



UNIVERSIDAD
TECNOLÓGICA DE HONDURAS

Erick Fernando Robles Cruz	202110010322
Irvin Leonel Lanza Meza	201610110496
Zohet Abigaíl Aguirre moreno	202210040039
Paola Abigail Peña Hernández	202330080046
Luis Enrique Cruz Gámez	202210010814
Addys Espinoza Calderon	202210010176

Asignatura: Arquitectura De Computadoras

Catedrático: Ricardo Enrique Lagos

Fecha de entrega: 11 de agosto de 2024

Informe Técnico

• Introducción al Proyecto:

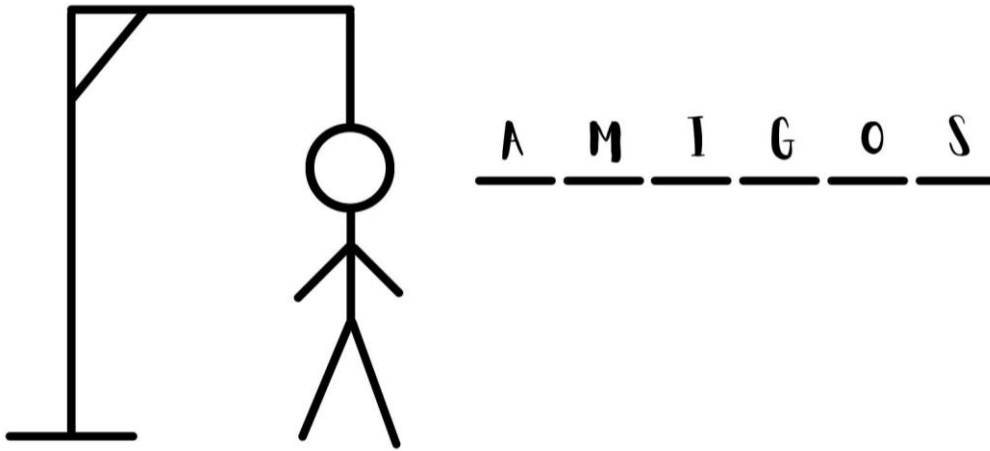
1. Descripción breve del juego del ahorcado y sus objetivos.

El ahorcado (también llamado colgado) es un juego para dos o más jugadores. Un jugador piensa, en una palabra, frase u oración y el otro trata de adivinarla según lo que sugiere por letras o dentro de un cierto número de oportunidades.

En su descripción general: Usando una fila de guiones, se representa la palabra a adivinar, dando el número de letras, números y categoría. Si el jugador adivinador sugiere una letra o número que aparece en la palabra, el otro jugador la escribe en todas sus posiciones correctas. Si la letra o el número sugerido no ocurre en la palabra, el otro jugador saca un elemento de la figura de hombre palo ahorcado como una marca de conteo.

El juego termina cuando:

- ✓ El jugador adivinador completa la palabra, o adivina la palabra completa correctamente
- ✓ El otro jugador completa el diagrama



Este diagrama es, de hecho, diseñado para parecerse a un hombre ahorcado. A pesar de que han surgido debates sobre el gusto cuestionable de esta imagen, todavía está en uso actualmente (2024). Una alternativa común para maestros es dibujar un árbol de manzanas con diez manzanas, borrar o tachar las manzanas a medida que se agotan las adivinanzas.

La naturaleza exacta del diagrama difiere; algunos jugadores dibujan la horca antes de jugar y dibujan las partes del cuerpo del hombre (tradicionalmente la cabeza, luego el torso, luego los brazos y las piernas de uno en uno). Algunos jugadores comienzan con sin esquema en absoluto, y elaboran los elementos individuales de la horca como parte del juego, dándole al jugador más posibilidades efectivas medida de adivinar.

Objetivos del Proyecto

Objetivo General:

- El objetivo de este proyecto es desarrollar una versión digital del juego del ahorcado utilizando Python, Django y Docker.

Objetivos específicos:

Para el desarrollo del juego:

- Implementar la lógica del juego del ahorcado en Python.
- Crear una interfaz de usuario web para el juego utilizando el framework Django.
- Permitir a los usuarios interactuar con el juego a través de su navegador web.

En la Optimización y Eficiencia:

- Optimizar el rendimiento del juego para asegurar una experiencia de usuario fluida.
- Implementar prácticas de eficiencia en el uso de recursos, tanto en el backend como en el frontend.

En la Implementación con Docker:

- Utilizar Docker para contenerizar la aplicación, facilitando su despliegue y portabilidad.
- Asegurar que el entorno de desarrollo y producción sea consistente y fácil de gestionar.

En la Experiencia de Usuario:

- Diseñar una interfaz intuitiva y atractiva que mejore la experiencia de juego.
- Incluir características adicionales como puntuaciones, niveles de dificultad y estadísticas de juego.

- **Análisis del Problema:**

1. Identificación y análisis del problema.

Identificación del Problema:

- Aislamiento del entorno de desarrollo: Es muy importante evitar conflictos de dependencias y garantizar que el proyecto funcione de manera consistente en diferentes entornos, por lo que se requiere el uso de Docker para aislar el entorno de desarrollo en que se trabaja.
- Implementación eficiente del juego: Crear un juego del ahorcado con Django implica desarrollar una lógica de juego complejo, gestionar la interacción del usuario, y almacenar información importante del juego de manera eficiente.
- Optimización del rendimiento: Se debe minimizar el uso de recursos del sistema, como la memoria y el tiempo de CPU, para asegurar que el juego funcione de manera fluida, incluso en sistemas con recursos limitados.
- Experiencia de usuario: La interfaz web debe ser intuitiva y atractiva que ofrezca una buena experiencia al usuario, lo que puede requerir el uso de tecnologías web como HTML, CSS, y JavaScript.
- Pruebas y depuración: Garantizar que el juego funcione correctamente en todos los escenarios posibles requiere un enfoque exhaustivo en pruebas y depuración.

Análisis del Problema:

- Entorno de desarrollo inconsistente: Sin un entorno de desarrollo aislado, el proyecto podría sufrir de inconsistencias debido a diferencias en las configuraciones del sistema operativo, lo que podría dificultar el desarrollo y la depuración.
- Bajo rendimiento: Sin optimización, el juego podría consumir más recursos de los necesarios, lo que podría llevar a tiempos de respuesta lentos y una mala experiencia del usuario.

- Complejidad en la implementación: La lógica del juego del ahorcado puede ser compleja, especialmente cuando se trata de manejar entradas del usuario y almacenar el estado del juego de manera eficiente.
- Riesgos en la experiencia de usuario: Una interfaz mal diseñada puede resultar en una experiencia de usuario deficiente, lo que afectaría la aceptación y el éxito del juego.

- **Pasos del Proyecto:**

1. Detalles sobre la implementación del juego.

Primer Paso: Configuración del Entorno de Desarrollo con Docker

- **Objetivo:** Asegurar que el desarrollo del juego se realice en un entorno controlado y consistente, evitando problemas de compatibilidad entre diferentes sistemas operativos.
- **Acciones:**
 - **Dockerfile:** Crear un archivo Dockerfile para definir el entorno necesario, incluyendo Python, Django, y otras dependencias necesarias para el proyecto.
 - **Docker Compose:** Utilizar un archivo docker-compose.yml para orquestar los contenedores necesarios. Esto incluirá la configuración del servidor web de Django y cualquier otro servicio de ser necesario como bases de datos.

Segundo Paso: Implementación del Juego del Ahorcado con Django

- **Objetivo:** Desarrollar la lógica del juego del ahorcado dentro del marco de Django, asegurando que el juego sea interactivo y se gestione correctamente la interacción del usuario.
- **Acciones:**
 - **Proyecto Django:** Iniciar un nuevo proyecto Django dentro del contenedor Docker, lo que servirá como la base para el desarrollo del juego.
 - **Aplicación Django hangman:** Crear una aplicación específica llamada hangman que contenga toda la lógica del juego. Esta aplicación manejará las reglas del juego, como la selección de la palabra a adivinar, el seguimiento de las letras utilizadas, y la determinación del estado del juego (ganado/perdido).
 - **Modelos de Django:** Definir modelos en Django para almacenar la información relevante del juego, como la palabra seleccionada,

las letras que el usuario ha adivinado, y el estado del juego en curso.

- **Vistas y Plantillas HTML:** Desarrollar vistas en Django para gestionar las interacciones del usuario con el juego. Estas vistas enviarán y recibirán datos a través de formularios y actualizarán la interfaz de usuario (UI) según corresponda. Las plantillas HTML se encargarán de la presentación del juego al usuario.

Tercer Paso: Optimización del Rendimiento y Eficiencia de Recursos

- **Objetivo:** Asegurar que el juego funcione de manera eficiente y rápida, incluso en sistemas con recursos limitados.
- **Acciones:**
 - **Optimización del Código Python:** Revisar y optimizar el código para eliminar redundancias y mejorar la eficiencia. Esto puede incluir la optimización de bucles, la reducción de operaciones costosas y la simplificación de la lógica donde sea posible.
 - **Estructuras de Datos:** Utilizar estructuras de datos eficientes, como diccionarios y conjuntos, para manejar la lógica del juego, permitiendo operaciones rápidas y con bajo consumo de memoria.
 - **Gestión de Recursos:** Implementar técnicas para minimizar el uso de recursos del sistema, como la memoria y el tiempo de CPU. Esto puede incluir la optimización de algoritmos y la gestión eficiente de entradas/salidas.

Cuarto Paso: Desarrollo Web con Django

- **Objetivo:** Crear una interfaz de usuario interactiva y atractiva para el juego del ahorcado, utilizando las capacidades de Django para manejar la lógica del lado del servidor.
- **Acciones:**

- **Integración de Lógica y Vistas:** Conectar la lógica del juego con las vistas de Django, de manera que las acciones del usuario (como adivinar una letra) se reflejen inmediatamente en la interfaz del juego.
- **Formularios Django:** Utilizar formularios para recoger la entrada del usuario (las letras que adivina), procesar esta entrada y actualizar el estado del juego.
- **Interfaz de Usuario:** Diseñar una UI que sea intuitiva y fácil de usar, utilizando HTML y CSS. Si es necesario, se puede incluir JavaScript para mejorar la interactividad, como la actualización en tiempo real de la interfaz.

Quinto Paso: Pruebas y Depuración

- **Objetivo:** Asegurar que el juego funcione correctamente en todos los escenarios posibles y que cualquier problema se detecte y solucione rápidamente.
- **Acciones:**
 - **Pruebas Unitarias:** Implementar pruebas unitarias para cada parte del juego, asegurando que cada función individual funcione correctamente.
 - **Pruebas de Integración:** Realizar pruebas de integración para asegurar que los diferentes componentes del juego (lógica, vistas, interacción con el usuario) funcionen juntos sin problemas.
 - **Depuración:** Utilizar herramientas de depuración para identificar y corregir errores en el código. Además, realizar un análisis de rendimiento para detectar posibles cuellos de botella y optimizar el código según sea necesario.

- **Desarrollo de la interfaz gráfica con Python Django**

Objetivo:

El objetivo principal fue crear una interfaz intuitiva y atractiva para el juego de ahorcado, permitiendo a los usuarios interactuar de manera sencilla tanto desde la página de inicio como durante el juego.

Estructura de la interfaz:

La interfaz se compone de dos páginas principales:

Página de Inicio (index.html): Contiene el menú con opciones para seleccionar avatar, ingresar el nombre de usuario, iniciar el juego, y acceder a la ayuda.

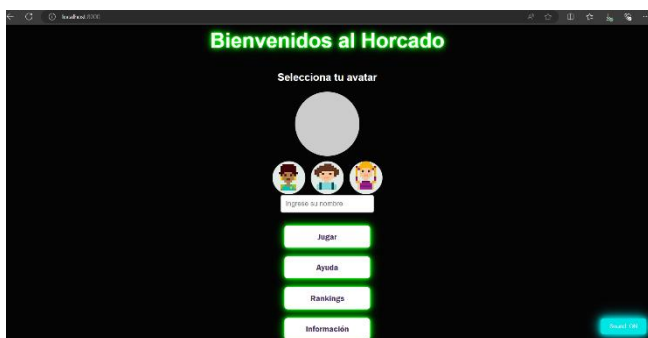
Página del Juego (hangman.html): Muestra la interfaz del juego de ahorcado, donde el usuario puede adivinar la palabra oculta y ver el progreso visual del ahorcado.

- **Diseño:**

El diseño fue pensado para ser simple pero atractivo, usando colores y fuentes que hacen que el juego sea agradable de jugar. Se priorizó la usabilidad, permitiendo una experiencia fluida y accesible.

Experiencia de usuario (UX):

La interfaz es responsiva, permitiendo su uso en diferentes dispositivos, y se añadieron modales para mensajes de ayuda y error, mejorando la interacción del usuario.



- **Documentación del Código:**

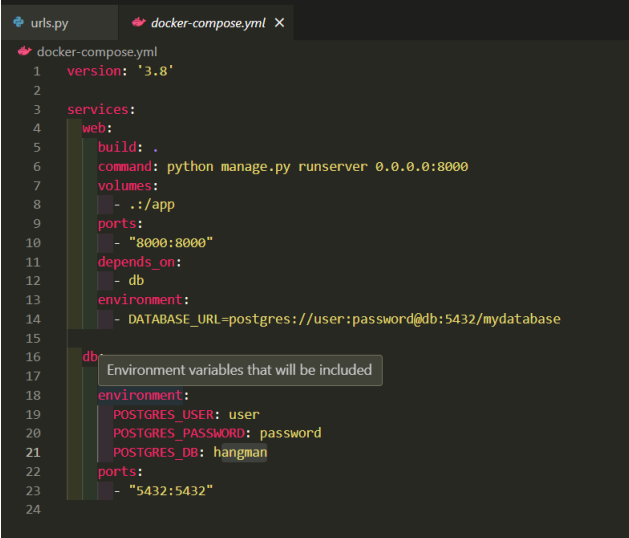
El proyecto está organizado de la siguiente manera:

juego_hangman/
 hangman/
juego_hangman/
 static/
 templates/
docker-compose.yml
Dockerfile
manage.py
requirements.txt

templates/: Contiene los archivos HTML para la interfaz.

static/: Incluye las imágenes de los avatares y otros recursos estáticos.

Docker-compose.yml



```
1  version: '3.8'
2
3  services:
4    web:
5      build: .
6      command: python manage.py runserver 0.0.0.0:8000
7      volumes:
8        - ./app
9      ports:
10       - "8000:8000"
11      depends_on:
12        - db
13      environment:
14        - DATABASE_URL=postgres://user:password@db:5432/mydatabase
15
16    db:
17      Environment variables that will be included
18      environment:
19        POSTGRES_USER: user
20        POSTGRES_PASSWORD: password
21        POSTGRES_DB: hangman
22      ports:
23        - "5432:5432"
24
```

Index.html

Es el cuerpo del programa es la pantalla principal

```
urlspy index.html X
templates > index.html > ...
1 <html lang="en">
2 <body>
304 <div class="button-container">
305 <button id="rankings">Rankings</button>
306 <button id="info">Información</button>
307 </div>
308
309 <!-- Error Modal HTML -->
310 <div id="errorModal" class="modal">
311 <div class="modal-content" non-red>
312 <span class="close" id="closeErrorModal">×</span>
313 <p>Por favor, ingrese su nombre para jugar.</p>
314 </div>
315 </div>
316
317 <!-- Modal HTML -->
318 <div id="helpModal" class="modal">
319 <div class="modal-content">
320 <p>¿Cómo jugar?</p>
321 <p>Como jugar?</p>
322 <p>El objetivo del juego es adivinar la palabra secreta letra por letra. Tienes un número limitado de intentos.</p>
323 </div>
324 </div>
325
326 <!-- Rankings Modal HTML -->
327 <div id="rankingsModal" class="modal">
328 <div class="modal-content">
329 <span class="close" id="closeRankingsModal">×</span>
330 <p>Rankings</p>
331 <table id="rankingsTable" style="width: 100%; color: #fff; border-collapse: collapse;">
332 <thead>
333 <tr>
334 <th>Puntuación</th>
335 <th>Palabra</th>
336 </tr>
337 </thead>
338 <tbody>
339 <tr>
340 <td>100</td>
341 <td>Girafa</td>
342 </tr>
343 <tr>
344 <td>90</td>
345 <td>Zebra</td>
346 </tr>
347 <tr>
348 <td>80</td>
349 <td>Elefante</td>
350 </tr>
351 <tr>
352 <td>70</td>
353 <td>Tigre</td>
354 </tr>
355 <tr>
356 <td>60</td>
357 <td>Fútbol</td>
358 </tr>
359 <tr>
360 <td>50</td>
361 <td>Tennis</td>
362 </tr>
363 <tr>
364 <td>40</td>
365 <td>Italia</td>
366 </tr>
367 <tr>
368 <td>30</td>
369 <td>Brasil</td>
370 </tr>
371 <tr>
372 <td>20</td>
373 <td>Doctor</td>
374 </tr>
375 <tr>
376 <td>10</td>
377 <td>Ingeniero</td>
378 </tr>
379 <tr>
380 <td>0</td>
381 <td>Chef</td>
382 </tr>
383 </tbody>
384 </table>
385 </div>
386 </div>
```

Play.html

Se diseña la interfaz del juego

```
urlspy play.html X
templates > hangman > play.html > ...
1 <html lang="es">
2 <head>
3 <meta>
300 </head>
301 <body>
302 <h1>AHORCADO</h1>
303 <p id="welcome-message"></p>
304 <p>Encuentra la palabra oculta - Ingrese una letra</p>
305 <div class="game-container">
306 <svg height="250" width="200" class="figure-container">
307 <!-- Cabeza -->
308 <circle cx="140" cy="70" r="20" class="figure-part">
309 <!-- Cuerpo -->
310 <div class="figure-part">
311 <div class="figure-part">
312 <div class="figure-part">
313 <div class="figure-part">
314 <div class="figure-part">
315 <div class="figure-part">
316 <div class="figure-part">
317 <div class="figure-part">
318 <div class="figure-part">
319 <div class="figure-part">
320 <div class="figure-part">
321 </div>
322 </div>
323 <div class="wrong-letters-container">
324 <div id="wrong-letters"></div>
325 </div>
326 <div class="word" id="word"></div>
327 </div>
328
329 <!-- Contenedor del final -->
330 <div class="popup-container" id="popup-container">
331 <div class="popup">
332 <div id="final-message"></div>
333 <button id="play-button">Juega otra vez</button>
334 <button id="index-button">Regresar al Inicio</button>
335 </div>
336 </div>
```

Script.js

Es donde esta almacenados las interacciones de la pagina web

```
const wordEl = document.getElementById('word');
const wrongLettersEl = document.getElementById('wrong-letters');
const playAgainBtn = document.getElementById('play-button2');
const popup = document.getElementById('popup-container');
const notification = document.getElementById('notification-container');
const finalMessage = document.getElementById('final-message');
const figureParts = document.querySelectorAll('.figure-part');

const selectionPopup = document.getElementById('selection-popup');
const selectionButtons = document.querySelectorAll('.selection-button');
const randomCategoryBtn = document.getElementById('random-category');
const indexButton = document.getElementById('index-button');
const timerElement = document.getElementById('timer');

let selectedWord = '';
const correctLetters = [];
const wrongLetters = [];
let playable = true;
let timerInterval;
let timeRemaining = 60; // Tiempo en segundos


// words by category
const categories = {
  animal: ['tiger', 'elephant', 'giraffe', 'zebra'],
  deporte: ['soccer', 'basketball', 'tennis', 'cricket'],
  pais: ['france', 'spain', 'italy', 'brazil'],
  profesion: ['doctor', 'engineer', 'artist', 'chef']
};

// Randomly selects a word from a given category
function getRandomWord(category) {
  const words = categories[category];
  return words[Math.floor(Math.random() * words.length)];
}

// Display hidden word
```

Style.css

Es donde esta almacenado los estilos de la pagina web.

```
he > # style.css > 
1 *
2 {
3   box-sizing: border-box;
4 }
5
6 body {
7   background-color: #000000;
8   color: #fff;
9   font-family: Arial, Helvetica, sans-serif;
10  display: flex;
11  flex-direction: column;
12  align-items: center;
13  justify-content: center;
14  height: 100vh;
15  margin: 0;
16  overflow: hidden;
17 }
18
19 h1 {
20   font-size: 3em;
21   margin-bottom: 50px;
22   color: #fff;
23   text-shadow: 0 0 5px #000, 0 0 10px #000, 0 0 20px #000;
24 }
25
26 .game-container {
27   padding: 20px 30px;
28   position: relative;
29   margin: auto;
30   height: 350px;
31   width: 450px;
32 }
33
34 .figure-container {
35   fill: transparent;
36   stroke: #fff;
37   stroke-width: 4px;
```

- **Conclusiones Técnicas:**

Durante el desarrollo del juego de ahorcado, surgieron varios desafíos técnicos que requerían soluciones innovadoras y prácticas. A continuación, se describen los principales desafíos y cómo se abordaron:

Sincronización del Temporizador con la Lógica del Juego:

Uno de los desafíos más significativos fue la implementación del temporizador en la página del juego. El temporizador debía sincronizarse perfectamente con la lógica del juego, de modo que cuando llegara a cero, el juego se detuviera y el usuario perdiera automáticamente. Para resolver este problema, se utilizó JavaScript para gestionar el temporizador en el frontend, mientras que se enviaban solicitudes periódicas al servidor para mantener el estado del juego actualizado.

Manejo de Sesiones y Persistencia de Datos:

Dado que los usuarios pueden salir y volver al juego, era crucial que el estado del juego se mantuviera consistente a lo largo de las sesiones. Para lograr esto, se aprovechó la funcionalidad de sesiones en Django, permitiendo almacenar la información del usuario, como su nombre, avatar, y progreso en el juego, directamente en el servidor.

Optimización de Rendimiento:

La eficiencia y velocidad de carga del juego fueron aspectos clave para garantizar una buena experiencia de usuario. Al utilizar Docker, fue posible aislar la aplicación en contenedores, lo que permitió un control preciso sobre el entorno de ejecución y minimizó los conflictos de dependencias.