KAIROS

Documentación

PROYECTO DE INGENIERÍA WEB

NOMBRE DEL EQUIPO, MIEMBROS Y PERFILES

Inicialmente, el equipo optó por no adoptar un nombre específico, priorizando la flexibilidad y adaptación ante la indefinición del problema y las áreas involucradas. A medida que el proyecto avanzó, se eligió el nombre "KAIROS", el mismo que se asignó a nuestro asistente deportivo. Este nombre fue seleccionado por su significado: el momento oportuno para realizar algo, lo cual representó la oportunidad y visión de este proyecto.

Previo al desarrollo, se establecieron bases teóricas y se definieron roles y perfiles basados en los intereses y habilidades de cada miembro. Los perfiles presentaban características diversas, buscando un equipo equilibrado y complementario. Esta diversidad se detalla en la tabla adjunta, donde se asocia al miembro su perfil, personalidad y posibles actividades. Pese a la asignación de roles inicial, se decidió que todos los miembros contribuirían al proyecto de forma equitativa y se involucrarían en todas las etapas clave para fomentar el aprendizaje mutuo y transversal.

TEMA Y OBJETIVOS DEL PROYECTO

Para el desarrollo del proyecto, se consideraron diversas ideas, optando finalmente por la implementación de una solución integral que combina Inteligencia Artificial (IA), desarrollo web y gestión de bases de datos.

La funcionalidad central se logró mediante la implementación de una API que conecta el modelo de IA de Gemini con una interfaz de usuario personalizada. Asimismo y como parte secundaria, utilizó una base de

datos para almacenar el historial e información de los usuarios, permitiendo que el asistente trabaje con contexto.

Se planteó un asistente, denominado KAIROS, que se enfocó en el ámbito deportivo. A través de un prompt maestro y cargas de texto específicas, se le otorgaron funcionalidades diversas, tales como la generación de rutinas personalizadas, apoyo motivacional y emocional, y recomendaciones de ejercicios y dietas. Su uso está estrictamente restringido al ámbito deportivo y no sustituye la consulta médica. El usuario tiene libertad de interacción, incluyendo modos especializados como el psicológico o el competencia.

A continuación, se presenta la información detallada, abordando desde la definición de objetivos hasta el desarrollo paso a paso del proyecto.

Objetivo del Proyecto

El objetivo principal de este proyecto es la creación de una aplicación web funcional como entrenador personal inteligente, enfocada en:

- Sugerencia y personalización de rutinas de entrenamiento.
- Apoyo emocional y motivacional.
- Activación de un modo competencia para maximizar el rendimiento.

Todo esto se logra gracias a la Inteligencia Artificial, la cual constituye la base tecnológica central del proyecto.

Bloques de Desarrollo

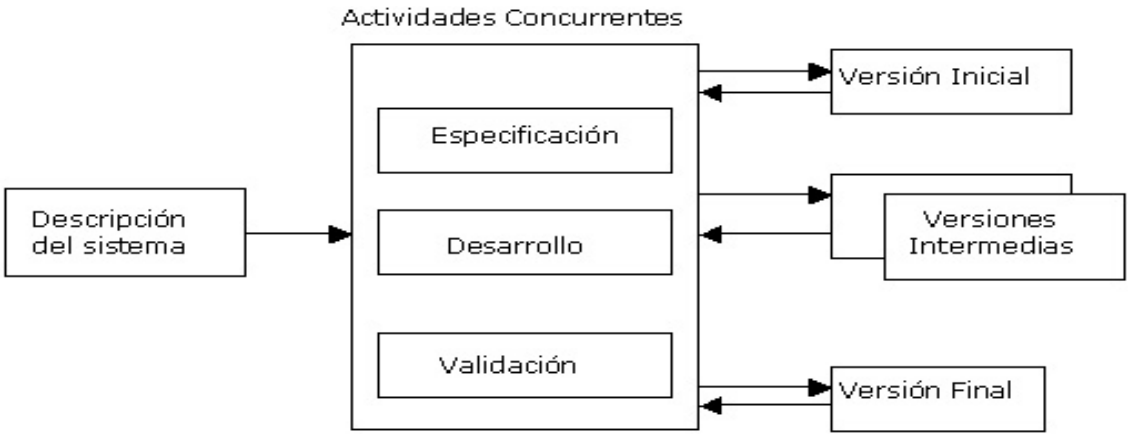
El proyecto se llevó a cabo durante el trimestre de estudio y se desarrolló mediante los siguientes bloques de trabajo:

- modelos de desarrollo de software
- preguntas sobre el tema del proyecto
- actividades/responsabilidades
- Levantamiento de requerimientos
- Daily Scrum
- Sprint 1
- Diagramas lógicos
- Diagramas de secuencia
- Diagrama de caso de uso

MÉTODOS DE DESARROLLO DE SOFTWARE

El método seleccionado para este proyecto fue el **método de desarrollo evolutivo**. Elegimos este modelo debido a su gran adaptabilidad, ya que al tratarse de un proyecto basado en inteligencia artificial era probable que surgieran desafíos que dificultan el cumplimiento de los objetivos planteados inicialmente.

Otro motivo para optar por este enfoque es que permite incorporar mejoras sin afectar significativamente el tiempo total del proyecto. Gracias a este método, podemos añadir nuevas funciones y utilizar herramientas que no habíamos considerado al inicio, ampliando así el alcance y la calidad del desarrollo.



Este método también nos presentó algunas dificultades, ya que al inicio del proyecto contábamos con una gran cantidad de ideas. Conforme avanzamos, nos dimos cuenta de que no disponíamos del tiempo

necesario para implementarlas todas o que algunas representaban un nivel de complejidad mayor, especialmente porque requerían más recursos de hardware.

Por esta razón, decidimos limitar el alcance y enfocarnos únicamente en desarrollar una página web, con el fin de optimizar el tiempo de trabajo. Asimismo, optamos por utilizar herramientas como **Django** para agilizar la gestión de la base de datos, la cual inicialmente tampoco estaba contemplada en el proyecto.Gracias al **método evolutivo**, pudimos avanzar mientras realizamos ajustes, reducimos ideas, fortalecemos las más importantes y añadimos nuevas o mejores herramientas. En definitiva, este método fue fundamental, ya que no nos frenó y nos brindó la flexibilidad necesaria para llevar a cabo el proyecto con éxito.

PREGUNTAS SOBRE EL TEMA DEL PROYECTO

Al inicio de este proyecto se nos hicieron las siguientes preguntas

¿Qué es la Inteligencia Artificial?

La inteligencia artificial (IA) es una tecnología que permite a las computadoras y máquinas simular el aprendizaje humano, la comprensión, la resolución de problemas, la toma de decisiones, la creatividad y la autonomía.

¿Para qué se ocupa la IA hoy en día?

La inteligencia artificial se ha utilizado en distintos campos como la robótica, las ciencias de la computación, las finanzas, la salud, los sistemas de transporte autónomos, el mundo de los videojuegos y las comunicaciones. En estos entornos, las máquinas son capaces de manejar grandes cantidades de datos que les permiten desde identificar y comprender comandos verbales e imágenes, hasta realizar cálculos y acciones complejas con una gran rapidez.

Estos sistemas, en consecuencia, sirven para percibir su entorno y relacionarse con él, así como también para que actúen con un objetivo específico, después de una recopilación y procesamiento de datos muy exhaustiva. Es decir, se trata de tecnología aplicada para la solución de tareas en el mercado.

Menciona los principales temas de estudio de la Inteligencia Artificial.

Los principales temas de estudio de la Inteligencia Artificial (IA) abarcan diversas disciplinas que buscan crear sistemas capaces de realizar tareas que normalmente requerirían inteligencia humana, como el aprendizaje y el razonamiento.estos son algunos ejemplos de algunas de sus aplicaciones: Machine learning o aprendizaje automático,Fuzzy logic o lógica difusa, Vida artificial, Sistemas expertos, Data Mining o minería de datos, Redes Bayesianas, Ingeniería del conocimiento, Redes neuronales artificiales, Sistemas reactivos, Sistemas basados en reglas

¿Todas las IA son comerciales o cualquier persona puede desarrollar

una?

No, no todas las IA son comerciales; cualquier persona con los conocimientos y herramientas adecuados puede desarrollar una. El mundo de la inteligencia artificial incluye modelos tanto propietarios y comerciales (como los desarrollados por grandes empresas) como de código abierto (open source), que fomentan la colaboración y la accesibilidad.

¿Te interesa algún tema en particular relacionado a la IA? ¿Cuál?

redes neuronales, minería de datos, vida artificial y sistemas relativos

Menciona los lenguajes de programación más asociados con la IA.

Los lenguajes más asociados con la IA son Python (el más dominante por su ecosistema de librerías como TensorFlow, PyTorch), R (para análisis estadístico), Java, C++ (para rendimiento), y lenguajes más específicos como Lisp y Prolog (históricos y lógicos), además de Julia (alto rendimiento) y JavaScript (para IA en navegadores)

Estas preguntas fueron las primeras en aparecer a lo largo de nuestro proyecto aunque estas preguntas solo eran respecto a que es una IA y para verificar nuestro conocimiento solo las mismas serían el inicio para saber que conocimientos nos hacen falta a partir de este punto.

ACTIVIDADES/RESPONSABILIDADES

Durante este proyecto realizamos actividades grupales y actividades individuales las cuales fueron registradas y repartidas de forma en la que todos participamos.

Para mejorar el entendimiento de las actividades se realizó una tabla y marcamos quien serían los responsables de la misma forma marcamos las actividades que realizamos todos.

LEVANTAMIENTO DE REQUERIMIENTOS

Durante el primer SPRINT de este proyecto realizamos el levantamiento de requerimientos donde pusimos que es lo que llevaría nuestra interfaz y cuales serían nuestros elementos más importantes.

En nuestro primer levantamiento de requerimientos fue la primera idea que teníamos del proyecto de los elementos queremos o que pensábamos que eran los necesarios en nuestra primera versión.

En esta primera versión no teníamos un buen comprendimiento de este formato por lo cual tuvimos una mala primera versión la cual se corrige en la segunda versión gracias a que la profesora nos hizo notar nuestros errores y cómo arreglar los mismos.

En la segunda versión agregamos los errores de formato cometidos de la misma forma podemos añadir algunos cambios y desarrollamos algunos puntos, añadimos nuevos elementos y a su vez podemos ver cuales serían nuestros primeros elementos del interfaz y con esto pudimos desarrollar lo primeros pasos en este proyecto.

En nuestra tercera versión añadimos y agregamos elementos recomendados durante las asesorías con la profesora ya que gracias a estas asesorías pudimos notas algunas deficiencias y errores en los elementos de las versiones anteriores , también pudimos hacer el formato de mejor forma y con organización

enlace al documento

DAILY SCRUM

Durante este proyecto realizamos los daily scrum para llevar a cabo un seguimiento del proyecto, notar si el equipo realizaba actividades o investigación fuera del tiempo de la clase o si seguía realizando algún avance del proyecto o si en sus tiempos libre veía o hacía algo para el proyecto.

nota hay algunos lagunas de días en los daily scrum debido a que no siempre se realizaban o por olvido, también no en todos los daily scrum participan todos sea por falta los días de que realizó o por llegar tarde ya que estos mismos sólo se realizaban en los primeros 10 minutos de las clases y se entregaban al terminar el tiempo

SPRINT 1

Durante este sprint fue la presentación de nuestro proyecto en el cual pudimos presentar por primera a Kairos y que es lo que lo compone.

1. el primer elemento que pudimos explicar fue la presentación fue cuál es la función de kairos y de que sería capaz al terminar el proyecto.

2. el segundo elemento fue la personalidad que deseamos que tuviere la ia y cómo podría interactuar con el usuario.

3. la interfaz y que es lo que contiene, en estas diapositivas mostramos cómo sería la interacción con el usuario y todas las funciones que tienen estas interfaz.

4. En esta parte explicamos cuáles serían las herramientas con los que trabajaríamos. El backend que usaremos sería DJANGO , la base de datos la trabajamos en MYSQL, la IA con la que trabajamos serie GEMINI.

5. En esta parte expresamos los siguientes pasos para el siguiente sprint y cuáles serán los principales puntos en nuestro siguiente avance


Home
About
Contact
Próximo sprint...
• Configuración del ambiente de Django.
• Diseño y construcción de las tablas de la BD.
• Implementación de las rutas de autenticación.
• Integración de Gemini API y prueba de conexión.

DIAGRAMA LÓGICO

Utilizamos este diagrama lógico para mostrar cómo se conectarán las herramientas que emplearemos en nuestro proyecto. En este caso, nuestra página web incluye una interfaz, Django, una base de datos y un motor de IA. Gracias a este diagrama podemos explicar de manera más clara cómo interactúan estos elementos entre sí.

Además, nos sirve como guía para definir y mantener la estructura de conexiones entre nuestras herramientas, y así asegurar que seguimos fielmente la idea y el diseño planteado para la página.

Esta es la descripción del paso a paso del diagrama lógico:

1. El usuario decide utilizar KAIROS e interactúa con la interfaz.

2. La interfaz envía la solicitud del usuario hacia Django para realizar una tarea. Por ejemplo: ¿Genera una rutina aumentar músculo?.

3. La consulta ORM del servidor de Django obtiene los datos relevantes del usuario.Esto de forma que lo codificado en Python permite la interacción con la BD.

4. La base de datos devuelve los datos al servidor (Django) para poder utilizar correctamente Kairos.

5. El prompt maestro (api files) junto con la base de datos, generan un archivo de tipo json.

6. El archivo JSON se envía como solicitud al motor de IA (Gemini) para ser procesado.

7. El motor de IA(Gemini) procesa la solicitud enviada en JSON y devuelve una respuesta en JSON a Django.

8. La respuesta de Gemini se envía a Django que a su vez la envía a la base de datos para así almacenar el historial de conversación.

9. La respuesta que recibió Django una vez almacenada en la bd, y procesada se envía a la interfaz.

10. La interfaz muestra la respuesta. Por ejemplo: ¿Rutina generada? o mensaje motivacional al usuario.

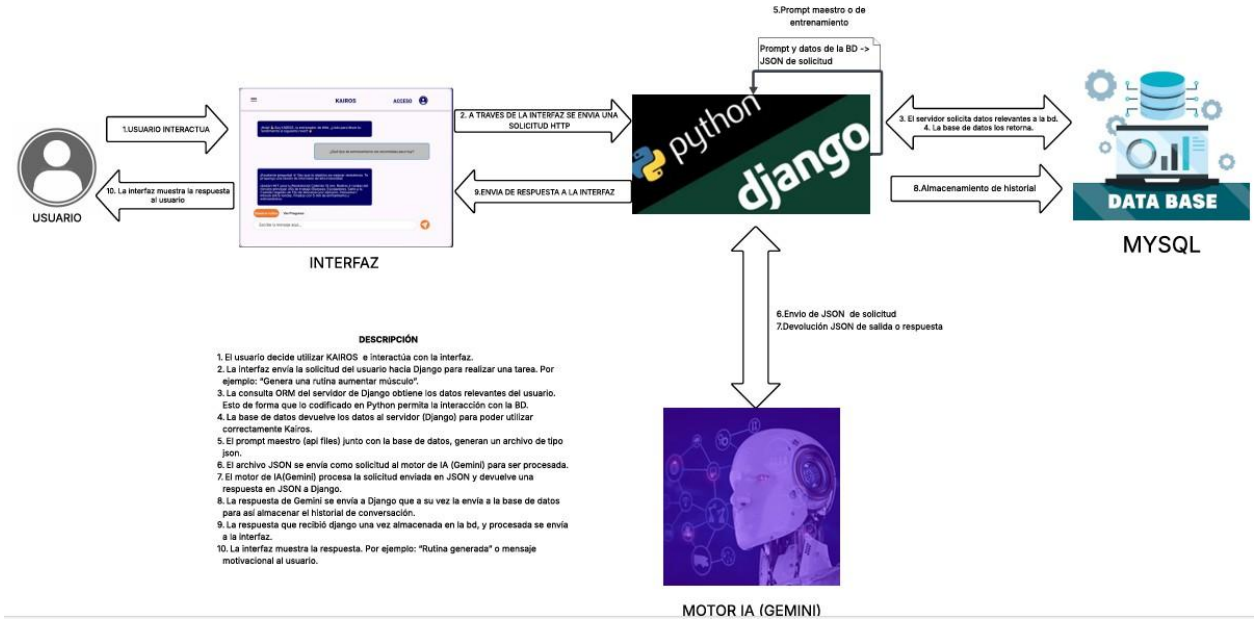


DIAGRAMA DE SECUENCIA

Con este diagrama de secuencia buscamos mostrar cómo funciona la interacción entre nuestras herramientas y qué ocurre detrás de cada proceso. Al igual que con el diagrama lógico, el objetivo es que este recurso nos sirva como guía para definir y mantener la estructura de conexiones entre nuestras herramientas, asegurando así que seguimos fielmente la idea y el diseño planteados para la página.

Paso a paso del diagrama de secuencia:

Inicio de sesión / autenticación de cuenta

1. El usuario entra a la página.
   El navegador solicita la página al servidor (Django).
2. El servidor devuelve la interfaz (HTML).
   Django retorna el HTML que muestra el formulario/interfaz para interactuar con el chat.
3. Mostrar interfaz al usuario.
   Ya en su navegador, el usuario ve el chat, botones y opciones de configuración
4. El usuario accede o se registra en su cuenta.
   Cuando hace login o registro, el sitio envía la solicitud a la BD para validar credenciales.

5. Validación en la base de datos.
   La BD comprueba los datos y devuelve si son correctos; el sistema confirma la sesión y responde al usuario.

Interacción con el chat

1. Usuario escribe en el chat y envía una pregunta.
   El mensaje viaja del navegador al servidor vía una petición HTTP.
2. El servidor procesa la pregunta.
   Puede preparar datos, combinar configuración del usuario y construir un *prompt* para la IA.
3. Enviar la petición a la IA (Gemini) o servicio externo.
   El sistema realiza una solicitud HTTP a Gemini con el prompt y la configuración.
4. La IA procesa y devuelve una respuesta.
   Gemini responde con texto (o JSON) que el servidor recibe.
5. Mostrar la respuesta al usuario.
   El servidor envía la respuesta al navegador y ésta se muestra en la interfaz del chat.

Perfil y configuración del comportamiento del chat

1. Acceder al perfil / solicitar datos del usuario.
   Si el usuario entra a su perfil, el sitio pide a la BD sus datos y los muestra.
2. El usuario puede ajustar la configuración del chat.
   - Seleccionar motivación, psicología, modo competencia, etc.
   - Cada cambio actualiza cómo se construyen los prompts (cambia el "comportamiento del chat").
3. Enviar prompt con la configuración actualizada.

   Tras cambiar algo, el sistema incorpora esa configuración en las futuras solicitudes a la IA.

Validación, recopilación y procesamiento de datos

1. Verificar datos y recopilar información.
   El sistema puede verificar o enriquecer datos del usuario antes de enviarlos a la IA (por ejemplo, para personalizar respuestas).
2. Cargar y procesar datos estáticos o del proyecto.
   El servidor puede leer archivos del proyecto, procesarlos y modificar el comportamiento del chat según esos datos.
3. El backend responde con JSON o mensajes según la ruta.
   Algunas rutas devuelven JSON (APIs), otras retornan páginas HTML. Django organiza estas respuestas.

preguntas frecuentes

1. Preguntas frecuentes (FAQ).
   El sistema puede mostrar FAQs en la interfaz para ayudar al usuario sin necesidad de usar la IA.…
2. Loop — interacción continua.
   El diagrama indica que el intercambio chat → procesamiento → respuesta se repite cuantas veces quiera el usuario (bucle).

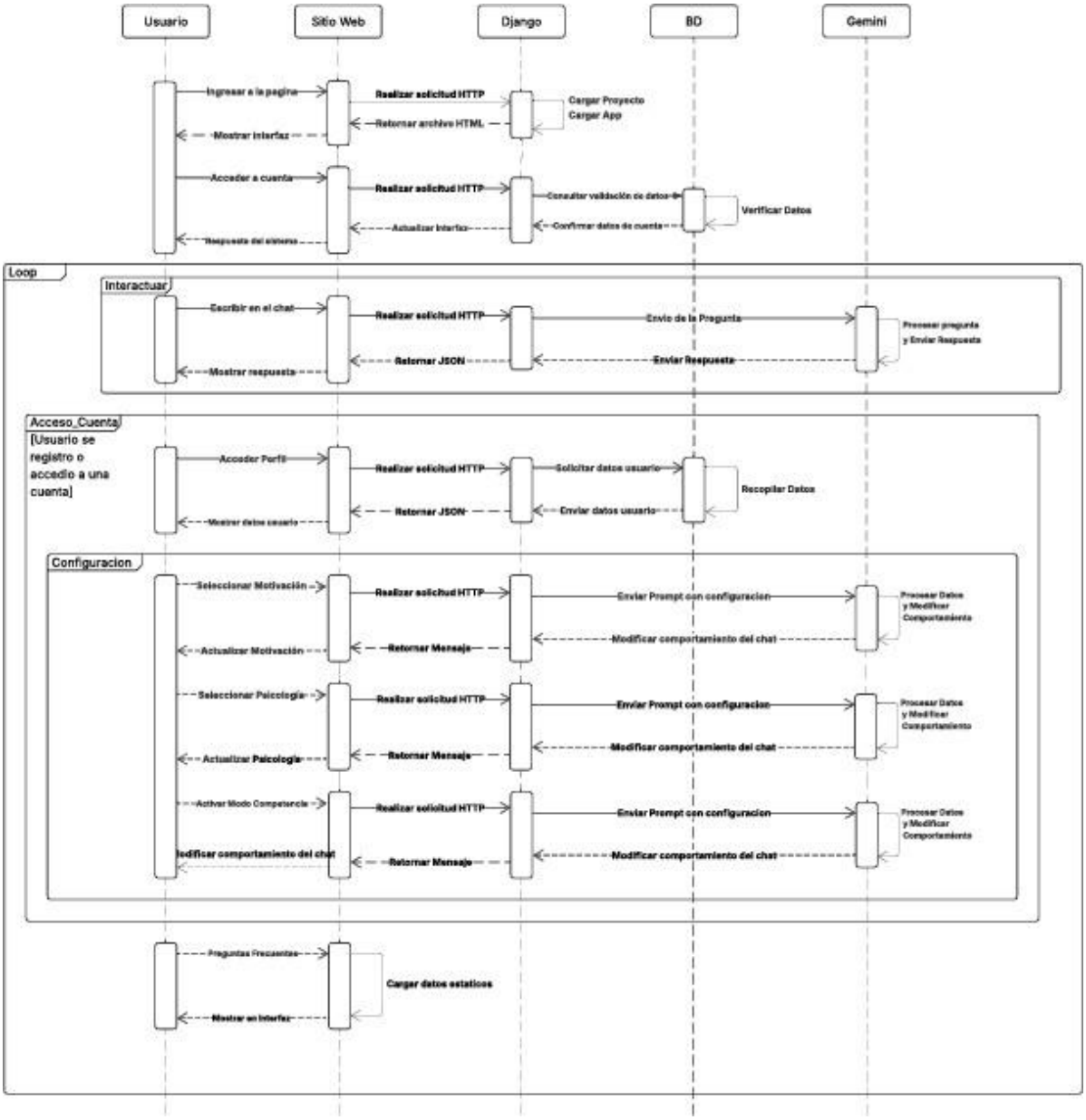


DIAGRAMA DE CASO DE USO

El diagrama de caso de uso se presenta a continuación para modelar y definir las interacciones clave que pueden realizar los dos actores principales del sistema: el usuario y el administrador.

Este modelo es fundamental para visualizar la relación entre la interfaz, el usuario y administrador del proyecto, KAIROS (la IA). Al identificar estas funciones, pudimos verificar que las interacciones y el alcance del sistema cumplieran efectivamente con los objetivos y requisitos funcionales establecidos para el proyecto.

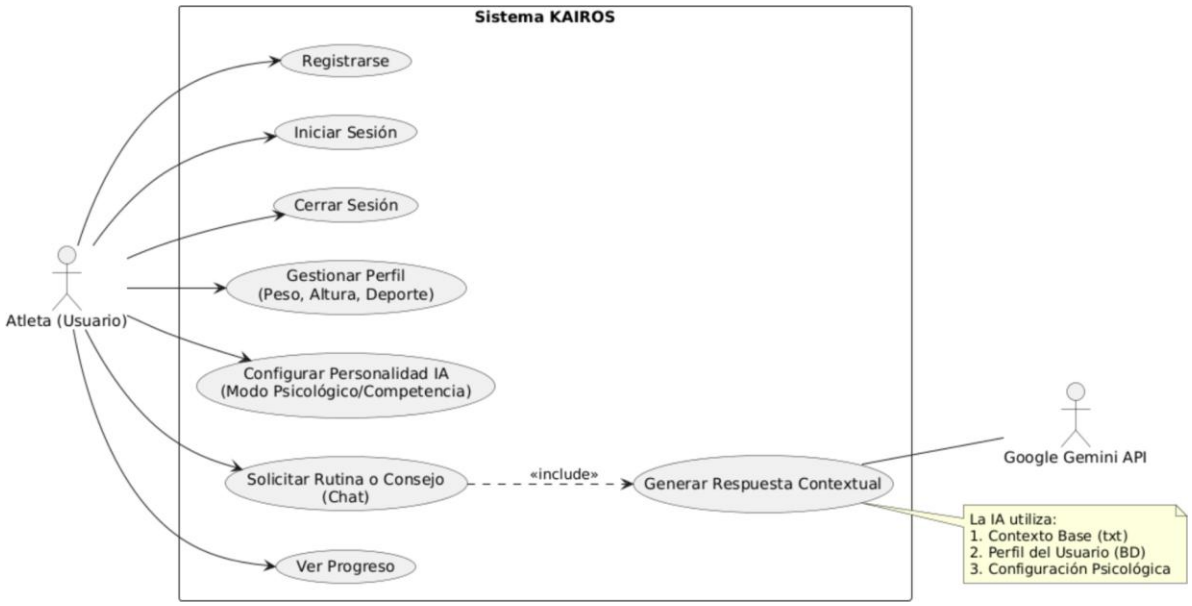
Elementos del diagrama de casos de uso 1

1. Los Actores

- **Atleta (Usuario):** Es el actor principal. Representa a la persona que usa la aplicación para entrenar. Sus acciones inician todos los procesos principales.
- **Google Gemini API:** Es un actor secundario (o externo). El usuario no habla "directamente" con él, pero KAIROS lo requiere para funcionar. KAIROS le envía los datos (en forma de prompt) y Gemini devuelve la respuesta.

2. Los Casos de Uso (Qué hace el sistema)

- **Acceso (Login/Registro):** Son requeridos para que el sistema pueda guardar sus objetivos y datos relevantes.
- **Gestión de Perfil (Métricas):** Aquí es donde el usuario define "quién es físicamente" (Peso, Altura, Deporte, Nivel). Esto es vital para que la IA no de consejos genéricos.
- **Configuración de Personalidad (Modo Psicológico):** Permite al usuario decirle al sistema "cómo quiere ser tratado" (Analítico, Desafiante o Empático) y si está en modo Competencia.
- **Solicitar Rutina (Chat):** Es el núcleo de la app. El usuario escribe un mensaje (input).
- **Generar Respuesta Contextual:** Este es un proceso interno. Cuando el usuario pide una rutina, el sistema automáticamente recopila el perfil, la configuración psicológica y el archivo de texto (prompt maestro), y se lo envía a Gemini para crear la respuesta.



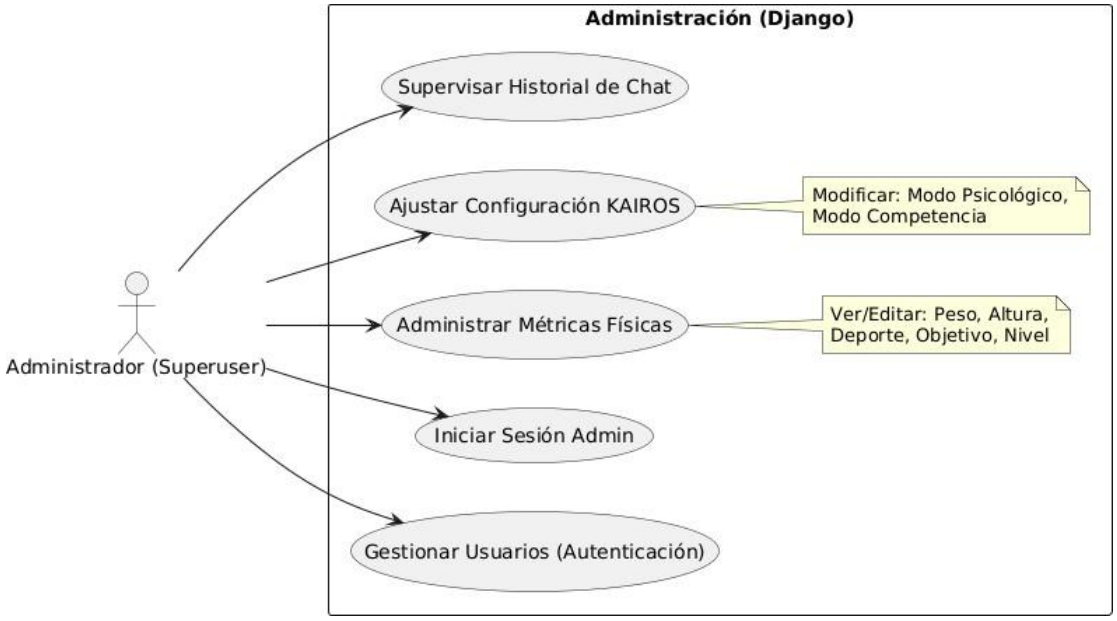
Elementos del diagrama de casos de uso 2

1. Los Actores

- **Administrador:** Es el actor principal, este con privilegios de administrador. No hace uso de la IA para entrenar, sino para gestionar datos, supervisar el funcionamiento del sistema u moderar por medio de Django.

2. Los Casos de Uso (Qué hace el sistema)

- **Inicio de Sesión (administrador):** Acceso exclusivo al superusuario, redirige al control interno como a la base de datos.
- **Gestión de usuarios (Autenticación):** Uso de Django para la administración de cuentas. Puede permitir que el administrador elimine o introduzca nuevos usuarios.
- **Administración de Métricas:** Control sobre el modelo de datos. El administrador puede visualizar y corregir errores en la información.
- **Configuración de KAIROS:** Permite la reestructuración de configuración, también la modificación de parámetros de comportamiento de la IA como el modo psicológico y competencia.
- **Supervisión de historial de chat (respuesta/solicitud):** Permite al administrador leer las interacciones almacenadas en el modelo para verificar y ajustar las respuestas de la IA.



DISEÑO DE INTERFAZ

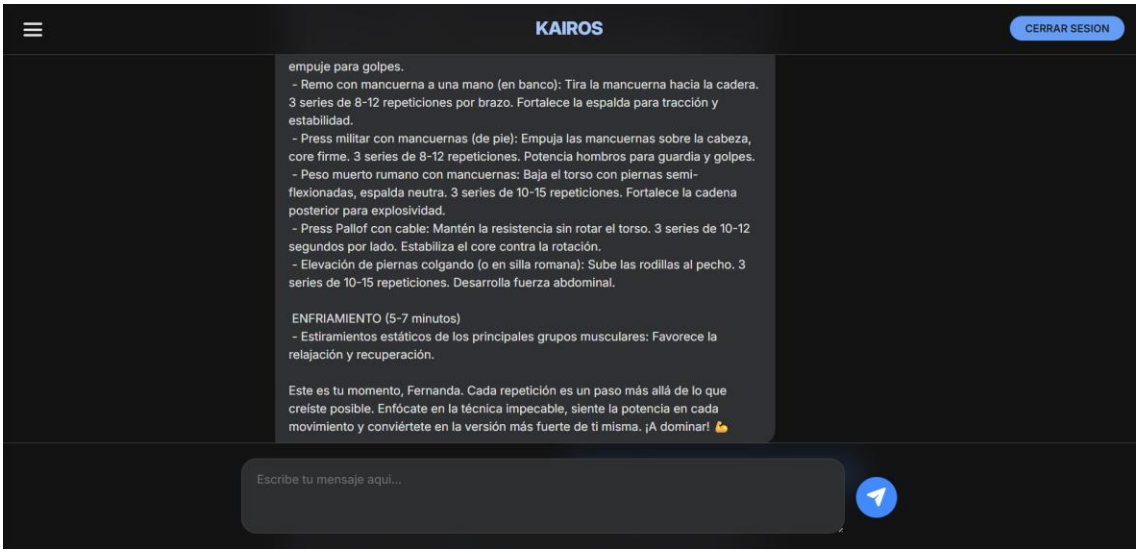
Al definir el diseño del proyecto, el equipo optó por adoptar un modelo de interacción familiar para el usuario, tomando como base la experiencia de usuario ofrecida por asistentes de IA y plataformas de chat existentes (como ChatGPT, Gemini, DeepSeek, e iMessage de iOS).

Se priorizó un diseño sencillo e intuitivo que garantizara la visibilidad y accesibilidad a todas las funcionalidades del sistema.

Estructura de la Interfaz Principal

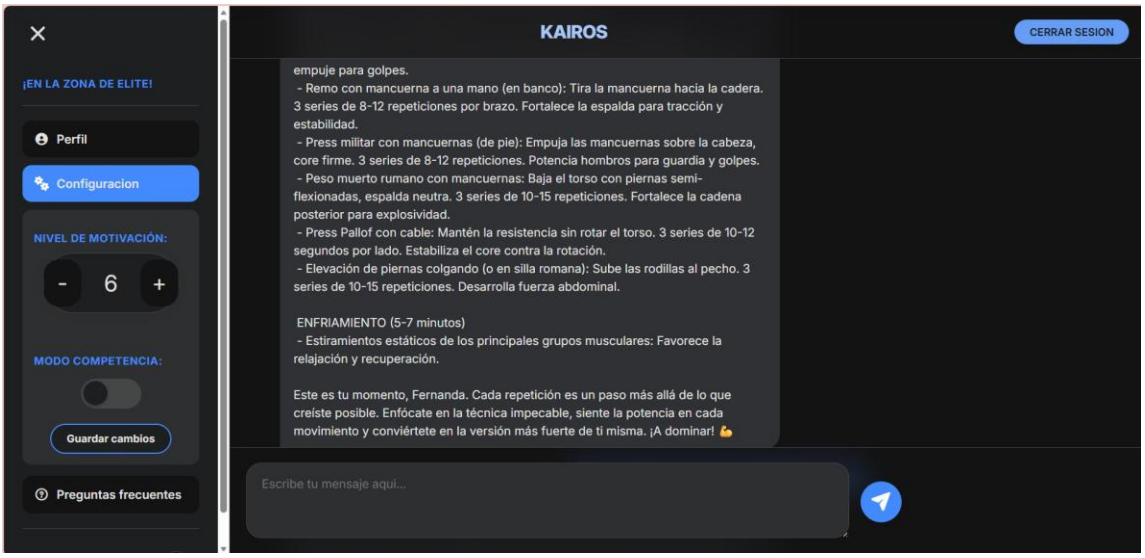
La interfaz principal del asistente se estructuró de la siguiente manera:

- Header: Contiene el nombre del asistente, las opciones de autenticación y un menú hamburguesa para navegación secundaria.
- Cuerpo Principal: Muestra un mensaje de bienvenida y el campo de ingreso de texto con su botón de envío para interactuar con KAIROS.



El menú hamburguesa se despliega para mostrar secciones como:

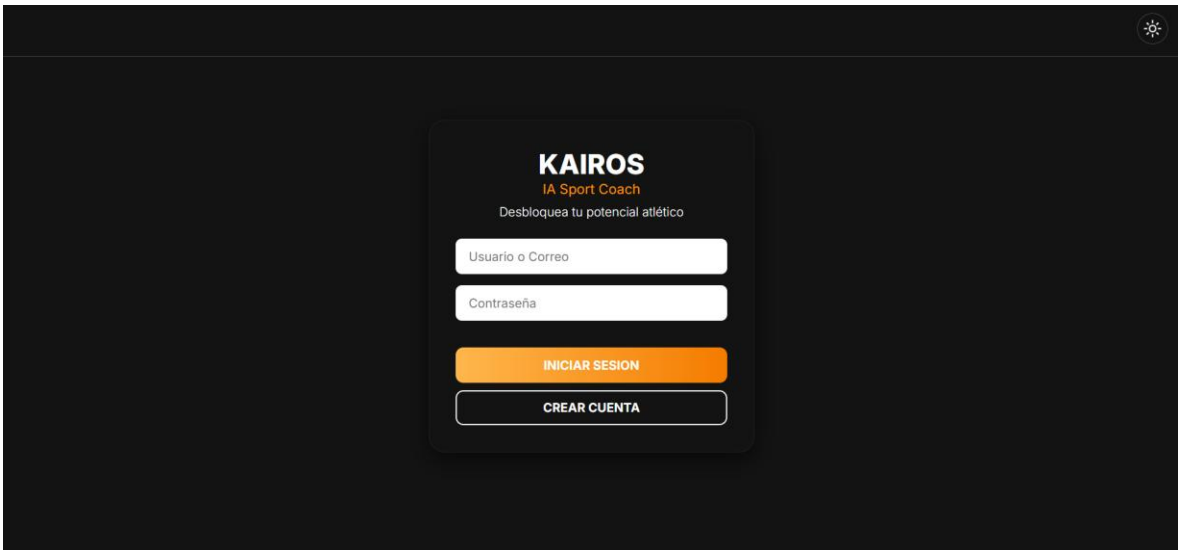
1. Métricas Físicas: Permite visualizar los datos introducidos por el usuario (peso, altura, etc.) y ofrece la opción de actualización.
2. Opciones de Configuración: Incluye el modo Psicológico (gestionado con un control stepper) y el modo Competencia (activación on/off mediante un switch).
3. Preguntas Frecuentes (FAQ): Sección de apoyo al usuario.



Interfaces de Acceso

Se desarrollaron dos interfaces de acceso fundamentales:

- Inicio de Sesión: Permite el acceso del usuario mediante la validación de correo electrónico y contraseña.
- Registro: Facilita la captura inicial de todos los datos necesarios para la personalización del asistente (peso, altura, objetivo deportivo, deporte principal, entre otros).



Diseño Visual y Paleta de Colores

Todas las interfaces mantuvieron una coherencia visual y un diseño similar, contemplando un modo oscuro para todo. La paleta de colores priorizó el azul y el blanco (o negro para el modo oscuro) porque estos colores aportan seriedad y estabilidad, esto complementado con detalles en naranja. Este color naranja fue elegido específicamente para asociarse con la energía, el dinamismo y la motivación inherentes al ámbito deportivo.

CRONOGRAMA

El cronograma que elaboramos al inicio del proyecto fue modificándose y generando nuevas versiones conforme avanzaba el desarrollo, ya que fuimos añadiendo más actividades. La primera versión quedó incompleta e ineficiente, por lo que utilizamos el cronograma como una herramienta de planificación que nos permitió organizar las tareas durante las semanas que dura el trimestre. Además, nos ayudó a dividir la carga de trabajo, lo que hizo que el progreso fuera más eficiente.

El cronograma cuenta con tres versiones.

- **Primera versión:** Se realizó antes del primer sprint, por lo que nos enfocamos únicamente en la fase inicial del proyecto sin considerar suficientemente los elementos futuros.

- **Segunda versión:** Fue creada durante el segundo sprint. En esta ampliamos las actividades, pero inicialmente cometimos el error de realizar demasiadas tareas en conjunto. Con el consejo de la profesora, logramos dividir mejor el trabajo entre los integrantes del equipo.
- **Tercera versión (final):** En esta última versión organizamos de forma más adecuada los tiempos restantes del trimestre para completar el tercer sprint. Nos enfocamos en integrar las partes desarrolladas individualmente y unirlas para presentar la versión final de Kairos.

Gracias al cronograma, pudimos visualizar el avance del proyecto y evitar sobrecargarse al final, logrando una mejor gestión del tiempo y del trabajo.

Enlace al archivo

SPRINT 2

En este sprint nos enfocamos en desarrollar el código necesario para avanzar en el proyecto. Gracias a la división de actividades entre los integrantes, pudimos trabajar de manera más organizada y lograr un mayor progreso.

Al finalizar el sprint, presentamos a la profesora los avances obtenidos y cómo seguimos el cronograma. Ella nos brindó retroalimentación, señalando qué aspectos aún nos faltaban y qué debíamos añadir o mejorar.

contenido simplificado del formato del sprint 2

¿Por qué es valioso este sprint?

Este sprint es importante porque nos permitirá crear las bases de datos donde guardaremos información del perfil del usuario y su historial. También nos ayudará a aprender lo necesario sobre Django para manejar el login, conectar la interfaz con el backend y trabajar con la API.

Además, ya contamos con el archivo que define el comportamiento de Kairos, así que en este sprint nos enfocaremos en que funcione correctamente como asistente deportivo, siguiendo sus reglas y características.

¿Qué entregables podemos obtener al finalizar este sprint?

Al terminar el sprint, deberíamos tener:

- **La base de datos funcional**, con sus tablas creadas en MySQL mediante las migraciones de Django, almacenando los datos de forma correcta.

- **El modelo de Inteligencia Artificial funcionando**, cumpliendo su rol como Kairos, respetando sus lineamientos, limitaciones y apoyando al usuario con rutinas y acompañamiento psicológico.

- **El servidor Django conectado con la interfaz**, permitiendo navegar entre páginas e iniciando las primeras pruebas de conexión con la API.

¿Cómo realizaremos el trabajo para lograr estos resultados?

Antes de comenzar, ya contamos con una idea clara de la arquitectura de Kairos, lo que nos permite dividir el proyecto en diferentes capas técnicas y trabajar en paralelo. Para lograrlo, necesitaremos comunicación constante entre los integrantes y buena organización.

Una vez que todos terminen el curso de Django, podremos avanzar con la conexión del frontend con el backend. Al mismo tiempo, completaremos la base de datos para enlazarla con Django y con la interfaz web.
Con estas bases listas, estaremos en condiciones de realizar las primeras pruebas con la API, siempre que el tiempo del sprint lo permita.

punto a mejor y faltantes a mejor dado por la profesora

Cian

-No puedo ver los Daily Scrum, ¿hicieron DS durante la semana 7?

Diagrama de secuencia

- Las líneas de vida inician cuando empieza la interacción.

- ¿Qué es el sistema? Sitio web, aplicación móvil o de escritorio ¿qué?

- ¿El sistema de dónde saca la interfaz? Falta un servidor web en caso de que sea un sitio web.

- Este diagrama lo comentamos en clase y no veo que haya una segunda versión.

- ¿Qué hace GEMINI? Le falta una línea de proceso.

- ¿Cómo interactúan el sistema y la BD? ¿Por qué la BD no es parte del sistema?

- ¿Gemini qué recibe para configurarse en los distintos modos chat, motivación, psicología, modo competencia? En la rutina personalizada al menos recibe datos del usuario.

Diagrama lógico

- Falta numeración.

- En el diagrama de secuencia el orden de interacción es distinto a lo que menciona aquí.

- Actualizar el diagrama lógico con las tecnologías o lenguajes que utilizarán.

- Descripción del diagrama lógico, conforme a la numeración.

Diagrama de desarrollo

- Obtener de aquí la planeación del Sprint 2.

- Veo que tienen planificadas el resto de semanas, es importante considerar lo que se logró y lo que no se logró (por ejemplo: estableciendo periodos más largos para alcanzar los objetivos o dando por terminadas actividades que se pensaba tomarían más tiempo).

SPRINT 3

Durante este último sprint unimos todos los trabajos individuales y completamos la fase final del proyecto. Esta etapa es clave porque es cuando comenzamos a ver el resultado del esfuerzo realizado durante todo el trimestre. También es aquí donde identificamos los errores y las partes faltantes del código. Gracias a esta fase final, pudimos corregir detalles menores y concentrarnos únicamente en los elementos más importantes del proyecto.

contenido simplificado del sprint 3

¿Por qué es valioso este sprint?

En este sprint finalmente podremos unir todas las partes del proyecto: la interfaz, la base de datos, Django y la API.
Los objetivos principales son:

1. **Conectar la base de datos con Django** y comprobar que la información se guarde correctamente.

2. **Unir Django con la interfaz**, verificando que los datos viajen bien entre ambas sin generar errores.

3. **Conectar la interfaz con la API**, asegurándonos de que el usuario pueda interactuar correctamente con la IA.

4. **Guardar las respuestas de la IA y las interacciones del usuario** en la base de datos.

5. **Revisar que todo funcione correctamente**, evitando fallos y errores finales.

¿Qué entregables podemos obtener al finalizar este sprint?

Al finalizar este sprint se espera **completar el proyecto**, logrando que el sitio web funcione según las especificaciones.
Esto incluye la integración total del sistema y una demostración final de su funcionamiento.

¿Cómo realizaremos el trabajo para entregar el incremento?

Como las partes individuales ya están terminadas, en este sprint trabajaremos de manera conjunta usando GitHub.
GitHub nos permitirá compartir, modificar y unir el código de todos en un solo lugar, facilitando el trabajo en equipo y la integración de cada componente.

PRUEBAS DE SOFTWARE

Realizaremos pruebas a este proyecto para verificar sus funciones y su funcionamiento de la misma forma se espera que al concluir las pruebas el proyecto ya pueda ser dado como terminado, en caso de que las pruebas no sean positivas se espera que se realicen cambios y reparaciones en el proyecto.
En esta prueba de software lo dividiremos en los siguientes elementos

- interfaz principal
- interfaz de inicio de sesión
- interfaz de registro

Interfaz principal

En esta interfaz es la primer interfaz que el usuario verá por lo cual es la que tiene mayor número de subcomponentes

- área de chat
- campo de texto
- botón de envío de texto
- menú de hamburguesa

Área de chat

En este subcomponente en grandes rasgos se trata la interacción usuario a ia en esta parte de la interfaz es la parte que muestra los mensajes generados por la ia(la respuesta a la interacción con el usuario). Verificaremos las siguientes pruebas para comprobar su uso de misma forma el objetivo de estas pruebas de expresar qué elementos están contenidos en este subcomponente.

Conexión api-interfaz	La API responde los mensajes que envía el usuario	La API responde a las peticiones del usuario
Generar rutina	La IA genera una rutina especializada contemplando el nivel, deporte y objetivo del usuario	Da recomendaciones según el nivel, objetivo, etc.
Modo psicológico	La IA es capaz de detectar el sentido del humor del usuario y le brinda atención y apoyo de acuerdo a su nivel establecido.	Detecta frustración, calma y aborda al usuario dependiendo el estado de ánimo y su configuración.
Mensajes correctos y concisos	Las respuestas mostradas al usuario no son muy largas o aburridas, además de evitar la sobreexplotación de ciertos ejercicios, además de mostrar formato compatible.	Se mejoró este rubro, ya da mensajes con formato compatible, limpio y concisas
Validación del prompt maestro en api	La IA toma el rol establecido en el prompt maestro y trabaja de acuerdo a él.	La IA toma el rol de KAIROS
Respuesta dentro del tiempo esperado	La IA debe responder en menos de 5 segundos en caso de tener conexión estable	El promedio de respuesta es de 7 segundos
Contenido seguro y sin bloqueo	La IA debe detectar que el usuario está frustrado o habla de deportes de contacto sin que este active el filtro de censura	Se puede hablar sobre algunos temas sin que se active el filtro de censura (por ejemplo golpes en el deporte de boxeo).
Guardar historial y api lo detecte	Se le pueden realizar preguntas sobre cosas del pasado a la IA y esta responde correctamente	No lo hace aún
Recepción de información o texto en el campo	Recibe la información planteada por el usuario para su envío posterior, el cual se activa con un botón	El campo permite el ingreso de texto con éxito, incluso si es amplio

Estas son las unidades que componen esta subcomponente, gracias a esta pruebas notamos que aunque la mayoría de nuestro elementos ya son aptos para la versión final también notamos algunos puntos donde debemos mejorar y otros donde deberemos replantear la forma en la que se realizaron.

Campo de texto

En este subcomponente se trata de donde el usuario podrá escribir.
realizamos la siguiente prueba para comprobar el uso adecuado de esta unidad

Recepción de información o texto en el campo	Recibe la información planteada por el usuario para su envío posterior, el cual se activa con un botón	El campo permite el ingreso de texto con éxito, incluso si es amplio

Botón de envío de texto

En este subcomponente se trata de un botón con el que el usuario podrá interactuar con la interfaz, de la misma forma este botón es el que hará funcionar todo el proceso del área de chant gracias a que conteste botón enviará lo que el usuario escriba en el campo de texto
realizamos la siguiente prueba para comprobar el funcionamiento de este botón (unidad)

Envío de texto	Al hacer clic o presionar enter, la información se envía al backend para que la IA de una respuesta	La interfaz hace el envío del texto al hacer clic en el botón

Menú de hamburguesa

En este subcomponente se trata de un menú el cual te dará más opciones que puede realizar nuestra interfaz, también nos permite interactuar con este menú para mejorar la interacción con el usuario, de la misma forma también nos permite activar el modo competencia.
Verificaremos las siguientes pruebas para comprobar su uso de misma forma el objetivo de estas pruebas de expresar qué elementos están contenidos en este subcomponente.

Activación de despliegue en todas las secciones	Al hacer clic el botón cambia de estado en la interfaz para mostrar un menú con opciones a configurar o datos del usuario	EAl hacer clic en el menú hamburguesa, se muestra con éxito
Actualización modo psicológico	Actualiza al nuevo nivel que el usuario decida (debe presionar botón guardar cambios)	Para procesar los datos está lista la bd, pero aún no se ha implementado en la interfaz
Actualización modo competencia	El usuario puede alternar la configuración como un interruptor para activar o desactivar este modo (debe presionar botón guardar cambios)	Para procesar los datos está lista la bd, pero aún no se ha implementado en la interfaz
Botón guardar cambios	Al hacer clic, se captura la modificación, valida y almacena en la bd	Para procesar los datos está lista la bd, pero aún no se ha implementado

		en la interfaz

Concluimos las pruebas como errores debido a que aun no eramos capaces de implementar estas unidades en la interfaz por lo cual Activación modo psicológico, actualización modo competencia, botón de guardar cambio.

Interfaz de inicio de sesión

En esta parte de la interfaz el usuario podrá entrar a su perfil por lo cual tendrá acceso a su historial y a su proceso en este componente se compone de estos siguientes subcomponentes:

- campo de correo
- campo para contraseña
- botón de inicio de sesión

CAMPO DE CORREO

En este campo el usuario pondrá su correo electrónico ya antes registrado en la página web.
Se realizarán las siguientes pruebas para verificar la validez del correo electrónico

Recepción de información o texto en el campo	Recuadro visual que permite la introducción de texto.	El campo permite el ingreso de información
Verificación de datos	Se valida el dato de correo introducido, es esencial que contenga '@'	Con éxito se verifica el formato de correo
Envio a la base de datos	La información se envía con una petición POST al servidor Django	Se hace un envío correcto a la bd
Verificación de correo registrado	Django recibe los datos y consulta las tablas de la bd para confirmar registro	La verificación en la bd es correcta, permite encontrar los correos registrados

Campo para contraseña

En este subcomponente el usuario podrá introducir su contraseña ya antes registrado
Se realizarán las siguientes pruebas para verificar la validez de la contraseña .

Recepción de la contraseña	Recuadro visual que captura la entrada oculta con asteriscos	El campo permite el ingreso de texto y se muestra formato seguro de asteriscos
Envío a la base de datos	La contraseña se envía al servidor Django y la procesa	El envío de la contraseña a la bd es exitoso

Validación de coincidencia con el correo	Django con la contraseña hace una comparación con el registro de la bd, y si coinciden, permite el acceso.	La validación de relación con el correo y lo registrado a la bd se da con éxito

Botón de inicio de sesión

En este subconjunto su función será enviar la información antes introducida en los subconjuntos “campo de correo” y “campo de contraseña”.
Se realizarán las siguientes pruebas que enviar la información para su verificación

Clic para activar el proceso	Detecta la interacción del usuario e inicia la recolección de datos	Previo al clic se inicia el proceso de validación
Verifica que los campos estén completos para acceder	Se hace una revisión de que todos los campos se hayan llenado	El sistema es capaz de verificar que todos los campos fueron ingresados
Ejecución	Los datos validados se envian a Django y procesa la solicitud, busca los datos en la bd y si todo es correcto, autoriza el acceso y dirige a la página principal	Los datos se logran enviar al backend y verificar, en caso de que sea correcto, permite el inicio de sesión

Interfaz de registro

En esta parte de la interfaz permitirá al usuario poder registrarse. en esta parte el usuario tendrá que introducir la información requerida para poder mejorar su interacción con kairos de la misma manera al registrarse las funciones principales de kairos .
En este componente consta de los siguientes subconjuntos:

- Campo de Correo
- Campo de Contraseña
- Campos de Métricas (peso, altura, edad)
- Selector (nivel, objetivo, deporte)
- Botón de Crear Cuenta

Campo de correo

En este subconjunto el usuario meterá un correo electrónico válido de la misma forma en este subconjunto también evitará el duplicado de cuentas con el mismo correo.
Estas son las unidades a las que se realizará las pruebas para registrará el correo de forma adecuada.

Recepción de información o	Entrada de texto que valida que lo ingresado contenga '@'	Con éxito permite que se ingrese en el campo un texto con formato de

texto en el campo		correo
Envío a la base de datos	El texto ingresado se envía en una solicitud POST para ser procesado por la vista del servidor para el registro	Se envia con éxito el correo ingresado al servidor
Verificación de duplicidad	Consulta a la tabla de usuarios para asegurarse de que el correo no esté registrado ya	Se hace la consulta de forma adecuada para evitar duplicidades

Campo de Contraseña

En este subconjunto el usuario introducirá su contraseña sin ninguna restricción
estos son las unidades para la prueba

Recepción de contraseña	El campo captura la clave deseada y oculta los elementos por asteriscos	Permite de forma adecuada el ingreso de una clave
Almacenamiento	El servidor toma la contraseña y la encripta para ser guardada con correspondencia al usuario	Después del clic, el sistema logra encriptar y almacenar la contraseña con su relación

Campo de Métrica(peso,altura,edad)

En este subconjunto el usuario necesitará introducir las siguientes unidades que son vitales para mejorar la experiencia y poder establecer rutinas adecuadas y personalizadas para el usuario
estos son las unidades para la prueba :

Recepción de datos numéricos	Los campos aceptan valores numéricos para construir el perfil del usuario	Se logra introducir información numérica de las métricas
Almacenamiento	Django recibe los datos y los almacena en la tabla de métricas	Las métricas se logran almacenar de forma adecuada para después mostrarse

Selector(nivel,objetivo,deporte)

En este subconjunto el usuario elegirá las opciones predeterminadas para completar el registro.
Estas unidades serán puedas a pruebas:

Selección de opción	El usuario elige una opción predefinida en una lista desplegable	Se logra hacer una selección de cada parámetro

Mapeo de datos	El servidor recibe los datos y los asocia según su propósito. Por ejemplo, el nivel psicológico lo coloca en un rango y lo asocia a un arquetipo	El servidor logra recibir la información
Almacenamiento	Django almacena los datos en la bd una vez hizo las asociaciones correspondientes	Django almacena de forma adecuada los datos y relaciones

Botón de Crear Cuenta

En este subconjunto solo se trata de un botón para guardar la información , verificará si los datos proporcionados son válidos y creará un nuevo usuario.
Estos son las unidades que se pondrán a pruebas:

Clic para iniciar registro	Recopila los datos necesarios para iniciar el registro	Se prepara el entorno de registro
Validación de campos	Se verifica que los campos se encuentren llenos y correctos	Se logran hacer las verificaciones para completar el registro
Creación de registro en la bd	Se ejecuta una transacción en donde se crea el usuario junto con sus datos vinculados	Se completa el registro y los datos pasan a ser almacenados en la bd

Enlace a documento "pruebas de software"

REGISTRO DE LAS SOLUCIONES DE LOS PROBLEMAS DE LAS PRUEBAS DE SOFTWARE

Se registraron algunas fallas durante las pruebas de software las siguientes pruebas fueron las no concluyentes o fallas:

Respuesta dentro del tiempo esperado	La IA debe responder en menos de 5 segundos en caso de tener conexión estable	El promedio de respuesta es de 7 segundos

Se realizó prueba independiente de esta unidad al final del proyecto ya con página web funcional.

Respuesta dentro del tiempo esperado	La IA debe responder en menos de 5 segundos en caso de tener conexión estable	El promedio de respuesta es de 3 segundos o menos

En esta falla aún no eramos capaces de realizar esta parte por falta de implementación

Guardar historial y api lo detecte	Se le pueden realizar preguntas sobre cosas del pasado a la IA y esta responde correctamente	No lo hace aún

Se realizó prueba independiente de esta unidad al final del proyecto ya con página web funcional.

Guardar historial y api lo detecte	Se le pueden realizar preguntas sobre cosas del pasado a la IA y esta responde correctamente	si la IA responde correctamente y es capaz de ver el historial

En esta falla se debió la no implementación de esta unidad de la misma forma al no implementarla no sabíamos si existían más complicaciones o fallas

Actualización modo psicológico	Actualiza al nuevo nivel que el usuario decida (debe presionar botón guardar cambios)	Para procesar los datos está lista la bd, pero aún no se ha implementado en la interfaz

Se realizó prueba independiente de esta unidad al final del proyecto ya con página web funcional.

Actualización modo psicológico	Actualiza al nuevo nivel que el usuario decida (debe presionar botón guardar cambios)	Para procesar los datos está lista la bd, implementado en la interfaz y funciona

En esta falla se debió la no implementación de esta unidad de la misma forma al no implementarla no sabíamos si existían más complicaciones o fallas

Actualización modo competencia	El usuario puede alternar la configuración como un interruptor para activar o desactivar este modo (debe presionar boton guardar cambios)	Para procesar los datos está lista la bd, pero aún no se ha implementado en la interfaz

Se realizó prueba independiente de esta unidad al final del proyecto ya con página web funcional.

Actualización modo competencia	El usuario puede alternar la configuración como un interruptor para activar o desactivar este modo (debe presionar botón guardar cambios)	Para procesar los datos está lista la bd, se ha implementado en la interfaz y funciona

En esta falla se debió la no implementación de esta unidad de la misma forma al no implementarla no sabíamos si existían más complicaciones o fallas

Botón guardar cambios	Al hacer clic, se captura la modificación, valida y almacena en la bd	Para procesar los datos está lista la bd, pero aún no se ha implementado en la interfaz

Se realizó prueba independiente de esta unidad al final del proyecto ya con página web funcional.

Botón guardar cambios	Al hacer clic, se captura la modificación, valida y almacena en la bd	Para procesar los datos está lista la bd, se ha implementado en la interfaz y funciona

CÓDIGO DEL PROYECTO

A continuación se detallan los bloques lógicos más relevantes del controlador principal.

REGISTRO:

Ubicacion: Funcion kairos_register.

Descripción: Este bloque contiene el proceso necesario para que el usuario pueda crear su cuenta, esto por medio de un formulario en el cual el usuario envía sus datos para posteriormente ser almacenados en la base de datos.

#Recibimos los datos por medio del formulario, estos datos seran los correspondientes a las metricas del usuario.

```
        username = request.POST['user']
        email = request.POST['email']
        password = request.POST['password']
        peso = float(request.POST.get('peso', 0) or 0)
        altura = float(request.POST.get('altura', 0) or 0)
        edad = int(request.POST.get('edad', 0) or 0)
        nivel_texto = request.POST.get('nivel')
        objetivo_texto = request.POST.get('objetivo')
        deporte_texto = request.POST.get('deporte')
#Verificamos que el correo no este en uso
        if User.objects.filter(email=email).exists():
            messages.info(request, 'El correo ya está en uso')
            return redirect('register')
        else:
            try:
                with transaction.atomic():
#Creamos el usuario y almacenamos sus datos
                    user = User.objects.create_user(username=username, email=email, password=password)
                    user.save()
#Creamos un espacio en la base de datos para almacenar las metricas fisicas y la configuracion del usario
                    MetricasFisicas.objects.create(
                        usuario=user,
```

```
                peso=peso,
                altura=altura,
                edad=edad,
                nivel=nivel_texto,
                objetivo=objetivo_texto,
                deporte=deporte_texto
            )
            Configuracion.objects.create(
                usuario=user,
                nivel_psico=4,
                modo_psicologico=1,
                modo_competencia=False
            )
#Enviar al usuario a la intrefaz principal, de otra forma mostrar mensaje de error
            messages.success(request, 'Registro exitoso')
            return redirect('login')
        except Exception as e:
            print(f"Error en registro: {e}")
            messages.error(request, 'Error al crear el perfil.')
            return redirect('register')
    else:
        #Enviar los datos predefinidos a la interfaz de registro.
        niveles = MetricasFisicas.NIVELES
```

```
    objetivos = MetricasFisicas.OBJETIVOS
    deportes = MetricasFisicas.DEPORTES
    return render(request, 'kairos_register.html', {
        'niveles': niveles,
        'objetivos': objetivos,
        'deportes': deportes
    })
```

CONSTRUCCIÓN DEL PERFIL DE USUARIO

Ubicación: Función kairos_main.

Descripción: Este bloque es la esencia del proyecto, ya que ayuda a la personalización. Aquí se obtienen los datos de la base de datos y se transforman en lenguaje natural que se enviará a la api, esto permite que Gemini tenga en cuenta siempre los datos del usuario antes de responder.

CÓDIGO COMENTADO:

```
try: if metricas and config: # Traducimos los booleanos y números de la BD a lenguaje natural para la IA
modo_txt = "ACTIVADO (Sé estricto)" if config.modo_competencia else "DESACTIVADO (Sé amable)"

    if config.modo_psicologico == 1: nivel_desc = "Nivel 1 (Técnico)"
  elif config.modo_psicologico == 2: nivel_desc = "Nivel 2 (Desafiante)"
  else: nivel_desc = "Nivel 3 (Empático)"

  # La IA recibirá esta "informacion" oculta antes de la pregunta del usuario
  contexto_usuario = f"""
  DATOS DEL ATLETA:
  - Nombre: {request.user.username}
  - Deporte: {metricas.deporte}
  - Nivel: {metricas.nivel}
  - Objetivo: {metricas.objetivo}
  - Peso/Altura: {metricas.peso}kg / {metricas.altura}m
  - MODO COMPETENCIA: {modo_txt}
  - MODO PSICOLÓGICO: {nivel_desc}
  """
else:
  contexto_usuario = "Usuario sin perfil configurado."
```

except Exception as ex_perfil: print(f"Error creando contexto usuario: {ex_perfil}")

LOGIN:

Ubicacion: Funcion kairos_login.

Descripción: Este bloque contiene el proceso necesario para que el usuario pueda acceder a su cuenta (creada previamente en la interfaz de registro), accediendo a la base datos, y verificando los datos por medio de un formulario.

#Recibimos los datos por medio del formulario

```
    user_input = request.POST['user']

    password = request.POST['password']

    username_login = user_input
```

#Dependiendo de si `user_input` tiene `@`, podremos verificar si es un correo o un nombre de usuario.

```
    if '@' in user_input:

        try:

            user_obj = User.objects.get(email=user_input)

            username_login = user_obj.username

        except User.DoesNotExist:

            username_login = None
```

#Realizamos la autentificación, si el usuario existe enviamos al usuario a la interfaz principal con su cuenta activa, de otra forma enviaremos un mensaje de alerta.

```
    user = None

    if username_login:

        user = auth.authenticate(username=username_login, password=password)

    if user is not None:
```

```
            auth.login(request, user)
            return redirect('main')
        else:
            messages.info(request, 'Usuario, Correo o Contraseña inválida')
            return redirect('login')
```

GESTIÓN DE MEMORIA CONVERSACIONAL (HISTORIAL)

Ubicación: Función kairos_main, antes de configurar el modelo.

Descripción: Gemini por defecto no tiene memoria, por lo que este bloque se encarga de recuperar las últimas interacciones de la base de datos, las convierte en estructura de tipo JSON que es el formato que requiere Google y las envía, de modo que la conversación sea fluida.

CÓDIGO COMENTADO:

```
historial_para_gemini = [] try: # Recuperamos los últimos 26 mensajes para no saturar el contexto (Token limit management) historial_db = Historial.objects.filter(usuario=request.user).order_by('-id')[:26]

# Invertimos el orden (reversed) para que la IA lea cronológicamente, de lo contrario pensará que el orden fue presente-pasado.
for h in reversed(historial_db):
    historial_para_gemini.append({"role": "user", "parts": [h.mensaje_usuario]})
    historial_para_gemini.append({"role": "model", "parts": [h.respuesta_ia]})


except: pass
```

CONFIGURACIÓN DE SEGURIDAD (SAFETY SETTINGS)

Ubicación: Función kairos_main, configuración del modelo.

Descripción: Dado que la aplicación trata temas deportivos (boxeo, artes marciales, fatiga muscular), es necesario ajustar los filtros de seguridad de la API para evitar falsos positivos de "violencia" o "daño físico".

La aplicación puede tratar temas deportivos (como artes marciales, heridas en entrenamiento, etc.) es necesario ajustar los filtros de seguridad de la API para evitar falsos positivos de "violencia" o "daño físico", lo que desactiva la API momentáneamente.

CÓDIGO COMENTADO:

safety_settings = [{"category": "HARM_CATEGORY_HARASSMENT", "threshold": "BLOCK_NONE"}, {"category": "HARM_CATEGORY_HATE_SPEECH", "threshold": "BLOCK_NONE"}, {"category": "HARM_CATEGORY_SEXUALLY_EXPLICIT", "threshold": "BLOCK_NONE"}, {"category": "HARM_CATEGORY_DANGEROUS_CONTENT", "threshold": "BLOCK_NONE"},]

INICIALIZACIÓN Y LLAMADA A LA API Ubicación: Función kairos_main.

Descripción: Aquí se instancia el cliente de Gemini con el prompt maestro (Contexto Base + Perfil Usuario) y se inicia la sesión de chat con el historial.

Nota: Se hizo uso del modelo gemini 2.5 flash por su rapidez, aunque podríamos cambiar de modelo si se requiere mayor análisis.

model = genai.GenerativeModel(model_name='models/gemini-2.5-flash', # Fusionamos el prompt maestrp (archivo txt) con los datos del usuario (BD) system_instruction=f"{contexto_base}\n\n{contexto_usuario}")

Se inicia el chat enviando el historial recuperado de la base de datos.

chat = model.start_chat(history=historial_para_gemini)

Se envía la solicitud del usuario

response = chat.send_message(mensaje_usuario, safety_settings=safety_settings)

BLOQUE 5: PERSISTENCIA DE DATOS Y MANEJO DE ERRORES Ubicación: Bloque try-except final y función intera_histo.

Descripción: Garantiza que la respuesta se guarde para futuras referencias y maneja posibles caídas de la API o bloqueos de seguridad.

try: respuesta_ia = response.text except Exception: # Fallback en caso de que la IA rechace responder por políticas de seguridad respuesta_ia = "Lo siento, mi sistema de seguridad se activó por error. Intenta reformular."

GUARDA LA INTERACCIÓN EN EL HISTORIAL.

intera_histo(request.user, mensaje_usuario, respuesta_ia)

CONCLUSIONES

Para el trabajo en equipo, en un principio, considero que fue complicado. La diversidad de ideas y experiencias dificultó la comunicación inicial. Sin embargo, esto eventualmente se transformó en un beneficio, ya que cada miembro aportó una visión distinta. Conforme avanzó el proyecto, logramos adaptarnos, entendernos mejor y todos colaboramos. Personalmente, me sentí cómoda y satisfecha porque pude aprender algo nuevo de cada uno de mis compañeros.

En cuanto al tema del proyecto, un asistente deportivo con IA me pareció muy interesante y relevante. La Inteligencia Artificial es una herramienta necesaria hoy en día y debemos aprender a convivir con ella e integrar en nuestras soluciones. Aunque el desarrollo fue enriquecedor y conocí nuevas herramientas, me hubiese gustado tener el tiempo para programar la lógica de la IA desde cero para entender sus fundamentos a mayor profundidad. Sobre los resultados, en forma general, pienso que obtuvimos buenos resultados. El hecho de haber dedicado tiempo suficiente a los diseños y a definir bien los requerimientos ayudó a que, al final, pudiéramos presentar lo que esperábamos. El producto final cumple con el objetivo, tiene la calidad esperada y refleja el esfuerzo del equipo.

Técnicas

En cuanto a los resultados, lo que sí se logró es que se alcanzó la meta de tener un asistente deportivo funcional conectado correctamente a una base de datos relacional, lo que nos permite gestionar el historial de chat y los archivos enviados por el usuario. Logramos crear una interfaz de usuario intuitiva, sencilla y amigable, implementando con éxito funcionalidades estéticas como el modo oscuro. Sobre lo que no se logró completar, es que faltó tiempo para realizar un entrenamiento más exhaustivo de la IA y pruebas de estrés adicionales, lo que resulta en que el modelo, aunque funcional, podría ser más ágil y optimizado en sus tiempos de respuesta.

Como trabajo a futuro, para futuras iteraciones, el trabajo se centraría en la optimización de la velocidad de respuesta del modelo y en hacer el diseño aún más minimalista. Basándonos en las sugerencias recibidas, sería ideal agregar un calendario de rutinas y recursos multimedia para mostrar videos o

imágenes de las posturas de los ejercicios. También es fundamental implementar capas de seguridad más robustas para la protección de los datos.

Dicho esto, en clase me hubiera gustado ver o analizar ejemplos prácticos de la creación de proyectos desde cero, observando el ciclo de vida completo, incluyendo cómo se construye una IA desde sus cimientos. En cuanto a mi formación técnica, me quedé con el interés de profundizar mucho más en teoría avanzada de bases de datos y en ciberseguridad. También me hubiera gustado probar otros lenguajes de programación que quizás ofrecieran un mayor rendimiento para este tipo de sistemas.

Finalmente, mi contribución principal fue en el Backend. Me encargué específicamente del diseño y la normalización de la base de datos, así como de la creación de los modelos en Django y la programación de la lógica de las vistas (views) relacionada con la base de datos. Adicionalmente, colaboré un poco en la creación del prompt maestro para las respuestas y personalidad del asistente, y apoyé con ajustes puntuales en el HTML y CSS del frontend.

BIBLIOGRAFÍA

- **IBM – Inteligencia Artificial**
  Stryker, C., & Kavlakoglu, E. (s. f.). *¿Qué es la inteligencia artificial o IA?* IBM Think. https://www.ibm.com/mx-es/think/topics/artificial-intelligence IBM
- **APD – Inteligencia Artificial**
  APD. (s. f.). *Técnicas de la inteligencia artificial: ¿Cuáles son y para qué se utilizan?* https://www.apd.es/tecnicas-de-la-inteligencia-artificial-cuales-son-y-para-que-se-utilizan/ URL inválido
- **DataCamp – Lenguajes de programación para IA**
  DataCamp. (s. f.). *Los 10 mejores lenguajes de programación de IA: guía de iniciación para principiantes.* https://www.datacamp.com/es/blog/ai-programming-languages DataCamp
- **Developer.mozilla.org – Introducción a Django**
  Mozilla Developer Network. (s. f.). *Introducción a Django.* https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/Django/Introduction MDN Web Docs
- **Django Documentation**
  Django Software Foundation. (s. f.). *Django documentation (versión 6.0).* https://docs.djangoproject.com/en/6.0/ Django Project
- **Ferrovial – Inteligencia artificial**
  Ferrovial. (s. f.). *Qué es la inteligencia artificial y cómo la aplicamos.* https://www.ferrovial.com/es/recursos/inteligencia-artificial/ URL inválido
- **Google AI – Gemini API (documentación principal)**
  Google AI. (s. f.). *Gemini API documentation.* https://ai.google.dev/gemini-api/docs?hl=es-419 Google AI for Developers

- **Google AI – Gemini API (clave de API)**
  Google AI. (s. f.). *Cómo usar claves de API de Gemini.* https://ai.google.dev/gemini-api/docs/api-key?hl=es-419 Google AI for Developers
- **Python.org – Documentación de Python**
  Python Software Foundation. (s. f.). *Python documentation.* https://www.python.org/doc/ Python.org