



# HAIA - Wordle con ChatGPT y Stable Diffusion

Por Daniel de Castro Isasi y Toni Blasco Calafat



# **Parte 1: Desarrollo de la aplicación web**

# Introducción

## Wordle

C	R	A	V	E
B	L	A	C	K
A	L	I	B	I
A	L	B	U	M

Variaciones →

- Frase en la que tendremos que acertar las palabras
- La frase vendrá dada por ChatGPT
- Como pista, se mostrarán 3 imágenes generadas por Stable Diffusion

# Preparación del framework

---

- Utilizaremos JavaScript y NodeJS para realizar una aplicación web
- Necesitaremos los siguientes paquetes →
  - Express, body-parser y cors → Nos permiten establecer la estructura básica de la web, acceder a la información que necesitamos y formatearla para que sea legible.
  - Is-word → Comprobación de que las palabras escritas existan
  - Node-cmd → Permite ejecutar comandos desde Node



# Preparación básica

```
const path = require('path');
const express = require('express');
const app = express();
const isWord = require('is-word');
const cors = require("cors");
const bodyParser = require('body-parser');
const port = 5000;
const spaWords = isWord("spanish");

app.use(cors())
app.use(bodyParser.json())

app.use(express.static(__dirname + "/static"));

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'static/index.html'));
});

app.listen(port, () => {
  console.log(`Now listening on port ${port}`);
});
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Wordle</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div id="game">
    <div id="theme" class="theme">
    </div>
    <div id="images" class="images">
    </div>
  </div>
  <script src="wordle.js" type="module"></script>
</body>
</html>
```

# Desarrollo del wordle

- Generaremos una variable para saber el estado de la app en todo momento. Necesitaremos los atributos:
  - Secret → Frase a adivinar
  - Grid → Tamaño de la cuadrícula del wordle
  - imagesLoaded → Indica cuando han sido generadas las imágenes
  - currentRow, currentCol, currentWord → Posición del cursor y palabra actual
  - won, lost → Indican si se ha acabado la partida

```
const state = {  
  secret: sentence,  
  grid: Array(6).fill().map(() =>  
    | Array(sentence.length).fill(''),  
  imagesLoaded: false,  
  currentRow: 0,  
  currentCol: 0,  
  currentWord: '',  
  won: false,  
  lost: false,  
}; <- #6-15 const state =
```

# Desarrollo del wordle



- Pasos a seguir →
  - Generación de la frase y las imágenes correspondientes → Proceso explicado más adelante
  - Dibujamos la cuadrícula en la web
  - Registramos las teclas pulsadas por el usuario y actuamos en consecuencia
- Una vez generadas las imágenes → drawGrid()

```
function drawGrid(){
  game.removeChild(document.getElementById('theme'));
  const grid = document.createElement('div');
  grid.className = 'grid';
  grid.style = `grid-template-columns: repeat(${state.secret.length}, auto);`

  for(let i = 0; i < 6; i++){
    for(let j = 0; j < state.secret.length; j++){
      drawBox(grid, i, j);
    }
  } <- #74-78 for(let i = 0; i < 6; i++)

  game.appendChild(grid);
} <- #68-81 function drawGrid()
```

# Desarrollo del Wordle



- Registro de las teclas → `document.body.onkeydown`
- Dependiendo de la tecla pulsada:
  - “Espacio” → Se pasa a la siguiente columna y se resetea “currentWord”
  - “Borrar” → Borraremos la última letra de “currentWord”. Si está vacía se pasará a la anterior palabra
  - “Enter” → Entregamos la frase y comienza el proceso de validación
  - Otros → Se comprueba que es una letra. En caso de que lo sea la añadimos a “currentWord”
- Después de cada una ejecutamos `updateGrid()` → Actualiza la web con los datos almacenados en el estado



# Desarrollo del Wordle

¿Se ha acertado la frase?

- Comprobamos que todas las palabras existan → Paquete is-word

```
async function isSentenceValid(sentence){
  if(state.currentCol !== state.secret.length-1) return;
  const url = new URL(`http://localhost:${PORT}/check`)
  let response = await fetch(url, {
    method: "post",
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({sentence})
  }) <- #129-136 let response = await fetch
  let res = await response.json()

  if(!res.valid.includes(false)) return true;

  return res.valid;
} <- #126-142 async function isSentenceValid(sentence)
```

```
app.post('/check', (req, res) => {
  let valid = [];
  const { sentence } = req.body;
  sentence.split(' ').forEach(word => {
    valid.push(spaWords.check(word));
  })
  res.send({ valid: valid });
}); <- #59-67 app.post
```

# Desarrollo del Wordle

¿Se ha acertado la frase?

- Si alguna palabra no existe la marcamos



GATS	EN	LA	TERRAZA

# Desarrollo del Wordle

¿Se ha acertado la frase?

- Si todas existen pasamos a comprobar si forman parte de la frase a adivinar

```
function revealSentence(guess){
  const row = state.currentRow;
  const animation_duration = 500;
  for(let i = 0; i < state.secret.length; i++){
    const box = document.getElementById(`box${row}${i}`)
    const word = box.textContent;

    setTimeout(() => {
      if(word.toUpperCase() === state.secret[i].toUpperCase()){
        box.classList.add('right');
      } else if(state.secret.map(function(x){return x.toUpperCase();}).includes(word.toUpperCase())){
        box.classList.add('wrong');
      } else{
        box.classList.add('empty');
      }
    }, (i * animation_duration) / 2); <- #152-160 setTimeout

    box.classList.add('animated');
    box.style.animationDelay = `${(i * animation_duration) / 2}ms`;
  } <- #148-164 for(let i = 0; i < state.secret.length; i++)
```


# Desarrollo del Wordle

¿Se ha acertado la frase?

- Si la palabra es correcta la marcamos en verde
- Si está en una posición incorrecta la marcamos en amarillo



GATO	LA	EN	TERRAZA



GATO	LA	EN	TERRAZA
GATO	EN	LA	VENTANA

¡Felicidades, has ganado!

Reiniciar



# Parte 2: Uso de IA



# Interacción con ChatGPT

- Uso de la librería openai
- Escoger una key correspondiente a nuestra cuenta, para usar los diferentes métodos
- Llamada a **ChatCompletion.create**
  - Se le pasa modelo, el rol (usuario) y el prompt.
  - Devuelve la respuesta a través de un json



## Interacción con Stable Diffusion

- Eliminación de acentos y otros caracteres para facilitar la vida al usuario.
- Descarga del modelo en local.
- Llamada a *from\_pretrained* para escoger el modelo.
- Uso de *torch\_dtype = auto* para que se adapte al ordenador correspondiente.
- Transferir el modelo a la GPU o a la CPU en caso de que no esté disponible
- Uso del método *pipe* para la creación de imagen, se le adjunta tanto el prompt como número de inferencias, en nuestro caso 60.
- Devuelve la imagen generada, con PIL se hace un *resize* a una imagen 200x200 y se guarda en local.



## Interacción con Stable Diffusion (II)

- Uso de la librería *Replicate* para la llamada de Stable Diffusion (tiene modelos de IA en la nube)
- Se hace uso de un token de acceso
- Se aplica este token haciendo uso de la librería *os*, convirtiéndola en una variable de entorno con *os.environment*.
- Se obtiene el modelo a escoger mediante *replicate.model.get*, en nuestro caso la versión de Stable Difussion.
- Luego a partir de ese modelo se escoge la versión que se prefiera con *versions.get*. En nuestro caso la 2.1.
- Con el uso de *version.predict* se puede hacer uso ya del modelo escogido para la generación de imágenes.
- Se devuelve la imagen como una URL. Todas las URL se añaden a una Lista que después se trata en el Post del *index.js*.





## Interacción con Dall-e

- Uso de la misma librería que con ChatGPT, `openai`, para la implementación
- Se llama al método `Image.create` y se pasa como parámetro el *prompt*, el número de imágenes y el tamaño al que se quiere, se devuelve la imagen como una URL. Todas las URL se añaden a una Lista que después se trata en el Post del `index.js`.



## Inclusión de un script python en Node js (I)

- Método *spawn* de nodejs, este método genera un subproceso del cual es posible llamar al script de python.
- `stdout.on`, para leer el buffer con todos los datos de salida del fichero python
- Es importante añadir la librería `sys` y `sys.stdout.flush()` al final del script para forzar a los datos salir del buffer.



## Inclusión de un script python en Node js (II)

- En el caso de Dall-e y Stable Diffusion se hace uso de *cmd.run* de la librería *node-cmd*
- Permite ejecutar en un shell las diferentes sentencias, en nuestro caso el script python. (devuelve URL)
- Se trata esto en el push del index.js donde se escogen la URL y se muestran posteriormente en pantalla.



# Conclusiones

- El papel de la IA puede ser muy variado y que aunque se ha utilizado su potencial para la realización de un juego,
- Ofrecen gran creatividad
- La creación del HTML y el script ayuda a aplicar diferentes tecnologías que como profesionales de la informática ayuda a desenvolverse y romper mano viendo una aplicación real de una inteligencia.