



Universidad Espiritu Santo

Modalidad en Línea

Ingeniería en Ciencias de la Computación

DISEÑO DE SOFTWARE

Docente : Mgtr. Vanessa Alexandra Jurado

Estudiantes : Ariel Alejandro García Coronado

Patricio David Fierro García

Richard Mathew Guach Aguilar

Ángeles Salome Zambrano

TEMA

Proyecto: Sistema de seguimiento de hábitos saludables: App para registrar hábitos diarios como hidratación, actividad física, sueño, con reportes visuales y recordatorios.

Índice

1. Definición de Roles y Responsabilidades	4
2. Cronograma de Trabajo	4
3. Bitácora de Decisiones	5
4. Evidencia De Reuniones y Acuerdos.	5
5. Planificación de Ramas y Flujo de Trabajo Colaborativo en Github.....	6
5.1 Estructura de Ramas.....	6
5.2 Herramientas y Prácticas Adoptadas.....	6
6. Descripción General del Sistema	7
7. Identificación de Actores y Funcionalidades	7
7.1 Actores Principales	7
7.2 Funcionalidades Clave	8
8. Requisitos Funcionales y no Funcionales	8
8.1 Requisitos Funcionales	9
8.2 Requisitos no Funcionales	9
9. Estilo Arquitectónico Adoptado.....	10
9.1 Justificación de la Arquitectura.....	11
10. Diagrama de Componentes o Módulos Principales	12
11. Diagramas de Clases, Secuencia y/o Estados	13
<i>Figura 12.1 Diagrama de Clases</i>	15
12. Modelo de Datos	18
<i>Figura 13.1. Modelo de Datos</i>	19

13.	Prototipo Funcional en Alta Fidelidad	20
14.	Control de Versiones en Github con Ramas por Integrante.....	20
15.	Documentación del Código y README del Proyecto.....	21
16.	Bibliografía	22
17.	Anexos.....	23
17.1	Cronograma de Trabajo	23
17.2	Evidencia de Reuniones	24

1. Definición de Roles y Responsabilidades

A continuación, se presenta una tabla que detalla los roles y responsabilidades de cada integrante del equipo en el proyecto. Esta división de tareas ha sido fundamental para asegurar una colaboración efectiva y la correcta ejecución de las actividades en cada fase del desarrollo del sistema. Cada miembro del equipo ha asumido responsabilidades clave que corresponden a sus áreas de especialización, contribuyendo al logro de los objetivos establecidos y al cumplimiento de los plazos del proyecto.

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.

Tabla 1. Roles y responsabilidades del equipo

2. Cronograma de Trabajo

El equipo ha empleado Monday.com como la plataforma central para la planificación, gestión y seguimiento de las tareas del proyecto. Se ha implementado una vista Kanban, complementada con

un calendario semanal, para organizar las actividades de manera eficiente. Las tareas se estructuran según los entregables asignados a cada fase del proyecto, lo que permite una visibilidad clara del progreso, asegura la trazabilidad de los procesos y facilita un control riguroso sobre el avance en cada etapa del desarrollo. Para más información revisar los Anexos.

3. Bitácora de Decisiones

Las decisiones clave del proyecto fueron cuidadosamente registradas en el Wiki del repositorio de GitHub, lo que facilita la documentación y el seguimiento de los acuerdos alcanzados a lo largo del desarrollo. Cada entrada contiene información detallada, como la fecha en que se tomó la decisión, los participantes involucrados en la discusión, el tema tratado, las decisiones adoptadas y las acciones a seguir. Este registro permite mantener una trazabilidad clara de las decisiones y asegura que todos los miembros del equipo tengan acceso a la información relevante para el avance del proyecto.

4. Evidencia De Reuniones y Acuerdos.

Las reuniones del equipo se realizaron a través de Google Meet y se documentaron mediante capturas de pantalla, además, los acuerdos y decisiones relevantes se documentaron en el grupo de WhatsApp, lo que facilitó una comunicación continua y aseguró la trazabilidad de las conversaciones y resoluciones tomadas.

Primera Reunión

La primera reunión del equipo se celebró el lunes 14 de febrero a las 20:00 horas. En este primer encuentro, se presentó a los miembros del equipo y se discutió la asignación de roles y las responsabilidades específicas que cada integrante asumiría dentro del proyecto.

Para acceder a información más específica sobre las reuniones y los acuerdos establecidos, se

sugiere revisar los Anexos, los cuales contienen las capturas de pantalla y los registros pertinentes.

5. Planificación de Ramas y Flujo de Trabajo Colaborativo en Github

El equipo ha adoptado el modelo Git Flow para estructurar el desarrollo del proyecto de manera eficiente y colaborativa. Este modelo facilita un manejo preciso de las ramas a través de Git, empleando la terminal y Visual Studio Code como ambiente principal de desarrollo.

5.1 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 destinada a correcciones urgentes realizadas directamente sobre la rama main.

5.2 Herramientas y Prácticas Adoptadas

Durante esta fase, se implementaron diversas herramientas y metodologías que promovieron tanto la colaboración como la calidad del código. Se utilizó Git a través de la terminal y Visual Studio Code como entorno principal de desarrollo. Además, se implementaron las siguientes prácticas:

Revisión del código mediante Pull Requests, asegurando que todo cambio pase por una validación previa antes de integrarse en las ramas develop o main.

Activación de Branch Protection Rules en GitHub, para evitar modificaciones directas en las ramas protegidas y garantizar la estabilidad de las versiones entregables.

6. Descripción General del Sistema

El sistema propuesto es una aplicación destinada al registro y monitoreo de hábitos saludables, con el objetivo de promover el bienestar integral de los usuarios. Esta herramienta permite a los usuarios hacer un seguimiento de diversos hábitos relacionados con su salud física y emocional, tales como la hidratación, la actividad física, el sueño, la alimentación y el estado de ánimo. A través de un enfoque integral, el sistema proporciona a los usuarios los recursos necesarios para tomar decisiones informadas y mejorar su calidad de vida, fomentando una mejor gestión de su bienestar general.

7. Identificación de Actores y Funcionalidades

En esta sección, se identifican los actores principales que interactúan con el sistema, así como las funcionalidades clave que permiten a la aplicación cumplir con los objetivos de seguimiento de hábitos saludables. Los actores y las funcionalidades están estrechamente relacionados, ya que cada actor tiene un rol que desempeñar dentro de las funcionalidades del sistema, lo que garantiza una experiencia de usuario completa y eficiente.

7.1 Actores Principales

Los actores son los componentes del sistema que interactúan directamente con la aplicación,

ya sea de forma activa o pasiva, para lograr sus objetivos. Los actores principales en el sistema incluyen:

Usuario Final: Interactúa con la aplicación para registrar y consultar hábitos. Además, tiene la capacidad de establecer metas y configurar alertas personalizadas.

Sistema: Administra los datos ingresados por el usuario y presenta los resultados visuales a través de gráficos e indicadores.

Módulo de Notificaciones: Ofrece notificaciones automáticas basadas en las configuraciones del usuario, tales como advertencias sobre hábitos no registradas u objetivos diarios no alcanzados.

7.2 Funcionalidades Clave

Las funcionalidades clave del sistema son las acciones y servicios que la aplicación ofrece para permitir a los usuarios registrar y seguir sus hábitos de manera efectiva. Estas funcionalidades abarcan varios aspectos del bienestar físico y emocional del usuario. Las principales funcionalidades del sistema incluyen:

Registro diario de hábitos: agua, actividad física, sueño, alimentación y estado de ánimo.

Configuración de metas y alertas personalizadas.

Visualización del progreso mediante gráficos e indicadores.

Personalización de hábitos según las necesidades del usuario.

Almacenamiento local de datos de hábitos saludables.

8. Requisitos Funcionales y no Funcionales

Esta sección presenta los requerimientos que definen el comportamiento y las cualidades que debe cumplir el sistema propuesto. Se dividen en dos categorías principales: requisitos

funcionales, que describen las acciones que el sistema debe ser capaz de realizar, y requisitos no funcionales, que establecen los estándares de calidad y condiciones operativas que debe respetar el sistema.

8.1 Requisitos Funcionales

Los requerimientos funcionales definen las habilidades que el sistema debe proporcionar para alcanzar sus metas. Tal como señala (Visure Solutions, 2023) , estos requisitos reflejan las propiedades observables del sistema y permiten verificar su funcionalidad mediante pruebas directas. Para el presente proyecto, se han identificado los siguientes requisitos funcionales clave:

Registro de hábitos: El sistema permitirá al usuario registrar diariamente información relacionada con sus hábitos de hidratación, ejercicio, sueño, alimentación y estado de ánimo.

Establecimiento de metas: Los usuarios podrán definir metas personalizadas para cada hábito, lo que les permitirá visualizar y medir su progreso con mayor claridad.

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

Visualización del progreso: Se ofrecerán reportes gráficos que representen el avance del usuario en los distintos hábitos, facilitando el análisis de su bienestar general.

Almacenamiento local de datos de hábitos saludables: El sistema deberá guardar los datos ingresados por el usuario de manera local en un archivo JSON para asegurar la persistencia de la información entre sesiones.

8.2 Requisitos no Funcionales

Los requisitos no funcionales (NFR, por sus siglas en inglés) establecen los criterios que determinan cómo debe funcionar el sistema, más allá de sus funcionalidades. De acuerdo con (Visure Solutions, 2024), estos requerimientos se centran en elementos como el desempeño, la seguridad, la facilidad de uso y la escalabilidad del software. Para este sistema, se han definido los siguientes:

Usabilidad: La interfaz debe ser intuitiva, clara y accesible para usuarios de distintos niveles técnicos, favoreciendo una experiencia fluida y agradable.

Rendimiento: El sistema debe garantizar tiempos de respuesta cortos y una ejecución eficiente, incluso cuando el volumen de datos aumente.

Seguridad: La información del usuario será manejada conforme a estándares de privacidad y protección de datos, evitando accesos no autorizados.

Escalabilidad: La arquitectura del sistema debe permitir la incorporación de nuevos usuarios y funcionalidades sin afectar su desempeño.

9. Estilo Arquitectónico Adoptado

Para el desarrollo del sistema de seguimiento de hábitos saludables se adoptó el estilo arquitectónico Modelo-Vista-Controlador (MVC), ampliamente utilizado en aplicaciones interactivas debido a su capacidad para separar claramente las responsabilidades entre la presentación, la lógica de negocio y el acceso a los datos. Este enfoque promueve un desarrollo modular, facilitando el mantenimiento, la escalabilidad y la posibilidad de incorporar nuevas funcionalidades sin afectar el resto del sistema.

Este patrón arquitectónico ha sido ampliamente adoptado en diversas aplicaciones debido a su capacidad para mejorar la mantenibilidad y escalabilidad del software. Según (Necula, 2024), el patrón MVC permite una separación clara de responsabilidades, lo que facilita el desarrollo paralelo y la reutilización de componentes en aplicaciones web y móviles.

A continuación, se describen los componentes que integran este modelo:

Modelo: Representa la capa de lógica de negocio y manejo de datos. En este sistema, se encarga de procesar la información relacionada con los hábitos del usuario y de almacenarla en un archivo estructurado (data.json), asegurando su integridad y persistencia.

Vista: Corresponde a la interfaz gráfica desarrollada con Tkinter, mediante la cual el usuario interactúa con la aplicación. Presenta los campos de entrada, botones, reportes y gráficos de forma clara y amigable.

Controlador: Actúa como intermediario entre la vista y el modelo. Gestiona los eventos generados por el usuario (como la introducción de datos o la solicitud de un reporte), coordinando la lógica de validación y actualización del modelo, así como la visualización de los resultados.

9.1 Justificación de la Arquitectura

La elección del patrón MVC se fundamenta en los siguientes beneficios:

Separación de responsabilidades: lo cual mejora la organización del código y facilita su mantenimiento.

Escalabilidad: permitiendo ampliar el sistema, por ejemplo, añadiendo nuevos hábitos o funcionalidades, sin alterar otras capas.

Reutilización de componentes: es posible modificar las vistas sin necesidad de cambiar la lógica del modelo ni del controlador, lo que permite mantener una estructura modular y flexible.

Facilidad de prueba y depuración: al poder testear cada componente por separado.

Estas características convierten al MVC en un enfoque ideal para sistemas centrados en la interacción del usuario y la visualización dinámica de datos, como lo es esta aplicación de seguimiento de hábitos.

10. Diagrama de Componentes o Módulos Principales

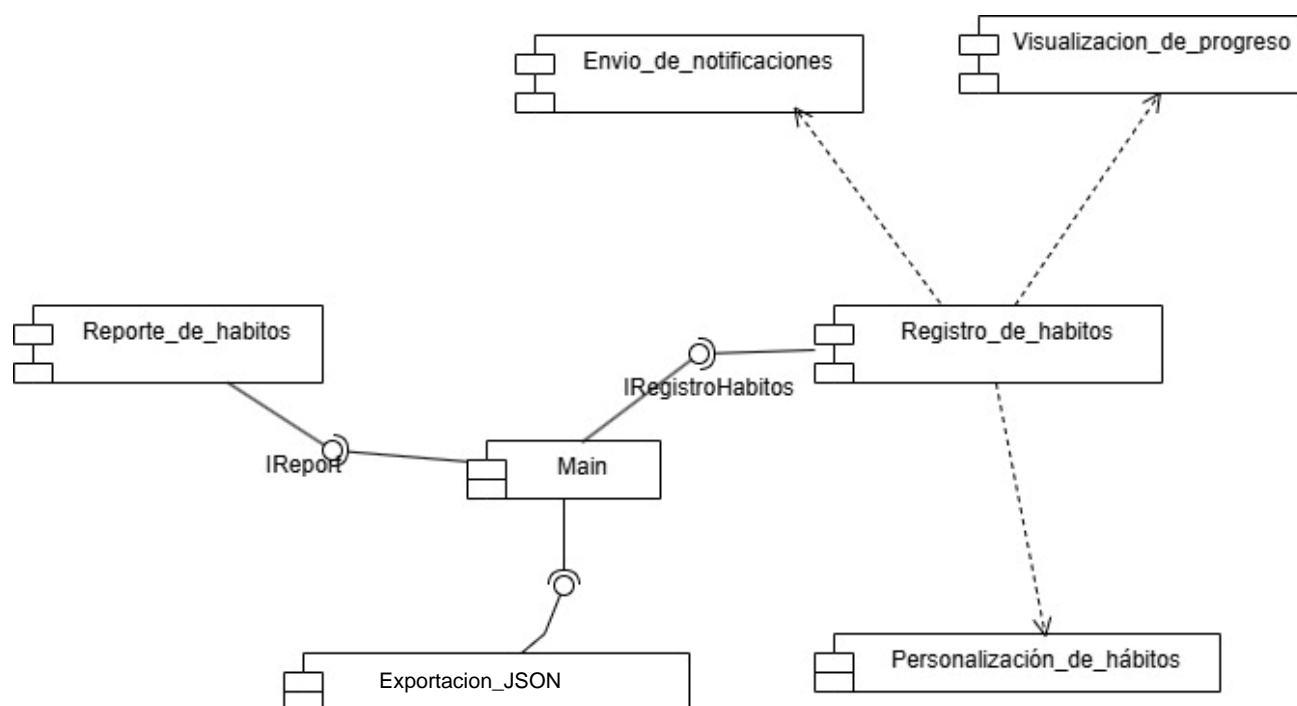


Figura 10.1 Diagrama de componentes del Sistema de Seguimiento de Hábitos Saludables

El diagrama de componentes presentado describe la arquitectura modular del Sistema de Seguimiento de Hábitos Saludables, desglosando las partes funcionales que lo integran y las interfaces que facilitan su interacción. El sistema consta de varios módulos que interactúan entre ellos a través de interfaces establecidas. El componente principal, conocido como **Main**, desempeña el papel de coordinador del sistema y punto de acceso central, conectándose con las interfaces **IRegistroHabitos** para la anotación de hábitos y **IReport** para la creación de informes.

El módulo **Registro_de_habitos** es responsable de almacenar los hábitos diarios del usuario, implementando la interfaz **IRegistroHabitos** y manteniendo relaciones de uso con los componentes **Envio_de_notificaciones**, **Visualización_de_progreso** y **Personalización_de_hábitos**. Por su parte, **Reporte_de_habitos** genera informes a partir de los datos almacenados y se comunica con **Main** a

través de la interfaz IReport.

Los componentes Envío_de_notificaciones, Visualización_de_progreso y Personalización_de_hábitos ofrecen funcionalidades complementarias como alertas, gráficos de avance y configuración personalizada de hábitos, respectivamente. Las interfaces IRegistroHabitos e IReport permiten que el componente Main interactúe con los demás módulos sin acoplarse directamente a sus implementaciones, lo cual promueve un diseño modular, versátil y fácilmente extensible.

Las relaciones entre los componentes se representan mediante líneas punteadas para indicar dependencias o uso de servicios, y líneas con círculos para la provisión o implementación de interfaces. Esta estructura modular mejora la escalabilidad, la mantenibilidad y la reutilización del sistema, permitiendo además una evolución progresiva sin comprometer su integridad funcional.

11. Diagramas de Clases, Secuencia y/o Estados

En esta sección se describen los modelos gráficos que representan diferentes aspectos del sistema de registro de hábitos saludables. Los diagramas permiten visualizar y comprender la estructura estática, el comportamiento dinámico y la interacción entre los componentes del software. Se incluyen diagramas de clases para definir la organización y relaciones entre objetos, diagramas de secuencia para ilustrar el flujo de eventos en los procesos clave, y diagramas de estados para describir los posibles estados y transiciones de los objetos relevantes.

Diagrama de Clases:

El Diagrama de Clases describe la estructura estática del sistema, evidenciando las clases fundamentales que participan en la implementación de la documentación de hábitos saludables, sus

características, procedimientos y las conexiones entre ellas. Este diagrama permite visualizar cómo se organizan y se comunican los componentes clave para cumplir con los requerimientos funcionales.

El sistema está compuesto por clases que representan a los usuarios, sus hábitos, los registros de cumplimiento diario, así como componentes encargados de la interfaz gráfica y la persistencia de datos. La clase Usuario contiene una lista de hábitos (Habito), cada uno de los cuales mantiene sus respectivos registros (Registro). Además, se incluye una clase GestorDeDatos que se encarga de guardar y recuperar la información, y una clase InterfazGrafica que representa la interacción con el usuario mediante la interfaz desarrollada con Tkinter. Por último, la clase HistorialView se encarga de mostrar visualizaciones gráficas del progreso.

Las relaciones reflejan asociaciones de composición y dependencia entre las clases, tales como la relación de uno a muchos entre Usuario y Habito, y entre Habito y Registro, así como las dependencias entre la interfaz gráfica y las clases encargadas de la lógica y persistencia del sistema.

Este modelo facilita la extensibilidad del sistema, permitiendo incorporar nuevas funcionalidades o mejorar la interfaz sin afectar el núcleo de la lógica de negocio.

A continuación, se presenta el diagrama de clases del sistema. Este diagrama permite modelar la estructura estática de la aplicación, mostrando las clases que intervienen, sus atributos, métodos principales y las relaciones entre ellas.

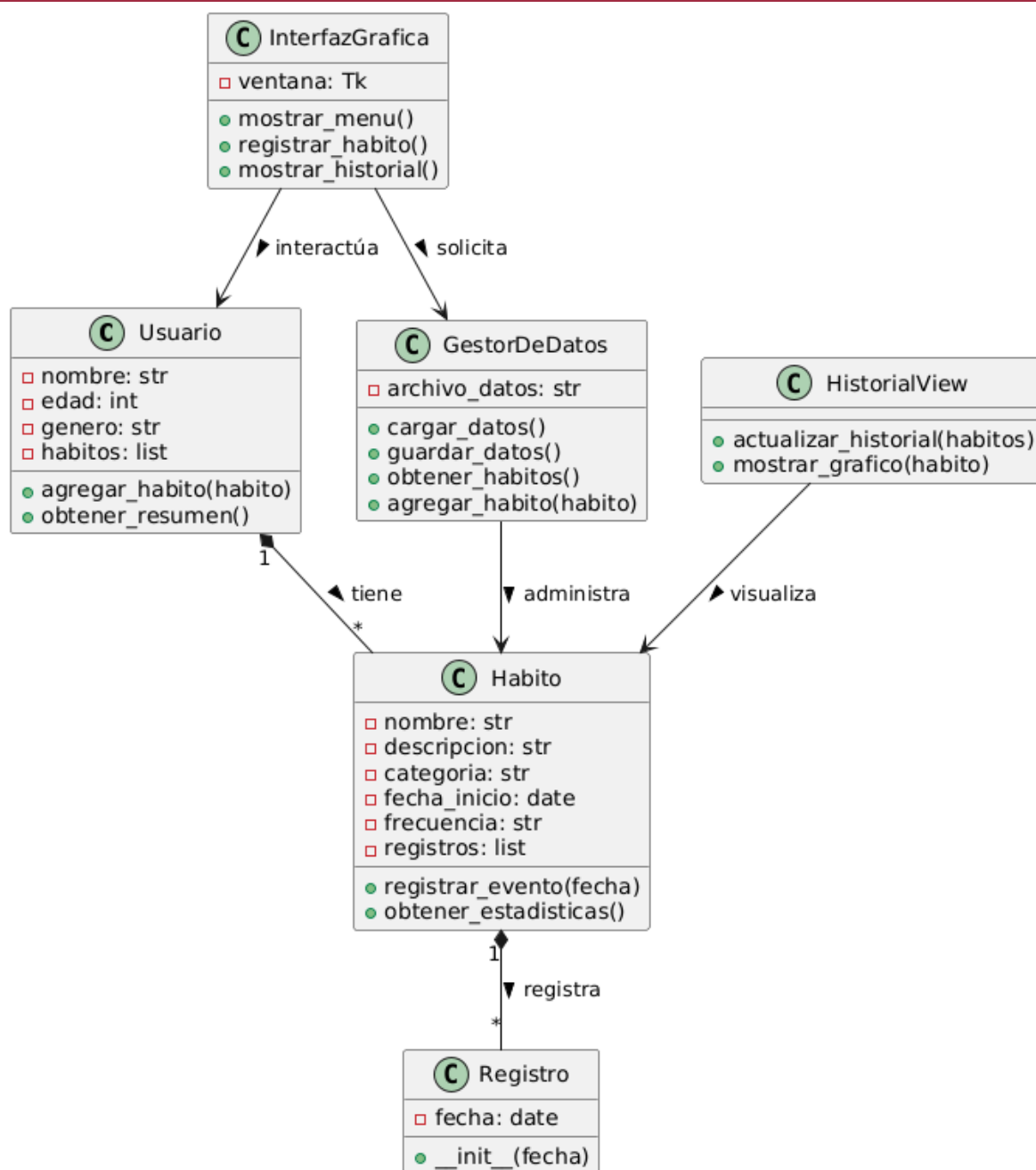


Figura 12.1 Diagrama de Clases

Diagrama de Secuencia:

A continuación, se presenta el diagrama de secuencia correspondiente al caso de uso "Registrar un nuevo hábito", el cual describe la interacción dinámica entre los diferentes componentes del sistema a lo largo del tiempo.

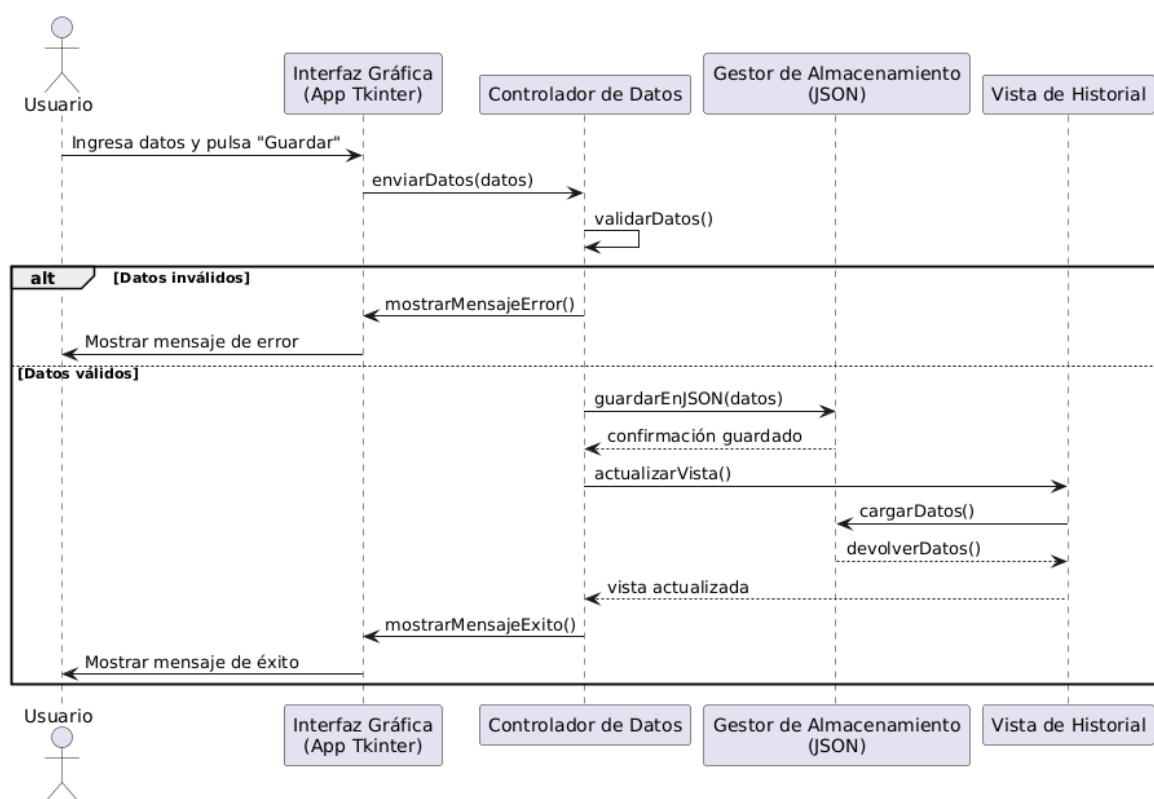


Figura 12.2. Diagrama de Secuencia

Este diagrama permite visualizar el flujo de mensajes entre el usuario, la interfaz gráfica (construida en Tkinter), el controlador de datos, el gestor de almacenamiento (basado en archivos JSON) y la vista del historial. El proceso inicia cuando el usuario ingresa los datos de un nuevo hábito y pulsa el botón de guardar. La interfaz envía esta información al controlador, el cual se encarga de validar los datos.

En caso de que la validación falle, se muestra un mensaje de error al usuario. Si los datos son correctos, el controlador procede a almacenarlos en un archivo JSON. Posteriormente, se actualiza la vista del historial para reflejar los nuevos datos registrados. Finalmente, se notifica al usuario con un mensaje de éxito.

Este diagrama resulta fundamental para entender el comportamiento del sistema frente a una de sus funcionalidades principales, así como para asegurar una correcta coordinación entre las capas

de presentación, lógica y persistencia de datos.

Diagrama de Estados:

A continuación, se presenta el diagrama de estados correspondiente a la entidad Hábito, el cual modela el comportamiento dinámico asociado al ciclo de vida de un hábito dentro del sistema. Este diagrama permite visualizar cómo cambia el estado de un hábito en función de las acciones del usuario.

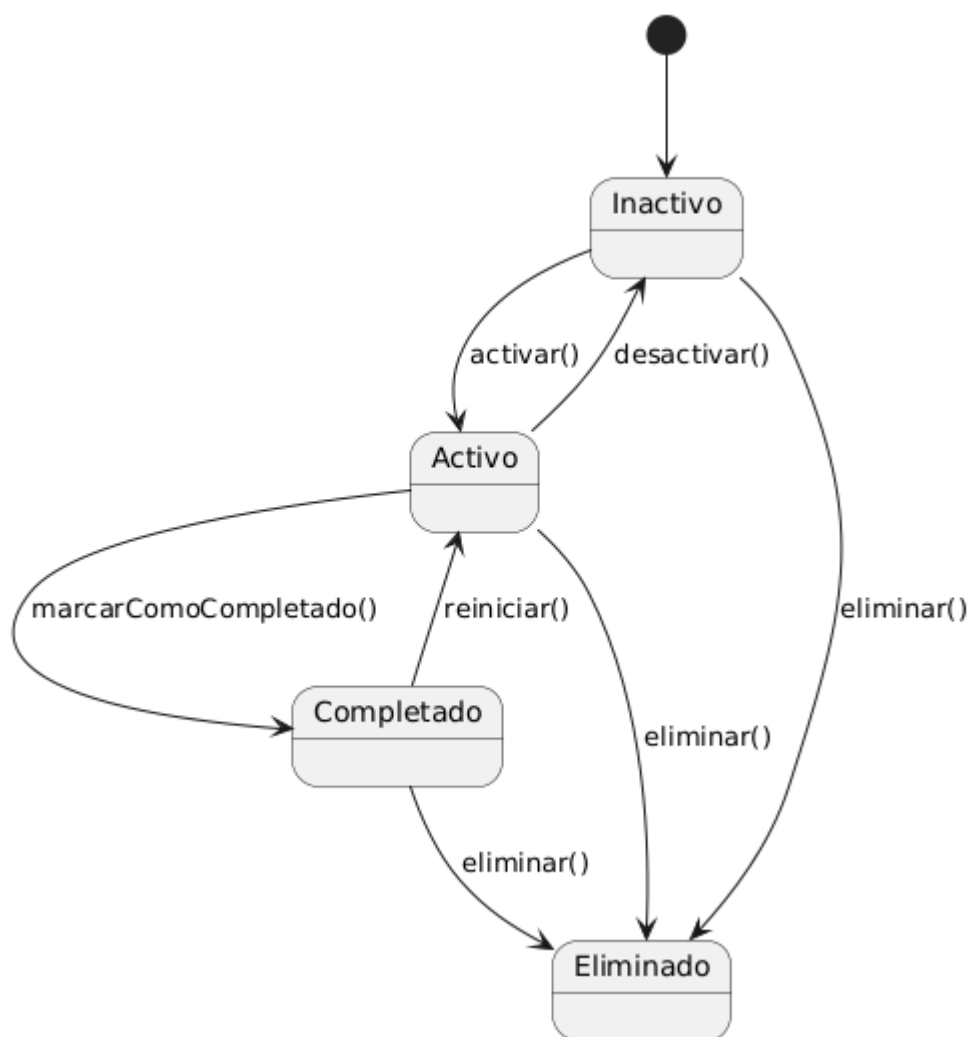


Figura 12.3. Diagrama de Estados

Inicialmente, un hábito se encuentra en estado Inactivo, lo que indica que ha sido creado, pero aún no se ha comenzado a seguir. Al ser activado por el usuario, transita al estado Activo, permitiendo

su seguimiento diario. Una vez cumplida la meta establecida, el hábito pasa al estado Completado. Si el usuario decide detener el seguimiento temporalmente, puede volver a dejar el hábito en estado Inactivo. Asimismo, es posible reiniciar un hábito desde el estado Completado, lo que representa la intención de volver a trabajar sobre él. Finalmente, el hábito puede ser eliminado desde cualquiera de sus estados, lo que lo lleva al estado Eliminado, considerado como un estado final en este ciclo.

Este diagrama de estados complementa la lógica de negocio del sistema, al permitir representar de forma explícita los posibles caminos que puede seguir un hábito, y refuerza la implementación orientada a objetos que considera la transición de estados como parte esencial del modelo de dominio.

12. Modelo de Datos

El modelo de datos del sistema se compone de tres entidades fundamentales: Usuario, Hábito y Reporte, las cuales representan los elementos esenciales para el registro y seguimiento de hábitos saludables.

Entidades

Usuario: Esta entidad representa al individuo que interactúa con el sistema para registrar y monitorear sus hábitos de salud. Contiene los atributos edad y género, que permiten identificar características demográficas relevantes para el análisis y personalización de la información almacenada.

Hábito: Define un conjunto de registros diarios relacionados con los hábitos saludables del usuario. Incluye los atributos cuantitativos agua (cantidad de vasos de agua consumidos), sueño (horas de sueño) y actividad (minutos de actividad física), además del atributo temporal fecha, que indica el día específico del registro. Para mantener la vinculación con el usuario correspondiente, la entidad incorpora los atributos edad y género replicados desde Usuario, configurando así una relación

implícita de pertenencia.

Reporte: Representa los documentos generados que contienen resúmenes o análisis de los hábitos registrados, almacenados en formato PDF. Contiene atributos descriptivos como `fechaGeneracion` (fecha en que se creó el reporte) y `nombreArchivo` (nombre del archivo generado). De manera similar, incorpora las características de edad y género para vincular cada informe con el usuario que lo produjo.

Relaciones

El modelo establece que un usuario puede registrar múltiples hábitos a lo largo del tiempo, estableciendo una relación uno a muchos entre Usuario y Hábito.

De igual forma, un usuario puede generar múltiples reportes, estableciendo una relación uno a muchos entre Usuario y Reporte. La relación entre Hábito y Usuario, y entre Reporte y Usuario, se implementa mediante la inclusión de atributos comunes (edad y género), sin utilizar una clave primaria o clave foránea explícita, garantizando así la asociación directa sin dependencia de identificadores únicos. A continuación, se presenta un diagrama visual del modelo de datos.

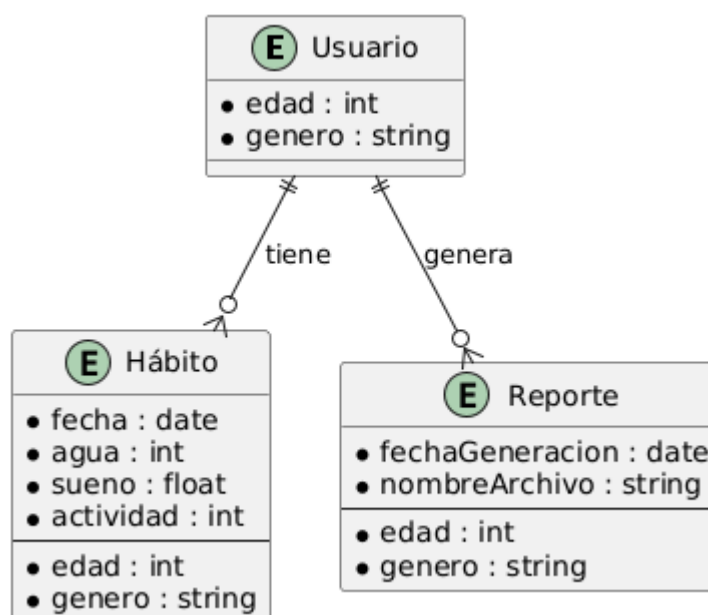


Figura 13.1. Modelo de Datos

13. Prototipo Funcional en Alta Fidelidad

El prototipo para utilizar comprende la aplicación de las características fundamentales del sistema, en particular, el registro de hábitos diarios.

Para esta fase, se ha implementado en Python lo siguiente:

Registro de hábitos: Los usuarios pueden ingresar información sobre su consumo de agua, actividad física, descanso, y estado de ánimo. Cada hábito se almacena en un objeto de la clase Hábito y se guarda en un archivo JSON para su persistencia.

Visualización de hábitos: Mediante el uso de la biblioteca Matplotlib, se producen diagramas de barras que ilustran el avance de los hábitos con el transcurso del tiempo.

Interfaz Gráfica: La interfaz de usuario está desarrollada con Tkinter, permitiendo al usuario interactuar con el sistema de manera intuitiva, ingresando datos y visualizando resultados. La interfaz incluye formularios para ingresar hábitos y botones para guardar y visualizar los datos.

14. Control de Versiones en Github con Ramas por Integrante

El desarrollo del prototipo se gestionó utilizando GitHub como plataforma de control de versiones. Cada miembro del equipo trabajó en su propia rama, y el flujo de trabajo siguió el modelo Git Flow, donde las ramas principales son main y develop. Las ramas individuales fueron integradas a la rama develop, y las funcionalidades completadas se fusionaron a la rama main después de ser revisadas y aprobadas.

Este enfoque permitió mantener un historial claro de cambios, facilitar la revisión de código y asegurar la trazabilidad de las contribuciones. Todo el trabajo de desarrollo y colaboración está disponible en el repositorio de GitHub:

<https://github.com/Arsiel15/seguimiento-habitos-equipo4>

15. Documentación del Código y README del Proyecto

La documentación del código se realizó utilizando comentarios detallados en cada módulo y función, explicando la lógica implementada y su propósito. Además, se creó un archivo README.md que proporciona una descripción general del proyecto, las instrucciones para ejecutar el prototipo, y los pasos necesarios para contribuir al código. El archivo README incluye detalles sobre las dependencias del sistema, como las bibliotecas Tkinter, Matplotlib, y ReportLab, así como instrucciones para instalar y ejecutar el proyecto en diferentes entornos.

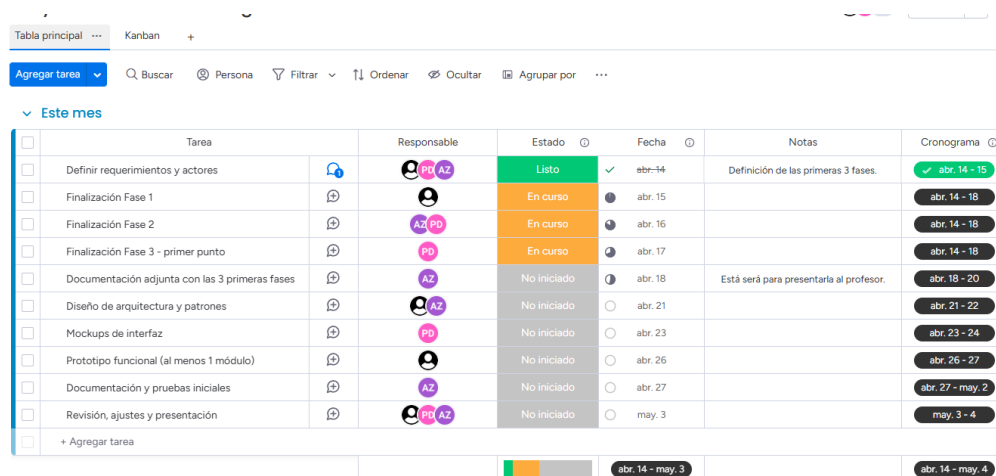
16. Bibliografía

- Necula, S. (29 de Abril de 2024). *Exploring The Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications*. Obtenido de https://www.researchgate.net/publication/380197155_Exploring_The_Model-View-Controller_MVC_Architecture_A_Broad_Analysis_of_Market_and_Technological_Applications
- 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/>
-

17. Anexos

17.1 Cronograma de Trabajo

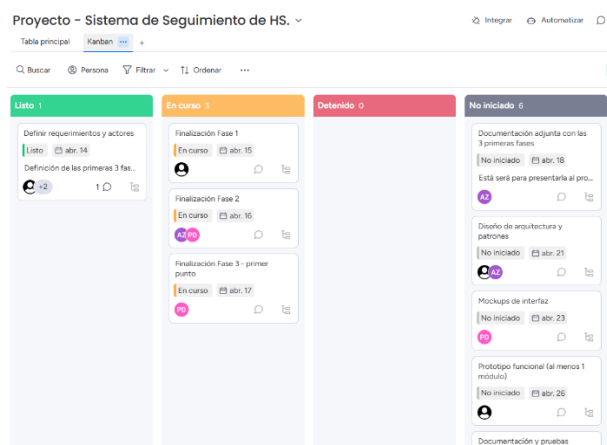
Imagen que muestra la organización de tareas por columnas, categorizadas por fase o estado (Pendiente, En curso, Completado, etc.).



Tarea	Responsable	Estado	Fecha	Notas	Cronograma
Definir requerimientos y actores	FR, AZ	Listo	abr-14	Definición de las primeras 3 fases.	✓ abr. 14 - 15
Finalización Fase 1	FR	En curso	abr. 15		abr. 14 - 18
Finalización Fase 2	AZ, PO	En curso	abr. 16		abr. 14 - 18
Finalización Fase 3 - primer punto	PO	En curso	abr. 17		abr. 14 - 18
Documentación adjunta con las 3 primeras fases	AZ	No iniciado	abr. 18	Está será para presentarla al profesor.	abr. 18 - 20
Diseño de arquitectura y patrones	FR, AZ	No iniciado	abr. 21		abr. 21 - 22
Mockups de interfaz	PO	No iniciado	abr. 23		abr. 23 - 24
Prototipo funcional (al menos 1 módulo)	FR	No iniciado	abr. 26		abr. 26 - 27
Documentación y pruebas iniciales	AZ	No iniciado	abr. 27		abr. 27 - may. 2
Revisión, ajustes y presentación	FR, AZ	No iniciado	may. 3		may. 3 - 4

Figura 19.1.1 Vista de Tabla Principal del Cronograma en Monday.com

Captura que evidencia el seguimiento individual y grupal de las tareas, incluyendo responsables y progreso.



Lista 1	En curso 3	Detenido 0	No iniciado 6
Definir requerimientos y actores Listo Definición de las primeras 3 fas...	Finalización Fase 1 En curso Finalización Fase 2 En curso Finalización Fase 3 - primer punto En curso		Documentación adjunta con las 3 primeras fases No iniciado Está será para presentarla al pro... Diseño de arquitectura y patrones No iniciado Mockups de interfaz No iniciado Prototipo funcional (al menos 1 módulo) No iniciado Documentación y pruebas iniciales

Figura 19.1.2 Seguimiento de tareas y responsables por fase

17.2 Evidencia de Reuniones

Captura de pantalla de la Primera Reunión (14 de febrero)

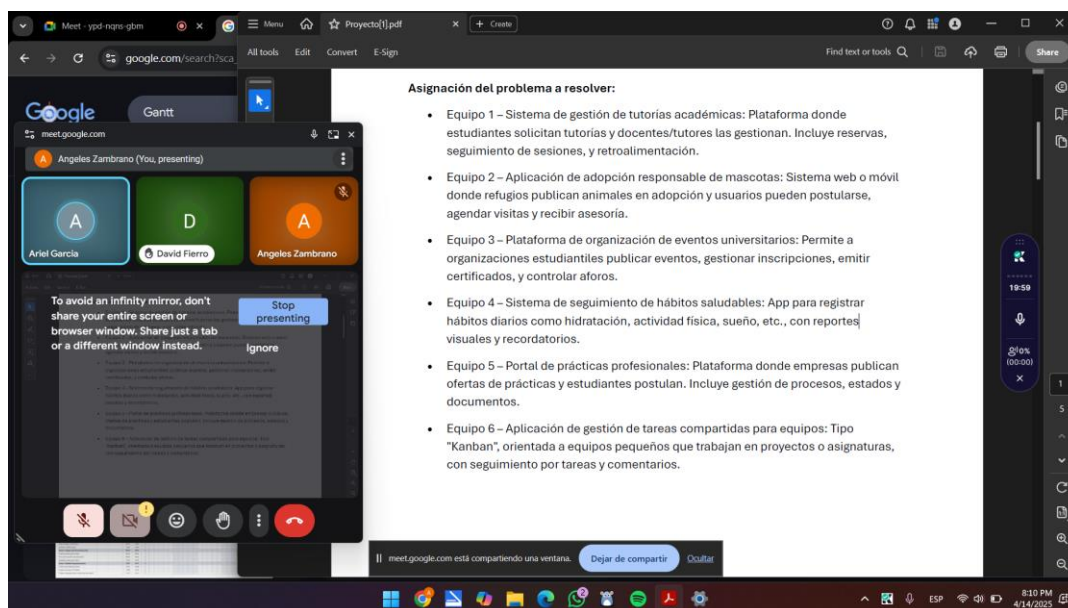


Figura 19.2.1: Reunión inicial vía Google Meet donde se discutieron roles y planificación del proyecto.