GRUPO#3

MEMORY GAME

INTEGRANTES

2

Fatima Isabel Briones

1

Ingrys Dariela Santamaria

3

Kensy Rosio Rodriguez

4

Denilson Josue Galo

1

Jhonathan Josue Cruz

2

Erikson Rodriguez Paz

4

Patrik Mijail Mendoza

3

Oscar Alonso Reyes

CONTEXTUALIZACION DEL JUEGO MEMORY GAME

MEMORY GAME es un juego clásico de memoria donde los jugadores deben encontrar pares de cartas idénticas.

- Es ampliamente utilizado en educación, entrenamiento cognitivo y entretenimiento.

- En este proyecto, su implementación se utiliza para aplicar conceptos de arquitectura y organización de computadoras.

RELEVANCIA EN ARQUITECTURA DE COMPUTADORAS

- Permite aplicar principios de eficiencia de recursos, rendimiento y diseño de sistemas.

- Utiliza tecnologías como Docker, que permiten un entorno aislado y portátil.

- Refuerza el entendimiento del manejo de memoria, procesamiento y estructuras de datos eficientes.

OBJETIVOS

01

Diseñar e implementar el juego MEMORY GAME usando Django y Docker.

02

Optimizar el rendimiento y uso eficiente de recursos del sistema.

03

Aplicar los conceptos fundamentales de la arquitectura de computadoras en un proyecto práctico.

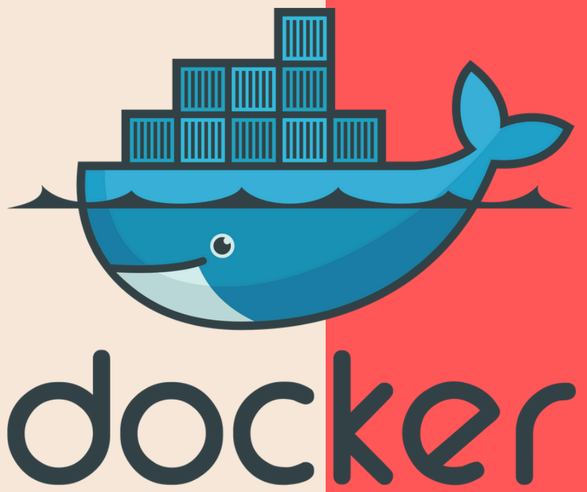
VINCULACIÓN CON ARQUITECTURA DE COMPUTADORAS

- Uso de contenedores para aislamiento y portabilidad.
- Optimización de algoritmos y estructuras de datos.
- Evaluación del rendimiento del sistema a través de análisis de recursos.

Fases del Desarrollo

- Fase de configuración del entorno con Docker:

Se creó un entorno de desarrollo aislado mediante la herramienta Docker, utilizando archivos Dockerfile y docker-compose.yml para definir y orquestar los servicios necesarios, como el servidor web de Django.



- Fase de desarrollo de la interfaz gráfica:

Se diseñó una interfaz web intuitiva y responsiva utilizando HTML, CSS, Bootstrap y JavaScript. Esta interfaz permite al jugador interactuar fluidamente con el tablero de cartas.

- **FASE DE CONFIGURACIÓN DEL ENTORNO CON DOCKER**

SE CREÓ UN ENTORNO DE DESARROLLO AISLADO MEDIANTE LA HERRAMIENTA DOCKER, UTILIZANDO ARCHIVOS DOCKERFILE Y DOCKER-COMPOSE.YML PARA DEFINIR Y ORQUESTAR LOS SERVICIOS NECESARIOS, COMO EL SERVIDOR WEB DE DJANGO.

- **FASE DE IMPLEMENTACIÓN DEL JUEGO CON DJANGO:**

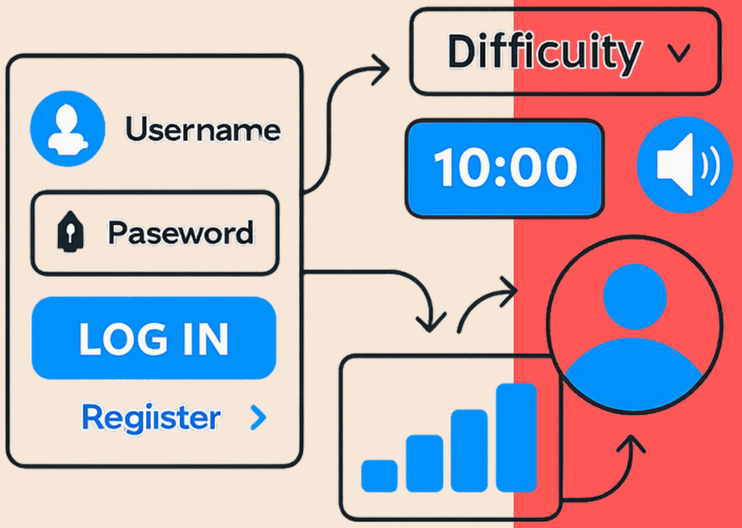
SE DESARROLLÓ LA LÓGICA PRINCIPAL DEL JUEGO CREANDO UNA APLICACIÓN DJANGO LLAMADA 'MEMORY_GAME', INCLUYENDO LOS MODELOS PARA GESTIONAR LAS CARTAS, USUARIOS, ESTADÍSTICAS Y PROGRESO DEL JUGADOR.

- Fase de desarrollo de la interfaz gráfica:

Se diseñó una interfaz web intuitiva y responsiva utilizando HTML, CSS, Bootstrap y JavaScript. Esta interfaz permite al jugador interactuar fluidamente con el tablero de cartas.

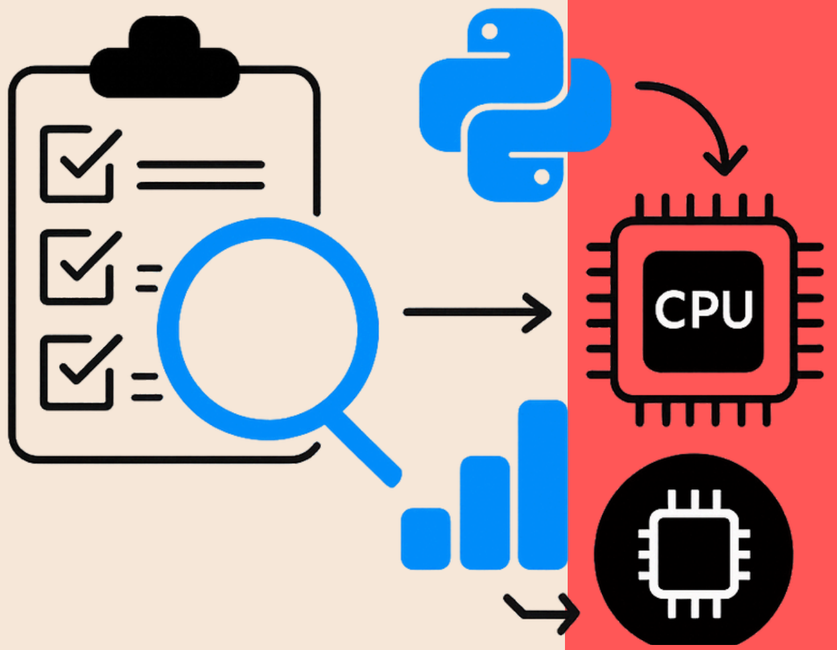
- Fase de integración de funcionalidades:

Se integraron características como login y registro de usuarios, selección de nivel de dificultad (básico, medio, avanzado), temporizador, efectos de sonido, estadísticas y perfil del jugador.



- Fase de pruebas y optimización:

Se realizaron pruebas exhaustivas del sistema y se aplicaron técnicas de optimización tanto en la lógica de Python como en el uso de recursos (memoria, CPU), garantizando el rendimiento del juego.



IMPORTANCIA DE CADA FASE EN RELACIÓN CON LA MEJORA DE LA ORGANIZACIÓN Y ARQUITECTURA DEL SISTEMA

- LA CONFIGURACIÓN CON DOCKER PERMITIÓ AISLAR EL ENTORNO DE DESARROLLO, GARANTIZANDO PORTABILIDAD, REPLICABILIDAD Y EFICIENCIA EN EL USO DE RECURSOS DEL SISTEMA.

- LA IMPLEMENTACIÓN CON DJANGO FACILITÓ UNA ORGANIZACIÓN MODULAR DEL CÓDIGO, MEJORANDO LA MANTENIBILIDAD Y ESCALABILIDAD.

- EL DISEÑO DE LA INTERFAZ MEJORÓ LA EXPERIENCIA DE USUARIO Y SEPARÓ LA LÓGICA DE PRESENTACIÓN DE LA LÓGICA DEL JUEGO.

- LA INTEGRACIÓN DE FUNCIONALIDADES PERMITIÓ REFLEJAR UNA ARQUITECTURA ROBUSTA CON SEPARACIÓN CLARA DE RESPONSABILIDADES.

- LAS PRUEBAS Y OPTIMIZACIÓN AYUDARON A DEPURAR ERRORES Y ASEGURAR UN COMPORTAMIENTO ÓPTIMO DEL SISTEMA BAJO DIFERENTES CONDICIONES.

Conexión explícita entre las fases del desarrollo y los conceptos teóricos aprendidos en clase.

- VIRTUALIZACIÓN Y CONTENEDORES: SE UTILIZÓ DOCKER COMO EJEMPLO PRÁCTICO DE VIRTUALIZACIÓN LIGERA, PERMITIENDO UNA EJECUCIÓN EFICIENTE Y AISLADA DEL SISTEMA.

- GESTIÓN DE RECURSOS: SE OPTIMIZARON ALGORITMOS Y ESTRUCTURAS DE DATOS PARA MEJORAR EL RENDIMIENTO DEL JUEGO, APLICANDO PRINCIPIOS DE EFICIENCIA EN EL USO DE CPU Y MEMORIA.

- ARQUITECTURA CLIENTE-SERVIDOR Y NIVELES DE ABSTRACCIÓN: EL USO DE DJANGO PERMITIÓ TRABAJAR CON MÚLTIPLES CAPAS (MODELOS, VISTAS, PLANTILLAS) SIGUIENDO UNA ARQUITECTURA ORGANIZADA.

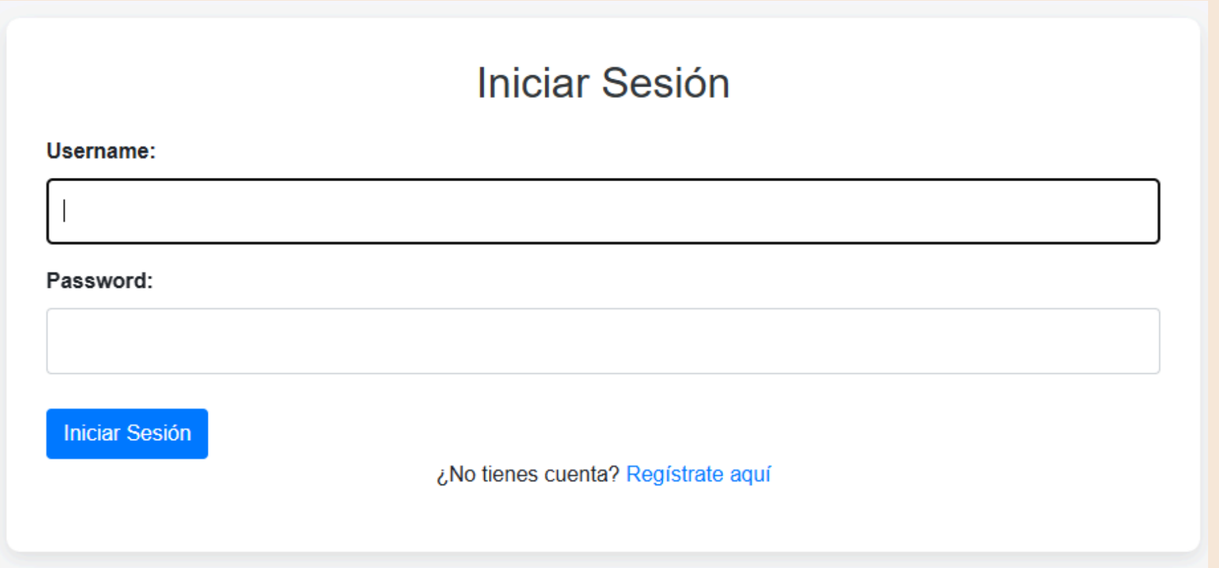
3

2

- DESARROLLO MODULAR: SE ESTRUCTURÓ EL CÓDIGO DEL SISTEMA EN MÓDULOS, SIGUIENDO PRINCIPIOS DE ORGANIZACIÓN LÓGICA DEL SOFTWARE VISTOS EN CLASE.

- LA INTEGRACIÓN DE FUNCIONALIDADES PERMITIÓ REFLEJAR UNA ARQUITECTURA ROBUSTA CON SEPARACIÓN CLARA DE RESPONSABILIDADES.

- LAS PRUEBAS Y OPTIMIZACIÓN AYUDARON A DEPURAR ERRORES Y ASEGURAR UN COMPORTAMIENTO ÓPTIMO DEL SISTEMA BAJO DIFERENTES CONDICIONES.



Aqui mostramos la interfaz de la pantalla de inicio de sesión del proyecto Memory Game. En el centro se encuentra un formulario con el título "Iniciar Sesión".

El formulario incluye dos campos de entrada: uno para el nombre de usuario (Username) y otro para la contraseña (Password).

Debajo de los campos, hay un botón azul con el texto "Iniciar Sesión", y un enlace que dice "¿No tienes cuenta? Regístrate aquí", permitiendo al usuario ir a la pantalla de registro. El diseño es limpio y centrado, ideal para una interfaz inicial de usuario.

Justo aqui mostramos la pantalla de bienvenida tras iniciar sesión en el juego Memory Game. Se presenta un saludo personalizado con el nombre del usuario (“Hola, Iasbel”) y una invitación a seleccionar un nivel de dificultad.

Se ofrecen tres opciones:

- Nivel Básico: 10 intentos, 8 cartas, 60 segundos (color azul claro).
- Nivel Medio: 8 intentos, 12 cartas, 90 segundos (color amarillo suave).
- Nivel Avanzado: 6 intentos, 16 cartas, 120 segundos (color rojo claro).

Debajo de las opciones, hay dos botones: uno turquesa que dice “Ver mi Perfil” y otro gris para “Cerrar Sesión”. La interfaz es clara, amigable y con un diseño visual que facilita la navegación.

Aqui mostramos la pantalla del perfil del usuario dentro del juego Memory Game, específicamente el de “Iasbel”.

En la parte superior se presentan las estadísticas generales del jugador:

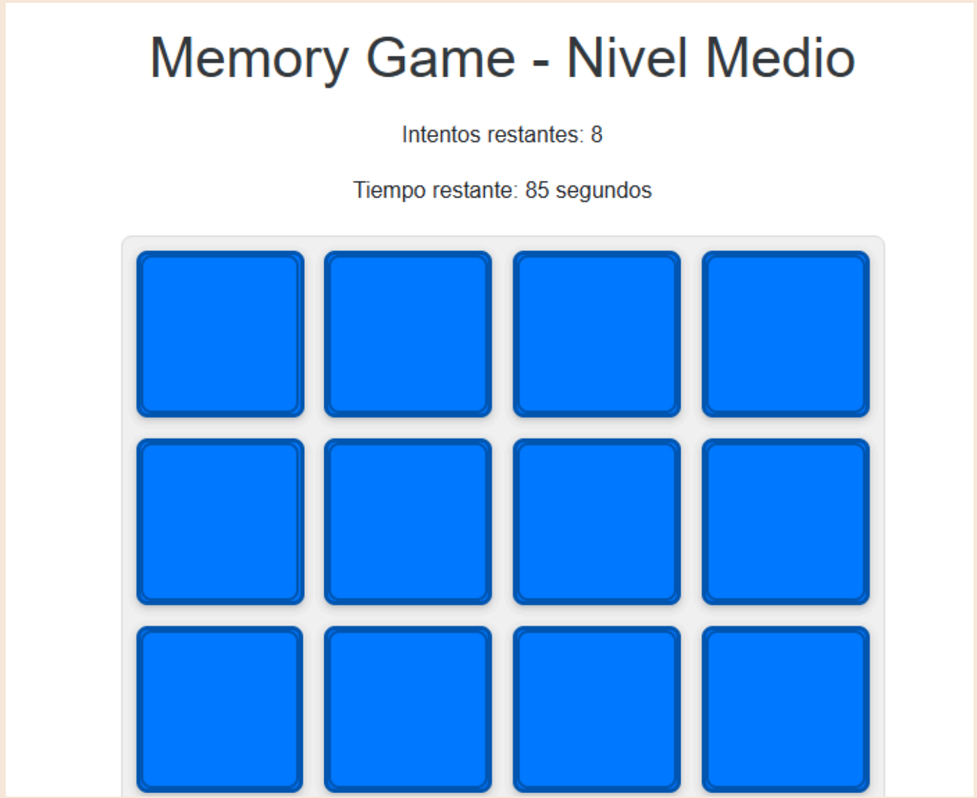
- Victorias Totales: 1
- Derrotas Totales: 0
- Partidas Jugadas: 1
- Nivel más jugado: N/A

Debajo se encuentra el historial de partidas, el cual incluye columnas para la fecha/hora, nivel, duración en segundos y el resultado. En este caso, se registra una partida en nivel Básico, con duración de 13 segundos y resultado de "Victoria" (marcado con una etiqueta verde).
Al final, se ofrecen dos botones: "Volver a Selección de Nivel" y "Cerrar Sesión", para continuar o salir del perfil.

Aqui se muestra el inicio del juego de memoria (Memory Game) en su nivel básico.

En el centro se observa una cuadrícula de 8 tarjetas azules (4 en la fila superior y 4 en la inferior), todas boca abajo, listas para ser seleccionadas.
En la parte superior, el texto indica:
Intentos restantes: 10
Tiempo restante: 58 segundos

El objetivo del juego es encontrar los pares de tarjetas iguales antes de que se acaben los intentos o el tiempo. El diseño es sencillo y claro, ideal para un nivel inicial.



Interfaz del juego de memoria correspondiente al Nivel Medio, que presenta una cuadrícula de 12 tarjetas ocultas organizadas en tres filas y cuatro columnas.

El sistema indica que el usuario dispone de 8 intentos restantes y 85 segundos para completar la actividad.

Pantalla del Nivel Avanzado del juego de memoria, con una cuadrícula de 16 tarjetas ocultas dispuestas en cuatro filas y cuatro columnas.

El jugador cuenta con 6 intentos restantes y 120 segundos para completar el desafío.

Presentación del juego funcional