UNIVERSIDAD DE ESPECIALIDADES ESPÍRITU SANTO

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA EN COMPUTACIÓN

TÍTULO DE LA TAREA

Sistema de seguimiento de hábitos saludables.

Nombre de los autores

García Coronado Ariel Alejandro

Zambrano Vera Ángeles Salome

Fierro García Patricio David

Guach Aguilar Richard Matthew

Asignatura y paralelo

Diseño de Software

Docente

Mgtr. Vanessa Alexandra Jurado Vite

Fecha de entrega

21/4/2025

Samborondón-Ecuador

Índice

Sistema de Seguimiento de Hábitos Saludables

Objetivo General

Desarrollar una aplicación destinada al registro y seguimiento de hábitos saludables, aplicando los principios y buenas prácticas del diseño de software. Permitiendo a los usuarios monitorear rutinas diarias relacionadas con su bienestar físico y emocional, tales como hidratación, actividad física, sueño, alimentación y estado de ánimo. Asimismo, se busca fomentar el desarrollo de soluciones colaborativas utilizando herramientas como GitHub y metodologías ágiles, priorizando la calidad técnica, el modularidad y la experiencia del usuario.

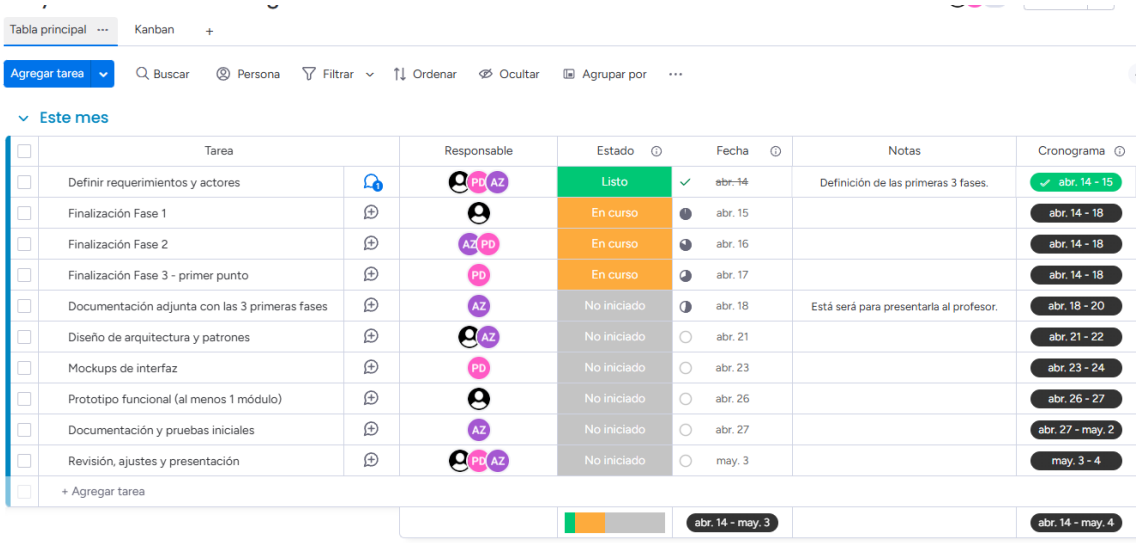
Fase 1: Planificación y coordinación

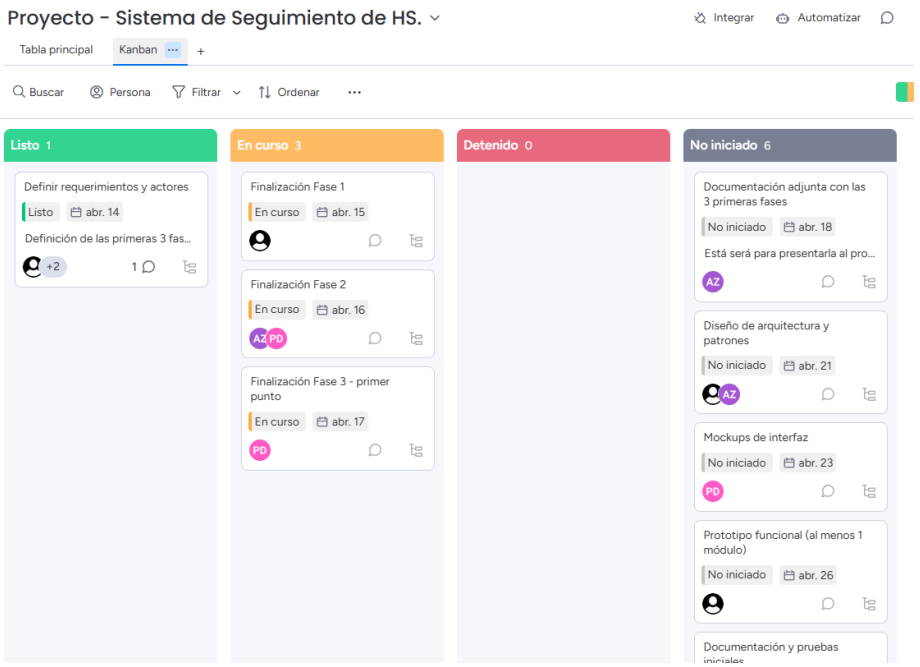
Definición de roles y responsabilidades.

Rol	Integrante	Responsabilidades Principales
Líder de Proyecto	Ángeles Zambrano	Coordinación general del equipo, validación de entregables, gestión del repositorio en GitHub.
Desarrollador Full Stack	Ariel García	Maquetación, desarrollo de interfaces, consumo de APIs, lógica del sistema, desarrollo del backend (API REST), y gestión de base de datos.
Diseñador UX/UI	David Fierro	Diseño de mockups, creación de pantallas, y mejora de la experiencia de usuario.
Documentador / QA	Richard Guach	Elaboración de bitácoras, ejecución de pruebas, y documentación técnica y funcional del sistema. .

Cronograma de trabajo

El equipo utiliza Monday.com como plataforma principal para la planificación, gestión y seguimiento de tareas, implementando una vista Kanban reforzada con un calendario semanal. Las actividades se organizan por entregables correspondientes a cada fase del proyecto, lo que facilita la visibilidad del avance, la trazabilidad de los procesos y un control efectivo del progreso en cada etapa del desarrollo.





Bitácora de decisiones

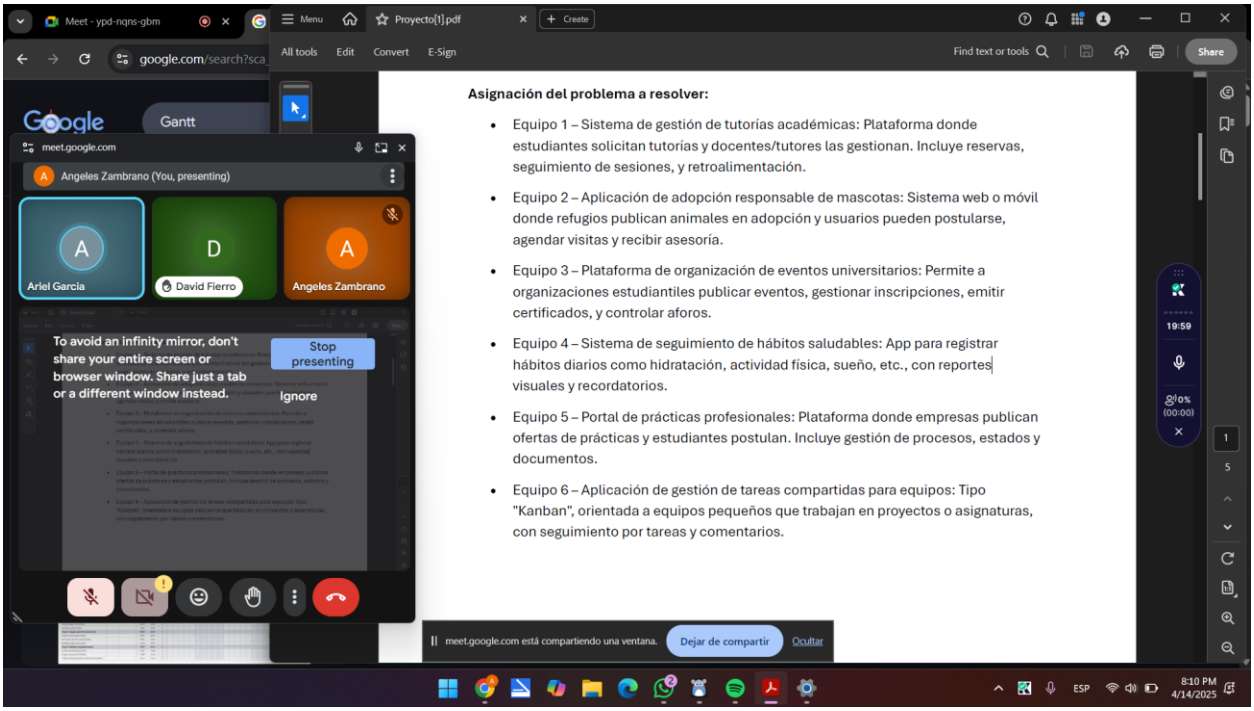
Las decisiones clave del proyecto son registradas en el Wiki del repositorio de GitHub. Cada entrada contiene la fecha, los participantes, el tema discutido, las decisiones tomadas y las acciones próximas.

Evidencia de reuniones y acuerdos.

Las reuniones se realizaron a través de Google Meet, y se documentaron mediante capturas de pantalla. Los acuerdos y decisiones también fueron registrados en el grupo de WhatsApp del equipo, lo que permitió mantener una comunicación continua y trazabilidad de las discusiones.

Primera Reunión

La primera reunión del equipo se llevó a cabo el lunes 14 de febrero a las 20:00 horas. Durante este encuentro inicial, se realizó la presentación de los integrantes y se discutió la asignación de roles, así como las responsabilidades específicas que asumiría cada miembro dentro del proyecto.



Planificación de ramas y flujo de trabajo colaborativo en GitHub

Modelo adoptado: Git Flow

El equipo ha optado por implementar el modelo Git Flow para estructurar el desarrollo de manera eficiente y colaborativa. Las ramas serán gestionadas a través de Git usando la terminal, con Visual Studio Code como entorno principal de desarrollo.

Estructura de ramas:

- **main**: Rama estable, destinada a versiones listas para producción o entrega.
- **develop**: Rama principal donde se integran las nuevas funcionalidades antes de su despliegue en producción.
- **feature/<nombre>**: Rama dedicada al desarrollo de nuevas funcionalidades específicas (por ejemplo, feature/login).
- **bugfix/<nombre>**: Rama para la corrección de errores detectados antes del lanzamiento.
- **release/<versión>**: Rama para la preparación y pruebas previas al lanzamiento.
- **hotfix/<urgente>**: Rama para correcciones urgentes que se realizan directamente sobre la rama **main**.

Herramientas y prácticas adoptadas

Durante esta fase, se implementaron diversas herramientas y metodologías que promovieron la colaboración y la calidad del código:

- Uso de Git por terminal y Visual Studio Code como entorno de desarrollo principal.
- Revisión del código mediante Pull Requests, garantizando que todo cambio pase por una validación previa antes de integrarse en develop o main.
- Activación de Branch Protection Rules en GitHub, evitando modificaciones directas a ramas protegidas y asegurando la estabilidad de las versiones entregables.

Fase 2: Análisis del problema y requerimientos

Descripción general del sistema

El sistema propuesto es una aplicación enfocada en el registro y monitoreo de hábitos saludables. La aplicación ayudará a los usuarios a monitorear hábitos relacionados con su bienestar físico y emocional. Permitirá realizar un seguimiento de hábitos como hidratación, actividad física, sueño, alimentación y estado de ánimo, todo con el fin de mejorar el bienestar general del usuario.

Identificación de Actores y Funcionalidades

Actores Principales:

Usuario Final: Interactúa con la aplicación para registrar y consultar hábitos. También puede establecer metas y configurar alertas.

Sistema: Administra los datos ingresados por el usuario y presenta los resultados visuales.

Módulo de Notificaciones: Enviará recordatorios automáticos conforme a la configuración del usuario.

Funcionalidades Clave:

- Registro diario de hábitos (agua, actividad, sueño, alimentación, estado de ánimo).
- Configuración de metas y alertas.
- Visualización del progreso mediante gráficos e indicadores.
- Personalización de hábitos.
- Sincronización y respaldo en la nube.

Requisitos funcionales y no funcionales

Los requerimientos del sistema se dividen en funcionales y no funcionales. Los requerimientos funcionales definen las acciones y comportamientos específicos que el sistema debe realizar, mientras que los no funcionales especifican las condiciones de calidad que debe cumplir el sistema.

Requisitos Funcionales

Los requisitos funcionales describen las capacidades concretas que debe poseer un sistema, es decir, las operaciones o procesos que el software debe ejecutar para satisfacer las necesidades del usuario. De acuerdo con (Visure Solutions, 2023), estos requisitos reflejan las propiedades perceptibles del sistema, y su implementación resulta esencial para asegurar que la solución desarrollada cumpla con los objetivos establecidos.

En este contexto, se han identificado los siguientes requisitos funcionales esenciales:

Registro de hábitos: El usuario tendrá la posibilidad de documentar información diaria acerca de su ejercicio, hidratación, descanso, dieta y humor.

Establecimiento de metas: Los usuarios tendrán la posibilidad de definir objetivos únicos para cada hábito, lo que les facilitará monitorear su avance.

Notificaciones: La aplicación enviará alertas cuando el usuario no haya registrado un hábito o para recordarle sus metas diarias.

Visualización del Progreso: Los usuarios tendrán acceso a gráficos y reportes que muestran su progreso en los distintos hábitos.

Sincronización: Los datos se sincronizarán entre dispositivos y se almacenarán en la nube.

Requisitos No Funcionales

Los requisitos no funcionales (NFR, por sus siglas en inglés) son atributos de calidad que definen cómo debe comportarse un sistema, más allá de las funciones que debe realizar. De acuerdo con (Visure Solutions, 2024), estos requisitos establecen limitaciones y estándares relacionados con el rendimiento, la seguridad, la usabilidad, la confiabilidad y otros aspectos que impactan directamente en la experiencia del usuario y en la calidad del software.

En el contexto de este proyecto, se han identificado los siguientes requisitos no funcionales esenciales:

Usabilidad: Es fundamental que la aplicación sea sencilla de manejar, con una interfaz clara e intuitiva que facilite una navegación suave.

Rendimiento: El sistema debe ser ágil y rápido, con tiempos de respuesta mínimos.

Seguridad: Los datos del usuario deben ser manejados de forma segura, respetando las normativas de privacidad y protección de datos.

Escalabilidad: La aplicación debe tener la capacidad de afrontar un incremento en el número de usuarios sin comprometer su rendimiento.

Fase 3: Diseño Arquitectónico

Estilo Arquitectónico Adoptado

Para este proyecto, se optó por la arquitectura Modelo-Vista-Controlador (MVC), la cual simplifica la separación de las capas de presentación, lógica de negocio, y el acceso a datos. Esta perspectiva posibilita un mantenimiento y escalabilidad más eficaces conforme el sistema progresa.

Modelo: Administra la base de datos y la lógica de negocio, garantizando que la información del usuario sea procesada y guardada de manera correcta.

Vista: Es la interfaz de usuario que presenta la información al usuario final de manera comprensible y fácil de entender.

Controlador: Se desempeña como un enlace entre el modelo y la vista, administrando las interacciones del usuario y actualizando la vista en base a las modificaciones en los datos.

Justificación de la Arquitectura

Se seleccionó el modelo MVC ya que posibilita un desarrollo más ordenado y modular, lo cual promueve la escalabilidad, el mantenimiento y la incorporación de nuevas funciones en el futuro. Además, este modelo promueve la separación de responsabilidades, lo que incrementa la calidad y el desempeño del sistema.

Bibliografía

- Visure Solutions, I. (2023). Obtenido de https://visuresolutions.com/es/blog/requerimientos-funcionales/
- Visure Solutions, I. (2024). Obtenido de https://visuresolutions.com/es/blog/non-functional-requirements/