CONSEJERÍA DE EDUCACIÓN
Comunidad de Madrid

IES ENRIQUE TIERNO GALVAN

Parla

CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Curso 2024/2025

Proyecto DAM

TITULO: Zenhabits

Alumno: Alba Almoril Benito

Tutor: Julián Parra Perales

Junio de 2025

CONTENIDO (índice en proceso)

PUEDES MIRAR TODO, tanto código como lo que no. Quiero una reunión urgente, me veo muy mal.

REPOSITORIO PÚBLICO EN GITHUB: https://github.com/albealvant/zenhabits-project

1. Contexto de la aplicación

1.1. Análisis del problema

La sociedad actual lleva un ritmo de vida rápido, el cual muchas veces dificulta el hecho de mantener costumbres saludables y ser productivos.

Según investigaciones recientes (Duhigg, 2012), la creación y mantenimiento de hábitos demandan constancia, repetición y motivación duradera. Sin embargo, la mayoría de las personas desisten en pocas semanas debido a la falta de disciplina, el olvido o la falta de motivación. A esto se le añade la escasez de herramientas eficaces que respondan a esa necesidad.

Hay numerosas aplicaciones en el mercado enfocadas en la administración de tareas (como Todoist o Trello) o hábitos (como Habitica o HabitNow), sin embargo, muchas de estas tienen restricciones: su utilización se basa únicamente en la conexión a internet, poseen interfaces poco intuitivas o no producen el compromiso adecuado.

1.2. Solución y objetivo

En respuesta a la necesidad de la sociedad actual de mantener hábitos saludables y de alcanzar metas personales en un mundo demasiado rápido y ajetreado; propongo el desarrollo de *ZENHABITS*, una aplicación multiplataforma diseñada para mejorar la productividad personal gracias a una combinación de gestión y "gamificación".

ZENHABITS no solo ofrece herramientas de organización; además, busca motivar al usuario a través de recompensas como logros, lo cual fomentará la constancia y el compromiso a largo plazo. También permitirá gestionar hábitos y metas de forma sencilla e intuitiva. Cada vez que el usuario complete/mantenga un hábito, será recompensado mediante logros y avances visibles en un personaje personalizable. Este personaje gastará energía, subirá de nivel o perderá vida según el grado de cumplimiento de las tareas, funcionando como una especie de reflejo del progreso personal.

El objetivo de este proyecto es que, ZENHABITS, consiga mantener la motivación y constancia de los usuarios ofreciéndoles una experiencia agradable, intuitiva y divertida. Además, como objetivo personal, aspiro a que el desarrollo de esta aplicación me proporcione nuevas habilidades, herramientas y lenguajes para mi crecimiento personal.

2. Análisis

El desarrollo técnico y general de ZENHABITS, se basa en que es una aplicación que ofrece una interfaz atractiva y multiplataforma, junto con un sistema de almacenamiento híbrido: combina el almacenamiento de datos local y sincronización en la nube. Para asegurar esta cumpla con las necesidades de los usuarios y mantenga una buena calidad, es imprescindible realizar un análisis detallado de los requisitos que guiarán todo el proceso de diseño e implementación.

2.1. Análisis de requisitos

Buscamos identificar las funcionalidades esenciales del sistema y las condiciones bajo las que se debe operar. Para identificarlas, haremos el análisis de requisitos, hay dos tipos:

2.1.1. Requisitos funcionales

- Crear, modificar y eliminar hábitos y metas.
- Asignar notificaciones para recordar hábitos y metas.
- Desbloquear logros en base al cumplimiento de hábitos y metas.

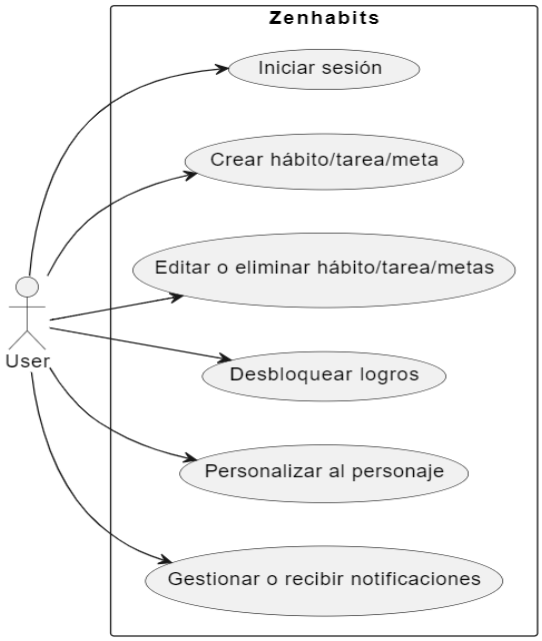
2.1.2. Requisitos no funcionales

- Usabilidad
  - La interfaz será intuitiva, clara y sencilla, permitiendo que cualquier usuario pueda manejarla sin necesidad de formación previa.
  - La experiencia de usuario se verá reforzada por un diseño visual minimalista, con iconografía reconocible y navegación fluida.
  - Se contempla una guía o ayuda básica accesible para apoyar al usuario.
- Portabilidad
  - La aplicación debe estar disponible para distintos tipos de dispositivos, incluyendo móviles y escritorio, y adaptarse correctamente a diferentes tamaños de pantalla mediante diseño responsivo.
- Fiabilidad
  - El sistema debe permitir su uso sin conexión a internet, garantizando que los datos esenciales sigan disponibles localmente.

- Se deben evitar pérdidas de información durante la sesión del usuario.

- <u>Seguridad</u>
  - El sistema garantizará la protección de datos personales y credenciales, asegurando que solo el usuario legítimo pueda acceder a su información.
  - El acceso estará protegido mediante mecanismos seguros de autenticación.
- <u>Eficiencia</u>
  - El sistema debe responder de forma rápida y fluida a las interacciones del usuario, incluso en dispositivos con recursos limitados.

2.2. Casos de uso



Los casos de uso, derivados de los requisitos funcionales, muestran la manera en que los "actores" interactúan con el sistema, estableciendo las funcionalidades principales desde el punto de vista de quien utiliza la aplicación.

En el caso de ZENHABITS, el actor principal es el usuario, el cual es la persona que interactúa con dicha aplicación. Este puede iniciar sesión; añadir, editar o eliminar hábitos o metas; desbloquear logros; personalizar a su personaje/avatar y gestionar o recibir notificaciones.

3. <u>Diseño</u>

3.1. Modelo arquitectónico

ZENHABITS ha sido diseñada para ofrecer una solución multiplataforma eficiente, escalable y segura. La propuesta busca ser funcional tanto en dispositivos móviles (Android e iOS) como en escritorio (Windows, macOS y Linux) y web (navegadores modernos) en fases futuras.

La solución sigue un modelo cliente-servidor. Se estructura internamente bajo los principios de Clean Architecture, permitiendo separación clara de toda su estructura. Además, se utilizarán prácticas de diseño adaptativo para garantizar interfaces "responsive" en distintos dispositivos y tamaños de pantalla.

3.1.1. Componentes

- Cliente Multiplataforma (Flutter + Dart)

  La aplicación se desarrolla con Flutter en Dart, lo que posibilita la compatibilidad multiplataforma (Android, iOS, escritorio (Windows, macOS y Linux) y web. También gestiona la interfaz de usuario, la lógica de presentación, el almacenamiento local a través de SQLite y la sincronización de datos por medio del protocolo HTTP seguro utilizando JWT para autenticación.

- API Backend (Rust + Axum)

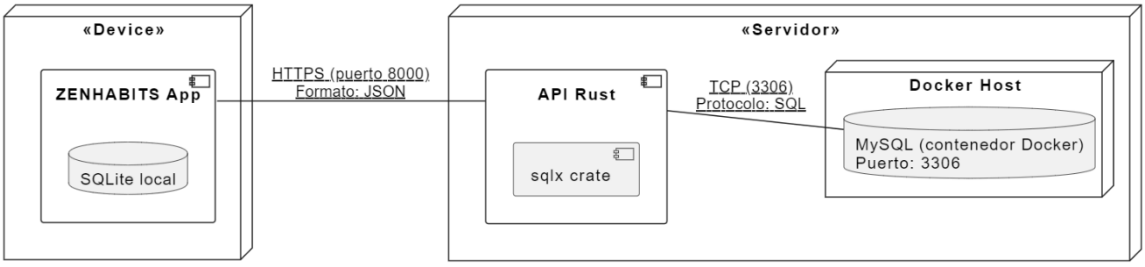
  La API está implementada en Rust utilizando el "framework" Axum para crear endpoints seguros y eficientes. Además, se utiliza SQLx para interactuar de manera asíncrona y segura con la base de datos MySQL alojada en la nube y la autenticación de usuarios se realiza mediante JSON Web Tokens (JWT).

- Base de Datos en la Nube (MySQL Docketizada)

  La base de datos MySQL se ejecuta dentro de un contenedor Docker, lo cual garantiza un entorno aislado, reproducible y escalable. Esta base de datos almacena de forma persistente y segura la información relativa a usuarios, hábitos, metas, logros y configuraciones, asegurando su integridad y consistencia.

- Servicios de Nube y Virtualización

  A futuro, se plantea ampliar el uso de Docker para incluir, además de la base de datos MySQL, otros componentes del sistema, como la API en Rust, consiguiendo una arquitectura más modular, escalable y sencilla de desplegar. Además, también se plantea la integración de notificaciones.



3.2. Plan de pruebas

El plan de pruebas define la estrategia que se seguirá para validar el correcto funcionamiento de ZenHabits antes de su puesta en producción. Su objetivo es asegurar que la aplicación cumpla con los requisitos funcionales, no funcionales y proporcione una buena experiencia de usuario.

3.2.1. Pruebas Funcionales

- Gestión de contenidos: Verificar que el usuario puede crear, editar y eliminar hábitos y metas, y que dichos cambios se reflejan tanto en la interfaz como en la persistencia de datos.
- Notificaciones y logros: Asegurar que al asignar recordatorios estos se generan correctamente y, al completar acciones definidas, se desbloquean los logros asociados.
- Personalización del personaje: Confirmar que la modificación del avatar (nivel o atributos) se actualiza de forma coherente y persistente.

3.2.2. Pruebas No Funcionales

- Compatibilidad multiplataforma: Realizar pruebas en diferentes dispositivos y resoluciones (Android, iOS, escritorio y web) para garantizar que la experiencia del usuario es consistente y los contenidos se adaptan correctamente.
- Funcionamiento offline: Comprobar la disponibilidad de datos mediante el almacenamiento local, asegurando que la aplicación funciona sin conexión y sincroniza correctamente cuando se restablece la conectividad.
- Seguridad: Evaluar el mecanismo de autenticación y autorización basado en JWT, verificando que sólo usuarios autorizados pueden acceder a la información.
- Rendimiento: Ejecutar pruebas de carga en la API y del sistema general para asegurar que la aplicación responde de forma fluida, incluso en condiciones de uso intensivo o en dispositivos con recursos limitados.

3.2.3. Pruebas Técnicas

- Pruebas de API (usando Postman):
    - Validación de Endpoints: Comprobar que cada endpoint responde según lo esperado: operaciones CRUD (crear, leer, actualizar, eliminar) sobre hábitos, metas, y demás datos.
    - Manejo de Errores y Seguridad: Verificar que las solicitudes no autorizadas se rechazan y que el manejo de errores es correcto (por ejemplo, mensajes de error claros cuando se envían datos incorrectos).
    - Pruebas de Carga y Rendimiento: Simular múltiples peticiones concurrentes para evaluar el tiempo de respuesta, la estabilidad y la robustez de la API (esto se puede hacer con un script en Python)
- Pruebas de Base de Datos:
    - Integridad de Datos: Comprobar que las operaciones realizadas desde la API (insertar, actualizar o eliminar) se reflejan correctamente en la base de datos y se mantienen las relaciones y restricciones definidas.
    - Persistencia y Sincronización: Verificar que la base de datos (MySQL en contenedor Docker) maneja correctamente la persistencia de la información y permite su recuperación sin errores.
    - Pruebas de Recuperación: Simular escenarios de fallo (por ejemplo, la pérdida de conexión) y comprobar que, al restablecerse, la sincronización se reanuda sin pérdida de datos.
    - Escalabilidad: Ejecutar pruebas que simulen un aumento en el volumen de transacciones, asegurando que la base de datos mantiene la integridad y rendimiento esperado.

3.2.4. Pruebas de Interfaz y Experiencia de Usuario (UI/UX)

- Usabilidad: Realizar pruebas con usuarios reales, donde se observe la facilidad para navegar por la aplicación, localizar funciones y realizar acciones de manera intuitiva.
- Verificar que la interfaz se ajusta adecuadamente a diferentes tamaños y orientaciones de pantalla.
- Asegurar que cada interacción (como pulsar botones o realizar gestos) produce respuestas visuales o sonoras que confirmen la acción realizada.

Una vez completadas todas estas pruebas de manera satisfactoria y se corrijan las incidencias detectadas, el sistema podrá considerarse preparado para su despliegue en producción. La documentación de cada fase de pruebas, junto con informes de rendimiento y de usabilidad, servirá para dar el visto bueno final al sistema.

4. Base de datos

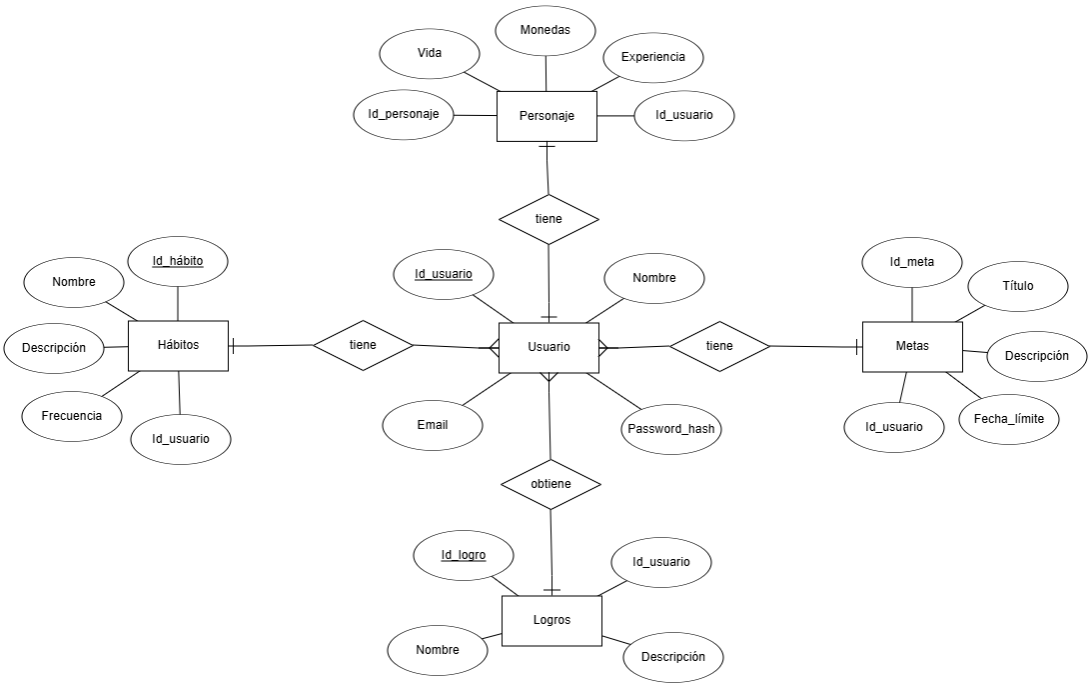
4.1. Contexto de la BD

La base de datos es una parte muy importante de una aplicación, ya que permite la persistencia de la información relacionada con hábitos, metas, configuraciones y logros de los usuarios.
En el caso de ZENHABITS se utilizarán dos entornos de almacenamiento:

- Almacenamiento local (SQLite): Permite el funcionamiento offline de la aplicación desde cualquier dispositivo.
- Almacenamiento en la nube (MySQL): Centraliza los datos para su sincronización y disponibilidad desde múltiples dispositivos a través de la API en Rust.

Como se menciona en apartados anteriores, la gestión de la base de datos remota se realizará mediante SQLx en Rust, proporcionando operaciones seguras, asincrónicas y eficientes. Además, el servidor de base de datos en MySQL estará dockerizado, facilitando su despliegue, escalabilidad y mantenimiento. Esta estructura híbrida de almacenamiento garantiza una experiencia de usuario fluida y continua, independientemente de la disponibilidad de red.

4.2. Entidades y MER (modelo Entidad Relación)



- **Usuario:** Representa a cada usuario registrado en la aplicación.
  - Atributos: Id_usuario, Nombre, Email, Password_hash.
- **Personaje:** Representa el avatar o personaje que evoluciona según los progresos del usuario.
  - Atributos: Id_personaje, Vida, Monedas, Experiencia, Id_usuario (clave foránea).
- **Hábitos:** Representa los hábitos que el usuario desea mantener.
  - Atributos: Id_hábito, Nombre, Descripción, Frecuencia, Id_usuario (clave foránea).
- **Tareas:** Representa las tareas que el usuario debe realizar cada día.
  - Atributos: Id_tarea, Nombre, Descripción, Completada, Id_usuario (clave foránea). HE DECIDIDO QUITARLO, tengo que actualizarlo
- **Metas:** Representa metas u objetivos mayores que el usuario desea alcanzar.
  - Atributos: Id_meta, Título, Descripción, Fecha_límite, Id_usuario (clave foránea).
- **Logros:** Representa los logros desbloqueados por los usuarios al cumplir objetivos o alcanzar hitos dentro de la aplicación.
  - Atributos: Id_logro, Nombre, Descripción, Id_usuario (clave foránea).

Usuario → puede tener múltiples → Hábitos, Tareas, Metas, Logros. *(1:N)*

Usuario → posee → Personaje *(1:1)*

Usuario → obtiene → Logros al cumplir tareas, hábitos o metas. *(1:N)*

5. Implementación

Para el desarrollo de ZENHABITS se han utilizado tecnologías actuales de desarrollo multiplataforma y backend, con almacenamiento local y en la nube (docketizado).

5.1. Entorno de desarrollo

- Entornos de desarrollo: Visual Studio Code, Android Studio, Docker Desktop y Git.
- Sistemas operativos: Android e iOS (para pruebas multiplataforma).

5.2. Tecnologías

Elemento	Tecnología	Función principal
App multiplataforma	Flutter (Dart)	UI adaptativa para Android, iOS, Web y Escritorio
Base de datos local	SQLite	Persistencia local de datos
API Backend	Rust + Axum	Creación de endpoints seguros y eficientes
ORM y acceso a datos	SQLx	Conexión segura y asíncrona con MySQL
Base de datos nube	MySQL	Gestión relacional de datos
Seguridad	JWT	Autenticación basada en tokens
Contenedores	Docker	Aislamiento y despliegue de servicios

Elemento	Tecnología	Función principal
Control de versiones	Git	Gestión de versiones, ramas y seguimiento de código
Plataforma de desarrollo	GitHub	Almacenamiento y gestión del proyecto
Pruebas de API	Postman	Testeo de peticiones y respuestas HTTP

- **Flutter (Dart)**: Framework para crear una app única que funcione en Android, iOS, web y escritorio en Dart, que es el lenguaje de programación para Flutter.
- **SQLite**: Motor de base de datos ligero para guardar datos localmente sin conexión.
- **Rust + Axum**: Backend seguro y rápido, ideal para crear APIs REST
- **Cargo**: Herramienta de gestión para proyectos en Rust.
- **SQLx**: Librería Rust para trabajar con bases de datos SQL de forma segura.
- **MySQL**: Sistema relacional de datos ejecutado en contenedor para facilitar despliegue y mantenimiento.
- **JWT**: Mecanismo de autenticación usando tokens cifrados.
- **Docker**: contenedores para MySQL y despliegue futuro de la API.
- **Git**: Control de versiones para el desarrollo del código.
- **GitHub**: Plataforma para almacenar el código en un repositorio y gestionar el proyecto.
- **Postman**: pruebas manuales de la API REST.

6. Interfaz

La interfaz de *ZENHABITS* se ha diseñado pensando en la simplicidad, accesibilidad y eficiencia de uso. El objetivo es ofrecer una experiencia coherente en todas las plataformas (Android, iOS, Web y Escritorio), aunque las primeras referencias visuales se basen en el diseño móvil.

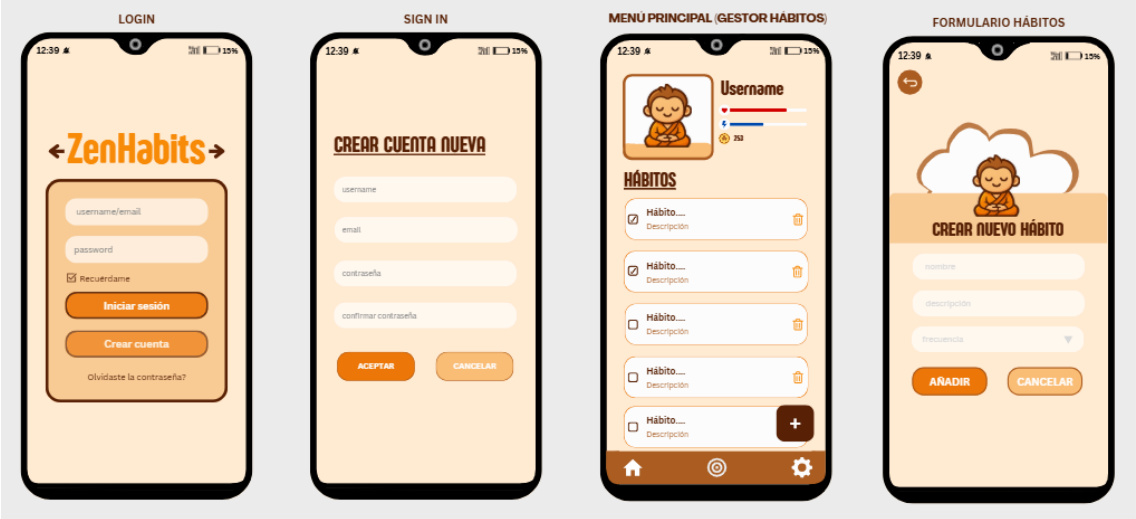
6.1. GUI

Su Interfaz Gráfica de Usuario presenta una estética limpia y minimalista, que facilita la navegación del usuario. Los componentes visuales están organizados de forma jerárquica para priorizar las acciones más comunes, como consultar hábitos, crear nuevas tareas y gestionar metas personales.

6.1.1. UI

La Interfaz de Usuario (UI) está compuesta por diferentes pantallas o vistas:

- Pantalla de Login/Registro: Introducción de credenciales para acceso seguro.
- Pantalla Principal Gestión de hábitos: Muestra hábitos activos y acceso rápido a crear nuevos hábitos.
    - Pantalla con el formulario de creación de hábitos.
- Pantalla de Gestión de tareas: Se podrán visualizar y navegar al formulario para añadir nuevas tareas.
    - Pantalla con el formulario de creación de tareas.
- Pantalla de Gestión de metas: Se podrán visualizar y navegar al formulario para añadir nuevas metas.
    - Pantalla con el formulario de creación de metas.
- Menú de navegación inferior: Accesos rápidos a hábitos, tareas, metas y perfil de usuario.



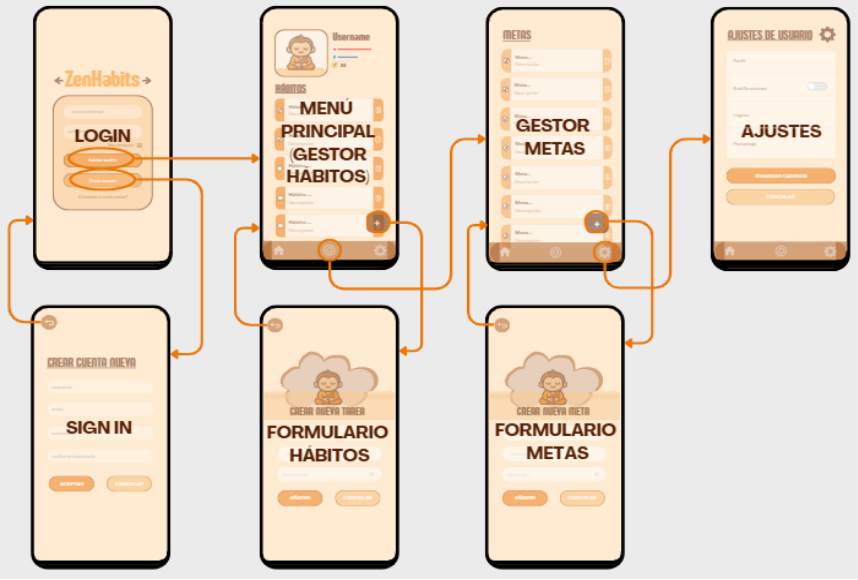
6.1.2. UX

La experiencia de usuario en *ZENHABITS* se centra en la simplicidad, la claridad y la motivación del usuario. Está diseñada para:

- Los formularios cortos y guiados, con validaciones claras para evitar errores.
- Navegación intuitiva: Todos los accesos principales están siempre visibles gracias a una barra de menú inferior y a botones flotantes.
- Retroalimentación inmediata: Cada acción muestra una respuesta (visual) que confirma su éxito o advierte en caso de error.

6.2. Diagrama de navegación

Este diagrama representa la estructura y flujo entre las pantallas principales de la aplicación en su versión móvil. Aunque en escritorio/web se adaptará con menús laterales u otras disposiciones, la lógica de navegación se mantiene.



6.3. Características visuales

(AQUÍ AÑADIRÉ ESTILOS ENTRE OTRAS COSAS)

6.4. Manual de usuario (se hará aparte)

(¿Se puede hacer en markdown o es obligatorio en word?)

7. Ejecución de pruebas - Informes

8. Puesta en producción

9. Elementos destacables del desarrollo

9.1. Innovaciones y problemas

10. Conclusiones

El desarrollo del proyecto ha generado un notable nivel de agobio debido a la falta de tiempo y a la dificultad para identificar qué aspectos reducir. Además, se han producido numerosos errores relacionados con la base de datos y el uso de Floor en Flutter, lo que ha complicado el avance y aumentado la carga de trabajo.

11. Anexos

Sigo sin saber que poner en los anexos

12. Bibliografía