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)

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, una buena organización de tareas 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, tareas y metas de forma sencilla e intuitiva. Cada vez que el usuario complete una tarea o 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

El análisis de requisitos permite identificar las funcionalidades esenciales del sistema y las condiciones bajo las que se debe operar.

2.1.1. Requisitos funcionales

Estos son las funciones específicas que el sistema debe realizar, es decir, *QUÉ* puede hacer el usuario con la aplicación:

- Crear, modificar y eliminar hábitos, tareas y metas.
- Asignar notificaciones para recordar hábitos y tareas.
- Desbloquear logros en base al cumplimiento de hábitos, tareas y metas.
- Personalizar al personaje virtual, cuyo estado depende del progreso del usuario en cuanto a sus acciones.

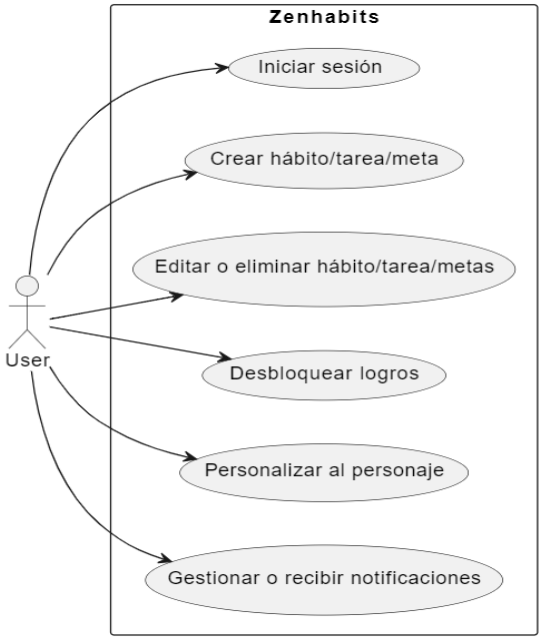
2.1.2. Requisitos no funcionales

Estos definen las características generales del sistema, así como las restricciones técnicas o de diseño:

- Compatibilidad multiplataforma con dispositivos Android e iOS mediante Flutter.
- Funcionamiento "offline" mediante almacenamiento local con SQLite.

- Una interfaz intuitiva, minimalista y adaptada a distintos tamaños de pantallas ("responsive").
- Seguridad de la API garantizada mediante autenticación con tokens JWT ("Json Web Tokens").
- Realización de pruebas con dispositivos físicos para validar la experiencia del usuario.

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, tareas o metas; desbloquear logros; personalizar a su personaje/avatar y gestionar o recibir notificaciones.

3. Diseño

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.

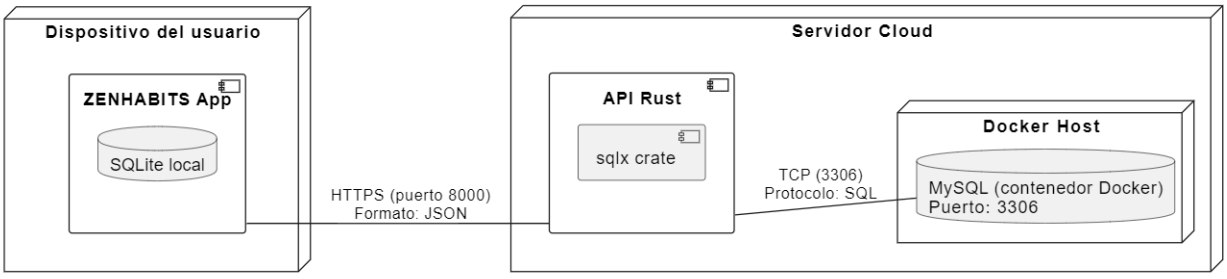
3.1.1. Componentes

- 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, tareas, 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.

(ESTÁ EN PROCESO)

4. Base de datos

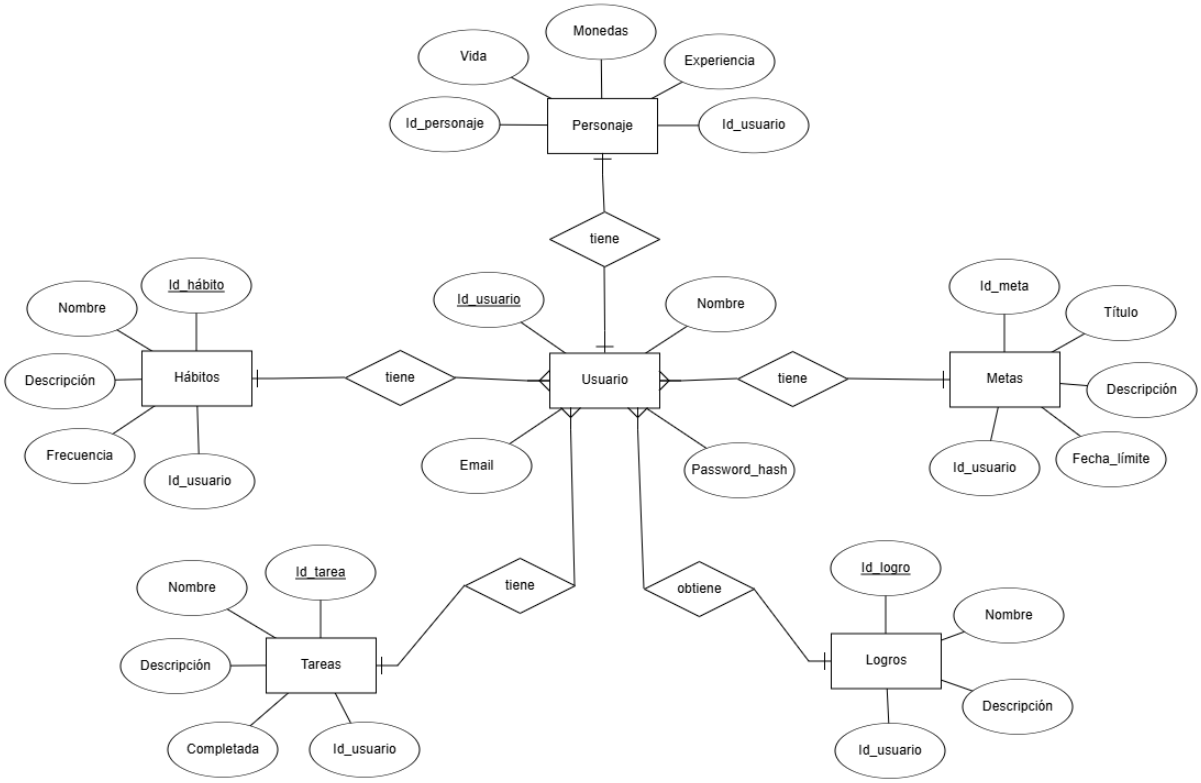
4.1. Contexto de la BD

La base de datos es una parte muy importante de una aplicación, ya que permite la persistencia de la información relacionada con hábitos, tareas, 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).
- **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.
- Sistemas operativos: Android e iOS (para pruebas multiplataforma).

5.2. Tecnologías

Componente	Tecnología	Descripción
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	Framework web rápido y seguro
Base de datos nube	MySQL	Gestión relacional de datos
Seguridad	JWT	Autenticación y autorización segura
Contenedores	Docker	Despliegue de API en producción

- **Flutter SDK**: desarrollo de la aplicación cliente.
- **Dart**: lenguaje de programación para Flutter.
- **Rust + Cargo**: desarrollo del backend.
- **Axum**: framework web en Rust para construcción de APIs REST.
- **sqlx (crate Rust)**: acceso a la base de datos MySQL desde la API de forma asincrónica.
- **Docker**: contenedores para MySQL y despliegue futuro de la API.
- **Postman**: pruebas manuales de la API REST.
- **GitHub**: control de versiones.

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. IxD

(NO SE QUE PONER AQUÍ)

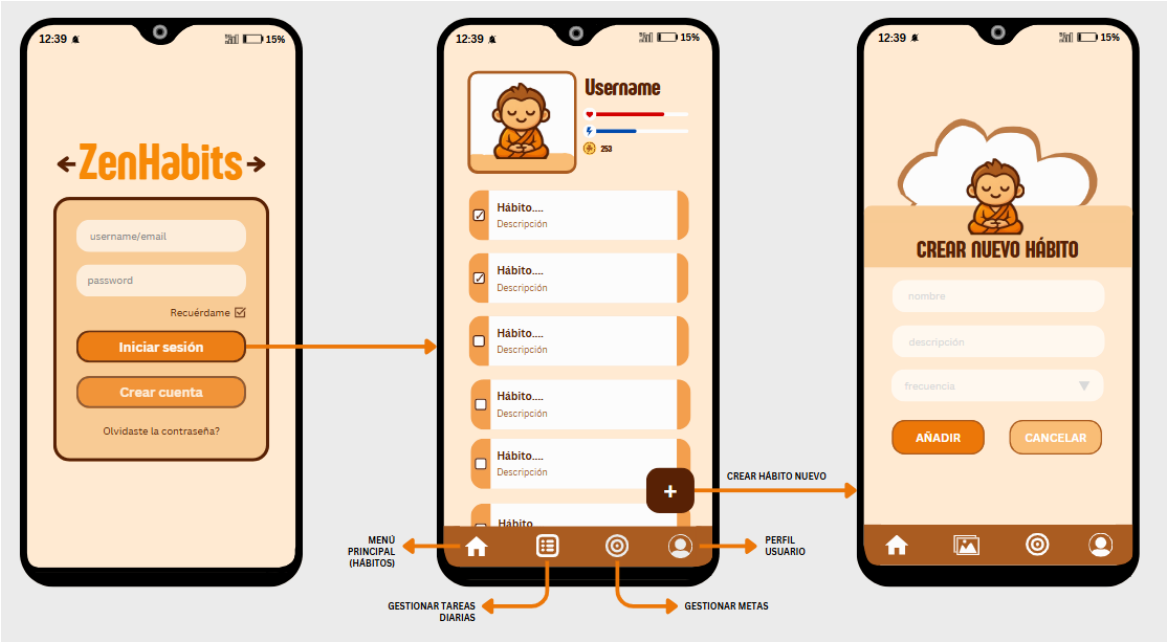
6.1.3. UX

6.2. Diagrama de navegación

Este ilustra cómo se relacionan las pantallas principales de la aplicación.

Este diseño será adaptado a pantallas más grandes como escritorio y web (por ejemplo, usando menús laterales).

(No sé si debo hacerlo más amplio, aunque luego solo funcionen los hábitos, es decir, hacer todas las pantallas incluyendo metas y tareas incluso el perfil para la gestión de notificaciones. Además, no sé si debería simplificarlo y que fuera menos visual y colorido)



6.3. Características visuales

(AQUÍ AÑADIRÉ ESTILOS ENTRE OTRAS COSAS)

6.4. Manual de usuario

7. Ejecución de pruebas - Informes

8. Puesta en producción

9. Elementos destacables del desarrollo

9.1. Innovaciones y problemas

10. Conclusiones

11. Anexos

(NO SE MUY BIEN QUE DEBO AÑADIR AQUÍ)

12. Bibliografía