalActivitiesTime,  
 caffeineConsumption,  
 screenTime,  
 stressLevel  
 );  
  
 Toast.*makeText*(registrorutina.this, "Datos guardados correctamente", Toast.*LENGTH\_SHORT*).show();  
 finish(); // Cerrar la actividad  
 }  
});

* Pruebas unitarias y de integración

Captura de pantalla de computadora

Descripción generada automáticamente

# 4.1 Historia de Usuario 03

* Diseñar la interfaz de configuración de alarma**.**
* <LinearLayout  
   xmlns:android="http://schemas.android.com/apk/res/android"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:orientation="vertical"  
   android:padding="16dp"  
   android:background="@drawable/fondo">  
    
    
   <!-- Selector de la hora para la alarma -->  
   <TimePicker  
   android:id="@+id/timePickerAlarm"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:timePickerMode="clock"  
   android:layout\_gravity="center"/>  
    
   <!-- Texto que muestra la hora sugerida para acostarse -->  
   <TextView  
   android:id="@+id/tvRecommendedSleepTime"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Hora sugerida para acostarse"  
   android:layout\_gravity="center"  
   android:padding="10dp"/>  
    
   <!-- Texto que muestra la hora para dormir máxima -->  
   <TextView  
   android:id="@+id/tvMaxSleepTime"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Hora para dormir máxima"  
   android:layout\_gravity="center"  
   android:padding="10dp"/>  
    
   <!-- Botón para seleccionar el tono de alarma -->  
   <Button  
   android:id="@+id/btnSelectTone"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Seleccionar Tono de Alarma"  
   android:layout\_gravity="center"/>  
    
   <!-- Botón para confirmar la alarma -->  
   <Button  
   android:id="@+id/btnSetAlarm"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:text="Confirmar Alarma"  
   android:layout\_gravity="center"/>  
   <Button  
   android:id="@+id/btnCancelarAlarma"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:backgroundTint="#F44336"  
   android:text="Cancelar"  
   android:layout\_gravity="center"/>  
    
  </LinearLayout>

# 5. Burndown Chart

El siguiente gráfico representa el progreso del sprint, comparando las horas restantes con los días de ejecución.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SPRINT BURNDOWN CHART | | | | | | | | | | | | |
|  |
| Backlog ID | Historia de Usuario | INICIAL ESTIMADO | 11-Abr | 12-Abr | 13-Abr | 14-Abr | 15-Abr | 16-Abr | 17-Abr | 18-Abr | 19-Abr | 20-Abr |  |
| DIA 0 | DIA 1 | DIA 2 | DIA 3 | DIA 4 | DIA 5 | DIA 6 | DIA 7 | DIA 8 | DIA 9 | DIA 10 |  |
| 1.2.6 | Historia de usuario 2.1 Inicio de sesión | 5 |  | 1 | 1 |  | 1 |  | 1 |  | 1 |  |  |
| 2.1.1 | Historia de Usuario 2.1 Registro y Gestion de rutina | 5 |  |  | 1 | 1 |  | 1 |  | 1 | 1 |  |  |
| 2.1.2 | 6 |  |  |  | 1 | 1 | 1 | 1 | 1 | 1 |  |  |
| 2.1.3 | 4 |  |  |  |  |  | 1 | 1 |  | 1 | 1 |  |
| 2.1.4 | 6 |  |  |  | 1 |  | 1 |  | 1 | 1 | 2 |  |
| 2.1.5 | 6 |  |  |  |  | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 2.2.1 | Historia de Usuario 2.2 Modificiación de rutina | 4 |  |  |  |  |  |  | 1 | 1 | 1 | 1 |  |
| 2.2.2 | 4 |  | 2 |  |  |  |  |  | 1 | 1 |  |  |
| 2.2.3 | 4 |  |  |  |  | 1 | 1 | 1 | 1 |  |  |  |
| 2.2.4 | 6 |  |  |  |  |  |  | 1 | 1 | 2 | 2 |  |
| 2.2.5 | 6 |  |  |  |  | 1 | 1 | 1 | 1 | 1 | 2 |  |
| 1.3.1 | Historia de usuario 3.1 Configuración y Gestión de alarma | 5 |  |  |  |  | 1 | 1 | 1 | 1 | 1 | 2 |  |
| Tiempo Trabajado | | 61 | 61 | 58 | 56 | 53 | 48 | 40 | 32 | 22 | 11 | 0 |  |
| Tiempo Estimado | | 61 | 55 | 49 | 43 | 37 | 31 | 24 | 18 | 12 | 6 | 0 |  |

# 6. Análisis del Sprint

# Durante el Sprint 02, el equipo abordó el diseño y desarrollo de funcionalidades clave para la aplicación, centradas en la rutina diaria del usuario y la configuración de alarmas personalizadas. El objetivo era mantener la coordinación lograda en el Sprint 01, pero con una planificación más técnica y detallada para prevenir bloqueos.

 Se diseñaron e implementaron las interfaces para el registro y modificación de rutinas, así como la pantalla de gestión de alarmas.

 Se incorporaron validaciones relacionadas con el tiempo de actividades y lógica para controlar conflictos horarios.

 Se logró implementar de forma inicial el almacenamiento local de rutinas mediante SQLite.

# . Dificultades Técnicas Encontradas

 **Problemas con el almacenamiento local de rutinas:** La lógica para guardar y modificar actividades en SQLite fue más compleja de lo previsto, especialmente al tratar de mantener la integridad de los datos al realizar modificaciones.

 **Validaciones de tiempo:** Implementar restricciones para evitar traslapes de horarios, tiempos negativos o mal ingresados demandó mayor cantidad de pruebas y ajustes.

 **Cancelación de cambios:** La opción para deshacer o cancelar ediciones no fue completamente funcional en los primeros intentos, ya que requería manejar estados intermedios en memoria que no estaban contemplados al inicio.

Debido a estas dificultades, varias tareas quedaron en espera durante los primeros días del sprint. El equipo debió **intensificar el trabajo en los días finales** para completar las historias comprometidas. Este esfuerzo permitió:

* Reorganizar el esquema de almacenamiento de datos en SQLite.
* Implementar una lógica más robusta para validaciones y control de errores.
* Realizar pruebas funcionales mínimas que aseguraran estabilidad en el flujo de rutina y alarmas.

# 7. Conclusiones

El Sprint 02 permitirá consolidar la funcionalidad central de la aplicación relacionada con la gestión personalizada de la rutina diaria del usuario. Se buscará avanzar significativamente en el diseño, implementación y validación de los formularios para registrar, editar y modificar la rutina, así como en la configuración inicial de alarmas personalizadas.

Sprint 03 - SLEEPAPP

**Fecha:** 14/04/2025

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

Durante el Sprint 03 se trabajará en el desarrollo de funcionalidades clave que mejorarán la experiencia de uso de la aplicación SleepApp. En esta etapa se priorizará la **configuración y gestión avanzada de alarmas**, así como el **acceso intuitivo al menú de navegación**, permitiendo a los usuarios personalizar sus horarios de alerta de acuerdo con sus rutinas, y desplazarse fácilmente entre las distintas secciones de la aplicación. Se abordarán tareas como la selección de hora para alarmas, personalización de tonos, validación de funcionamiento offline, así como pruebas unitarias y de integración. Asimismo, se desarrollará el diseño e implementación del menú de navegación, incluyendo animaciones, configuración de rutas internas y validación de sesión para asegurar que solo los usuarios autenticados puedan acceder a las funcionalidades internas.

# 3. Lineamientos de Diseño

Durante el **Sprint 02**, se desarrollará una versión más completa de la aplicación móvil **SleepApp**, enfocada en incorporar funcionalidades relacionadas con la **rutina diaria del usuario** y la **configuración de alarmas personalizadas**. El objetivo de este sprint será permitir que los usuarios gestionen su planificación diaria (horas de estudio, trabajo, recreación y descanso), y que la aplicación pueda utilizar esta información para futuras recomendaciones sobre el sueño:

* Registro y gestión de rutina diaria del usuario
* Edición y modificación de la rutina registrada
* Almacenamiento local de datos para permitir el uso sin conexión
* Diseño de interfaces para rutina y alarma

Estas funcionalidades permitirán que en los siguientes sprints se integre el módulo de análisis de patrones de sueño, alarmas inteligentes, y recomendaciones personalizadas, basadas en los datos que el usuario haya proporcionado a través de estas rutinas.

# 2. Objetivo del Sprint

Se implementarán las funcionalidades de configuración y gestión de alarmas personalizadas, permitiendo a los usuarios seleccionar la hora de la alarma y personalizar los tonos de notificación. Asimismo, se desarrollará el funcionamiento sin conexión mediante almacenamiento local, y se diseñará e integrará un menú de navegación funcional que estará disponible solo para usuarios autenticados.

# 3. Historias de Usuario y Tareas

* Historia de Usuario 3.1: Configuración y gestión de alarma.

Descripción: Como usuario, quiero configurar alarmas personalizadas según mi rutina, para recibir en el momento adecuado.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tarea | Responsable | Horas |
| 3.1.2 | Implementar selección de hora de alarma | Frontend Dev | 6 |
| 3.1.3 | Permitir personalización de tonos de alarmas | Backend Dev | 6 |
| 3.1.4 | Implementar funcionamiento sin conexión | Backend Engineer | 5 |
| 3.1.5 | Pruebas unitarias y de integración | QA Enginner | 6 |

* Historia de Usuario 4.1: Acceder al menú de Navegación

Descripción: Como usuario, quiero acceder fácilmente al menú de navegación de la aplicación para moverme entre las diferentes funcionalidades sin dificultad.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tarea | Responsable | Horas |
| 4.1.1 | Diseñar la interfaz del menú de navegación | UX/UI Designer | 4 |
| 4.1.2 | Agregar animaciones y transiciones para una mejor UX | Frontend Dev | 4 |
| 4.1.3 | Configurar rutas y vinculación de cada sección | Backend Dev | 4 |
| 4.1.4 | Implementar validación de sesión (el menú solo es visible si el usuario está autenticado) | Backend Engineer | 6 |
| 2.1.5 | Pruebas de navegación y usabilidad | QA Enginner | 6 |

# 4. Evidencias de Código

A continuación, se incluirán fragmentos del código implementado para las historias de usuario.

# 4.1 Historia de Usuario 03

* Implementación selección de hora de alarma

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@drawable/fondo">  
  
  
 <!-- Selector de la hora para la alarma -->  
 <TimePicker  
 android:id="@+id/timePickerAlarm"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:timePickerMode="clock"  
 android:layout\_gravity="center"/>  
  
 <!-- Texto que muestra la hora sugerida para acostarse -->  
 <TextView  
 android:id="@+id/tvRecommendedSleepTime"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hora sugerida para acostarse"  
 android:layout\_gravity="center"  
 android:padding="10dp"/>  
  
 <!-- Texto que muestra la hora para dormir máxima -->  
 <TextView  
 android:id="@+id/tvMaxSleepTime"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hora para dormir máxima"  
 android:layout\_gravity="center"  
 android:padding="10dp"/>  
  
 <!-- Botón para seleccionar el tono de alarma -->  
 <Button  
 android:id="@+id/btnSelectTone"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Seleccionar Tono de Alarma"  
 android:layout\_gravity="center"/>  
  
 <!-- Botón para confirmar la alarma -->  
 <Button  
 android:id="@+id/btnSetAlarm"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Confirmar Alarma"  
 android:layout\_gravity="center"/>  
 <Button  
 android:id="@+id/btnCancelarAlarma"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="#F44336"  
 android:text="Cancelar"  
 android:layout\_gravity="center"/>  
  
</LinearLayout>

* Permitir personalización de tonos de alarmas

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* Implementar funcionamiento sin conexión
* // Configurar la alarma al hacer clic en el botón  
   btnSetAlarm.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View v) {  
   int hour = timePickerAlarm.getHour();  
   int minute = timePickerAlarm.getMinute();  
    
   if (selectedToneUri != null) {  
   // Guardar la alarma en la base de datos con el tono seleccionado  
   Calendar calendar = Calendar.*getInstance*();  
   calendar.set(Calendar.*HOUR\_OF\_DAY*, hour);  
   calendar.set(Calendar.*MINUTE*, minute);  
   calendar.set(Calendar.*SECOND*, 0);  
    
   // Programar la alarma  
   Intent intent = new Intent(RegistrarAlarma.this, AlarmReceiver.class);  
   intent.putExtra("ALARM\_TONE", selectedToneUri.toString());  
    
   PendingIntent pendingIntent = PendingIntent.*getBroadcast*(  
   RegistrarAlarma.this, 0, intent, PendingIntent.*FLAG\_UPDATE\_CURRENT* | PendingIntent.*FLAG\_IMMUTABLE*);  
    
   AlarmManager alarmManager = (AlarmManager) getSystemService(*ALARM\_SERVICE*);  
   if (alarmManager != null) {  
   alarmManager.setExact(AlarmManager.*RTC\_WAKEUP*, calendar.getTimeInMillis(), pendingIntent);  
   Toast.*makeText*(RegistrarAlarma.this, "Alarma configurada para " + hour + ":" + minute, Toast.*LENGTH\_SHORT*).show();  
    
   // Enviar Intent a estadisticas  
   Intent updateIntent = new Intent(RegistrarAlarma.this, MenuPrincipal.class);  
   updateIntent.putExtra("UPDATE\_GRAPH", true);  
   startActivity(updateIntent); // O usa un método diferente si deseas regresar a la actividad actual  
   }  
   } else {  
   Toast.*makeText*(RegistrarAlarma.this, "Por favor, selecciona un tono de alarma", Toast.*LENGTH\_SHORT*).show();  
   }  
   }  
   });  
   btnCancelarAlarma.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   //Regresar al menu principal  
   Intent Regresar = new Intent(RegistrarAlarma.this,MenuPrincipal.class);  
   startActivity(Regresar);  
   }  
   });  
    
    
  }
* Pruebas unitarias y de integración

Captura de pantalla de computadora

Descripción generada automáticamente

Pantalla de juego de computadora

Descripción generada automáticamente con confianza media

# 4.2 Historia de Usuario 04

* Diseñar la interfaz del menú de navegación
* <?xml version="1.0" encoding="utf-8"?>  
  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
   xmlns:app="http://schemas.android.com/apk/res-auto"  
   xmlns:tools="http://schemas.android.com/tools"  
   android:id="@+id/main"  
   android:layout\_width="match\_parent"  
   android:layout\_height="match\_parent"  
   android:background="@drawable/fondo"  
   tools:context=".MenuPrincipal">  
    
    
   <ImageButton  
   android:id="@+id/btnSalir"  
   android:layout\_width="wrap\_content"  
   android:layout\_height="wrap\_content"  
   android:background="#00FFFFFF"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toEndOf="parent"  
   app:layout\_constraintHorizontal\_bias="0.847"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.076"  
   app:srcCompat="@drawable/iconosalida" />  
    
   <ImageButton  
   android:id="@+id/perfil"  
   android:layout\_width="60dp"  
   android:layout\_height="65dp"  
   android:background="#00FFFFFF"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toEndOf="parent"  
   app:layout\_constraintHorizontal\_bias="0.133"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.067"  
   app:srcCompat="@drawable/usuario" />  
    
   <TextView  
   android:id="@+id/Bienvenido"  
   android:layout\_width="247dp"  
   android:layout\_height="40dp"  
   android:text="BIENVENIDO &quot;CHRISTIAN&quot;"  
   android:textSize="20sp"  
   android:textStyle="bold"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toEndOf="parent"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.205" />  
    
   <CalendarView  
   android:id="@+id/calendarView"  
   android:layout\_width="317dp"  
   android:layout\_height="274dp"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toEndOf="parent"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.452" />  
    
   <TextClock  
   android:id="@+id/textClock"  
   android:layout\_width="170dp"  
   android:layout\_height="51dp"  
   android:textSize="34sp"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toEndOf="parent"  
   app:layout\_constraintHorizontal\_bias="0.568"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.747" />  
    
   <ImageButton  
   android:id="@+id/btnRutina"  
   android:layout\_width="91dp"  
   android:layout\_height="70dp"  
   android:layout\_marginStart="28dp"  
   android:background="#00FFFFFF"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toStartOf="@+id/textView3"  
   app:layout\_constraintHorizontal\_bias="0.315"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.888"  
   app:srcCompat="@drawable/rutina" />  
    
   <ImageButton  
   android:id="@+id/btnEstadistica"  
   android:layout\_width="95dp"  
   android:layout\_height="76dp"  
   android:background="#00FFFFFF"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toEndOf="parent"  
   app:layout\_constraintHorizontal\_bias="0.693"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.896"  
   app:srcCompat="@drawable/estadistic" />  
    
   <ImageButton  
   android:id="@+id/botonMostrarAlarma"  
   android:layout\_width="75dp"  
   android:layout\_height="78dp"  
   android:background="#00FFFFFF"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toEndOf="parent"  
   app:layout\_constraintHorizontal\_bias="0.407"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.898"  
   app:srcCompat="@drawable/clock" />  
    
   <ImageButton  
   android:id="@+id/btnVideos"  
   android:layout\_width="98dp"  
   android:layout\_height="80dp"  
   android:background="#00FFFFFF"  
   app:layout\_constraintBottom\_toBottomOf="parent"  
   app:layout\_constraintEnd\_toEndOf="parent"  
   app:layout\_constraintHorizontal\_bias="1.0"  
   app:layout\_constraintStart\_toStartOf="parent"  
   app:layout\_constraintTop\_toTopOf="parent"  
   app:layout\_constraintVertical\_bias="0.901"  
   app:srcCompat="@drawable/yoga" />  
    
  </androidx.constraintlayout.widget.ConstraintLayout>
* Agregar animaciones y transiciones para una mejor UX

<ImageButton  
 android:id="@+id/btnRutina"  
 android:layout\_width="91dp"  
 android:layout\_height="70dp"  
 android:layout\_marginStart="28dp"  
 android:background="#00FFFFFF"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/textView3"  
 app:layout\_constraintHorizontal\_bias="0.315"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.888"  
 app:srcCompat="@drawable/rutina" />

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Configurar rutas y vinculación de cada sección
* botonRutina.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   Intent botonRutina = new Intent(MenuPrincipal.this, registrorutina.class);  
   startActivity(botonRutina);  
   }  
  });  
    
  botonAlarma.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   Intent botonAlarma = new Intent(MenuPrincipal.this, RegistrarAlarma.class);  
   startActivity(botonAlarma);  
   }  
  });  
    
  botonEstadistica.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   Intent botonEstadistica = new Intent(MenuPrincipal.this, estadisticas.class);  
   startActivity(botonEstadistica);  
   }  
  });  
    
  botonVideos.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   Intent botonVideos = new Intent(MenuPrincipal.this, VideosActivity.class);  
   startActivity(botonVideos);  
   }  
  });
* Implementar validación de sesión (el menú solo es visible si el usuario está autenticado)

// Guardar el DNI del usuario actual para futuras consultas  
String dniUsuarioActual = cursor.getString(cursor.getColumnIndex("dni"));  
Intent ventanaPrincipal = new Intent(MainActivity.this, MenuPrincipal.class);  
ventanaPrincipal.putExtra("dniUsuarioActual", dniUsuarioActual); // Pasar el DNI a la siguiente actividad  
startActivity(ventanaPrincipal);

* Pruebas de navegación y usabilidad

Captura de pantalla de computadora

Descripción generada automáticamente

Pantalla de juego de computadora

Descripción generada automáticamente con confianza media

# 5. Burndown Chart

El siguiente gráfico representa el progreso del sprint, comparando las horas restantes con los días de ejecución.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SPRINT BURNDOWN CHART | | | | | | | | | | | | |
|  |
| Backlog ID | Historia de Usuario | INICIAL ESTIMADO | 21-Abr | 22-Abr | 23-Abr | 24-Abr | 25-Abr | 26-Abr | 27-Abr | 28-Abr | 29-Abr | 30-Abr |  |
| DIA 0 | DIA 1 | DIA 2 | DIA 3 | DIA 4 | DIA 5 | DIA 6 | DIA 7 | DIA 8 | DIA 9 | DIA 10 |  |
| 3.1.2 | Historia de usuario 3.1 Configuración y Gestión de alarma | 6 | 2 | 1 | 1 |  | 1 | 1 |  |  |  |  |  |
| 3.1.3 | 6 |  | 2 | 1 | 1 | 1 |  |  |  |  |  |  |
| 3.1.4 | 5 |  |  |  |  | 1 | 1 | 1 | 1 | 1 |  |  |
| 3.1.5 | 6 |  |  |  |  |  | 1 | 1 | 1 | 1 | 2 |  |
| 4.1.1 | Historia de Usuario 4.1 Acceder al meú de navegación | 6 |  | 1 |  | 1 |  |  | 1 | 1 | 2 | 1 |  |
| 4.1.2 | 4 |  |  |  | 1 |  | 1 |  | 1 | 1 | 2 |  |
| 4.1.3 | 4 |  |  |  |  |  |  |  |  | 2 | 2 |  |
| 4.1.4 | 6 | 1 | 1 |  | 1 | 1 | 1 | 1 |  |  |  |  |
| 4.1.5 | 6 |  |  |  |  |  |  | 1 | 1 | 2 | 2 |  |
| Tiempo Trabajado | | 49 | 46 | 41 | 39 | 35 | 32 | 28 | 23 | 18 | 9 | 0 |  |
| Tiempo Estimado | | 49 | 44 | 39 | 34 | 29 | 25 | 20 | 15 | 10 | 5 | 0 |  |

# 6. Análisis del Sprint

# Durante el Sprint 03, el equipo se enfocó en el desarrollo de funcionalidades relacionadas con la configuración avanzada de alarmas y el acceso al menú de navegación. Estas tareas exigieron una coordinación más estrecha entre las áreas de diseño, frontend y backend, especialmente por la necesidad de mantener una experiencia de usuario fluida tanto online como offline.

 Se implementaron pantallas funcionales para la gestión y personalización de alarmas, incluyendo la selección de tonos.

 Se desarrolló el acceso al menú de navegación, con control de visibilidad solo para usuarios autenticados.

 Se agregaron animaciones al menú, mejorando la experiencia visual del usuario.

# Dificultades Técnicas Encontradas

 **Almacenamiento offline de alarmas:** La lógica para que las alarmas funcionen sin conexión, especialmente en cuanto a notificaciones locales persistentes, presentó conflictos en dispositivos con diferentes versiones de Android.

 **Autenticación de sesión:** La validación para mostrar el menú de navegación exclusivamente a usuarios autenticados generó incompatibilidades temporales con la gestión de sesiones en SQLite.

 **Animaciones del menú:** Las animaciones requerían compatibilidad con distintas resoluciones y dispositivos, lo cual implicó ajustes en diseño responsivo y consumo de recursos.

 **Tonos personalizados:** La integración de archivos de audio personalizados exigió permisos adicionales y pruebas de compatibilidad con formatos de audio.

Debido a los obstáculos técnicos y la alta dependencia entre tareas, varias funcionalidades quedaron parcialmente implementadas durante la primera mitad del sprint. El equipo debió **reorganizar prioridades y trabajar intensamente en los días finales**, lo cual permitió:

* Resolver los errores de permisos y notificaciones en alarmas offline.
* Estabilizar la lógica de autenticación para controlar el acceso al menú.
* Ajustar animaciones y realizar pruebas en distintos dispositivos

# 7. Conclusiones

El Sprint 03 permitirá consolidar funcionalidades clave de la aplicación SleepApp, enfocadas en la personalización de la experiencia del usuario a través de la configuración de alarmas y la navegación fluida por la aplicación. Se buscará avanzar de forma significativa en la implementación de alarmas personalizadas según la rutina del usuario, la gestión offline de estas configuraciones, así como en el diseño e integración del menú de navegación. Estos avances fortalecerán la usabilidad general de la aplicación y prepararán el terreno para incorporar funcionalidades más avanzadas en los siguientes sprints, como las alarmas inteligentes y el análisis automático de patrones de sueño.

Manual de Usuario Final — SleepApp

**Fecha: 18/06/2025**

**Autor: Christian Castro Usto**

**Versión:** 1.0

# 1. Introducción

Bienvenido a SleepApp, una aplicación móvil desarrollada para ayudarte a gestionar y mejorar tu rutina diaria y tus hábitos de sueño de manera personalizada, simple y eficiente. Este manual tiene como objetivo guiarte paso a paso en el uso de la aplicación, explicando cada una de sus funcionalidades: desde el registro y la configuración de tu rutina diaria, hasta la personalización de alarmas y el acceso a recursos de relajación. SleepApp está diseñada para brindarte una experiencia intuitiva y funcional que contribuya a tu bienestar y descanso.

# 2. Requisitos del Sistema

Para acceder y utilizar correctamente la aplicación móvil SleepApp, necesitas cumplir con los siguientes requisitos:

* Un dispositivo móvil Android (versión 7.0 o superior).
* Almacenamiento interno disponible para guardar datos localmente mediante SQLite.
* Conexión a Internet estable para acceder a funciones como la descarga de videos de relajación y sincronización inicial de datos (aunque algunas funcionalidades están disponibles sin conexión).

Permisos habilitados para:

* Acceso a **notificaciones** (para activar alarmas).
* Acceso al **almacenamiento** (para guardar configuraciones y rutinas localmente)
* Reproducción de **contenido multimedia** (para videos de relajación).

# 3. Acceso a la Aplicación

## 3.1 Registro de Usuario

1. Abre la aplicación SleepApp en tu dispositivo móvil Android.
2. En la pantalla de inicio, presiona el botón **"Registrarse"**.
3. Completa el formulario con tus datos personales: DNI, nombres, apellidos, correo electrónico, contraseña, confirmación de contraseña y fecha de nacimiento.
4. Presiona el botón **"Crear cuenta"** para finalizar el registro.
5. Si los datos son válidos, la cuenta se guardará localmente en el dispositivo.

## 3.2 Inicio de Sesión

1. En la pantalla principal de la aplicación, presiona el botón **"Iniciar Sesión"**.
2. Ingresa tu **correo electrónico** y **contraseña** registrados previamente.
3. Presiona el botón **"Ingresar"** para acceder.
4. Si los datos son incorrectos, la aplicación mostrará un mensaje de error.

Si los datos son incorrectos, el sistema mostrará un mensaje de error.

# 4. Funcionalidades Principales

## 4.1 Registro de Rutina Diaria

1. Una vez iniciada la sesión, accede a la opción **"Registrar Rutina"** desde el menú principal.
2. Utiliza los controles deslizantes para ingresar la cantidad de horas dedicadas a trabajo, estudio, ejercicio, actividades recreativas, uso de pantallas, etc.
3. Indica también tu nivel de estrés y consumo de estimulantes.
4. Presiona el botón **"Guardar"** para almacenar la rutina de forma local

## 4.2 Configuración de Alarma

1. Ve a la sección **"Alarmas"** desde el menú de navegación.
2. Selecciona la **hora deseada** usando el selector de tiempo (TimePicker).
3. Pulsa el botón **"Seleccionar Tono"** para elegir un sonido de alarma personalizado.
4. Confirma la configuración presionando **"Confirmar Alarma".**
5. Puedes cancelar la alarma activa con el botón **"Cancelar".**

## 4.3 Visualización de Datos Estadísticos

1. Accede a la sección **"Estadísticas"** para visualizar gráficos sobre tus horas dormidas diarias y promedios semanales.
2. La información se genera automáticamente a partir de las rutinas registradas.
3. Los gráficos permiten identificar patrones de sueño y posibles mejoras.

## 4.4 Reproducción de videos de relajación

## Ingresa a la sección "Relajación" desde el menú principal.

## Elige entre los diferentes videos disponibles (almacenados localmente o en línea).

## Presiona el botón de reproducción para ver un video de forma individual.

## Solo se puede reproducir un video a la vez para evitar sobrecarga sensorial.

# 5. Buenas Prácticas de Uso

* Registra tu **rutina diaria con precisión**, indicando correctamente tus horarios de estudio, trabajo, ejercicio y descanso para obtener recomendaciones más acertadas.
* **Actualiza tu rutina** si cambian tus hábitos o actividades, para mantener la aplicación alineada con tu día a día.
* Configura tus **alarmas personalizadas** según tu planificación para ayudarte a mejorar tu higiene del sueño.
* **Consulta frecuentemente las estadísticas** para evaluar tu progreso y ajustar tu rutina si es necesario.
* Utiliza los **videos de relajación** en momentos de estrés o antes de dormir para favorecer el descanso.
* **Haz uso de la aplicación todos los días** para mantener un registro consistente y obtener mejores resultados.

# 6. Preguntas Frecuentes (FAQ)

**¿Qué sucede si cierro la aplicación o apago mi dispositivo?**

Los datos se almacenan localmente en tu dispositivo usando SQLite, por lo que **no se perderán** al cerrar la app.

**¿Puedo acceder a mi información desde otro dispositivo?**

En esta versión inicial, los datos se guardan de forma local, por lo tanto, **no es posible sincronizarlos entre diferentes dispositivos**.

**¿Puedo editar o eliminar mi rutina o alarmas?**

# Sí, puedes modificar tu rutina diaria desde la opción "Editar Rutina". Las alarmas también se pueden cancelar o reconfigurar.

# ¿Puedo ver más de un video de relajación a la vez?

# No. Para evitar distracciones, la aplicación solo permite reproducir un video a la vez.

# 7. Contacto y Soporte

# Para dudas, problemas técnicos o sugerencias sobre la aplicación, puedes escribir al correo de soporte del equipo de desarrollo: 📧 [soporte@sleepapp.com](mailto:soporte@sleepapp.com)

# 8. Cierre

Gracias por utilizar **SleepApp**. Esperamos que esta aplicación te ayude a mejorar tu calidad de descanso mediante una mejor gestión de tu rutina diaria y patrones de sueño.  
SleepApp es un proyecto en evolución constante. ¡Tu retroalimentación es muy valiosa para seguir mejorando la experiencia de usuario y añadir nuevas funcionalidades!

Manual Técnico — SleepApp

**Fecha**: 18/06/2025

**Autor**: Christian Castro Usto

**Versión**: 1.0

# 1. Introducción

Este manual técnico documenta los aspectos internos del desarrollo de **SleepApp**, una aplicación móvil desarrollada en Android para la gestión personalizada de la rutina diaria y la mejora del sueño. El documento está dirigido a desarrolladores, técnicos de soporte y profesionales interesados en comprender la **estructura, arquitectura, tecnologías utilizadas y el funcionamiento interno** del sistema.

SleepApp tiene como objetivo proporcionar una experiencia integral al usuario, permitiéndole registrar su rutina, configurar alarmas personalizadas, visualizar estadísticas de sueño y acceder a contenido de relajación, todo almacenado localmente mediante SQLite. Este manual ofrece una guía técnica completa para facilitar el mantenimiento, la evolución del sistema y su escalabilidad en futuras versiones.

.

# 2. Tecnologías Utilizadas

|  |  |
| --- | --- |
| Componente | Tecnología |
| Frontend (Interfaz) | Android SDK (Java) + XML (Material Design) |
| Lógica de Negocio | Java (Clases de control y validación) |
| Base de Datos | SQLite (almacenamiento local) |
| Persistencia de Datos | SQLite (sincronización local/offline) |
| Reproducción de Medios | Android MediaPlayer (videos de relajación) |
| Notificaciones/Alarmas | AlarmManager + NotificationManager de Android |

# 3. Estructura del Proyecto

# SleepApp/

# │

# ├── manifests/

# │ └── AndroidManifest.xml

# │

# ├── java/

# │ └── com.example.sleepapp/

# │ ├── MainActivity.java ← App.js

# │ ├── MenuPrincipal.java ← Pantalla de inicio

# │ ├── Registrar.java ← Registro.js

# │ ├── LoginActivity.java ← Login.js

# │ ├── RegistrarAlarma.java ← NuevaTarea.js

# │ ├── NotificacionesActivity.java ← Notificaciones

# │ ├── EstadisticasActivity.java ← Vista de estadísticas

# │ ├── GraficoBarras.java ← Gráfico barras

# │ ├── GraficoCircular.java ← Gráfico circular

# │ ├── AlarmReceiver.java ← Alarmas automáticas

# │ ├── ApagarAlarma.java ← Detener alarmas

# │ ├── Usuario.java ← Modelo de usuario

# │ ├── SqlLiteHelper.java ← Configuración de base de datos

# │ └── ... otros componentes

# │

# ├── res/

# │ ├── layout/

# │ │ ├── activity\_apagar\_alarma.xml

# │ │ └── activity\_estadisticas.xml

# │ │ └── activity\_listardatos.

# │ │ └── activity\_login.xml

# │ │ └── activity\_main.xml

# │ │ └── activity\_menu\_principal.xml

# │ │ └── activity\_registrar.xml

# │ │ └── activity\_registrar\_alarma.xml

# │ │ └── activity\_registrar\_rutina.xml

# │ └── drawable/

# │ └── ... imágenes, íconos

# │

# ├── test/

# │ └── com.example.sleepapp/

# │

# └── androidTest/

# └── com.example.sleepapp/

# 4. Arquitectura General

Arquitectura en **capas**:

* **Capa de Presentación:** Android (Java) + XML (Material Design) para la interfaz de usuario móvil.
* **Capa de Negocio:** Lógica implementada en Java dentro de las Activities y clases de control para el procesamiento de datos, cálculos de sueño y validaciones.
* **Capa de Modelo:** Clases Java que representan las entidades del sistema (Usuario, Rutina, Alarma, etc.), enlazadas a la base de datos SQLite.
* **Capa de Persistencia:** SQLite como base de datos local, con acceso a través de DAOs (UsuarioDao, RutinaDao, AlarmaDao) y gestionada mediante DatabaseHelper.

# 5. Descripción de Componentes

### 5.1 Frontend (Interfaz Móvil Android)

* **Lenguaje:** Java para Android (usando Android Studio).
* **Estilos:** XML con Material Design.
* **Comunicación con base de datos:** Mediante clases Java y DAOs conectadas a SQLite.
* **Componentes clave (Activities):**
  + Registrar.java: Formulario de registro de nuevos usuarios.
  + LoginActivity.java: Autenticación e inicio de sesión local.
  + RegistroRutina.java: Ingreso y edición de la rutina diaria del usuario.
  + RegistrarAlarma.java: Configuración de alarmas personalizadas.
  + VideosActivity.java: Reproducción de videos de relajación.
  + EstadisticasActivity.java: Visualización de datos estadísticos (horas dormidas, promedios).

### 5.2 Backend (Lógica de Negocio integrada en la app)

* **Tecnología:** Java (en las mismas Activities y clases auxiliares).
* **Gestión de operaciones CRUD:** Se realiza dentro de DatabaseHelper.java.
* **Validaciones:** Lógica incorporada para controlar:
  + Registro e inicio de sesión (verificación de datos).
  + Reglas para ingreso/modificación de rutinas y alarmas.
  + Validación de campos (tiempos, estados, formatos).
* **Funcionalidad offline:** Toda la aplicación funciona sin conexión, almacenando datos en SQLite local.

### 5.3 Base de Datos

* **Motor:** SQLite (local, embebido en la app).
* **Gestión de acceso:** Implementado mediante DAOs y DatabaseHelper.java.
* **Tablas del esquema:**
  + usuarios(id\_usuario, dni, nombre, apellidos, correo, contraseña, fecha\_nacimiento, fecha\_registro)
  + rutinas(id\_rutina, edad, hora\_trabajo, hora\_ejercicios, hora\_actividades\_recreativas, cantidad\_estimulantes, horas\_pantalla, cantidad\_estres, id\_usuario)
  + alarmas(id\_alarma, hora, tono, estado, id\_usuario)
  + relajacion(id\_video, titulo, url\_video, duracion, id\_usuario)
  + estadisticas(id\_estadistica, dia, horas\_dormidas, promedio, id\_rutina)

## 6. Instalación y Ejecución

## 6.1 Requisitos

 **Android Studio** (versión recomendada: Electric Eel o superior)

 **Java Development Kit (JDK) 8 o superior**

 **Dispositivo físico con Android 8.0+** o **emulador configurado**

 **Git** (para clonar el repositorio si es necesario)

## 6.2 Backend

## git clone https://github.com/ChrisCU16/SLEEPAPP\_TP2.git

## cd SLEEPAPP\_TP2

## 6.3 Abrir y ejecutar el proyecto en Android Studio

 Abrir Android Studio.

 Seleccionar "Open an existing project" y elegir la carpeta SLEEPAPP\_TP2.

 Esperar a que se sincronicen las dependencias Gradle.

 Conectar un dispositivo Android físico o iniciar un emulador.

 Hacer clic en **Run (▶)** para compilar y ejecutar la app.

## 6.4 Base de Datos

 No **requiere configuración externa.**

 La base de datos SQLite se crea automáticamente al ejecutar la aplicación por primera vez mediante la clase DatabaseHelper.java.

 Las tablas incluyen: usuarios, rutinas, alarmas, relajacion y estadisticas.

 Todos los datos se almacenan localmente en el dispositivo.

# 7. Seguridad

 Contraseñas **almacenadas de forma segura** utilizando funciones hash (como MessageDigest en Java).

 Validación **de entradas de usuario** en los formularios de registro e inicio de sesión.

 Restricción **de acceso a datos**: solo el usuario autenticado puede acceder a sus propias rutinas, alarmas o estadísticas.

 Protección **local**: uso de SharedPreferences para almacenar información mínima, evitando datos sensibles sin cifrado.

 En **futuras versiones**, se recomienda integrar autenticación segura en la nube (Firebase/Auth).

# 8. Buenas Prácticas

* **Código estructurado por capas (MVVM)**: separación clara entre interfaz, lógica y datos.
* **Modularización de funcionalidades**: clases independientes para cada módulo (Usuario, Rutina, Alarma, etc.).
* **Uso de constantes y recursos centralizados** para colores, strings y estilos.
* **Tratamiento de errores y validaciones** implementado antes de ejecutar operaciones críticas.
* **Persistencia local con SQLite**, con clases DAO bien definidas.

# 9. Futuras Mejoras

* **Sincronización en la nube con Firebase** para acceso desde múltiples dispositivos.
* **Funcionalidad de edición y eliminación** para rutinas y alarmas.
* **Notificaciones inteligentes** para mejorar la adherencia al sueño saludable.
* **Panel gráfico interactivo** para mostrar tendencias y estadísticas personalizadas.
* **Integración con sensores de sueño o pulseras inteligentes** para análisis más precisos.

# 10. Contacto

Para soporte técnico, sugerencias o colaboración, puedes escribir a:  
📧 **sleepapp.dev@proyecto.edu.pe**

Manual de Instalación y Configuración — SLEEPAPP

**Fecha**: 02/07/2025

**Autor**: Christian Castro Usto

**Versión**: 1.0

# 1. Introducción

Este documento proporciona las instrucciones detalladas para la **instalación, configuración y ejecución de la aplicación móvil SleepApp** en un entorno de desarrollo local. Está dirigido a **desarrolladores Android**, personal técnico y testers que necesiten implementar o mantener la solución en sus dispositivos o entornos de prueba. SleepApp está diseñada para ayudar a los usuarios a **gestionar su rutina diaria, configurar alarmas personalizadas y visualizar estadísticas del sueño**, funcionando completamente sin conexión mediante SQLite.

# 2. Requisitos del Sistema

**2.1 Requisitos de Software**

* Android Studio Flamingo o superior
* Java SDK 11 o superior
* Gradle 7.0 o superior
* Emulador Android o dispositivo físico con Android 8.0+
* **Git** (opcional, para clonar el repositorio)

**2.2 Requisitos de Hardware (mínimos)**

* **CPU:** Procesador de 2 núcleos (Intel i5 / AMD Ryzen 3 o superior)
* **RAM:** 8 GB (se recomienda 16 GB para un rendimiento fluido con Android Studio)
* **Almacenamiento:** 2 GB libres para el proyecto y dependencias
* **Resolución de pantalla:** 1366x768 o superior

# 3. Estructura del Proyecto

# SleepApp/

# │

# ├── manifests/

# │ └── AndroidManifest.xml

# │

# ├── java/

# │ └── com.example.sleepapp/

# │ ├── MainActivity.java ← App.js

# │ ├── MenuPrincipal.java ← Pantalla de inicio

# │ ├── Registrar.java ← Registro.js

# │ ├── LoginActivity.java ← Login.js

# │ ├── RegistrarAlarma.java ← NuevaTarea.js

# │ ├── NotificacionesActivity.java ← Notificaciones

# │ ├── EstadisticasActivity.java ← Vista de estadísticas

# │ ├── GraficoBarras.java ← Gráfico barras

# │ ├── GraficoCircular.java ← Gráfico circular

# │ ├── AlarmReceiver.java ← Alarmas automáticas

# │ ├── ApagarAlarma.java ← Detener alarmas

# │ ├── Usuario.java ← Modelo de usuario

# │ ├── SqlLiteHelper.java ← Configuración de base de datos

# │ └── ... otros componentes

# │

# ├── res/

# │ ├── layout/

# │ │ ├── activity\_apagar\_alarma.xml

# │ │ └── activity\_estadisticas.xml

# │ │ └── activity\_listardatos.

# │ │ └── activity\_login.xml

# │ │ └── activity\_main.xml

# │ │ └── activity\_menu\_principal.xml

# │ │ └── activity\_registrar.xml

# │ │ └── activity\_registrar\_alarma.xml

# │ │ └── activity\_registrar\_rutina.xml

# │ └── drawable/

# │ └── ... imágenes, íconos

# │

# ├── test/

# │ └── com.example.sleepapp/

# │

# └── androidTest/

# └── com.example.sleepapp/

# 4. Instalación del Entorno de Desarrollo

## 4.1 Clonación del Repositorio

## git clone https://github.com/ChrisCU16/SLEEPAPP\_TP2.git

## cd SLEEPAPP\_TP2

**4.2 Configuración del Proyecto en Android Studio**

1. **Abre Android Studio.**
2. En la pantalla de inicio, selecciona **"Open an Existing Project"**.
3. Navega hasta la carpeta clonada (SLEEPAPP\_TP2) y selecciónala.
4. Espera a que Gradle sincronice automáticamente el proyecto. Si no lo hace, ve al menú **File > Sync Project with Gradle Files**.
5. Asegúrate de tener configurado un **emulador Android** o un **dispositivo físico conectado vía USB** para la ejecución.

**4.3 Configuración de la Base de Datos**

* SleepApp utiliza **SQLite** como base de datos local, no necesita configuración externa.
* La clase DatabaseHelper.java se encarga de crear y gestionar las tablas automáticamente al ejecutarse la app.
* No es necesario ejecutar scripts SQL ni configurar archivos .env.

**4.4 Ejecución de la Aplicación**

1. En Android Studio, selecciona un dispositivo/emulador.
2. Haz clic en el botón **Run (▶️)** o presiona **Shift + F10**.
3. La aplicación se instalará y ejecutará automáticamente en el dispositivo seleccionado.

**5. Despliegue en Producción**

**5.1 Aplicación Móvil (APK)**

* Plataforma recomendada para distribución: **Google Play Store** o distribución manual del **APK**.
* Para generar el archivo APK de producción:
  1. En Android Studio, ve a **Build > Build Bundle(s) / APK(s) > Build APK(s)**.
  2. El archivo APK se generará en:  
     SLEEPAPP\_TP2/app/build/outputs/apk/release/
* Antes del despliegue, asegúrate de:
  1. Firmar el APK con una clave segura (usando un **keystore**).
  2. Activar el modo **release** para evitar logs de depuración innecesarios.

**5.2 Base de Datos**

* SleepApp utiliza **SQLite** de forma local. No requiere configuración adicional.
* Los datos del usuario se almacenan en el dispositivo y no necesitan un servidor externo.
* Asegúrate de manejar correctamente el ciclo de vida de la base de datos mediante la clase DatabaseHelper.

**6. Configuración Adicional**

**6.1 Seguridad**

* En producción:
  + Asegúrate de **encriptar contraseñas** antes de almacenarlas (recomendado uso de librerías como BCrypt para Android).
  + No almacenes contraseñas en texto plano.
  + Limita el acceso a recursos sensibles (como rutas internas o videos privados).
  + Implementa control de acceso local en SQLite para proteger los datos del usuario en caso de pérdida de dispositivo.

**6.2 Logs**

* Usa la clase Log de Android solo durante el desarrollo (Log.d, Log.e, etc.).
* En producción, desactiva o elimina logs sensibles.
* Para análisis posterior o debugging en producción, puedes considerar:
  + Firebase Crashlytics (opcional).
  + Exportación manual de logs por parte del usuario.

**7. Verificación de Instalación**

Antes de publicar tu aplicación o entregarla a los usuarios, asegúrate de que:

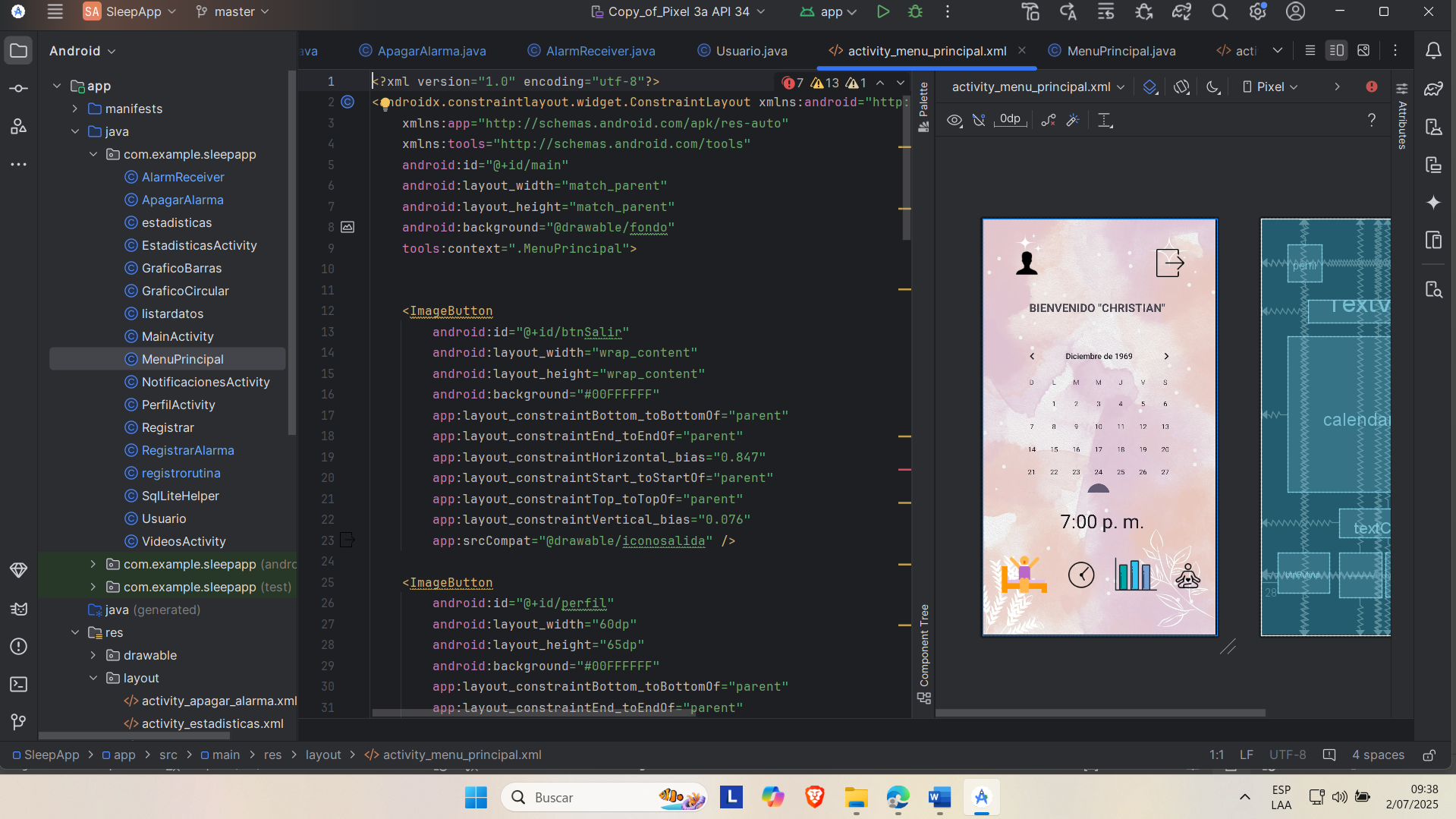
* Se puede **registrar** e **iniciar sesión** correctamente.
* Se pueden **ingresar rutinas** y **configurar alarmas**.
* Se muestra correctamente el **gráfico estadístico** con los datos de sueño.
* Se puede acceder a la **sección de relajación** y visualizar los videos.
* La aplicación **funciona sin conexión** gracias al uso de SQLite.

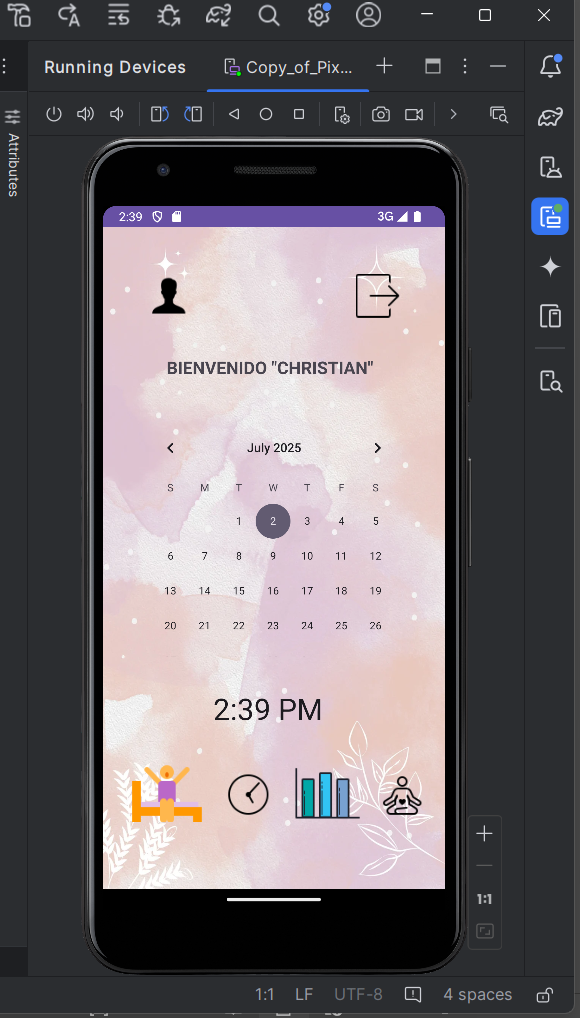
**8. Contacto Técnico**

Para soporte o asistencia técnica respecto a SleepApp, puedes comunicarte con el desarrollador a través del correo:  
📧 **73024009@continental.edu.pe**

**9. Anexos**

* Capturas de pantalla de la interfaz.





* Instrucciones para instalación en dispositivo físico (APK).

https://www.youtube.com/watch?v=eaOB-KS-Qjk&ab\_channel=MoureDevbyBraisMoure

* Estructura de la base de datos SQLite.

Diagrama, Esquemático

Descripción generada automáticamente

* Diagrama de arquitectura del sistema.

Diagrama

Descripción generada automáticamente

* Manual de usuario (ver documento separado)

https://github.com/ChrisCU16/SLEEPAPP\_TP2

Manual del Administrador del Sistema — SLEEPAPP

**Fecha**: 02/07/2025

**Autor**: Christian Castro Usto

**Versión**: 1.0

# 1. Introducción

# Este manual está destinado a los administradores y desarrolladores encargados del mantenimiento de SleepApp, una aplicación móvil que permite a los usuarios gestionar su rutina diaria, configurar alarmas personalizadas, visualizar estadísticas de sueño y acceder a contenido de relajación. Su propósito es describir las funciones administrativas, el control de usuarios, mantenimiento de la base de datos local (SQLite) y procedimientos recomendados para respaldo y monitoreo.

# 2. Acceso Administrativo

SleepApp no cuenta con un panel visual de administración. La gestión se realiza directamente sobre la base de datos local SQLite, ubicada en el almacenamiento interno del dispositivo Android, utilizando herramientas como Android Device File Explorer o scripts SQL dentro del código fuente. En futuras versiones, se planea incluir sincronización con la nube y un dashboard administrativo.

# 3. Gestión de Usuarios

## 3.1 Visualizar Usuarios

* Para consultar los registros de usuarios desde SQLite:

SELECT \* FROM usuarios;

## 3.2 Eliminar Usuario

DELETE FROM usuarios WHERE id\_usuario = X;

E Esto también eliminará registros asociados en tablas como rutina, alarma, relajación y estadística si están implementadas las restricciones ON DELETE CASCADE.

## 3.3 Reiniciar Contraseña (Manual)

* Modificar directamente la contraseña desde SQLite:

UPDATE usuarios SET contraseña = 'nueva\_contraseña' WHERE id\_usuario = X;

* **Nota:** No se utiliza encriptación por defecto, pero se recomienda implementarla en versiones futuras.

# 4. Gestión de Datos de la Rutina

## 4.1 Consultar Rutinas

SELECT \* FROM rutina WHERE id\_usuario = X;

## 4.2 Editar Rutina

## Modificar manualmente los valores desde SQLite si es necesario:

UPDATE rutina SET hora\_trabajo = 5, hora\_actividades\_recreativas = 2 WHERE id\_rutina = X;

# 5. Gestión de Alarmas

## 5.1 Ver Alarmas Activas

SELECT \* FROM alarma WHERE estado = 'activa';

## 5.2 Eliminar Alarmas

DELETE FROM alarma WHERE id\_usuario = X;

# 6. Respaldo y Recuperación de Base de Datos

**6.1 Respaldo**

Copiar el archivo .db o .sqlite ubicado en la carpeta de datos internos del dispositivo (por ejemplo: /data/data/com.sleepapp/databases/sleepapp.db).

Se recomienda usar **Android Studio Device File Explorer** o adb pull:

adb pull /data/data/com.sleepapp/databases/sleepapp.db ./backup/sleepapp\_backup.db

## 6.2 Restauración

adb push ./backup/sleepapp\_backup.db /data/data/com.sleepapp/databases/sleepapp.db

# 7. Monitoreo y Logs

# 7.1 Logs de Aplicación

# Se recomienda utilizar Logcat desde Android Studio para registrar eventos, errores y excepciones durante la ejecución

# Log.e("SleepApp", "Error al guardar rutina", e);

# 7.2 Revisión Manual

# Los errores se pueden identificar revisando el log de Android Studio mientras la app está corriendo.

# 8. Seguridad Administrativa

 Evitar publicar archivos .db o con datos sensibles en repositorios públicos.

 Proteger el acceso físico al dispositivo.

 Restringir el acceso al código fuente y a herramientas de depuración a personal autorizado.

 Agregar cifrado a las contraseñas en futuras versiones.

# 9. Automatización Recomendada

 Crear tareas automáticas (dentro del app) para limpiar datos antiguos como alarmas inactivas o estadísticas pasadas.

 Incluir respaldo automático semanal del archivo SQLite al almacenamiento externo (requiere permisos).

# 10. Cambios Previstos para el Rol de Administrador

 Implementación de sincronización en la nube.

 Dashboard de monitoreo estadístico por usuario.

 Gestión de accesos, recuperación de datos remotos y envío de notificaciones.

Manual de Mantenimiento — SLEEPAPP

**Fecha**: 02/07/2025

**Autor**: Christian Castro Usto

**Versión**: 1.0

# 1. Introducción

Este manual describe los procedimientos de mantenimiento preventivo y correctivo del sistema **SleepApp**, una aplicación móvil para la gestión del sueño y rutinas diarias. Su objetivo es garantizar la continuidad operativa, el rendimiento óptimo y la seguridad del sistema. Está dirigido al equipo técnico encargado del soporte, desarrollo y mantenimiento de la aplicación.

# 2. Objetivos del Mantenimiento

* Prevenir fallos antes de que ocurran (mantenimiento preventivo).
* Corregir errores detectados durante el uso de la app (mantenimiento correctivo).
* Asegurar la integridad de la información del usuario y la disponibilidad de funciones clave (registro, rutinas, alarmas, estadísticas).

# 3. Mantenimiento Preventivo

## 3.1 Tareas Semanales

 Revisar los registros de Logcat para detectar errores y excepciones frecuentes.

 Validar el correcto funcionamiento de los formularios de rutina y alarma.

 Comprobar el uso de almacenamiento del dispositivo (espacio ocupado por la base de datos local).

 Verificar integridad del archivo sleepapp.db ubicado en la carpeta de datos internos.

## 3.2 Tareas Mensuales

* Limpiar datos innecesarios (alarmas vencidas, rutinas incompletas):

DELETE FROM rutina WHERE fecha < date('now', '-30 days');

* Actualizar dependencias del proyecto si se usa Gradle (Android Studio):

./gradlew dependencies

* Ejecutar pruebas básicas en dispositivos con distintas versiones de Android (compatibilidad).

## 3.3 Tareas Trimestrales

* Revisar el rendimiento de la app (uso de RAM, batería) con Android Profiler.
* Aplicar pruebas de estrés para múltiples alarmas configuradas y datos simultáneos.
* Verificar posibles vulnerabilidades de dependencias (lint, npm audit si se usa código híbrido o bibliotecas externas).

# 4. Mantenimiento Correctivo

## 4.1 Gestión de Errores Comunes

* App **se cierra inesperadamente**: revisar logs de Logcat, buscar errores de null o acceso a DB.
* Alarma **no suena**: verificar permisos de notificación y uso del AlarmManager.
* Datos **no se guardan**: comprobar estado de la base de datos y manejo de excepciones en SQLiteOpenHelper.

## 4.2 Procedimiento de Corrección

1. Identificar el error mediante el reporte del usuario y Logcat.
2. Reproducir el error en un entorno de pruebas.
3. Corregir el código fuente o configuración.
4. Validar la solución y realizar pruebas.
5. Registrar el incidente en documentación interna o bitácora.

# 5. Respaldo de Información

## 5.1 Respaldo Manual

* Ubicar el archivo de base de datos en:  
  /data/data/com.sleepapp/databases/sleepapp.db
* Usar adb para extraerlo:

adb pull /data/data/com.sleepapp/databases/sleepapp.db ./respaldo/

## 5.2 Respaldo Programado

# Opcionalmente, se puede implementar en futuras versiones una función interna que copie automáticamente el archivo .db al almacenamiento externo semanalmente (requiere permisos WRITE\_EXTERNAL\_STORAGE).

# 6. Actualización del Sistema

**6.1 Aplicación**

* Actualizar dependencias desde Android Studio (build.gradle).
* Subir la nueva versión del APK a dispositivos de prueba.
* Verificar migraciones de base de datos si hay cambios en el esquema (se recomienda control de versión de DB).

**6.2 Base de Datos**

* Ejecutar scripts SQL para modificaciones en tablas.
* Validar compatibilidad de los datos previos con el nuevo esquema.

# 7. Documentación de Cambios

Registrar cada cambio importante en el archivo CHANGELOG.md bajo formato:

## [1.2.0] - 2025-06-18

### Agregado

- Módulo de configuración de alarmas personalizadas.

- Validación offline para el registro de rutinas.

### Corregido

- Error al guardar cambios en rutinas sin conexión.

# 8. Herramientas Recomendadas

* **Android Studio**: desarrollo y depuración.
* **SQLite Browser**: revisión manual de base de datos.
* **ADB**: acceso a archivos internos del dispositivo.
* **Logcat**: seguimiento de errores y rendimiento.
* **Postman** (futuro): si se incorpora backend remoto o sincronización.

# 9. Contacto de Soporte

# Para reportes técnicos o mantenimiento avanzado del sistema:

# 📧 73024009@continental.edu.pe

# 📞 +51 966384355

# 10. Conclusión

Este manual proporciona un conjunto de prácticas esenciales para mantener la aplicación **SleepApp** operativa, funcional y segura. La aplicación sistemática de estos procedimientos es clave para su sostenibilidad, confiabilidad y mejora continua.

Manual de Pruebas — SLEEPAPP

**Fecha**: 02/07/2025

**Autor**: Christian Castro Usto

**Versión**: 1.0

# 1. Introducción

Este manual describe el proceso de verificación y validación de la aplicación **SleepApp**, incluyendo los tipos de pruebas realizadas, los casos de prueba definidos, herramientas utilizadas y criterios de aceptación establecidos. Está dirigido al equipo de aseguramiento de la calidad (QA) y a los desarrolladores responsables de validar que el sistema funcione correctamente antes de su despliegue.

.

# 2. Objetivos

* Verificar que cada funcionalidad cumpla con los requisitos funcionales y no funcionales definidos.
* Identificar errores antes del lanzamiento y corregirlos a tiempo.
* Asegurar que la aplicación sea estable, usable y segura en diferentes entornos de ejecución.

# 3. Tipos de Pruebas Aplicadas

**3.1 Pruebas Unitarias**

* Validan clases y funciones individuales del backend (SQLite, DAO) y del frontend (actividades Android).
* Herramientas: **JUnit**, **Mockito**.

**3.2 Pruebas de Integración**

* Validan la interacción entre componentes como la vista (Activity) y la base de datos SQLite.

**3.3 Pruebas Funcionales**

* Verifican funcionalidades clave como registro, ingreso de rutina, configuración de alarmas, y reproducción de videos.

**3.4 Pruebas de Regresión**

* Se ejecutan al finalizar cada sprint para asegurar que nuevas funcionalidades no rompan las anteriores..

**3.5 Pruebas Manuales (Caja Negra)**

* Realizadas desde la perspectiva del usuario final usando dispositivos Android reales.
* Se valida la experiencia completa sin conocer la lógica interna.

# 4. Ambiente de Pruebas

|  |  |
| --- | --- |
| Componente | Versión |
| Android Studio | Flamingo+ |
| Java SDK | 11 |
| SQLite | nativo Android |
| Dispositivo | Android 10–13 |
| Emulator | Pixel 4 (API 30) |

# 5. Casos de Prueba

**5.1 Registro de Usuario**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Descripción | Entrada | Resultado Esperado | Estado |
| TC01 | Registro exitoso | Correo, contraseña válidos | Redirección al login | Aceptado |
| TC02 | Correo ya registrado | Correo existente | Mensaje de "Usuario ya registrado" | Aceptado |
| TC03 | Campos vacíos | Ninguno | Mensaje de "Campos requeridos" | Aceptado |

**5.2 Inicio de Sesión**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Descripción | Entrada | Resultado Esperado | Estado |
| TC04 | Login exitoso | Credenciales válidas | Acceso al menú principal | Aceptado |
| TC05 | Login fallido | Contraseña incorrecta | Mensaje de error de autenticación | Aceptado |

**5.3 Gestión de Rutinas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Descripción | Entrada | Resultado Esperado | Estado |
| TC06 | Registrar rutina diaria | Horarios de estudio, trabajo, etc. | Datos guardados y visibles | Aceptado |
| TC07 | Validar tiempos inconsistentes | Horario de descanso negativo | Mensaje de validación | Aceptado |
| TC08 | Modificar rutina | Cambios aplicados | Rutina actualizada correctamente | Aceptado |

# 5.4 Alarmas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Descripción | Entrada | Resultado Esperado | Estado |
| TC09 | Configurar nueva alarma | Hora, tono | Alarma almacenada en local | Aceptado |
| TC10 | Funcionamiento offline | Sin internet | Alarma sigue funcionando | Aceptado |

# 6. Resultados

Todas las pruebas realizadas al finalizar el Sprint 2 y durante el Sprint 3 han sido satisfactorias. Las funcionalidades principales como el acceso, registro de rutina y configuración de alarmas están operativas y estables en dispositivos de prueba.

# 7. Criterios de Aceptación

* El sistema permite registrar, editar y visualizar rutinas diarias.
* Las alarmas personalizadas funcionan correctamente, incluso sin conexión.
* El acceso a la aplicación (login/registro) funciona con validaciones.
* El sistema no presenta errores críticos durante la navegación normal.

# 8. Herramientas Utilizadas

* **Android Studio**: entorno principal de pruebas.
* **JUnit**: pruebas unitarias de clases lógicas y controladores.
* **Mockito**: pruebas con objetos simulados (mocked DAOs).
* **ADB + Logcat**: para detección de errores en ejecución.
* **Dispositivos físicos + emulador**: pruebas manuales.

# 9. Recomendaciones Futuras

* Integrar pruebas de interfaz automatizadas con **Espresso**.
* Implementar pruebas E2E con múltiples flujos de usuario.
* Validar comportamiento en dispositivos con batería optimizada y modo oscuro.
* Agregar testing para cambios en configuración del sistema (idioma, brillo, etc.).

# 10. Contacto QA

Para reportes de errores o coordinación de pruebas:

📧 73024009@continental.edu.pe

📞 +51 966384355

Registro de Mantenimiento y Plan de Mejoras Futuras — SleepApp

**Fecha**: 02/07/2025

**Autor**: Christian Castro Usto

**Versión**: 1.0

# 1. Introducción

Este documento consolida el historial de mantenimiento de la aplicación **SleepApp** y presenta un plan estructurado de mejoras futuras. Su propósito es garantizar la sostenibilidad del sistema, documentar acciones técnicas relevantes y trazar la hoja de ruta para su evolución con base en los requerimientos del usuario y el crecimiento funcional del sistema.

# 2. Registro de Mantenimiento

## 2.1 Historial de Mantenimientos Ejecutados

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Tipo | Descripción | Responsable |
| 2025-06-10 | Preventivo | Limpieza de rutinas antiguas y verificación de base de datos | Equipo Técnico |
| 2025-06-11 | Correctivo | Solución a error en la validación de usuario offline | Backend Dev |
| 2025-06-12 | Preventivo | Actualización de dependencias y librerías Android | QA / DevOps |
| 2025-06-13 | Correctivo | Restauración de SQLite tras pérdida de registros locales | Administrador App |
| 2025-06-14 | Preventivo | Revisión de logs, consumo de memoria y pruebas de rendimiento | DevOps |

# 3. Diagnóstico General

Tras las primeras iteraciones de desarrollo y despliegue, se identifican los siguientes puntos:

## 3.1 Fortalezas

* Arquitectura modular orientada a capas.
* Funcionalidad offline implementada con SQLite.
* Flujo de usuario claro e intuitivo.
* Diseño enfocado en mejorar hábitos de sueño.

## 3.2 Oportunidades de Mejora

* Falta de notificaciones personalizadas y recordatorios.
* Visualización limitada de estadísticas semanales.
* Interfaz no adaptada a todos los tamaños de pantalla.
* Ausencia de sistema de respaldo en la nube.
* Falta de perfil de usuario o historial completo de sueño.

# 4. Plan de Mejoras Futuras

## 4.1 Funcionalidades Prioritarias

|  |  |  |  |
| --- | --- | --- | --- |
| Mejora | Descripción | Prioridad | Versión Estimada |
| Historial de sueño | Registro semanal y mensual del tiempo real de descanso | Alta | v1.1 |
| Recomendaciones inteligentes | Sugerencias basadas en rutinas y horas dormidas | Alta | v1.2 |
| Notificaciones personalizadas | Alarmas con avisos según comportamiento del usuario | Alta | v1.2 |
| Sincronización en la nube | Respaldo automático y acceso desde otros dispositivos | Alta | v1.3 |
| Interfaz responsiva | Mejoras visuales para dispositivos con diferentes resoluciones | Media | v1.3 |
| Perfil del usuario | Datos del sueño, avatar y configuraciones personalizadas | Media | v1.4 |
| Modo oscuro | Compatibilidad visual para uso nocturno | Baja | v1.4 |

## 4.2 Acciones de Mantenimiento Programado

* **Backups automáticos** de la base de datos SQLite (mensual o tras actualización).
* **Revisión de logs** del sistema y consumo de recursos (mensual).
* **Actualización de librerías** y dependencias Android (trimestral).
* **Revisión del rendimiento general** y usabilidad de la app (trimestral).

# 5. Seguimiento y Actualización

* Este documento será actualizado al finalizar cada sprint o al liberar una nueva versión funcional.
* Se deberá mantener un registro de cambios (CHANGELOG) con mejoras aplicadas.
* Los ajustes en la hoja de ruta deben ser consensuados por el equipo técnico y validación del QA.

# 6. Contacto Responsable

# 📧 73024009@continental.edu.pe

# 📞 +51 966384355

# 7. Conclusión

El mantenimiento proactivo y la planificación de mejoras estratégicas garantizan que SleepApp evolucione con calidad, eficiencia y orientación al bienestar del usuario. Este documento servirá como referencia para las decisiones técnicas y el desarrollo de nuevas funcionalidades centradas en hábitos de sueño saludables.