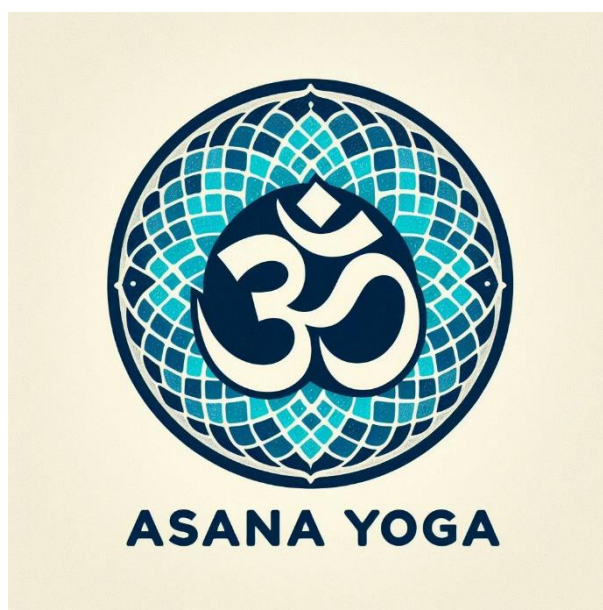


PROYECTO

AsanaYoga Enterprise

AsanaYoga



CICLO FORMATIVO DE GRADO SUPERIOR

Desarrollo de Aplicaciones Multiplataforma



I.E.S. «Venancio Blanco» SALAMANCA

AUTOR

Javier Mesonero Moro

Licencia.

Esta obra está bajo una licencia Reconocimiento-Compartir bajo la misma licencia 3.0 España de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

Resumen.

AsanaYoga Enterprise quiere crear una aplicación dedicada al mundo yoguístico, con la intención de dar a conocer al mundo todo el conocimiento relativo al yoga moderno y todas sus vertientes desde el punto de vista de un estudiante de Hatha Yoga (Yoga de la consciencia o autoconocimiento), y con esta intención nace AsanaYoga.

AsanaYoga es una aplicación diseñada específicamente para aquellos apasionados del yoga, con el ambicioso propósito de consolidar en un único espacio toda la información relevante acerca de las asanas, esas posturas que son el corazón de esta práctica ancestral. La esencia de la propuesta es clara y persuasiva: se busca crear un recurso integral y accesible que permita a los practicantes profundizar no solo en su conocimiento de las posturas, sino también en la experiencia colectiva de la comunidad yogui.

Un problema recurrente que se manifiesta en el vasto panorama de aplicaciones disponibles es la notable falta de plataformas que aborden de manera exhaustiva todas las variantes posibles de las posturas. Aquí es donde se destaca. La aplicación permitirá a los usuarios acceder a información detallada sobre cada postura: su nombre, el cómo realizarlas, las variantes que existen y las partes del cuerpo que involucran, a través de un listado en su pantalla principal, transformando así la práctica del yoga en una experiencia más rica y esclarecedora.

La demanda para esta aplicación es, sin duda, considerablemente amplia, abarcando desde aquellos yoguis que recién inician su camino en esta disciplina hasta instructores y centros de yoga que buscan mejorar sus métodos de enseñanza. En este sentido, se presenta no solo como una herramienta, sino como un catalizador para el crecimiento y la educación en la práctica del yoga.

Entre las características más sobresalientes de la aplicación se encuentra un ingenioso sistema de votación, denominado **shantis**, que permitirá clasificar las posturas de acuerdo con la evaluación y preferencia de los usuarios. Esta funcionalidad no solo facilitará la identificación de las posturas más valoradas por la comunidad, sino que también ayudará a moderar y validar el contenido nuevo que se integre a la plataforma.

Además, se implementará un sofisticado buscador que permitirá filtrar los asanas por una variedad de categorías, desde el tipo de postura (como las invertidas o de equilibrio) hasta las partes del cuerpo que se ven involucradas, ofreciendo una experiencia de navegación intuitiva y enriquecedora.

De cara al futuro, en una etapa posterior de desarrollo, se prevé la incorporación de innovaciones aún más atractivas, como la opción de crear clases personalizadas. Esto ampliará las posibilidades para los usuarios, adaptándose a las necesidades individuales y permitiendo que cada practicante trace su propio camino en la práctica del yoga.

En conclusión, **AsanaYoga** no solo promete transformar la manera en que los yoguis interactúan con el conocimiento y la práctica del yoga, sino que aspira a convertirse en un recurso esencial para todos aquellos que deseen sumergirse más profundamente en esta hermosa y enriquecedora disciplina.

Índice de contenido.

Licencia.....	2
Resumen.....	3
Índice de contenido.....	4
Índice de figuras.....	6
Necesidades del sector productivo y de la organización de la empresa.	7
Clasificación de Empresas Relacionadas con el Proyecto AsanaYoga.	7
a. Startups de Bienestar Digital.	7
b. Centros de Yoga y Fitness con Presencia Digital.	8
c. Empresas Tecnológicas Especializadas en Aplicaciones de Bienestar.....	8
Oportunidades de Negocio para AsanaYoga.	9
a. Demanda Creciente de Bienestar Digital.	9
b. Personalización y Creación de Comunidades.....	9
c. Expansión Internacional.	9
Obligaciones Fiscales, Laborales y de Prevención de Riesgos.....	9
a. Obligaciones Fiscales.	9
b. Obligaciones Laborales.	10
c. Prevención de Riesgos Laborales.....	10
d. Ayudas y Subvenciones.	10
e. Conclusión.....	10
Diseño del proyecto.	11
Definición o análisis de sistemas o de requisitos.....	11
Objetivos del Proyecto.....	11
Requisitos de Información.....	12
Requisitos Funcionales.....	13
Diseño del sistema.	23
Diseño de datos.....	23
Diseño de arquitectura.....	28
Diseño de interfaz.....	31
Diseño de componentes.	42
Diseño de despliegue.....	44
Implementación.	45
Pruebas.	59

Planificación del desarrollo y puesta en marcha del proyecto.	61
Definición de procedimientos de control y evaluación de la ejecución de proyectos.	66
Referencias y bibliografía.	69

Índice de figuras.

Figura 1: Registrar un usuario	14
Figura 2: Caso de uso Inicio de sesión	15
Figura 3: Caso de uso Buscar Posturas por Categoría	16
Figura 4: Caso de uso Visualizar Información de una Postura	17
Figura 5: Caso de uso Votar una Postura (Shantis)	18
Figura 6: Caso de uso Votar una Postura (Shantis)	19
Figura 7: Caso de uso Gestión de Perfil de Usuario	20
Figura 8: Modelo Entidad-Relación	23
Figura 9: Pantalla de Carga	31
Figura 10: Pantalla de Inicio de Sesión	32
Figura 11: Pantalla Principal	33
Figura 12: Pantalla de Búsquedas avanzadas	34
Figura 13: Pantalla Agregar Asana	35
Figura 14: Pantalla Asana Detalle	36
Figura 15: Pantalla de Crear Rutina	37
Figura 16: Pantalla de Perfil	38
Figura 17: Pantalla de Editar Perfil	39
Figura 18: Pantalla Rutina Personalizada	41
Figura 19: Diagrama de componentes	43
Figura 20: Diagrama de despliegue	44
Figura 21: Código Inicialización BBDD	47
Figura 22: Código Inicio Sesión	48
Figura 23: Código Registro Usuario	48
Figura 24: Código Insertar Usuario	49
Figura 25: Métodos de comprobación de usuario	49
Figura 26: Código para editar datos del usuario	50
Figura 27: Código para actualizr información en la BBDD	50
Figura 28: Código Creación de Asana	51
Figura 29: Código Insertar Asana en BBDD	51
Figura 30: Código Mostrar Asanas	52
Figura 31: Código Consulta Mostrar Asanas	52
Figura 32: Código Borrar asana	52
Figura 33: Código Borrar Asana BBDD	53
Figura 34: Código Votar Asana	53
Figura 35: Código Insertar Voto en BBDD	53
Figura 36: Código Contador de votos	54
Figura 37: Código Aplicar Filtros	54
Figura 38: Código Filtrar por ranking de votos	55
Figura 39: Código Crear Rutina	55
Figura 40: Código Insertar Rutina en BBDD	56
Figura 41: Código Mostrar Rutina	56
Figura 42: Código consulta Mostrar Asanas de Rutina	57
Figura 43: Código Eliminar Rutina	57
Figura 44: Código Eliminar Rutina de BBDD	58
Figura 45: Diagrama de Grantt	62

Necesidades del sector productivo y de la organización de la empresa.

Este estudio tiene como objetivo analizar las características organizativas y productivas del sector que está relacionado con la aplicación, para encuadrar el proyecto en el contexto del desarrollo de aplicaciones de bienestar y yoga.

Para ello he dividido este análisis de necesidades sobre el sector producto y la organización de la empresa de la siguiente manera:

Clasificación de Empresas Relacionadas con el Proyecto AsanaYoga.

El proyecto **AsanaYoga** pertenece al sector de las aplicaciones móviles de bienestar, particularmente aquellas dedicadas a la práctica del yoga y la salud física. Para analizar este sector, podemos clasificar las empresas relacionadas en tres categorías principales:

a. Startups de Bienestar Digital.

Este tipo de empresas emergen con el objetivo de ofrecer soluciones innovadoras en el ámbito de la salud y el bienestar mediante aplicaciones móviles. Ejemplos conocidos incluyen aplicaciones como **Blue Bamboo Yoga** y **Down Dog**.

- **Características organizativas:** son empresas pequeñas o medianas, con una estructura organizativa ágil. A menudo, trabajan con metodologías ágiles (Scrum, Kanban) para el desarrollo de software.
- **Servicios ofrecidos:** suscripciones a servicios de meditación, yoga y bienestar general. Ofrecen contenido premium, clases en línea y personalización de rutinas. La tarifa que suelen tener este tipo de aplicaciones suele variar entre los 3.99€ y 9.99€ al mes. Aunque también pueden venderse como licencias de un único pago, donde el precio varía dependiendo del proyecto.
- **Estructura organizativa:**
 - **Departamento de Desarrollo:** enfocado en el diseño y desarrollo de la app, manteniendo la base de datos, características como los filtros de búsqueda y el sistema de votación.
 - **Marketing:** encargado de atraer y retener usuarios mediante campañas en redes sociales, SEO y ASO (App Store Optimization).
 - **Atención al Cliente:** resolución de problemas técnicos y gestión de feedback de usuarios.
 - **UX/UI:** enfocados en mejorar la experiencia de usuario, asegurando que la app sea fácil de navegar y visualmente atractiva.
- **Necesidades más demandadas:**
 - Creación de contenido atractivo y relevante.
 - Mantener la retención de usuarios.
 - Innovación constante en la personalización de los servicios para cada usuario.
 - Una opción más accesible para las personas que trabajen y no puedan acudir a clases presenciales de yoga.

b. Centros de Yoga y Fitness con Presencia Digital.

Estos centros se están adaptando a la digitalización, creando plataformas que permiten a los usuarios acceder a clases virtuales. Ejemplos son **Glo**, **CorePower Yoga**, y **YogaWorks**.

- **Características organizativas:** empresas medianas con un fuerte componente presencial, que han ampliado su oferta a plataformas digitales.
- **Servicios ofrecidos:** clases de yoga en línea, tanto en directo como pregrabadas, seguimiento personalizado y recomendaciones basadas en el progreso del usuario.
- **Estructura organizativa:**
 - **Instructores de Yoga:** parte clave del contenido ofrecido en la aplicación, proporcionando clases y guías especializadas.
 - **Equipo Técnico:** desarrolladores y diseñadores responsables de la app.
 - **Marketing:** promoción de clases virtuales, suscripciones y eventos en línea.
 - **Atención al Cliente:** gestión de inscripciones, dudas técnicas y problemas con las suscripciones.
- **Necesidades más demandadas:**
 - Optimización de la experiencia del usuario para mejorar el acceso a clases.
 - Innovación en las ofertas de clases en línea y personalización.

c. Empresas Tecnológicas Especializadas en Aplicaciones de Bienestar.

Empresas que desarrollan aplicaciones o plataformas dedicadas al bienestar físico y han querido crear un subdepartamento especializado en yoga, taichí y este tipo de actividades deportivas, como **Fitbit**, **Nike Training Club**, y **Strava**. Su enfoque incluye tanto software como hardware (por ejemplo, dispositivos portátiles para medir el rendimiento físico).

- **Características organizativas:** estructuras grandes con múltiples departamentos que cubren investigación, desarrollo, marketing, y soporte técnico.
- **Servicios ofrecidos:** aplicaciones de fitness, yoga y meditación, con integración de dispositivos y recomendaciones basadas en los datos del usuario.
- **Estructura organizativa:**
 - **Investigación y Desarrollo:** innovación en nuevas características para mejorar el rendimiento y personalización.
 - **Departamento de Datos:** análisis de grandes volúmenes de datos de los usuarios para proporcionar insights y recomendaciones personalizadas.
 - **Marketing y Publicidad:** promoción de la aplicación y servicios a nivel global.
 - **Soporte Técnico:** resolución de problemas de hardware y software.

- **Necesidades más demandadas:**
 - Integración de tecnología de inteligencia artificial y machine learning para personalizar la experiencia del usuario.
 - Captación de datos para mejorar las recomendaciones y el rendimiento de las apps.

Oportunidades de Negocio para AsanaYoga.

El mercado de aplicaciones de bienestar y yoga ofrece una serie de oportunidades interesantes para un proyecto como **AsanaYoga**:

a. Demanda Creciente de Bienestar Digital.

Cada vez más personas buscan herramientas digitales para cuidar su salud mental y física. Las aplicaciones de yoga han experimentado un crecimiento notable, especialmente a raíz de la pandemia, donde la práctica en casa se volvió común.

- **Oportunidades:** AsanaYoga puede capitalizar esta tendencia ofreciendo un producto que no solo sirva para aprender y practicar yoga, sino que también cree una comunidad en la que los propios usuarios compartan conocimientos.

b. Personalización y Creación de Comunidades.

La posibilidad de personalizar tus propias variaciones de posturas y de clases de yoga (en una futura etapa de desarrollo) y acceder a recomendaciones basadas en el nivel del usuario son funciones valoradas por los usuarios. Este conjunto de implementaciones puede generar una gran ventaja competitiva.

- **Oportunidades:** monetizar a través de suscripciones premium que ofrezcan clases personalizadas, contenido exclusivo, o incluso colaboraciones con instructores reconocidos. Aunque la idea principal sería un sistema de anuncios muy permisivo, para que la experiencia del usuario sea enriquecedora y no se sature con tantos anuncios.

c. Expansión Internacional.

Las aplicaciones de bienestar suelen tener un público global. El yoga, en particular, tiene una comunidad internacional que busca practicar en distintos idiomas y culturas.

- **Oportunidades:** AsanaYoga puede ofrecer su contenido en varios idiomas para atraer a una audiencia internacional, utilizando la base de datos de asanas como un recurso universal.

Obligaciones Fiscales, Laborales y de Prevención de Riesgos.

a. Obligaciones Fiscales.

Para operar legalmente, las empresas que desarrollan y comercializan aplicaciones deben cumplir con las siguientes obligaciones fiscales:

- **IVA sobre servicios digitales:** en muchos países, los ingresos generados por servicios digitales están sujetos al Impuesto al Valor Agregado (IVA). Esto implica que AsanaYoga, al monetizar sus servicios, debe gestionar este impuesto.

- **Declaración de Ingresos:** los ingresos obtenidos de las tiendas de aplicaciones (Google Play, App Store) deben ser declarados a las autoridades fiscales.

b. Obligaciones Laborales.

Si se contrata personal para el desarrollo y mantenimiento de AsanaYoga, la empresa deberá cumplir con las normativas laborales:

- **Contratos de trabajo:** el personal técnico (desarrolladores, diseñadores, etc.) debe tener contratos adecuados.
- **Seguridad Social:** pago de contribuciones de seguridad social y otras prestaciones según la legislación local.

c. Prevención de Riesgos Laborales.

Aunque el desarrollo de software no conlleva grandes riesgos físicos, sí existen riesgos relacionados con la salud mental y el bienestar del equipo:

- **Prevención del estrés laboral y burnout:** implementar horarios flexibles, pausas activas y una cultura de trabajo saludable.
- **Riesgos ergonómicos:** proporcionar equipos adecuados para evitar problemas físicos relacionados con largas horas frente a la computadora.

d. Ayudas y Subvenciones.

Existen múltiples programas de ayuda para startups tecnológicas, especialmente aquellas enfocadas en el sector de la salud y el bienestar. En Europa, iniciativas como **Horizon Europe** o **NextGenerationEU** podrían ser fuentes de financiamiento para AsanaYoga. Estas subvenciones podrían cubrir:

- Investigación y desarrollo.
- Implementación de nuevas tecnologías (como inteligencia artificial para la personalización).
- Expansión a nuevos mercados.

e. Conclusión.

El proyecto **AsanaYoga** tiene un fuerte potencial en el sector de bienestar digital, especialmente debido a la creciente demanda de herramientas que permitan practicar yoga en cualquier lugar. Con una correcta estructura organizativa, un enfoque en la personalización, y una comunidad activa de usuarios, AsanaYoga puede diferenciarse de sus competidores por su enfoque en evitar bombardear a los usuarios con anuncios. No obstante, será necesario cumplir con las obligaciones fiscales y laborales, y aprovechar las ayudas disponibles para garantizar su éxito en el mercado.

Diseño del proyecto.

Las empresas en Salamanca, en su mayoría, se dedican al turismo, la hostelería, y otros servicios vinculados a la vida universitaria. Sin embargo, no es raro encontrarse con pequeñas startups tecnológicas que, en los últimos años, están tratando de abrirse paso en un mercado más competitivo. Este auge de nuevas empresas tecnológicas es una señal del cambio que está atravesando la ciudad, buscando aprovechar el talento joven que año tras año se gradúa en sus aulas. Aun así, el crecimiento es lento, y este cambio no está exento de desafíos.

Por otro lado, el nivel de vida en Salamanca es una cuestión intrigante. Aunque se podría pensar que una ciudad universitaria como esta gozaría de una economía vibrante, la realidad es que el nivel adquisitivo medio no es especialmente elevado si se compara con otras ciudades españolas. No obstante, esta situación también trae consigo una ventaja: el costo de vida es significativamente más bajo, lo que podría considerarse un alivio para aquellos que buscan emprender. Este hecho es particularmente relevante para proyectos como AsanaYoga, que pueden beneficiarse de este entorno donde los costos operativos no son tan elevados como en otras ciudades, y donde existe una creciente tendencia hacia lo digital y el bienestar personal, impulsada, en parte, por la búsqueda de equilibrio en la vida moderna.

Una vez aclarada la situación en la que se encuentra Salamanca con respecto al nivel de vida y demás, pasamos con el apartado principal, que lo he querido agrupar en pequeños apartados para una mayor claridad:

Definición o análisis de sistemas o de requisitos.

Objetivos del Proyecto.

A continuación, se detallan los objetivos principales del proyecto **AsanaYoga**:

Objetivo	Descripción
1. Crear una comunidad de aficionados al yoga	El proyecto busca establecer una comunidad donde los usuarios puedan compartir experiencias, conocimientos y opiniones sobre las posturas de yoga (asanas). Esto fomenta un entorno colaborativo en el que los usuarios aprenden y mejoran sus prácticas de yoga.
2. Crear un catálogo completo de posturas de yoga	La aplicación ofrecerá un repositorio exhaustivo de asanas, que incluirá su descripción, variantes, dificultad y las partes del cuerpo involucradas. Esto será útil tanto para principiantes como para usuarios avanzados que buscan mejorar su técnica o explorar nuevas posturas.
3. Facilitar la búsqueda de posturas por categorías	Una de las funcionalidades principales es un sistema de búsqueda avanzada que permite a los usuarios filtrar posturas por categorías como el tipo de asana, nivel de dificultad, o las partes del cuerpo implicadas. Esto ayudará a los usuarios a encontrar de manera eficiente las posturas que se ajusten a sus necesidades.
4. Permitir la personalización de prácticas y rutinas	Los usuarios podrán crear y personalizar sus propias rutinas de yoga, ajustando las asanas según sus preferencias y objetivos personales, lo que brindará una experiencia única y adaptada a cada usuario.
5. Implementar un sistema de votación (shantis)	Los usuarios tendrán la posibilidad de votar y valorar las posturas a través de un sistema de "shantis". Esto generará un ranking de popularidad de las posturas, ayudando a los usuarios a descubrir las asanas más valoradas por la comunidad.

Requisitos de Información.

Los siguientes requisitos de información detallan los datos que serán gestionados por la aplicación **AsanaYoga**, especificando qué campos almacenarán y con qué objetivos y requisitos funcionales se relacionan.

Requisito de Información	Campos a Almacenar	Objetivo Relacionado	Requisito Funcional Relacionado
Usuarios	Nombre de usuario, correo electrónico, contraseña, nivel de experiencia, asanas preferidas, votos emitidos	Crear comunidad de aficionados al yoga	Registro y gestión de usuarios, votación de posturas
Posturas (Asanas)	Nombre de la postura, descripción, variantes, nivel de dificultad, partes del cuerpo involucradas, imágenes, vídeos	Crear catálogo de posturas de yoga	Búsqueda avanzada, visualización de asanas
Partes del cuerpo	Nombre de la parte del cuerpo (erg., espalda, piernas), asanas relacionadas	Facilitar búsqueda por categorías	Filtrar posturas por parte del cuerpo
Votos (Shantis)	ID del usuario, ID de la postura, calificación asignada	Implementar sistema de votación	Valoración de posturas, generación de ranking
Rutinas personalizadas	ID del usuario, asanas seleccionadas, duración de la rutina, nivel de dificultad	Permitir la personalización de prácticas	Creación de rutinas y clases personalizadas

Requisitos Funcionales.

Para cumplir con los objetivos del proyecto, **AsanaYoga** incluirá diversas funcionalidades. A continuación, se describen las principales funciones, presentadas como casos de uso.

Casos de Uso en tabla:

Caso de Uso	Descripción
Registro de usuario	Los usuarios podrán registrarse en la aplicación proporcionando su nombre, correo electrónico y contraseña.
Inicio de sesión	El usuario podrá iniciar sesión en su cuenta para acceder a sus datos personales, rutinas y poder interactuar con la comunidad.
Buscar posturas por categoría	Los usuarios podrán realizar búsquedas avanzadas filtrando las posturas según su categoría, tipo de postura, nivel de dificultad o parte del cuerpo implicada.
Visualizar información de una postura	Al seleccionar una postura, el usuario podrá ver su descripción detallada, variantes, dificultad, imágenes y videos.
Votar una postura (Shantis)	Los usuarios podrán votar y valorar las posturas, lo que generará un ranking basado en la popularidad dentro de la comunidad.
Creación y edición de rutinas personalizadas	Los usuarios podrán crear y editar rutinas personalizadas, seleccionando las posturas que prefieran según sus objetivos personales.
Gestionar perfil de usuario	El usuario podrá modificar su perfil, incluyendo la actualización de sus preferencias y nivel de experiencia.

En las siguientes líneas se mostrarán un poco más en profundidad los casos de uso mencionados anteriormente, junto a su respectivo diagrama de flujo:

Registro de Usuario.

Descripción: un usuario no registrado podrá crear una cuenta en la aplicación proporcionando los datos necesarios.

Actor: usuario no registrado

Precondiciones: el usuario no debe tener una cuenta registrada en la aplicación.

Postcondiciones: el usuario queda registrado en la base de datos y puede iniciar sesión en la aplicación.

Flujo principal:

1. El usuario accede a la pantalla de registro en la aplicación.
2. Introduce sus datos, incluyendo nombre, correo electrónico, contraseña y nivel de experiencia en yoga.
3. El sistema valida que los datos proporcionados sean correctos (el correo no está ya registrado, la contraseña cumple con los requisitos mínimos de seguridad).
4. Si los datos son válidos, el sistema crea la cuenta y registra la información en la base de datos.
5. El sistema confirma el registro enviando un correo de confirmación al usuario.

Flujo alternativo:

- Si el correo ya está registrado, el sistema muestra un mensaje de error indicando que la cuenta ya existe.

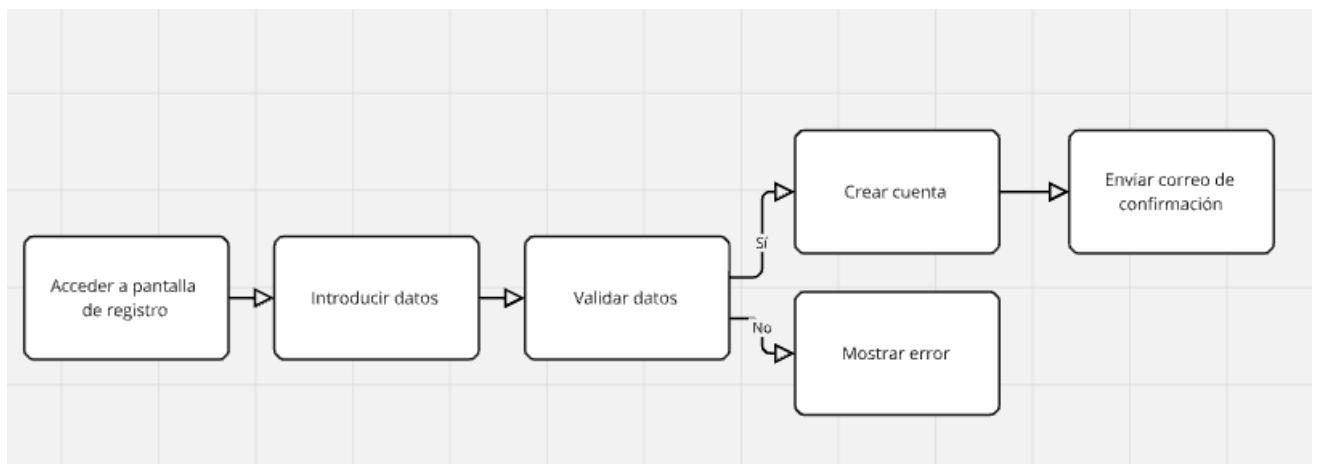


Figura 1: Registrar un usuario

Inicio de Sesión.

Descripción: un usuario registrado podrá iniciar sesión en la aplicación utilizando sus credenciales.

Actor: usuario registrado

Precondiciones: el usuario debe haber completado el registro y tener una cuenta activa.

Postcondiciones: el usuario accede a su cuenta y puede interactuar con todas las funcionalidades de la aplicación.

Flujo principal:

1. El usuario abre la aplicación e ingresa a la pantalla de inicio de sesión.
2. Introduce su correo electrónico y contraseña.
3. El sistema verifica que las credenciales sean correctas.
4. Si los datos son correctos, el sistema permite el acceso y carga la pantalla principal de la aplicación.

Flujo alternativo:

- Si las credenciales no coinciden, el sistema muestra un mensaje de error e invita al usuario a reintentar o a restablecer su contraseña.

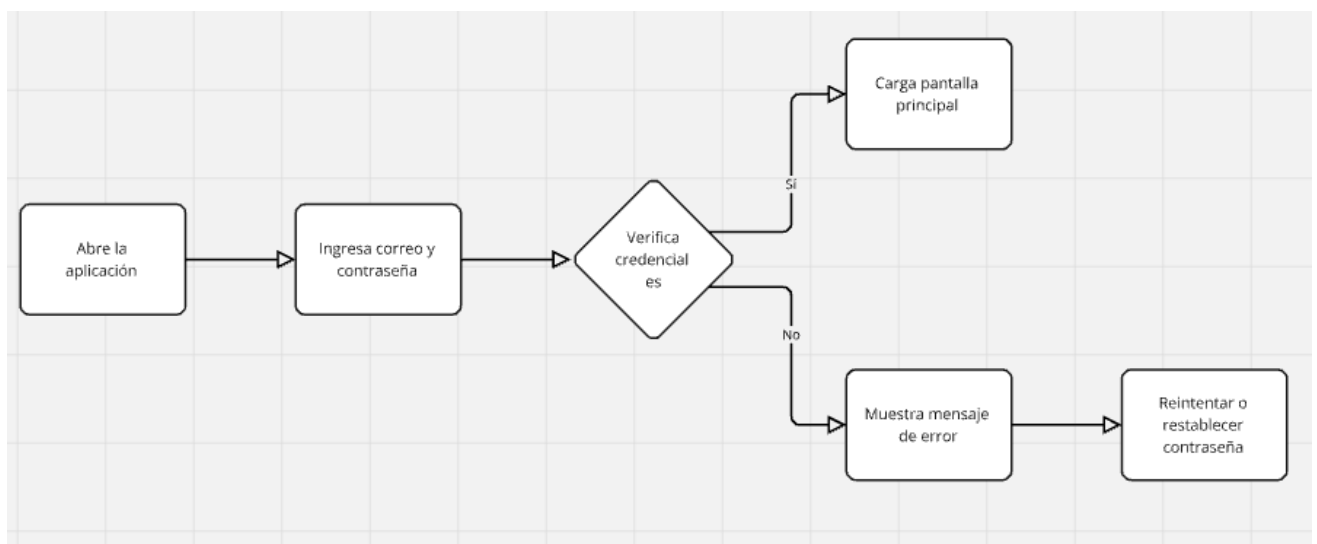


Figura 2: Caso de uso Inicio de sesión

Buscar Posturas por Categoría.

Descripción: el usuario puede buscar asanas aplicando filtros para encontrar posturas específicas según tipo, nivel de dificultad o parte del cuerpo involucrada.

Actor: Usuario registrado

Precondiciones: El usuario debe estar registrado e iniciado sesión en el sistema.

Postcondiciones: Se muestra una lista de posturas que coinciden con los filtros seleccionados.

Flujo principal:

1. El usuario accede a la pantalla de búsqueda en la aplicación.
2. Selecciona uno o varios filtros para refinar la búsqueda, como el tipo de asana (equilibrio, flexión), nivel de dificultad o parte del cuerpo implicada.
3. El sistema busca en la base de datos las posturas que coinciden con los filtros.
4. El sistema muestra los resultados de la búsqueda en formato de lista con imágenes y descripciones básicas.

Flujo alternativo:

- Si no hay coincidencias con los filtros seleccionados, el sistema muestra un mensaje indicando que no se encontraron resultados.

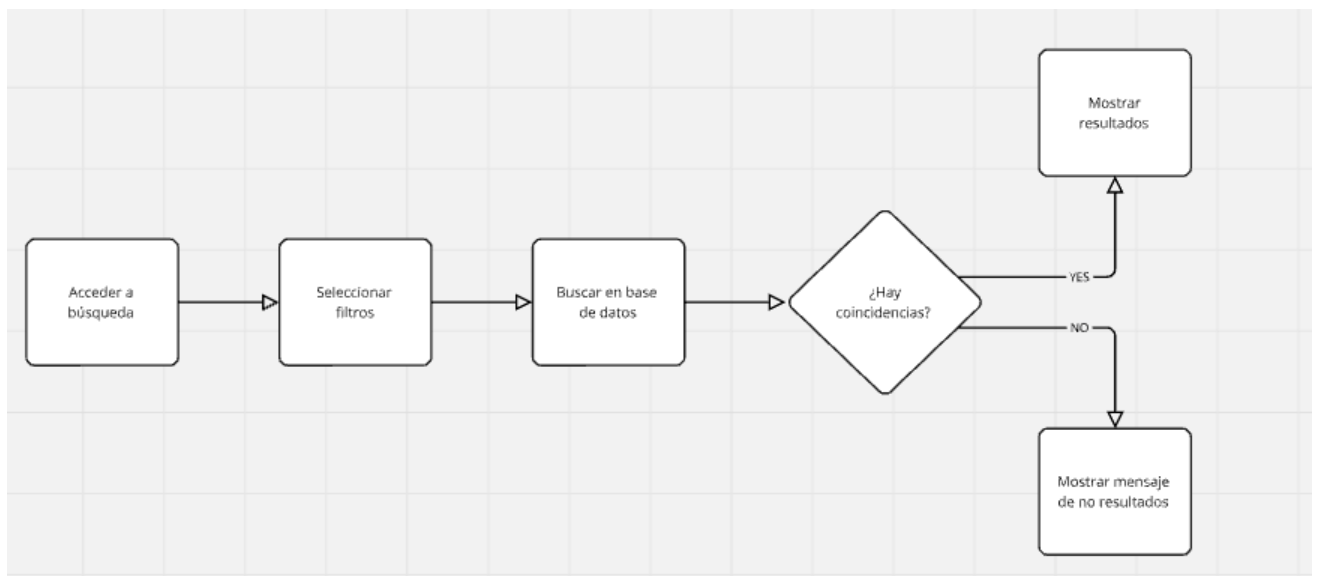


Figura 3: Caso de uso Buscar Posturas por Categoría

Visualizar Información de una Postura.

Descripción: el usuario puede seleccionar una postura de yoga para ver más detalles, como su descripción, variantes, dificultad, partes del cuerpo involucradas, imágenes y vídeos.

Actor: usuario registrado

Precondiciones: el usuario debe haber realizado una búsqueda o seleccionado una postura desde una lista (posturas más populares).

Postcondiciones: Se muestra toda la información detallada sobre la postura seleccionada.

Flujo principal:

1. El usuario selecciona una postura de yoga desde una lista de resultados o desde la pantalla principal.
2. El sistema carga y muestra la información detallada de la postura, incluyendo imágenes y vídeos si están disponibles.
3. El usuario puede ver variantes de la postura, el nivel de dificultad, las partes del cuerpo involucradas y otros detalles.

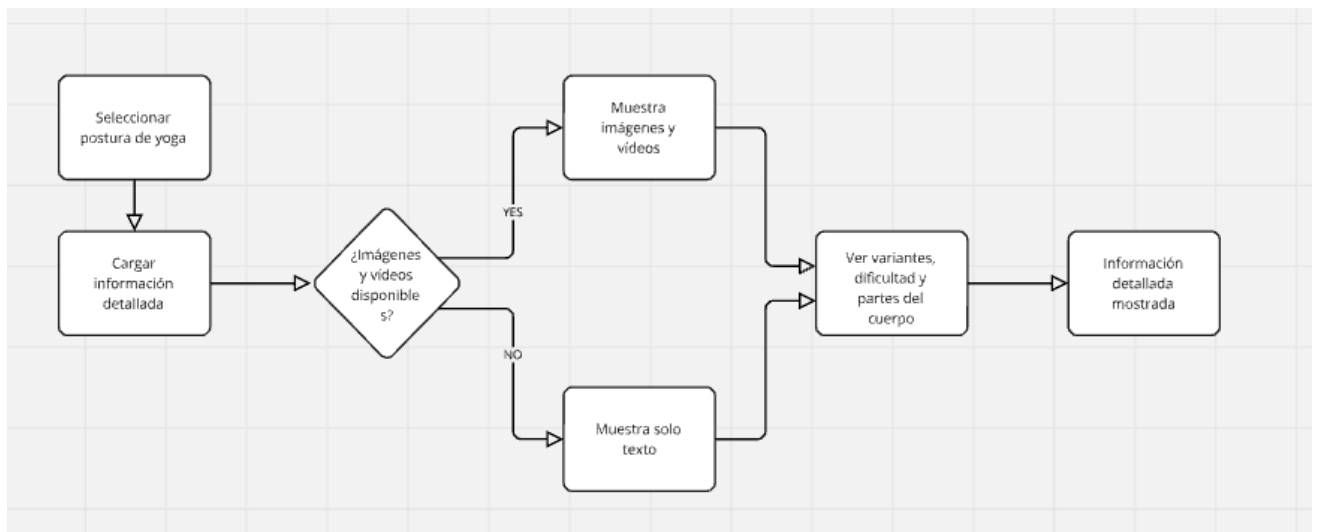


Figura 4: Caso de uso Visualizar Información de una Postura

Votar una Postura (Shantis).

Descripción: el usuario puede votar una postura de yoga, asignando una calificación que influye en el ranking general de las posturas más valoradas.

Actor: usuario registrado.

Precondiciones: el usuario debe haber visualizado la información de una postura.

Postcondiciones: la votación se guarda en la base de datos y se actualiza el ranking de posturas.

Flujo principal:

1. El usuario accede a la página de detalles de una postura.
2. Selecciona la opción de votar.
3. El sistema registra el voto del usuario en la base de datos.
4. El sistema actualiza el promedio de votos de la postura y su posición en el ranking general.

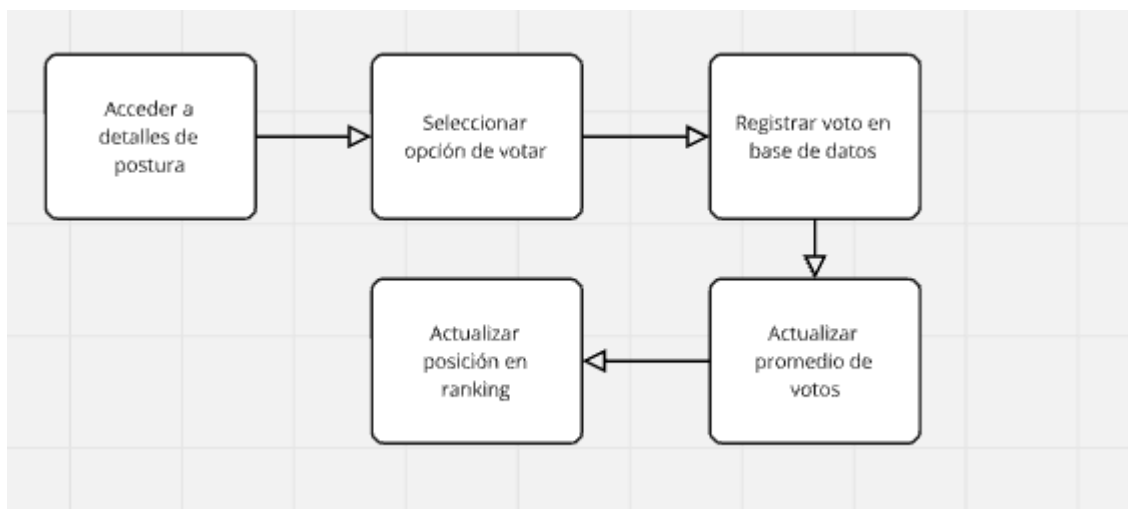


Figura 5: Caso de uso Votar una Postura (Shantis)

Creación y Edición de Rutinas Personalizadas.

Descripción: el usuario puede crear y personalizar rutinas de yoga seleccionando posturas específicas y ajustando la duración y nivel de dificultad de la rutina.

Actor: usuario registrado.

Precondiciones: el usuario debe haber iniciado sesión y tener acceso a las posturas en la aplicación.

Postcondiciones: se guarda la rutina personalizada del usuario en la base de datos y está disponible para ser ejecutada posteriormente.

Flujo principal:

1. El usuario accede a la sección de "Rutinas personalizadas".
2. Selecciona las posturas que desea incluir en la rutina.
3. Ajusta la duración y nivel de dificultad de la rutina.
4. El sistema guarda la rutina personalizada en la cuenta del usuario.
5. El usuario puede ver, modificar o eliminar la rutina en cualquier momento.

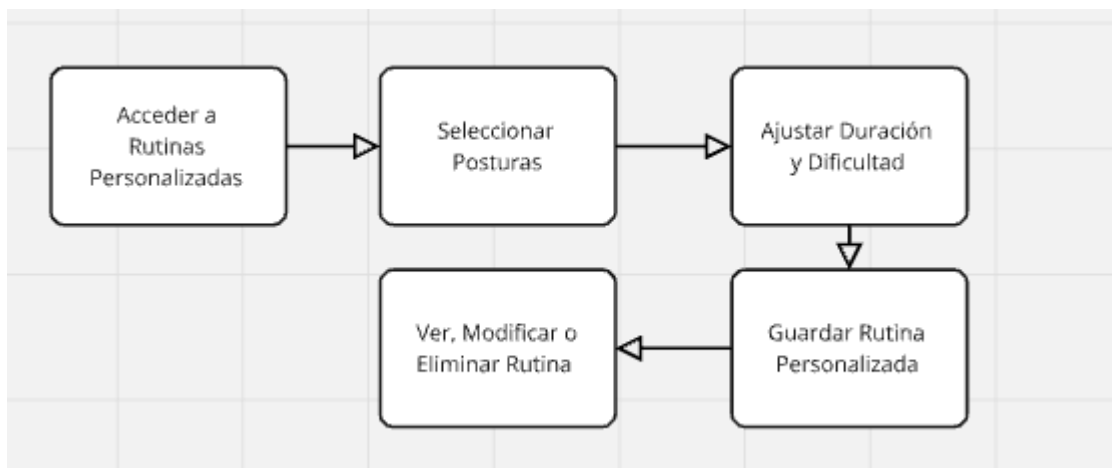


Figura 6: Caso de uso Votar una Postura (Shantis)

Gestión del Perfil de Usuario.

Descripción: el usuario puede modificar la información de su perfil, como su nombre, correo electrónico, nivel de experiencia o preferencias.

Actor: usuario registrado.

Precondiciones: el usuario debe haber iniciado sesión en su cuenta.

Postcondiciones: la información modificada del perfil se guarda y actualiza en la base de datos.

Flujo principal:

1. El usuario accede a la sección de "Perfil" en la aplicación.
2. Modifica los datos que desea actualizar, como nombre, correo, nivel de experiencia o preferencias.
3. El sistema valida los cambios y los guarda en la base de datos.
4. El sistema confirma que los cambios se han guardado correctamente.

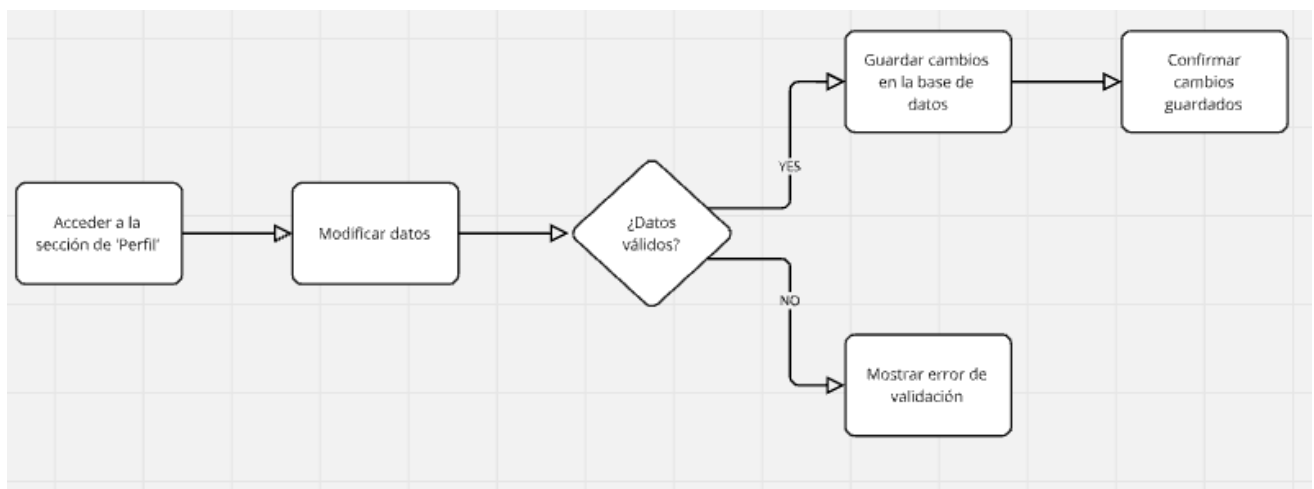


Figura 7: Caso de uso Gestión de Perfil de Usuario

Requisitos no funcionales.

En este apartado expondremos cualquier elemento que no tenga que ver con el código en sí, pero sí en la elaboración de proyecto, ya que son fundamentales para garantizar la calidad y el éxito del producto final. Aquí se detallan los requisitos no funcionales:

Tabla de Propiedades de la Aplicación.

Propiedad	Descripción
Interfaz intuitiva y amigable	La aplicación debe tener una interfaz fácil de navegar, con menús claros y una organización lógica de pantallas.
Compatibilidad multiplataforma	Compatible con Android e iOS, adaptándose a diferentes resoluciones y tamaños de pantalla.
Eficiencia de respuesta	Las consultas deben ejecutarse rápidamente, con un tiempo de respuesta inferior a 2 segundos en operaciones comunes.

Tabla de Cualidades de la Aplicación.

Cualidad	Descripción
Seguridad de la información	Implementación de medidas de seguridad, como cifrado de contraseñas y autenticación segura.
Escalabilidad	Soporte para el crecimiento en número de usuarios y cantidad de datos sin afectar el rendimiento de la aplicación.
Mantenimiento y actualización	Estructura modular que permita realizar modificaciones o añadir nuevas funcionalidades de forma eficiente.

Tabla de Normas y Estándares.

Norma/Estándar	Descripción
Estándares de usabilidad	Cumplir con guías de usabilidad de Android e iOS para una experiencia de usuario consistente y accesible.
Normativas de privacidad	Cumplir con normativas de privacidad como el RGPD, protegiendo los datos personales de los usuarios.
Gestión de accesibilidad	Seguir estándares de accesibilidad (WCAG 2.1), incluyendo texto alternativo en imágenes y etiquetas en formularios.

Tabla de Restricciones.

Restricción	Descripción
Restricciones de tiempo	Completar el desarrollo en un plazo de seis meses, con pruebas Alfa y Beta en un máximo de un mes.
Restricciones de coste	Ajustar el presupuesto para cubrir diseño, desarrollo, pruebas y licencias necesarias; optar por servidores de bajo costo si es necesario.
Restricciones de diseño	Seguir un diseño visual minimalista y en tonos neutros que transmitan tranquilidad, evitando sobrecarga visual.

Diseño del sistema.

Diseño de datos.

Aquí se muestran los datos representados en un modelo entidad-relación y unas tablas aclaratorias, junto a las relaciones correspondientes:

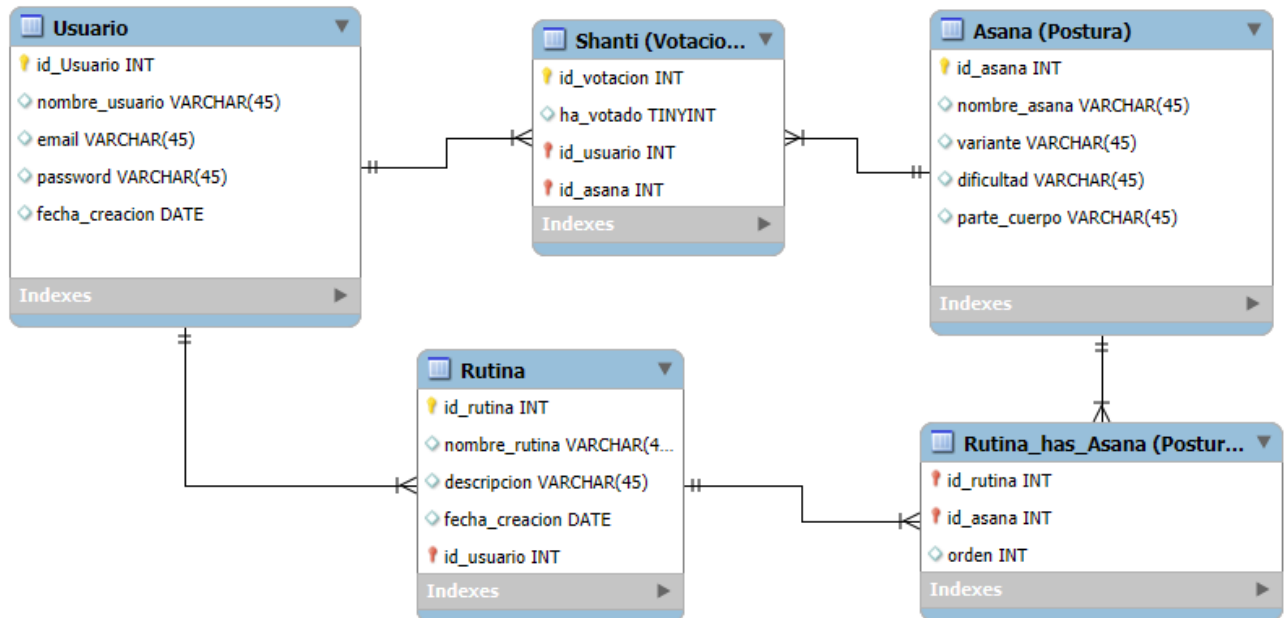


Figura 8: Modelo Entidad-Relación

Tabla Usuario.

- **Descripción:** esta tabla almacena la información básica de los usuarios registrados en la aplicación.

Campo	Tipo de dato	Clave	Descripción
id_usuario	INT	Primaria	Identificador único para cada usuario.
nombre_usuario	VARCHAR(45)	Única	Nombre elegido por el usuario para identificarse en la aplicación.
email	VARCHAR(20)	Única	Correo electrónico del usuario, utilizado para comunicación.
password	VARCHAR(255)	-	Contraseña cifrada del usuario para autenticación segura.
fecha_creacion	DATE	-	Fecha en la que el usuario creó su cuenta.

Tabla Asana (Postura).

- **Descripción:** contiene la información sobre cada asana o postura de yoga disponible en la plataforma.

Campo	Tipo de dato	Clave	Descripción
id_asana	INT	Primaria	Identificador único de cada postura de yoga.
nombre_asana	VARCHAR(45)	-	Nombre de la postura de yoga.
variante	VARCHAR(45)	-	Variante de la postura (opcional, en caso de que exista una variante).
dificultad	VARCHAR(10)	-	Nivel de dificultad de la postura.
descripcion	TEXT	-	Descripción de los beneficios físicos y mentales de la postura.
parte_cuerpo	VARCHAR(45)	-	Parte principal del cuerpo involucrada en la postura (ej., espalda, piernas).

Tabla Shanti (Votacion).

- **Descripción:** registra las votaciones o "shantis" que los usuarios hacen en cada asana para indicar su aprobación o preferencia.

Campo	Tipo de dato	Clave	Descripción
id_votacion	INT	Primaria	Identificador único de cada votación.
ha_votado	BOOLEAN(puse TINYINT porque no deja ponerlo)	-	Indica si el usuario ha votado o no.
id_usuario	INT	Foránea	Identificador del usuario que realiza la votación (clave foránea).
id_asana	INT	Foránea	Identificador de la asana votada (clave foránea).

Relación:

- **Usuario y Shanti:** relación de **1-N**.
(Un usuario puede realizar múltiples votaciones, pero cada votación pertenece a un solo usuario).
- **Asana y Shanti:** relación de **1-N**.
(Una asana puede tener múltiples votaciones, pero cada votación corresponde a una única asana).

Tabla Rutina.

- **Descripción:** Esta tabla guarda las rutinas creadas por los usuarios, que consisten en una colección de asanas en un orden determinado.

Campo	Tipo de dato	Clave	Descripción
id_rutina	INT	Primaria	Identificador único de cada rutina.
nombre_rutina	VARCHAR(45)	-	Nombre asignado por el usuario a la rutina.
descripcion	VARCHAR(255)	-	Descripción detallada de la rutina creada por el usuario.
fecha_creacion	DATE	-	Fecha en la que se creó la rutina.
id_usuario	INT	Foránea	Identificador del usuario que creó la rutina (clave foránea).

Relación:

- **Usuario y Rutina:** relación de **1-N**.

(Un usuario puede crear múltiples rutinas, pero cada rutina pertenece a un solo usuario).

Tabla Rutina_has_Asana (Postura).

- **Descripción:** tabla intermedia para gestionar la relación **N**

entre Rutina y Asana. Cada fila representa una postura en una rutina específica, permitiendo definir el orden de la postura dentro de la rutina.

Campo	Tipo de dato	Clave	Descripción
id_rutina	INT	Foránea	Identificador de la rutina en la que se incluye la postura (clave foránea).
id_asana	INT	Foránea	Identificador de la postura incluida en la rutina (clave foránea).
orden	INT	-	Orden de la postura dentro de la rutina.

Relación:

- **Rutina y Asana:** relación de **N-M**.

(Una rutina puede incluir múltiples asanas, y una asana puede formar parte de múltiples rutinas).

Resumen de Relaciones:**1. Relación Usuario - Shanti (Votacion):**

- **Tipo de Relación:** 1
- **Explicación:** un usuario puede hacer varias votaciones en diferentes asanas, pero cada votación pertenece a un solo usuario.
- **Cardinalidad:** 1 usuario puede realizar M votaciones.
- **Claves:** id_usuario en Shanti es clave foránea que referencia a Usuario.

2. Relación Asana - Shanti (Votacion):

- **Tipo de Relación:** 1
- **Explicación:** una asana puede recibir múltiples votaciones de distintos usuarios, pero cada votación está relacionada con una única asana.
- **Cardinalidad:** 1 asana puede tener M votaciones.
- **Claves:** id_asana en Shanti es clave foránea que referencia a Asana.

3. Relación Usuario - Rutina:

- **Tipo de Relación:** 1
- **Explicación:** un usuario puede crear múltiples rutinas, pero cada rutina pertenece a un solo usuario.
- **Cardinalidad:** 1 usuario puede tener M rutinas.
- **Claves:** id_usuario en Rutina es clave foránea que referencia a Usuario.

4. Relación Rutina - Asana (a través de Rutina_has_Asana):

- **Tipo de Relación:** N
- **Explicación:** una rutina puede estar compuesta de múltiples asanas, y una asana puede formar parte de múltiples rutinas.
- **Cardinalidad:** N rutinas pueden tener M asanas, y viceversa.
- **Claves:** id_rutina en Rutina_has_Asana referencia a Rutina, y id_asana referencia a Asana.

Diseño de arquitectura.

La arquitectura del sistema AsanaYoga está concebida bajo un modelo de capas múltiples, una estructura que no solo asegura la claridad en la asignación de responsabilidades, sino que también potencia tanto la escalabilidad como el mantenimiento a largo plazo del sistema. Esta arquitectura propone, capa por capa, un entramado que refleja un enfoque ordenado, pero adaptable, hacia el desarrollo de la aplicación. Veamos, entonces, las diferentes capas y sus roles respectivos:

Capa de Presentación (Frontend).

Esta capa constituye el punto de contacto entre el usuario y el sistema. Diseñada para ser intuitiva y accesible, proporciona una interfaz gráfica donde la simplicidad y la estética se combinan para facilitar la experiencia del usuario. A través de esta interfaz, los usuarios pueden navegar la aplicación, buscar asanas, personalizar sus rutinas e interactuar con la comunidad.

- Tecnologías: para lograr un entorno flexible y consistente en diversas plataformas, se contempla el uso de frameworks móviles como Flutter o React Native. Con ellos, se busca asegurar una experiencia homogénea tanto en Android como en iOS.
- Componentes:
 - Interfaz de Usuario: incluye pantallas de bienvenida, formularios de búsqueda, secciones dedicadas a la visualización de asanas, y espacios para la interacción comunitaria.

Componentes de UI: la estructura incluye botones, menús, formularios, listas, y otros elementos interactivos que mejoran la navegación y usabilidad de la aplicación.

Capa de Lógica de Negocio (Backend).

Aquí reside el núcleo de las operaciones de negocio de AsanaYoga. En esta capa se implementan las reglas y procesos que dan sentido a la aplicación, facilitando que las interacciones del usuario se transformen en respuestas eficaces y en funcionalidades tangibles.

- Responsabilidades:
 - Procesar las búsquedas avanzadas y aplicar los filtros necesarios para devolver resultados relevantes.
 - Administrar el sistema de votación, controlando el registro y la visualización de las puntuaciones (o "shantis") que los usuarios otorgan a cada asana.
 - Gestionar la creación y personalización de rutinas, adaptándose a las preferencias individuales de cada usuario.
- Tecnologías: dependiendo del frontend, Dart (si se utiliza Flutter) o Node.js serían adecuados para implementar una lógica ligera y en tiempo real, permitiendo además la posibilidad de funcionamiento offline.

- Componentes:
 - Gestor de Votaciones: controla el registro y actualización de las valoraciones que recibe cada asana.
 - Manejador de Rutinas: permite a los usuarios crear, modificar y gestionar sus secuencias de yoga.

Capa de Acceso a Datos (Data Layer).

Encargada de gestionar la base de datos, esta capa es crucial para almacenar y recuperar la información de manera eficiente. Se ha optado por SQLite, una elección pragmática y ligera para aplicaciones móviles, conocida por su equilibrio entre simplicidad y rendimiento.

- Responsabilidades:
 - Gestionar el almacenamiento de datos de asanas, usuarios, votaciones y rutinas, permitiendo un acceso rápido y eficaz a esta información.
 - Optimizar las consultas de datos para minimizar el tiempo de respuesta.
- Componentes:
 - Repositorio de Asanas: gestiona los datos relacionados con las posturas, incluidas variantes y beneficios.
 - Repositorio de Usuarios: almacena la información de cada usuario, como credenciales, historial de actividad y rutinas personalizadas.
 - Repositorio de Votaciones: administra la información sobre los votos que los usuarios otorgan a las asanas.

Capa de Infraestructura (Infrastructure Layer).

Esta capa soporta la infraestructura subyacente que permite el funcionamiento estable de la aplicación, incluyendo la comunicación con la base de datos y otros servicios externos.

- Responsabilidades:
 - Garantizar la conectividad con la base de datos SQLite y otros recursos.
 - Asegurar el funcionamiento de la aplicación en condiciones de red inestables, mediante la sincronización y almacenamiento en caché.

- Componentes:
 - Gestor de Red: supervisa las interacciones de la aplicación con la red, asegurando una experiencia óptima tanto online como offline.
 - Gestor de Sincronización: coordina la actualización de los datos entre el almacenamiento local y los servidores, especialmente en condiciones de red intermitente.

Arquitectura General: Cliente-Servidor.

La estructura del sistema se apoya en un modelo Cliente-Servidor:

- Cliente: la aplicación móvil que recopila la información del usuario y muestra datos relevantes.
- Servidor: procesa las solicitudes del cliente, aplica las reglas de negocio y gestiona el acceso a la base de datos.

Beneficios de esta Arquitectura.

- Escalabilidad: la arquitectura permite añadir nuevas funcionalidades en la capa de lógica de negocio sin comprometer el funcionamiento general.
- Mantenibilidad: separar la lógica de negocio de la presentación y el acceso a datos simplifica las tareas de mantenimiento y mejora del sistema.
- Rendimiento: al reducir el acceso innecesario a la base de datos y optimizar la comunicación entre capas, esta estructura potencia la rapidez y la eficiencia de la aplicación.

Diseño de interfaz

Pantalla de Inicio (Splash Screen).

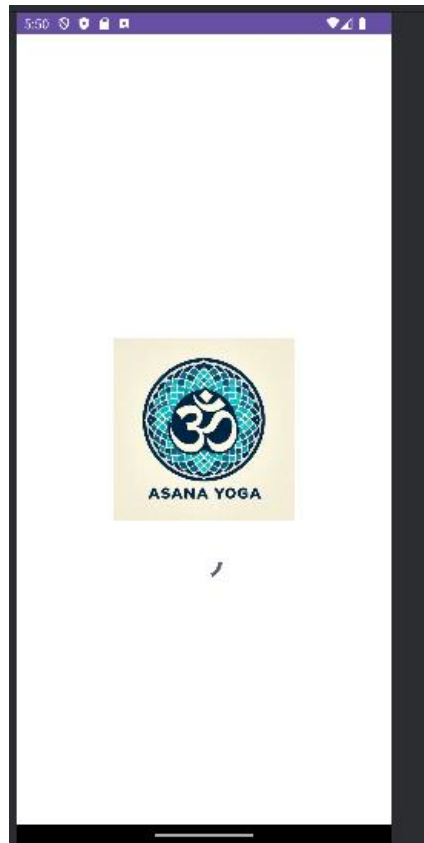



Figura 9: Pantalla de Carga

Desde el primer momento en que se inicia la aplicación, esta pantalla de bienvenida captura la atención del usuario. Una imagen central del logo de **AsanaYoga** se presenta aquí, en una pantalla que muestra, de forma breve pero significativa, la identidad de la aplicación. Además, una sutil animación de carga no solo indica al usuario que la aplicación está preparando su entorno, sino que también establece un ritmo inicial antes de sumergirse en la interfaz.

- **Elementos:**
 - **Logo:** colocado al centro, da una primera impresión clara y distintiva.
 - **Animación de carga:** un detalle visual dinámico que ofrece información inmediata sobre el progreso de carga.
- **Propósito:** lograr un primer impacto positivo y dar unos segundos para que el sistema prepare los elementos visuales y operativos de la interfaz principal.

Pantalla de Bienvenida / Registro / Inicio de Sesión.

Usuario

Email

Contraseña

INICIAR SESIÓN

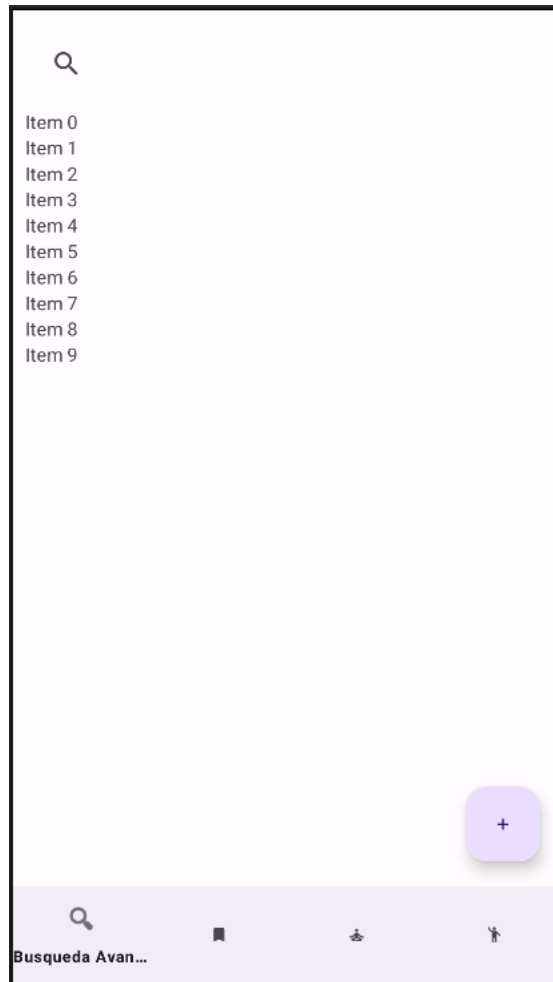
REGISTRARSE

Figura 10: Pantalla de Inicio de Sesión

En esta pantalla, el usuario tiene la posibilidad de ingresar a la aplicación, ya sea registrándose por primera vez o accediendo a su cuenta existente. Se trata de un espacio diseñado para facilitar esta entrada inicial, equilibrando tanto la seguridad como la accesibilidad.

○ **Elementos:**

- **Campo de texto para usuario:** permite que el usuario introduzca su username, esencial para la autenticación.
- **Campo de texto para email:** permite que el usuario introduzca su email.
- **Campo de texto para contraseña:** entrada segura para proteger los datos personales.
- **Botón "Iniciar Sesión":** lleva al usuario directamente a su cuenta.
- **Botón "Registrarse":** especialmente dirigido a nuevos usuarios.

Pantalla Principal (Dashboard).*Figura 11: Pantalla Principal*

Este es el núcleo de la experiencia del usuario, donde accede a todas las funciones clave de **AsanaYoga**. Cada elemento en esta pantalla está organizado para facilitar la navegación intuitiva y para guiar al usuario hacia los aspectos más destacados de la aplicación.

- **Elementos:**

- **Barra de búsqueda:** diseñada para localizar rápidamente asanas a través de palabras clave, nombres, o categorías.
- **Botones de navegación:** iconos que dirigen a las secciones de "Búsqueda Avanzada", "Rutinas", "Perfil" y "Asanas".
- **Botón Añadir Asana:** este botón flotante redirigirá a otra pantalla para agregar una asana.
- **Sección de asanas destacadas:** un carrusel que muestra las posturas más populares o nuevas, generando interés por descubrir.

Pantalla de Búsqueda Avanzada.

Search icon

Tipo de Asana

Item 1

Nivel de Dificultad

Item 1

Parte del Cuerpo

Item 1

Aplicar Filtros

Mostrar Ranking de posturas

Item 0

Item 1

Item 2

Item 3

Item 4

Item 5

Item 6

Item 7

Item 8

Item 9

Figura 12: Pantalla de Búsquedas avanzadas

La sección de "**Agregar Asana**" permite a los usuarios incorporar nuevas posturas a su lista personal o a la base de datos de la aplicación, ofreciendo una experiencia de usuario clara y amigable.

- **Elementos:**
 - **Campos de texto para introducir los datos de la asana:** espacio donde el usuario introduce los datos, como lo son nombre, descripción, variante, dificultad y parte del cuerpo
 - **Botón "Aplicar Cambios":** al presionarlo, se aplican los filtros.
 - **Botón "Ranking de asanas":** al presionarlo, muestra las asanas por orden de ranking, es decir, muestra las asanas más votadas.
 - **Lista de Búsqueda Avanzada:** aquí se mostraran las asanas que cumplan con los filtros establecidos.
- **Propósito:** permitir a los usuarios añadir nuevas posturas de yoga de manera intuitiva, asegurando que cada asana se guarde con sus características específicas para facilitar su futura consulta o inclusión en rutinas personalizadas. Este diseño minimalista y ordenado facilita el proceso de incorporación de datos, haciendo que la pantalla sea fácil de entender y usar.

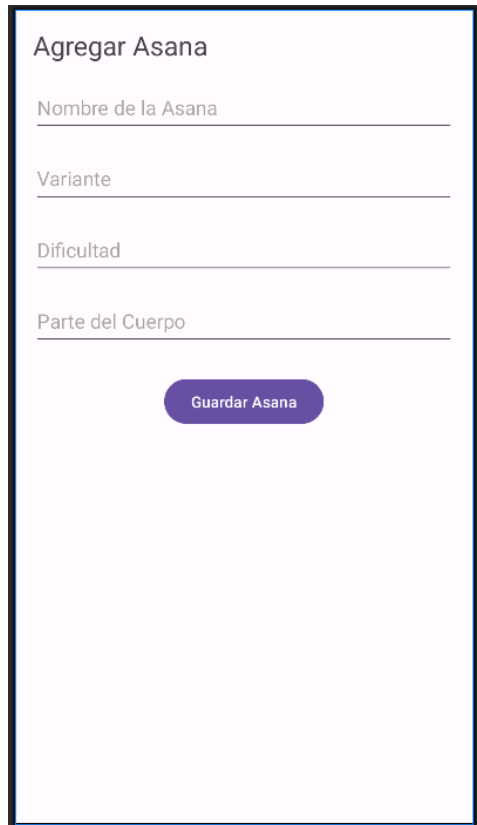
Pantalla Agregar Asana.

Figura 13: Pantalla Agregar Asana

La sección de "Agregar Asana" permite a los usuarios incorporar nuevas posturas a su lista personal o a la base de datos de la aplicación, ofreciendo una experiencia de usuario clara y amigable.

- **Elementos:**
 - **Campo de texto para nombre de asana:** espacio donde el usuario introduce el nombre de la postura.
 - **Desplegables de selección:** opciones para especificar detalles de la postura, como la variante, nivel de dificultad y área del cuerpo.
 - **Botón "Guardar asana":** guarda la postura con la información proporcionada y la añade a la base de datos.
- **Propósito:** permitir a los usuarios añadir nuevas posturas de yoga de manera intuitiva, asegurando que cada asana se guarde con sus características específicas para facilitar su futura consulta o inclusión en rutinas.

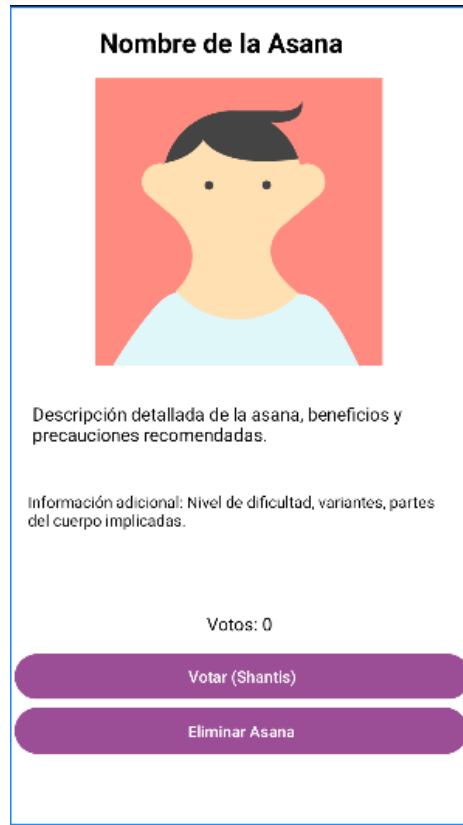
Pantalla de Detalle de Asana.

Figura 14: Pantalla Asana Detalle

Esta pantalla ofrece una descripción completa de cada asana seleccionada, con información que va más allá de una simple imagen y que busca instruir y orientar al usuario en su práctica.

- **Elementos:**
 - **Imagen de la asana:** una ilustración que facilita visualizar la postura.
 - **Nombre de la asana:** presentado como el título principal de la postura.
 - **Descripción:** una explicación detallada de la postura, sus beneficios y precauciones recomendadas.
 - **Información adicional:** datos sobre el nivel de dificultad, variantes, y partes del cuerpo implicadas.
 - **Botón de votación y contador (shantis):** una herramienta para valorar la postura, que fomenta la interacción de la comunidad.
 - **Botón Eliminar Asana:** Elimina la asana de la BBDD.
- **Propósito:** ofrecer un panorama completo sobre cada asana, promoviendo una experiencia de aprendizaje y personalización.

Pantalla de Creación de Rutina.

Crear Rutina

Nombre de la Rutina:

Escribe el nombre aquí

Buscar asanas

Asanas Disponibles

Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7

Rutina Actual

Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7

Guardar Rutina

Eliminar

Figura 15: Pantalla de Crear Rutina

Esta pantalla ofrece al usuario una herramienta intuitiva y eficiente para crear y personalizar sus propias rutinas de yoga, adaptándolas a sus metas y preferencias. La interfaz se ha diseñado para maximizar la simplicidad y la claridad visual, permitiendo una experiencia fluida al organizar las posturas.

Elementos:

- **Campo de nombre para la rutina:** permite al usuario asignar un título descriptivo a la rutina creada.
- **Buscador de asanas:** un campo interactivo que facilita la búsqueda de posturas específicas para añadir a la rutina.
- **Lista de asanas disponibles:** una sección que muestra las posturas que el usuario puede seleccionar y agregar a la rutina.
- **Sección de rutina actual:** presenta las asanas que ya han sido seleccionadas y organizadas en la rutina en creación, permitiendo su revisión y reordenamiento.
- **Botón "Guardar Rutina":** guarda la rutina personalizada en la base de datos, haciéndola accesible para futuras prácticas.
- **Botón "Eliminar":** ofrece la posibilidad de remover todas las posturas de la rutina actual en un solo paso, permitiendo reiniciar el proceso de personalización.

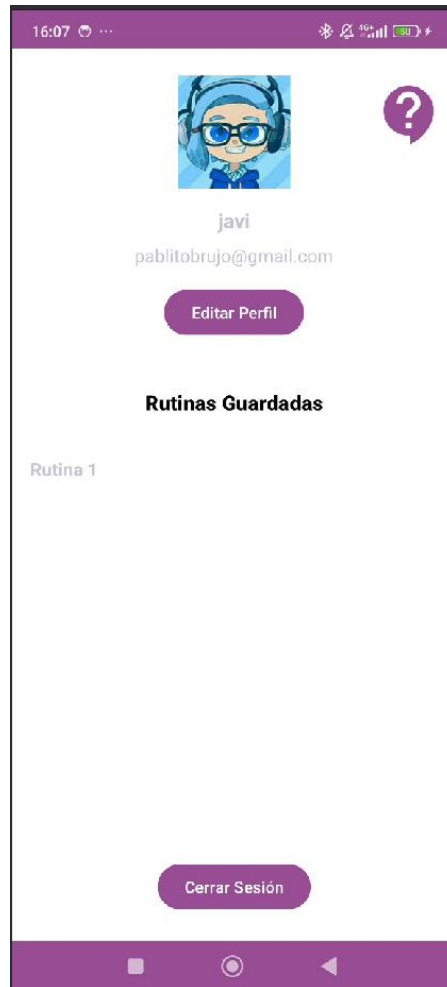
Pantalla de Perfil del Usuario.

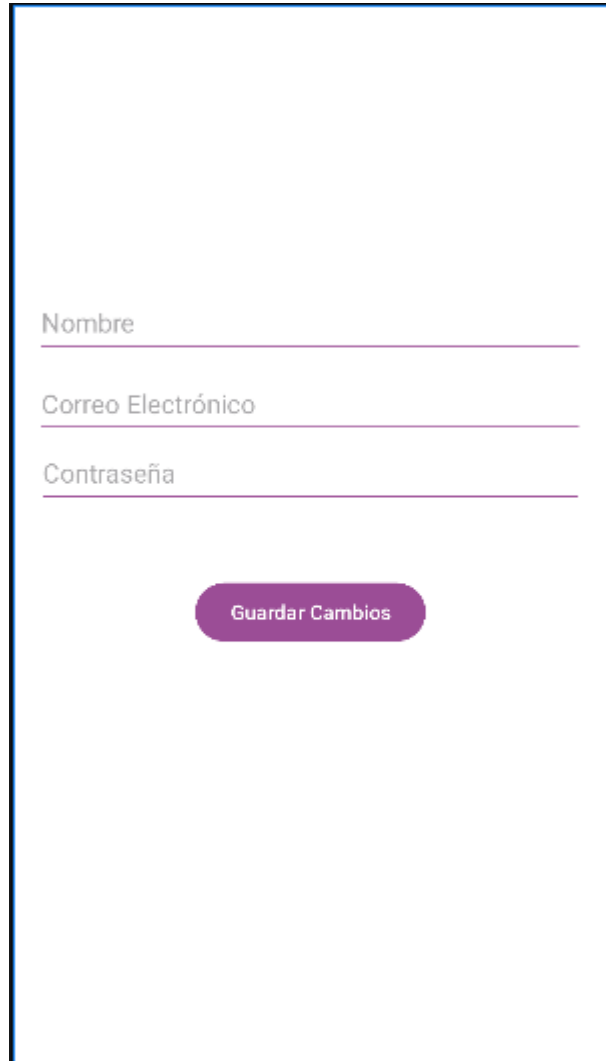
Figura 16: Pantalla de Perfil

Una pantalla que permite al usuario gestionar su información personal, visualizar sus rutinas guardadas, y ajustar las configuraciones de su cuenta.

- **Elementos:**

- **Foto de perfil:** imagen del usuario.
- **Botón Acerca de:** información sobre la app (versión, creador, información adicional)
- **Datos personales:** incluye nombre, email, y configuraciones adicionales.
- **Sección de rutinas guardadas:** una lista completa de todas las rutinas creadas por el usuario solo saldrá su nombre, para ver la rutina habrá que hacer clic en una rutina y te llevará a la "Pantalla de Rutinas Personalizadas".
- **Botón "Editar Perfil":** facilita la modificación de datos.
- **Botón "Cerrar Sesión":** opción para salir de la cuenta de usuario.

- **Propósito:** centralizar la gestión de la cuenta del usuario, ayudando a mantener su información y rutinas organizadas.

Pantalla de Editar Perfil.

The image shows a mobile application screen for editing a user profile. It features three input fields with placeholder text: 'Nombre', 'Correo Electrónico', and 'Contraseña'. Below these fields is a purple button with the text 'Guardar Cambios'.

Figura 17: Pantalla de Editar Perfil

Es la pantalla que se inicia cuando le das al botón Editar Perfil.

- **Elementos:**

- **Campos con los datos a editar:** se cargarán los datos del usuario en estos campos y podrás editar su username, su email y su contraseña
- **Botón “Guardar Cambios”:** botón para confirmar los cambios y así, actualizar los cambios de los datos del usuario.

Propósito:

Cambiar los datos del usuario de forma sencilla y de manera intuitiva.

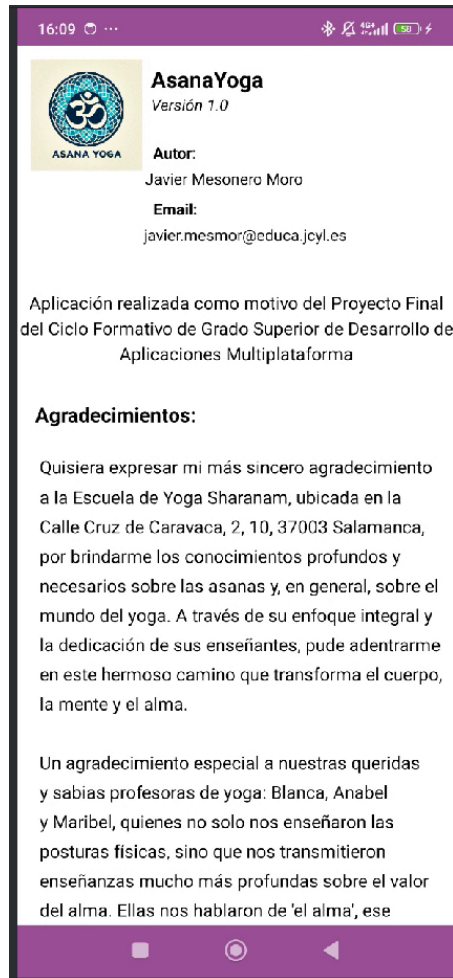
Pantalla de Acerca de.

Figura: Pantalla Acerca de

Una pantalla que permite al usuario ver información acerca de la app y de su creador:

- **Elementos:**
 - **Versión de la app:** versión 1.0.
 - **Creador:** javier Mesonero Moro.
 - **Email:** javier.mesmor@educa.jcyl.es.
 - **Propósito:** motivo del desarrollo (proyecto final del ciclo formativo).
 - **Agradecimientos:** mensaje extendido agradeciendo a colaboradores, profesores y la institución del yoga.

Propósito:

Centralizar la gestión de la cuenta del usuario, ofreciendo una forma organizada y accesible de visualizar la información sobre la aplicación, su creador, y mantener las rutinas y datos actualizados.

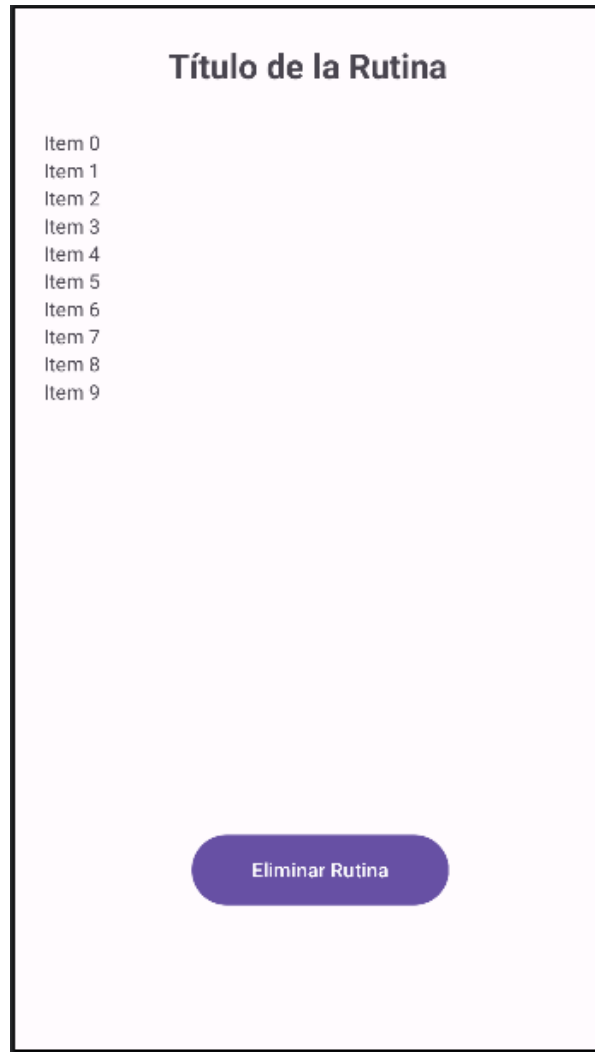
Pantalla de Rutinas Personalizadas.

Figura 18: Pantalla Rutina Personalizada

Una pantalla que permite al usuario ver sus rutinas personalizadas creadas.

- **Elementos:**
 - **Nombre de la rutina:** es como se llama la rutina que has creado.
 - **Lista de asanas:** listado de las asanas que conforman la rutina.
 - **Botón “Eliminar rutina”:** Elimina la rutina personalizada
- **Propósito:** una pantalla que permite al usuario visualizar los detalles de sus rutinas personalizadas creadas, incluyendo el nombre de la rutina y una lista de las asanas que la conforman. Su objetivo principal es ofrecer un acceso centralizado a la información de la rutina, facilitando la organización y visualización clara de los elementos asociados.

Diseño de componentes.

El diseño de componentes de **AsanaYoga** sigue un enfoque modular que permite la escalabilidad, el mantenimiento eficiente y la posibilidad de futuras expansiones. Cada componente desempeña una función específica, pero interactúa con otros módulos para brindar una experiencia de usuario fluida e integrada.

Componentes principales del sistema.

1. Componente de Autenticación de Usuario.

- **Descripción:** gestiona las operaciones de registro e inicio de sesión, asegurando la autenticidad de los usuarios y controlando el acceso a funcionalidades restringidas.
- **Funciones principales:**
 - Verificar credenciales de usuarios.
 - Almacenar información personal y perfiles en la base de datos.
 - Proporcionar permisos a usuarios autenticados.
- **Dependencias:** este componente se conecta a la base de datos de usuarios para almacenar y recuperar información de autenticación.

2. Componente de Gestión de Asanas.

- **Descripción:** administra el catálogo de posturas de yoga, proporcionando información completa de cada asana.
- **Funciones principales:**
 - Almacenar y mostrar asanas con detalles como nombre, variantes, nivel de dificultad y beneficios.
 - Incluir imágenes ilustrativas para guiar a los usuarios.
- **Dependencias:**
 - Utiliza la base de datos de asanas como fuente principal de información.
 - Proporciona datos al módulo de rutinas personalizadas y votación comunitaria.

3. Componente de Rutinas Personalizadas.

- **Descripción:** permite a los usuarios crear y editar rutinas de yoga personalizadas, seleccionando y organizando asanas según sus objetivos.
- **Funciones principales:**
 - Crear rutinas a partir del catálogo de asanas.
 - Guardar y editar rutinas personalizadas.
 - Proporcionar vistas para visualizar las rutinas creadas.
- **Dependencias:** este componente interactúa con el Componente de Gestión de Asanas para acceder a la lista de posturas disponibles.

4. Componente de Votación Comunitaria (Shantis).

- **Descripción:** facilita la interacción comunitaria mediante un sistema de votación, permitiendo valorar las asanas y generar estadísticas sobre popularidad.
- **Funciones principales:**
 - Registrar votos de los usuarios.
 - Mostrar puntuaciones en tiempo real.
 - Destacar las asanas mejor valoradas.
- **Dependencias:** se conecta con la base de datos de votación para almacenar los resultados y con el Componente de Gestión de Asanas para reflejar las valoraciones.

Figura.

El diseño modular de componentes, junto con sus conexiones y dependencias, se presenta en el **diagrama de componentes** que describe gráficamente cómo interactúan los módulos principales entre sí y con las bases de datos relevantes.

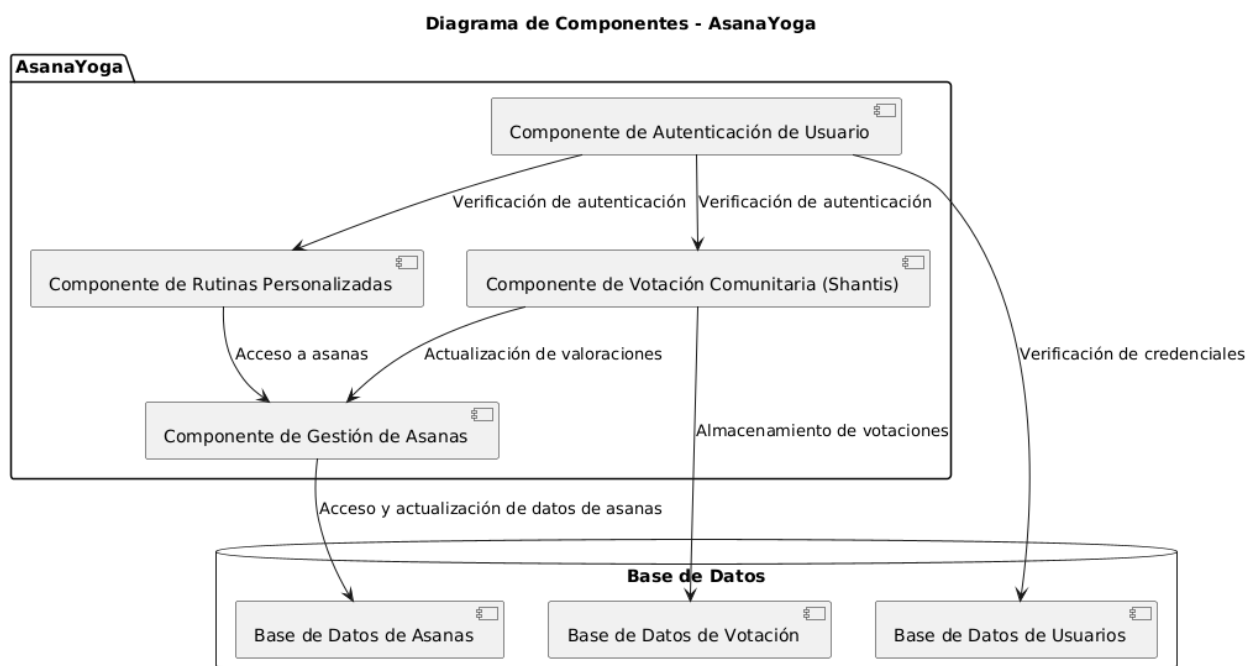


Figura 19: Diagrama de componentes

Diseño de despliegue.

El despliegue de **AsanaYoga** utiliza una arquitectura en la nube para garantizar la disponibilidad, escalabilidad y seguridad de los datos. La aplicación incluye:

1. **Base de Datos SQL Online:** aloja datos de usuarios, asanas y votaciones, permitiendo acceso en tiempo real y protegiendo la información con copias de seguridad automáticas.
2. **Sincronización en Tiempo Real:** el sistema de votación (Shantis) actualiza las valoraciones instantáneamente para ofrecer una experiencia fluida.
3. **Modelo Cliente-Servidor:** la app móvil interactúa con la base de datos a través de una API segura, que maneja la transmisión de datos entre el cliente y la nube.
4. **Gestión de Seguridad:** utiliza protocolos como OAuth 2.0 y encriptación para proteger credenciales y datos sensibles.

Beneficios:

- **Escalabilidad:** ajusta recursos según el crecimiento de usuarios.
- **Mantenimiento Simplificado:** centraliza la administración de datos y actualizaciones.
- **Alta Disponibilidad:** garantiza accesibilidad constante y mínimos tiempos de inactividad.

Este enfoque asegura un rendimiento óptimo, seguridad robusta y bases sólidas para futuras mejoras en AsanaYoga.

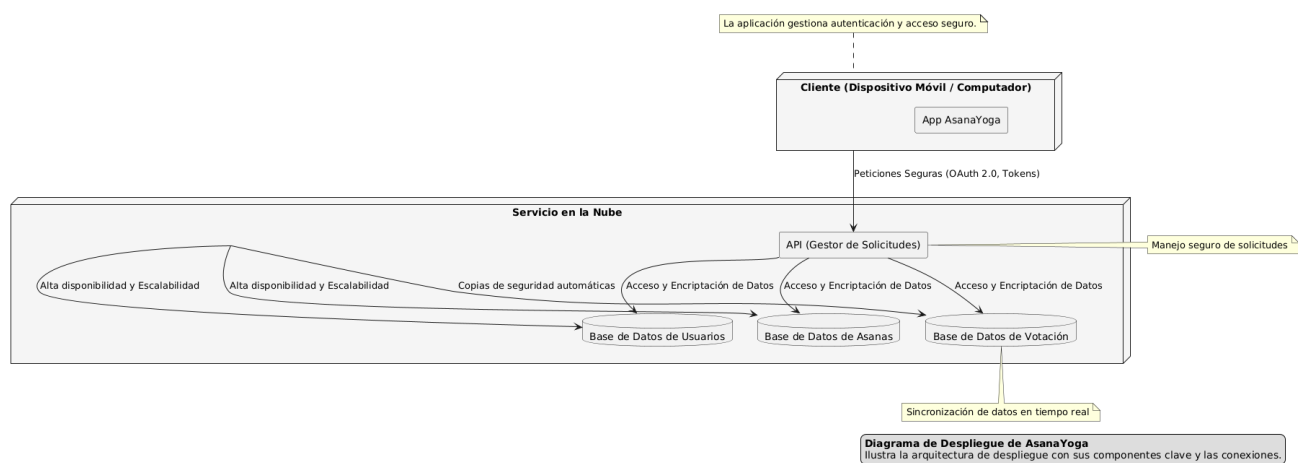


Figura 20: Diagrama de despliegue

Implementación.

Técnicas y herramientas.

Para la implementación del proyecto **AsanaYoga**, se han utilizado las siguientes herramientas, lenguajes de programación y recursos:

1. Java.

He elegido **Java** como lenguaje principal para el desarrollo de la aplicación debido a varias razones:

- Es uno de los lenguajes oficiales compatibles con **Android Studio**, lo que facilita la integración con las herramientas del entorno de desarrollo.
- Es un lenguaje versátil y ampliamente documentado, con numerosas librerías que permiten implementar funcionalidades avanzadas.
- Es el lenguaje en el que tengo mayor experiencia, adquirido durante los dos años de estudio, lo que asegura una implementación eficiente y confiable.

2. Android Studio.

Android Studio es el entorno de desarrollo integrado (IDE) oficial para la creación de aplicaciones Android, y ha sido clave en la implementación de este proyecto. Sus características principales incluyen:

- **Edición de código avanzada:** con funciones como resaltado de sintaxis, completado automático y refactorización, facilita un flujo de trabajo ágil.
- **Emulador integrado:** permite probar la aplicación en diferentes versiones de Android y configuraciones de dispositivos sin necesidad de hardware físico.
- **Diseñador de interfaces gráficas:** incluye una herramienta de diseño visual para crear y modificar interfaces de usuario arrastrando componentes, lo que simplifica la creación de pantallas.
- **Integración con Android SDK:** facilita el acceso a bibliotecas y APIs necesarias para el desarrollo de la aplicación.

3. SQLite.

SQLite es el sistema de gestión de bases de datos utilizado en **AsanaYoga**. Es ligero, eficiente y está perfectamente integrado en el entorno de desarrollo de Android. La conectividad con las BBDD online se hacen mediante APIs externas.

• Ventajas:

- Almacenamiento local para manejar datos como usuarios, asanas, rutinas y votaciones.
- No requiere configuración de servidor, lo que simplifica la implementación en dispositivos móviles.
- Permite consultas avanzadas mediante SQL para gestionar y recuperar información de manera eficiente.

4. Material Design Components.

Para garantizar una interfaz moderna y atractiva, se han utilizado componentes de **Material Design**, que permiten mantener un diseño uniforme y profesional. Estos componentes incluyen botones, barras de navegación, menús y animaciones que mejoran la experiencia del usuario.

5. Figma.

Figma se utilizó para el diseño inicial de la interfaz de usuario. Con esta herramienta:

- Se crearon prototipos interactivos que ayudaron a visualizar el flujo de navegación de la aplicación.
- Se diseñaron las pantallas principales, asegurando que cumplieran con principios de usabilidad y estética.

6. Git y GitHub

- **Git** se empleó para el control de versiones, permitiendo rastrear cambios en el código y garantizar que los desarrollos sean seguros y colaborativos.
- **GitHub** sirvió como repositorio remoto, facilitando la organización y documentación del proyecto.

Recursos Necesarios.

1. Hardware.

- Dispositivos Android para pruebas físicas, que permitan validar el funcionamiento de la aplicación en entornos reales.

2. Software.

- Se podrá descargar su aplicación desde Google Play Store, aunque por el momento, se puede descargar con su apk.

3. Equipo de Trabajo.

- **Desarrolladores de Android:** con experiencia en Java y en el uso de Android Studio.
- **Diseñadores UX/UI:** para crear interfaces visualmente atractivas y funcionales.

Implementación del código.

Aquí se enseñan los fragmentos del código más importantes de la aplicación:

Inicialización de la BBDD:

```
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_USUARIO_TABLE = "CREATE TABLE " + TABLE_USUARIO + " (" +
        COLUMN_USUARIO_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        COLUMN_USUARIO_NOMBRE + " TEXT NOT NULL UNIQUE, " +
        COLUMN_USUARIO_EMAIL + " TEXT NOT NULL UNIQUE, " +
        COLUMN_USUARIO_PASSWORD + " TEXT NOT NULL)";

    String CREATE_ASANA_TABLE = "CREATE TABLE " + TABLE_ASANA + " (" +
        COLUMN_ASANA_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        COLUMN_ASANA_NOMBRE + " TEXT, " +
        COLUMN_ASANA_VARIANTE + " TEXT, " +
        COLUMN_ASANA_DIFICULTAD + " TEXT, " +
        COLUMN_ASANA_PARTE_CUERPO + " TEXT, " +
        COLUMN_ASANA_DESCRIPCION + " TEXT);";

    String CREATE Rutina_TABLE = "CREATE TABLE " + TABLE_RUTINA + " (" +
        COLUMN_RUTINA_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        COLUMN_RUTINA_NOMBRE + " TEXT NOT NULL, " +
        COLUMN_RUTINA_DESCRIPCION + " TEXT, " +
        COLUMN_RUTINA_USUARIO_ID + " INTEGER NOT NULL, " +
        "FOREIGN KEY(" + COLUMN_RUTINA_USUARIO_ID + ") REFERENCES " + TABLE_USUARIO + "(" + COLUMN_USUARIO_ID + ") ON DELETE CASCADE ON UPDATE CASCADE");

    String CREATE_SHANTI_TABLE = "CREATE TABLE " + TABLE_SHANTI + " (" +
        COLUMN_SHANTI_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        COLUMN_SHANTI_HA_VOTADO + " BOOLEAN NOT NULL, " +
        COLUMN_SHANTI_USUARIO_ID + " INTEGER NOT NULL, " +
        COLUMN_SHANTI_ASANA_ID + " INTEGER NOT NULL, " +
        "FOREIGN KEY(" + COLUMN_SHANTI_USUARIO_ID + ") REFERENCES " + TABLE_USUARIO + "(" + COLUMN_USUARIO_ID + ") ON DELETE CASCADE ON UPDATE CASCADE, " +
        "FOREIGN KEY(" + COLUMN_SHANTI_ASANA_ID + ") REFERENCES " + TABLE_ASANA + "(" + COLUMN_ASANA_ID + ") ON DELETE CASCADE ON UPDATE CASCADE");

    String CREATE_Rutina_HAS_ASANA_TABLE = "CREATE TABLE " + TABLE_Rutina_HAS_ASANA + " (" +
        COLUMN_Rutina_HAS_ASANA_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        COLUMN_Rutina_HAS_ASANA_Rutina_ID + " INTEGER NOT NULL, " +
        COLUMN_Rutina_HAS_ASANA_ASANA_ID + " INTEGER NOT NULL, " +
        COLUMN_Rutina_HAS_ASANA_ORDEN + " INTEGER NOT NULL, " +
        "FOREIGN KEY(" + COLUMN_Rutina_HAS_ASANA_Rutina_ID + ") REFERENCES " + TABLE_Rutina + "(" + COLUMN_Rutina_ID + ") ON DELETE CASCADE ON UPDATE CASCADE, " +
        "FOREIGN KEY(" + COLUMN_Rutina_HAS_ASANA_ASANA_ID + ") REFERENCES " + TABLE_ASANA + "(" + COLUMN_ASANA_ID + ") ON DELETE CASCADE ON UPDATE CASCADE");
```

Figura 21: Código Inicialización BBDD

Inicio de sesión y registro

```
private void iniciarSesion() {
    String nombre_usuario = nombreUsuarioEditText.getText().toString().trim();
    String email = emailEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();

    if (nombre_usuario.isEmpty() || email.isEmpty() || password.isEmpty()) {
        Toast.makeText(context, this, text: "Por favor, completa todos los campos", Toast.LENGTH_SHORT).show();
        return;
    }

    if (db.comprobarUsuario(nombre_usuario, email, password)) {
        Toast.makeText(context, this, text: "Inicio de sesión exitoso", Toast.LENGTH_SHORT).show();
        int userId = db.obtenerUsuarioId(nombre_usuario, email);

        // Guardar el userId en SharedPreferences
        SharedPreferences sharedPreferences = context.getSharedPreferences("SesionUsuario", MODE_PRIVATE);
        sharedPreferences.edit().putInt("USER_ID", userId).apply();

        // Pasar a PantallaInicio
        Intent intent = new Intent(context, PantallaInicio.class);
        intent.putExtra("USER_ID", userId);
        startActivity(intent);
        finish();
    } else {
        Toast.makeText(context, this, text: "Credenciales incorrectas", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 22: Código Inicio Sesión

```
private void registrarUsuario() {
    String nombre_usuario = nombreUsuarioEditText.getText().toString().trim();
    String email = emailEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();

    if (nombre_usuario.isEmpty() || email.isEmpty() || password.isEmpty()) {
        Toast.makeText(context, this, text: "Por favor, completa todos los campos", Toast.LENGTH_SHORT).show();
        return;
    }

    if (!db.comprobarUsuario(nombre_usuario, email, password)) {
        if (db.anadirUsuario(nombre_usuario, email, password)) {
            Toast.makeText(context, this, text: "Registro exitoso", Toast.LENGTH_SHORT).show();
            iniciarSesion(); // Redirigir al perfil tras el registro
        } else {
            Toast.makeText(context, this, text: "Error al registrar usuario", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(context, this, text: "El nombre de usuario o el email ya existen", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 23: Código Registro Usuario

Aquí se muestran los métodos usados para insertar un usuario en la BBDD y las comprobaciones de inicio de sesión y registro:

```
//Metodo para agregar un nuevo usuario a la base de datos
1 usage
public boolean anadirUsuario(String nombre, String email, String password) {
    SQLiteDatabase db = this.getWritableDatabase();

    if (comprobarSiExisteNombreYEmail(nombre, email)) {
        return false;
    }

    ContentValues values = new ContentValues();
    values.put(COLUMN_USUARIO_NOMBRE, nombre);
    values.put(COLUMN_USUARIO_EMAIL, email);
    values.put(COLUMN_USUARIO_PASSWORD, password);
    long result = db.insert(TABLE_USUARIO, nullColumnHack: null, values);
    return result != -1;
}
```

Figura 24: Código Insertar Usuario

```
//Metodo para comprobar si el nombre de usuario o el email ya existen en la base de datos como medida preventiva, aunque ya tengo el metodo comprobarUsuario
1 usage
private boolean comprobarSiExisteNombreYEmail(String nombre, String email) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.query(TABLE_USUARIO, columns: null,
        selection: COLUMN_USUARIO_NOMBRE + "=? OR " + COLUMN_USUARIO_EMAIL + "=?",
        new String[]{nombre, email}, groupBy: null, having: null, orderBy: null);

    boolean exists = (cursor.getCount() > 0);
    cursor.close();
    return exists;
}

//Metodo para comprobar si el usuario existe en la base de datos para iniciar sesion y registrarse
2 usages
public boolean comprobarUsuario(String nombre_usuario, String email, String password) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(
        sql: "SELECT * FROM " + TABLE_USUARIO + " WHERE " +
            COLUMN_USUARIO_NOMBRE + "=? AND " +
            COLUMN_USUARIO_EMAIL + "=? AND " +
            COLUMN_USUARIO_PASSWORD + "=?",
        new String[]{nombre_usuario, email, password}
    );
    boolean exists = cursor.getCount() > 0;
    cursor.close();
    return exists;
}
```

Figura 25: Métodos de comprobación de usuario

Ahora se muestra el código encargado de actualizar los datos del usuario:

```
// Método para comprobar los campos de edición y actualizar el perfil
1 usage
public void comprobarEditarCampos(int userId){
    {
        String nuevoNombre = editarNombre.getText().toString().trim();
        String nuevoEmail = editarEmail.getText().toString().trim();
        String nuevaPassword = editarContraseña.getText().toString().trim();

        if (nuevoNombre.isEmpty() || nuevoEmail.isEmpty() || nuevaPassword.isEmpty()) {
            Toast.makeText(context: this, text: "Por favor, completa todos los campos.", Toast.LENGTH_SHORT).show();
            return;
        }

        boolean actualizado = asanaDatabase.actualizarUsuario(userId, nuevoNombre, nuevoEmail, nuevaPassword);
        if (actualizado) {
            Toast.makeText(context: this, text: "Perfil actualizado correctamente.", Toast.LENGTH_SHORT).show();
            finish();
        } else {
            Toast.makeText(context: this, text: "Error al actualizar el perfil. Intentalo de nuevo.", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Figura 26: Código para editar datos del usuario

```
// Método para actualizar información del usuario
1 usage
public boolean actualizarUsuario(int idUsuario, String nuevoNombre, String nuevoEmail, String nuevaPassword) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_USUARIO_NOMBRE, nuevoNombre);
    values.put(COLUMN_USUARIO_EMAIL, nuevoEmail);
    values.put(COLUMN_USUARIO_PASSWORD, nuevaPassword);

    int rowsAffected = db.update(TABLE_USUARIO, values, whereClause: COLUMN_USUARIO_ID + "=?", new String[]{String.valueOf(idUsuario)});
    return rowsAffected > 0;
}
```

Figura 27: Código para actualizar información en la BBDD

Aquí se muestra cómo se gestionan las asanas y los métodos CRUD:

```
// Metodo para agregar una nueva asana a la base de datos
1 usage
private void agregarAsana() {
    String nombre = nombreAsana.getText().toString().trim();
    String descripcion = descripcionAsana.getText().toString().trim();
    String variante = varianteAsana.getText().toString().trim();
    String dificultad = dificultadAsana.getText().toString().trim();
    String parteCuerpo = parteCuerpoAsana.getText().toString().trim();

    if (nombre.isEmpty() || dificultad.isEmpty()) {
        Toast.makeText(context, this, text: "Nombre y Dificultad son obligatorios", Toast.LENGTH_SHORT).show();
        return;
    }

    boolean isInserted = db.anadirAsana(nombre, variante, dificultad, parteCuerpo, descripcion);
    if (isInserted) {
        Toast.makeText(context, this, text: "Asana agregada exitosamente", Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(packageContext: AgregarAsana.this, PantallaInicio.class);
        startActivity(intent);
        finish();
    } else {
        Toast.makeText(context, this, text: "Error al agregar la asana", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 28: Código Creación de Asana

```
//Metodo para agregar una nueva asana a la base de datos
21 usages
public boolean anadirAsana(String nombre, String variante, String dificultad, String parteCuerpo, String descripcion) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_ASANA_NOMBRE, nombre);
    values.put(COLUMN_ASANA_VARIANTE, variante);
    values.put(COLUMN_ASANA_DIFICULTAD, dificultad);
    values.put(COLUMN_ASANA PARTE_CUERPO, parteCuerpo);
    values.put(COLUMN_ASANA_DESCRIPCION, descripcion);
    long result = db.insert(TABLE_ASANA, nullColumnHack: null, values);
    return result != -1;
}
```

Figura 29: Código Insertar Asana en BBDD

```
//Metodo para cargar las asanas en el RecyclerView
2 usages
private void cargarAsanas() {
    Cursor cursor = null;
    try {
        cursor = asanaDatabase.obtenerTodasLasAsanas();
        asanasList.clear();

        if (cursor != null) {
            while (cursor.moveToNext()) {
                String nombre = cursor.getString(cursor.getColumnIndexOrThrow(AsanaDatabase.COLUMN_ASANA_NOMBRE));

                Asana asana = new Asana(nombre);
                asanasList.add(asana);
            }
            asanaAdapter.notifyDataSetChanged();
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (cursor != null) {
            cursor.close();
        }
    }
}
```

Figura 30: Código Mostrar Asanas

```
//Metodo para obtener todas las asanas de la base de datos
5 usages
public Cursor obtenerTodasLasAsanas() {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.query(TABLE_ASANA, columns: null, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
}
```

Figura 31: Código Consulta Mostrar Asanas

```
// Método para eliminar la asana
1 usage
private void eliminarAsana() {
    boolean eliminado = db.eliminarAsana(asanaId);
    if (eliminado) {
        Toast.makeText(context: this, text: "Asana eliminada correctamente", Toast.LENGTH_SHORT).show();
        finish();
    } else {
        Toast.makeText(context: this, text: "Error al eliminar la asana", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 32: Código Borrar asana

```
// Método para eliminar una asana de la base de datos
1 usage
public boolean eliminarAsana(int idAsana) {
    SQLiteDatabase db = this.getWritableDatabase();
    int filasAfectadas = db.delete(TABLE_ASANA, whereClause: COLUMN_ASANA_ID + "=?", new String[]{String.valueOf(idAsana)});
    return filasAfectadas > 0;
}
```

Figura 33: Código Borrar Asana BBDD

Votar y Control de votos:

```
// Método para registrar un voto
1 usage
private void votarAsana() {
    boolean resultado = db.votarAsana(asanaId, context: this);
    if (resultado) {
        Toast.makeText(context: this, text: "¡Voto registrado!", Toast.LENGTH_SHORT).show();
        actualizarContadorVotos(); // Actualizar los votos tras registrar uno nuevo
    } else {
        Toast.makeText(context: this, text: "Ya has votado esta asana.", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 34: Código Votar Asana

```
//Método para votar una asana
1 usage
public boolean votarAsana(int idAsana, Context context) {
    // Obtener el ID del usuario desde las preferencias compartidas
    int idUsuario = context.getSharedPreferences(name: "SesionUsuario", Context.MODE_PRIVATE).getInt(key: "USER_ID", defValue: -1);

    // Configurar valores para la inserción
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_SHANTI_HA_VOTADO, true);
    values.put(COLUMN_SHANTI_USUARIO_ID, idUsuario);
    values.put(COLUMN_SHANTI_ASANA_ID, idAsana);

    // Intentar insertar el voto
    long resultado = db.insert(TABLE_SHANTI, nullColumnHack: null, values);

    // Comprobar el resultado e informar al usuario
    if (resultado != -1) {
        Toast.makeText(context, text: "¡Voto registrado!", Toast.LENGTH_SHORT).show();
        return true;
    } else {
        Toast.makeText(context, text: "Ya has votado esta asana.", Toast.LENGTH_SHORT).show();
        return false;
    }
}
```

Figura 35: Código Insertar Voto en BBDD

```

2 usages
private void actualizarContadorVotos() {
    SQLiteDatabase db = this.db.getReadableDatabase();
    String query = "SELECT COUNT(*) FROM " + AsanaDatabase.TABLE_SHANTI +
        " WHERE " + AsanaDatabase.COLUMN_SHANTI_ASANA_ID + "=?";
    Cursor cursor = db.rawQuery(query, new String[]{String.valueOf(asanaId)});

    if (cursor != null && cursor.moveToFirst()) {
        int votos = cursor.getInt( columnIndex: 0);
        votosContador.setText("Votos: " + votos);
        cursor.close();
    }
}

```

Figura 36: Código Contador de votos

Aquí se puede ver como se aplican los filtros en la Búsqueda Avanzada y como se obtiene el ranking de asanas por votación

```

// Metodo para aplicar los filtros al darle al boton de aplicar filtros
1 usage
private void aplicarFiltros() {
    String tipoSeleccionado = tipoSpinner.getSelectedItem().toString();
    String dificultadSeleccionada = dificultadSpinner.getSelectedItem().toString();
    String parteCuerpoSeleccionada = parteCuerpoSpinner.getSelectedItem().toString();

    Cursor cursor = asanaDatabase.obtenerTodasLasAsanas();
    asanasList.clear();

    if (cursor != null && cursor.moveToFirst()) {
        do {
            String nombre = cursor.getString(cursor.getColumnIndexOrThrow(AsanaDatabase.COLUMN_ASANA_NOMBRE));
            String descripcion = cursor.getString(cursor.getColumnIndexOrThrow(AsanaDatabase.COLUMN_ASANA_DESCRIPCION));
            String variante = cursor.getString(cursor.getColumnIndexOrThrow(AsanaDatabase.COLUMN_ASANA_VARIANTE));
            String dificultad = cursor.getString(cursor.getColumnIndexOrThrow(AsanaDatabase.COLUMN_ASANA_DIFICULTAD));
            String parteCuerpo = cursor.getString(cursor.getColumnIndexOrThrow(AsanaDatabase.COLUMN_ASANA_PARTE_CUERPO));

            // Filtra las asanas según los valores seleccionados en los Spinners
            if ((tipoSeleccionado.equals("Cualquiera") || variante.equalsIgnoreCase(tipoSeleccionado)) &&
                (dificultadSeleccionada.equals("Cualquiera") || dificultad.equalsIgnoreCase(dificultadSeleccionada)) &&
                (parteCuerpoSeleccionada.equals("Cualquiera") || parteCuerpo.equalsIgnoreCase(parteCuerpoSeleccionada))) {
                asanasList.add(new Asana(nombre, variante, dificultad, parteCuerpo, descripcion));
            }
        } while (cursor.moveToNext());
    }
    if (cursor != null) cursor.close();

    if (asanasList.isEmpty()) {
        Toast.makeText( context: this, text: "No se encontraron resultados", Toast.LENGTH_SHORT).show();
    }
    asanaAdapter.notifyDataSetChanged();
}

```

Figura 37: Código Aplicar Filtros

```
// Metodo para obtener el ranking de asanas por votos cuando pulsas el boton de ranking
1 usage
private void obtenerRankingAsanas() {
    Cursor cursor = asanaDatabase.obtenerRankingAsanasPorVotos();
    asanasList.clear();

    if (cursor != null && cursor.moveToFirst()) {
        do {
            String nombre = cursor.getString(cursor.getColumnIndexOrThrow(AsanaDatabase.COLUMN_ASANA_NOMBRE));
            // Agrega la asana al listado
            asanasList.add(new Asana(nombre));
        } while (cursor.moveToNext());
    }
    if (cursor != null) cursor.close();

    if (asanasList.isEmpty()) {
        Toast.makeText(context, this, text: "No se encontraron resultados", Toast.LENGTH_SHORT).show();
    }
    asanaAdapter.notifyDataSetChanged();
}
}
```

Figura 38: Código Filtrar por ranking de votos

Ahora pasamos a ver la lógica de Rutinas (crearlas, mostrarlas y borrarlas):

```
// Método para guardar la rutina
1 usage
private void guardarRutina() {
    if (asanasSeleccionadas.isEmpty()) {
        Toast.makeText(context, this, text: "La rutina está vacía. Añade al menos una asana.", Toast.LENGTH_SHORT).show();
        return;
    }

    String nombreRutinaTexto = nombreRutina.getText().toString().trim();
    if (nombreRutinaTexto.isEmpty()) {
        Toast.makeText(context, this, text: "Introduce un nombre para la rutina.", Toast.LENGTH_SHORT).show();
        return;
    }

    // Crear una nueva instancia de Rutina
    Rutina rutina = new Rutina(nombreRutinaTexto, new ArrayList<>(asanasSeleccionadas));

    boolean resultado = db.anadirRutina(rutina.getNombre(), rutina.generarDescripcion(), userId);

    if (resultado) {
        // Obtener el ID de la rutina recién guardada
        int rutinaId = db.obtenerRutinaId(rutina.getNombre(), userId);

        // Añadir cada asana a la rutina
        for (int i = 0; i < asanasSeleccionadas.size(); i++) {
            Asana asana = asanasSeleccionadas.get(i);
            int asanaId = db.obtenerAsanaId(asana.getNombre()); // Obtén el ID de la asana
            db.anadirAsanaARutina(rutinaId, asanaId, i); // Asocia la asana a la rutina
        }

        Toast.makeText(context, this, text: "Rutina guardada exitosamente", Toast.LENGTH_SHORT).show();
        limpiarRutinaActual();
    } else {
        Toast.makeText(context, this, text: "Error al guardar la rutina.", Toast.LENGTH_SHORT).show();
    }
}
}
```

Figura 39: Código Crear Rutina

```
//Metodo para agregar una nueva rutina a la base de datos
1 usage
public boolean anadirRutina(String nombre, String descripcion, int idUsuario) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_RUTINA_NOMBRE, nombre);
    values.put(COLUMN_RUTINA_DESCRIPCION, descripcion);
    values.put(COLUMN_RUTINA_USUARIO_ID, idUsuario);
    long result = db.insert(TABLE_RUTINA, nullColumnHack: null, values);
    return result != -1;
}

//Metodo para agregar una asana a una rutina, este metodo usa la tabla relacionada Rutina_has_Asana
1 usage
public void anadirAsanaARutina(int rutinaId, int asanaId, int orden) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_RUTINA_HAS_ASANA_RUTINA_ID, rutinaId);
    values.put(COLUMN_RUTINA_HAS_ASANA_ASANA_ID, asanaId);
    values.put(COLUMN_RUTINA_HAS_ASANA_ORDEN, orden);
    db.insert(TABLE_RUTINA_HAS_ASANA, nullColumnHack: null, values);
}
}
```

Figura 40: Código Insertar Rutina en BBDD

```
// Método para cargar las asanas de la rutina, cargo la rutina y las asanas por problemas de logica
1 usage
private void cargarAsanasDeRutina(int rutinaId) {
    Cursor cursor = db.getAsanasByRutina(rutinaId);

    // Limpiar la lista antes de añadir nuevas asanas
    asanasList.clear();

    if (cursor != null && cursor.getCount() > 0) {
        while (cursor.moveToNext()) {
            String nombre = cursor.getString(cursor.getColumnIndexOrThrow(AsanaDatabase.COLUMN_ASANA_NOMBRE));

            // Añadir la asana a la lista
            Asana asana = new Asana(nombre);
            asanasList.add(asana);

            // Log para ver qué datos estamos recibiendo
            Log.d("PantallaRutinaPersonalizada", "Asana: " + nombre);
        }
        cursor.close();
    } else {
        //Esto es para comprobar que no hay asanas en la rutina. Ya que mostraba las asanas en blanco
        // Log.d("PantallaRutinaPersonalizada", "No se encontraron asanas para esta rutina.");

        Toast.makeText(context: this, text: "No hay asanas en esta rutina.", Toast.LENGTH_SHORT).show();
    }

    // Notificar al adaptador que los datos han cambiado
    asanaAdapter.notifyDataSetChanged();
}
}
```

Figura 41: Código Mostrar Rutina


```

//Método para obtener las asanas de una rutina
1 usage
public Cursor getAsanasByRutina(int rutinaId) {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.rawQuery(
        sql: "SELECT a." + COLUMN_ASANA_NOMBRE + ", a." + COLUMN_ASANA_DESCRIPCION +
            " FROM " + TABLE_RUTINA_HAS_ASANA + " rha " +
            " JOIN " + TABLE_ASANA + " a ON rha." + COLUMN_RUTINA_HAS_ASANA_ASANA_ID + " = a." + COLUMN_ASANA_ID +
            " WHERE rha." + COLUMN_RUTINA_HAS_ASANA_RUTINA_ID + " = ?" +
            " ORDER BY rha." + COLUMN_RUTINA_HAS_ASANA_ORDEN,
        new String[]{String.valueOf(rutinaId)}
    );
}

```

Figura 42: Código consulta Mostrar Asanas de Rutina

```

// Método para confirmar la eliminación de la rutina
1 usage
private void confirmarEliminacion() {
    new AlertDialog.Builder(context: this)
        .setTitle("Eliminar Rutina")
        .setMessage("¿Estás seguro de que deseas eliminar esta rutina? Esta acción no se puede deshacer.")
        .setPositiveButton(text: "Eliminar", (dialog, which) -> eliminarRutina())
        .setNegativeButton(text: "Cancelar", listener: null)
        .show();
}

// Método para eliminar la rutina
1 usage
private void eliminarRutina() {
    boolean success = db.eliminarRutina(rutinaId);

    if (success) {
        Toast.makeText(context: this, text: "Rutina eliminada correctamente.", Toast.LENGTH_SHORT).show();
        // Redirigir al usuario a la pantalla anterior
        Intent intent = new Intent(packageContext: this, PantallaUsuario.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
        finish();
    } else {
        Toast.makeText(context: this, text: "Error al eliminar la rutina.", Toast.LENGTH_SHORT).show();
    }
}

```

Figura 43: Código Eliminar Rutina

```
// Metodo para eliminar una rutina y sus relaciones con las asanas
1 usage
public boolean eliminarRutina(int rutinaId) {
    SQLiteDatabase db = this.getWritableDatabase();

    try {
        // Inicia una transacción para garantizar consistencia en la eliminación
        db.beginTransaction();

        // Elimina la rutina de la tabla Rutina
        int deletedRutina = db.delete(
            TABLE_RUTINA,
            whereClause: COLUMN_RUTINA_ID + "=?",
            new String[]{String.valueOf(rutinaId)}
        );

        if (deletedRutina > 0) {
            // Si ambas operaciones fueron exitosas, confirma la transacción
            db.setTransactionSuccessful();
            return true;
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        // Finaliza la transacción
        db.endTransaction();
    }

    return false;
}
```

Figura 44: Código Eliminar Rutina de BBDD

Pruebas.

Pruebas Unitarias.

Las pruebas unitarias realizadas en **AsanaYoga** han evaluado cada componente de forma aislada para garantizar su correcto funcionamiento antes de la integración.

Pruebas de usuario.

1. **Navegación entre pantallas:** se verificó que al interactuar con los botones de navegación (por ejemplo, "Iniciar sesión", "Registrar usuario", o "Crear rutina"), el sistema redirige a la pantalla correspondiente sin errores. Todas las rutas de navegación fueron probadas con éxito.
2. **Validación de formularios:** los formularios para el inicio de sesión y el registro validan correctamente la información introducida, mostrando mensajes de error si un campo está vacío o si el formato del correo es incorrecto. Por ejemplo, se comprobó que al ingresar un correo no válido el sistema alerta al usuario con un mensaje claro.
3. **Interacción con la interfaz de usuario:** se verificó que los botones, campos de texto y demás elementos respondieran adecuadamente a las acciones del usuario, como seleccionar asanas para una rutina o votar una postura.

Pruebas por parte del servidor.

1. **Gestión de datos de usuarios:** se probó la funcionalidad de registro e inicio de sesión asegurando que los datos del usuario se almacenan correctamente en la base de datos SQLite y se recuperan sin errores.
2. **Comunicación con la base de datos:** se comprobó que las operaciones de lectura y escritura (crear rutinas, añadir asanas, registrar votos) funcionan correctamente. Por ejemplo, al agregar una nueva rutina, esta aparece inmediatamente en la lista del usuario.
3. **Integridad de datos:** se verificó que los datos relacionados con las asanas, como nombre, variantes y nivel de dificultad, se mantienen consistentes al ser almacenados y recuperados.

Pruebas de Integración.

Las pruebas de integración aseguraron que los módulos individuales interactuaran correctamente entre sí y con la base de datos.

1. **Flujo entre pantallas:** se realizó un flujo completo desde el inicio de sesión hasta la creación de una rutina, verificando que todos los módulos (Autenticación, Gestión de Asanas y Rutinas Personalizadas) interactúan de manera fluida.
2. **Actualización en tiempo real:** se probó el sistema de votación comunitaria, asegurando que, al votar por una postura, las valoraciones se actualizan correctamente y aparecen reflejadas en la interfaz del usuario.

3. **Sincronización de datos:** se comprobó que las rutinas creadas por el usuario contienen las asanas seleccionadas, validando que los datos de ambos módulos (Gestión de Asanas y Rutinas Personalizadas) se integren sin pérdida de información.

Pruebas de Sistema.

Estas pruebas evaluaron la aplicación en su conjunto para verificar que cumple con los requisitos funcionales y no funcionales.

1. **Pruebas de rendimiento:** se comprobó la velocidad de carga al consultar la base de datos con múltiples asanas almacenadas. Por ejemplo, se verificó que el tiempo de respuesta en la búsqueda de asanas no exceda los dos segundos.
2. **Pruebas de usabilidad:** se realizaron pruebas con usuarios finales para evaluar la facilidad de uso de la aplicación, identificando que la navegación y la interfaz son intuitivas.
3. **Pruebas de seguridad:** se verificó que los datos del usuario (como contraseñas) se almacenan de manera segura en SQLite y que los formularios de inicio de sesión no permiten entradas inseguras, como inyecciones de código.

Este conjunto de pruebas garantiza que **AsanaYoga** no solo cumple con los requisitos funcionales, sino que también ofrece una experiencia de usuario fluida y confiable.

Planificación del desarrollo y puesta en marcha del proyecto.

El desarrollo de **AsanaYoga** se ha estructurado en una planificación integral que abarca desde la concepción de la idea hasta su despliegue. Este enfoque se basa en fases bien definidas, con una secuencia lógica de tareas, dependencias claras y recursos asignados, lo que garantiza el cumplimiento de los objetivos del proyecto.

1. Fases de desarrollo y cronograma.

La ejecución de **AsanaYoga** se divide en las siguientes fases:

1.1. Planificación y definición de requisitos (2 semanas).

- **Descripción:**
 - Identificación de las funcionalidades clave, como la gestión de asanas, la personalización de rutinas y el sistema de votación comunitaria.
 - Diseño preliminar de la estructura de la base de datos, incluyendo entidades y relaciones principales.
 - Maquetación inicial de la interfaz, definiendo las pantallas principales y la navegación entre ellas.
- **Duración:** 2 semanas.
- **Recursos necesarios:**
 - Herramientas de diseño como Figma para la creación de prototipos.
 - Documentación técnica para definir requisitos funcionales y no funcionales.
- **Dependencias:**
 - Esta fase es esencial para establecer las bases del desarrollo; no se puede iniciar el desarrollo sin completarla.

1.2. Desarrollo del sistema (8 semanas).

- **Semana 1-2:** creación de la interfaz inicial con Android Studio y Material Design.
- **Semana 3-4:** implementación de la base de datos SQLite y desarrollo de la lógica de negocio, como la gestión de usuarios, asanas y rutinas.
- **Semana 5-8:** integración de los componentes desarrollados y optimización de la interfaz para mejorar la experiencia de usuario.
- **Recursos necesarios:**
 - Android Studio, Java para el desarrollo.
 - Herramientas de prueba como emuladores y dispositivos físicos.

1.3. Pruebas (4 semanas).

- **Semana 9-10:**
 - Ejecución de pruebas unitarias para validar cada componente por separado.
 - Pruebas de integración para asegurar la interacción correcta entre módulos.
- **Semana 11-12:**
 - Pruebas alfa con usuarios internos.
 - Pruebas beta con un grupo reducido de usuarios externos para obtener retroalimentación.
- **Recursos necesarios:**
 - Emuladores Android, dispositivos físicos y herramientas de seguimiento de errores como GitHub Issues.

1.4. Despliegue y puesta en marcha (2 semanas).

- **Descripción:**
 - Publicación de la aplicación en Google Play.
 - Revisión final en el entorno de producción para garantizar la funcionalidad y el rendimiento.
- **Duración:** 2 semanas.
- **Recursos necesarios:**
 - Cuenta de desarrollador en Google Play.
 - Dispositivos reales para pruebas finales.

A continuación, se muestra de manera más resumida, en forma de tabla, un pequeño resumen de la planificación acompañado de un diagrama de Grantt, para verlo más visualmente:

	Tareas a realizar	Duración	Comienza	Termina	Octubre	Noviembre	Diciembre	Enero
1	Planificación y definición de requisitos	2 semanas	31.09.24	15.10.24				
2	Desarrollo del sistema	8 semanas	16.10.24	11.12.24				
3	Pruebas	4 semanas	12.12.24	09.01.25				
4	Despliegue y puesta en marcha	2 semanas	10.01.25	24.01.25				

Figura 45: Diagrama de Grantt

Fase	Duración	Recursos necesarios	Dependencias
Definición de requisitos	2 semanas	Equipo de diseño y documentación	Ninguna
Desarrollo de interfaz	2 semanas	Android Studio, Material Design	Definición de requisitos
Implementación de la base de datos y lógica	2 semanas	SQLite, Java	Desarrollo de interfaz
Integración y optimización	4 semanas	Android Studio, herramientas de prueba	Implementación de base de datos y lógica
Pruebas	4 semanas	Emuladores, dispositivos físicos	Integración y optimización
Despliegue y puesta en marcha	2 semanas	Cuenta Google Play, recursos técnicos	Pruebas completas

2. Establecimiento de permisos y autorizaciones.

La aplicación solicita únicamente permisos estrictamente necesarios, en conformidad con el RGPD:

- **Permisos requeridos:**
 - **Cámara:** para permitir la subida de imágenes de rutinas personalizadas.
 - **Almacenamiento:** para guardar datos de usuario localmente.
 - **Red:** para sincronizar datos y realizar votaciones comunitarias.
- **Política de privacidad:**
 - Diseñada en conformidad con las normativas europeas, obliga al usuario a aceptarla antes de usar la aplicación.

3. Plan de prevención de riesgos

Se han identificado los principales riesgos y sus respectivas estrategias de mitigación:

1. Pérdida de datos sensibles:

- **Riesgo:** pérdida o robo de información de usuarios.
- **Mitigación:** uso de copias de seguridad automáticas, cifrado de datos y autenticación robusta.

2. Ataques cibernéticos:

- **Riesgo:** vulnerabilidades explotadas por hackers.
- **Mitigación:** implementación de pruebas de seguridad periódicas y actualizaciones constantes del sistema.

3. Problemas de rendimiento:

- **Riesgo:** saturación del sistema con un gran número de usuarios.
- **Mitigación:** optimización del código y pruebas de carga en etapas avanzadas.

4. Cumplimiento de normas de seguridad y ambientales.

La aplicación sigue estándares internacionales y nacionales:

- **Seguridad:** encriptación de datos y autenticación basada en OAuth 2.0.
- **Ambientales:** uso eficiente de recursos digitales y servidores optimizados para reducir el consumo energético.

5. Valoraciones económicas.

Aunque el proyecto AsanaYoga se desarrolla en un entorno educativo, se presentan los costos en un contexto profesional, considerando tanto costes directos como indirectos. El desglose es el siguiente:

Costes directos

- **Licencias de software y herramientas:**
 - Android Studio: gratis para uso personal/educativo.
 - Cuenta de desarrollador en Google Play: **25 €** (pago único).
 - Figma (plan profesional): **144 € al año**.
- **Hardware y dispositivos:**
 - Emuladores: incluidos en el software, sin coste adicional.
 - Dispositivos físicos para pruebas: **600 €** (móvil gama media-alta).
- **Recursos humanos:**
 - Equipo de desarrollo (2 desarrolladores, 2 meses, **1.500 €/mes cada uno**): **6.000 €**.
 - Diseñador UI/UX (1 mes, **2.000 €**): **2.000 €**.
 - Tester (2 semanas, **1.200 €**): **1.200 €**.

Costes indirectos

- **Infraestructura:**
 - Almacenamiento en la nube: **120 €/mes, 3 meses: 360 €**.
- **Imprevistos y mantenimiento inicial:** **500 €**.
- **Posibles expansiones del sistema y soporte:** estimación inicial incluida en imprevistos.

Resumen de Costes

El uso de herramientas gratuitas como Android Studio y SQLite ayuda a minimizar los costos iniciales, pero en un entorno profesional los gastos aproximados totales son:

Coste total estimado: 10.829 €.

6. Desarrollo y puesta en marcha.

El despliegue final incluye:

1. **Pruebas finales en dispositivos reales.**
2. **Publicación en la tienda oficial de Android.**
3. **Lanzamiento controlado con usuarios selectos antes de abrir la disponibilidad al público general.**

Definición de procedimientos de control y evaluación de la ejecución de proyectos.

El éxito de un proyecto como **AsanaYoga** depende no solo de la calidad de su diseño y desarrollo, sino también de una correcta evaluación y control de su ejecución. En este apartado se establecen los procedimientos necesarios para verificar si los objetivos planteados han sido alcanzados, analizar las soluciones implementadas, destacar aspectos relevantes y proyectar mejoras futuras.

1. Verificación del cumplimiento de objetivos.

1. Funcionalidades clave implementadas.

Funcionalidad	Descripción	Cumplimiento
Gestión de asanas	Verificación de que el catálogo incluya todas las posturas necesarias con variantes, beneficios y dificultad.	✓
Rutinas personalizadas	Validación de que los usuarios puedan crear, modificar y guardar rutinas de manera intuitiva.	✓
Votación comunitaria	Confirmación del funcionamiento del sistema de "Shantis" para interacción entre usuarios.	✓
Gestión de búsquedas	Sistema que permite buscar asanas por dificultad, beneficios o tipo	✓

2. Satisfacción de los usuarios finales.

Aspecto evaluado	Descripción	Cumplimiento
Encuestas en pruebas beta	Realización de encuestas para recopilar la experiencia general de los usuarios.	✓
Sugerencias registradas	Registro y análisis de sugerencias para incorporar mejoras en futuras versiones.	✓

3. Cumplimiento de requisitos no funcionales.

Requisito	Descripción	Cumplimiento
Rendimiento	Garantizar tiempos de respuesta menores a 2 segundos en consultas complejas.	✓
Seguridad	Confirmar el cifrado de datos sensibles y la conformidad con el RGPD.	✓

2. Evaluación de actividades mediante indicadores de calidad.

Para medir el desempeño de las actividades realizadas, se definen métricas que permitan un análisis objetivo:

- **Indicadores funcionales:**
 - Porcentaje de funcionalidades operativas después de las pruebas unitarias e integración (>90%).
 - Cantidad de errores críticos detectados durante las pruebas (debe ser menor al 5%).
- **Indicadores no funcionales:**
 - Disponibilidad de la aplicación durante las pruebas beta (>99%).
 - Opinión positiva en las encuestas (>85% de los usuarios satisfechos).

Estos indicadores permiten medir el éxito de cada etapa y detectar posibles áreas de mejora.

3. Registro, evaluación y solución de incidencias.

El manejo adecuado de incidencias es esencial para garantizar que el proyecto avance de manera fluida. Se establece un sistema organizado para este propósito:

1. **Registro de incidencias:**
 - Uso de herramientas como GitHub Issues para documentar problemas reportados durante el desarrollo y pruebas.
 - Clasificación de incidencias en **críticas**, **importantes** y **menores**, según su impacto.
2. **Evaluación y priorización:**
 - Las incidencias críticas (ejemplo: errores de conexión a la base de datos) se resuelven en un plazo máximo de 48 horas.
 - Las incidencias menores (ejemplo: ajustes en la interfaz) se agrupan y resuelven en las fases de optimización.
3. **Solución de incidencias:**
 - Asignación de responsabilidades específicas al equipo técnico para abordar cada problema.
 - Registro de soluciones implementadas para prevenir errores similares en futuras iteraciones.

4. Gestión de cambios en recursos y tareas.

El desarrollo de **AsanaYoga** implica adaptarse a imprevistos y ajustes. Se implementan procedimientos para gestionar cambios de manera efectiva:

- **Cambios en recursos humanos:**
 - Si un desarrollador abandona el equipo, se redistribuyen las tareas entre los miembros restantes.
- **Ajustes en el cronograma:**
 - Se reevalúan las fechas de entrega cuando una tarea excede el tiempo previsto, ajustando otras actividades sin comprometer los hitos finales.
- **Incorporación de nuevas herramientas:**
 - Si se identifican soluciones más eficientes durante el desarrollo, se analiza su viabilidad técnica y económica antes de adoptarlas.

5. Participación de los usuarios en la evaluación.

La retroalimentación de los usuarios es fundamental para identificar fortalezas y áreas de mejora. Se aplican los siguientes mecanismos:

- **Pruebas beta con usuarios seleccionados:**
 - Los usuarios proporcionan comentarios sobre la funcionalidad, usabilidad y diseño de la aplicación.
 - Se identifican posibles errores y sugerencias de mejora para futuras versiones.
- **Encuestas de satisfacción:**
 - Cuestionarios que evalúan aspectos como facilidad de uso, velocidad de respuesta y utilidad de las funciones.
- **Foros y canales de interacción:**
 - Espacios dentro de la aplicación para que los usuarios expresen sus opiniones y reporten problemas.

6. Conclusiones

La implementación de **AsanaYoga** ha cumplido con los objetivos principales: proporcionar una herramienta funcional, segura y atractiva para los entusiastas del yoga. Las soluciones adoptadas, como el uso de Android Studio, SQLite y Material Design, han demostrado ser adecuadas para los requerimientos del proyecto.

Líneas futuras

1. **Nuevas funcionalidades:** incorporar estadísticas personalizadas y un sistema de recompensas para fomentar el uso continuo.
2. **Ampliación del sistema de votación:** permitir comentarios junto a las valoraciones, incrementando la interacción comunitaria.

Referencias y bibliografía.

1. Desarrollo de aplicaciones móviles en Android:

- Google. (n.d.). *Developer Guides*. Disponible en <https://developer.android.com/guide>

2. Metodologías y prácticas para el desarrollo de software:

- Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide*.
 - Disponible en <https://scrumguides.org/>
- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

3. Diseño de aplicaciones de bienestar digital:

- Faster Capital. "Yoga startups y su impacto en el bienestar digital."
 - Disponible en <https://fastercapital.com/es/contenido/Equipos-de-yoga--Startups-y-equipos-de-yoga--encontrar-el-equilibrio-en-los-negocios.html>
- Noticias de Navarra. "Aplicaciones de yoga populares en la actualidad."
 - Disponible en <https://www.noticiasdenavarra.com/on-igandea-plus/2021/08/02/apps-yoga-populares-2122948.html>.

4. Cumplimiento de normativas y seguridad:

- Unión Europea. *Reglamento General de Protección de Datos (RGPD)*.
 - Disponible en <https://eur-lex.europa.eu/eli/reg/2016/679/oj>
- Consilium. "Horizon Europe y su apoyo a la innovación digital."
 - Disponible en <https://www.consilium.europa.eu/es/policies/horizon-europe/>.

5. Investigación y planificación de proyectos digitales:

- Next Generation EU. "Oportunidades para Startups."
 - Disponible en <https://nexteugeneration.com/>
- Chacon, S., & Straub, B. (2014). *Pro Git*. Apress.