*PROYECTO***:**

**Wordlabras**

|  |
| --- |
|  |

*Realizado por:* Adrián López Pastor

|  |  |
| --- | --- |
| http://alba9.es/imagenes/logo_galileo_1.jpg | **Desarrollo de Aplicaciones Web (DAW)** |
| **Valladolid, a 20 de** **Diciembre de 2022** |
| Nota: |  |

Índice

[1 Introducción 7](#_Toc122466540)

[1.1 Objeto de proyecto 7](#_Toc122466541)

[1.1.1 Breve Introducción 7](#_Toc122466542)

[1.1.2 Justificación 7](#_Toc122466543)

[1.1.3 Sentido del proyecto 7](#_Toc122466544)

[1.2 Lenguajes empleados 8](#_Toc122466545)

[1.2.1 Lenguajes 8](#_Toc122466546)

[1.2.2 IDEs 9](#_Toc122466547)

[1.3 Distribución 9](#_Toc122466548)

[1.3.1 Despliegue 9](#_Toc122466549)

[1.4 Requisitos de los clientes: 10](#_Toc122466550)

[1.5 Licenciamiento 10](#_Toc122466551)

[2 Recursos 11](#_Toc122466552)

[2.1 Hardware 11](#_Toc122466553)

[2.1.1 Para su desarrollo, Implementación y distribución 11](#_Toc122466554)

[2.2 Software 11](#_Toc122466555)

[2.2.1 Para su desarrollo, implementación y distribución 11](#_Toc122466556)

[2.3 Humanos 11](#_Toc122466557)

[2.4 Previsión económica del coste del proyecto 12](#_Toc122466558)

[2.4.1 Personal (pago por mano de obra) 12](#_Toc122466559)

[2.4.2 Coste de mantenimiento 12](#_Toc122466560)

[3 Descripción de la aplicación 13](#_Toc122466561)

[3.1 Funcionamiento general 13](#_Toc122466562)

[3.1.1 Diagrama de casos de uso 13](#_Toc122466563)

[3.1.2 Diagrama de secuencia 14](#_Toc122466564)

[3.2 Arquitectura 15](#_Toc122466565)

[3.2.1 Diseño de las bases de datos 15](#_Toc122466566)

[3.2.2 Arquitectura del sistema 17](#_Toc122466567)

[3.3 Interfaz 19](#_Toc122466568)

[3.3.1 Características generales 19](#_Toc122466569)

[3.3.2 Adaptación a dispositivos móviles 25](#_Toc122466570)

[3.3.3 Usabilidad/accesibilidad 32](#_Toc122466571)

[4 Autoevaluación y conclusiones 32](#_Toc122466572)

[4.1 Valoración del trabajo y dificultades encontradas 32](#_Toc122466573)

[4.2 Valoración de la herramienta o aplicación desarrollada 34](#_Toc122466574)

[4.3 Conclusiones finales 34](#_Toc122466575)

[4.3.1 Del diseño 34](#_Toc122466576)

[4.3.2 De la aplicación 35](#_Toc122466577)

[4.3.3 Del tiempo empleado 35](#_Toc122466578)

[4.3.4 De los conocimientos adquiridos 35](#_Toc122466579)

[5 Bibliografía 35](#_Toc122466580)

[6 HTML 37](#_Toc122466581)

[6.1 Index.html 37](#_Toc122466582)

[6.2 Wordlabras.html 38](#_Toc122466583)

[6.3 Registro.html 39](#_Toc122466584)

[6.4 Ayuda.html 40](#_Toc122466585)

[6.5 Stats.html / rank.html 41](#_Toc122466586)

[7 CSS 42](#_Toc122466587)

[7.1 Login.css 42](#_Toc122466588)

[7.2 Main.css 49](#_Toc122466589)

[7.3 Ayuda.css 6](#_Toc122466590)

[8 JavaScript 7](#_Toc122466591)

[8.1 Main.js 7](#_Toc122466592)

[8.2 Mysql.js (WIP) 14](#_Toc122466593)

[8.3 App.js (WIP) 16](#_Toc122466594)

[9 Manual de Juego 17](#_Toc122466595)

[9.1 Los pasos que seguiremos para poder usar la aplicación serán los siguientes: 19](#_Toc122466596)

[9.1.1 Registrarse 19](#_Toc122466597)

[9.2 Login 19](#_Toc122466598)

[9.3 Jugar 20](#_Toc122466599)

[9.4 Ganar 22](#_Toc122466600)

[9.5 Perder 22](#_Toc122466601)

[10 Pruebas del sistema 24](#_Toc122466602)

[10.1 Pruebas a realizar 24](#_Toc122466603)

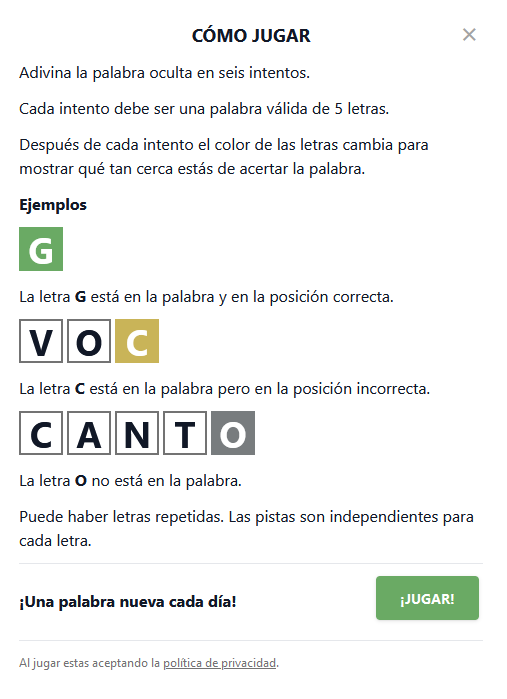
[10.1.1 Pruebas del Registro 24](#_Toc122466604)

[10.1.2 Pruebas del juego 2](#_Toc122466605)

# Introducción

## Objeto de proyecto

### Breve Introducción

El proyecto consiste en realizar una página web como el juego llamado “Wordle” que se ha hecho viral desde hace poco tiempo (en el tiempo en el que presento esta solicitud) de la página [https://wordle.danielfrg.com/](https://wordle.danielfrg.com/%20%20)  Dicho juego, se basa en tener una palabra de 5 letras diaria (cambia cada 24h) para todas las personas, esas personas tienen que adivinar esa palabra en 6 intentos. Está como aplicaciones y página web con muchas variantes (por ejemplo: Creación de retos, pones la palabra que quieran adivinar, el número de intentos y crea automáticamente la plantilla en el enlace que tienes que copiar y enviar a las personas que quieran. También existe en diferentes idiomas, de diferentes temáticas, por ejemplo, de Pokémon, etc…)

### Justificación

Realizo este proyecto por lo viral que se hizo/se ha hecho y lo divertido que es (al final es como un juego y es educativo). La innovación en este proyecto será contar con una base de datos en la que guardar/almacenar, gestionar y obtener los usuarios y con ello sus datos y estadísticas. Además, gracias a contar con una base de datos con los usuarios y sus resultados, haré un ranking de los mejores usuarios y su posición global respecto al resto de usuarios.

### Sentido del proyecto

Como he mencionado, ya existe una página web de éxito en el mercado que es en la que me he basado, aunque a partir de ahí hayan aparecido muchas más y lo realizo porque me parece divertido y entretenido de hacer.

## Lenguajes empleados

### Lenguajes

* HTML
  + HTML (Lenguaje de Marcas de Hipertexto, del inglés HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) o la funcionalidad/comportamiento (JavaScript).[[1]](#footnote-1)
* CSS
  + Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML (en-US) (incluyendo varios languages basados en XML como SVG, MathML o XHTML). CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios.[[2]](#footnote-2)
* JAVASCRIPT
  + JavaScript (JS) es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo, programación funcional). Lee más en acerca de JavaScript.[[3]](#footnote-3)
  + Librerías:
    - Node.js
    - Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google[[[4]](#footnote-4)]
    - Axios (utilizado para la conexión con la BBDD).
    - Axios es un Cliente HTTP basado en promesas para node.js y el navegador. Es isomorfico (= puede ejecutarse en el navegador y nodejs con el mismo código base). En el lado del servidor usa el modulo nativo http de node.js, mientras que en el lado del cliente (navegador) usa XMLHttpRequests.[[5]](#footnote-5)
* SQL
  + SQL es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.[[6]](#footnote-6)
  + Utilizando MySQL
    - MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos[[7]](#footnote-7)

### IDEs

* VisualStudio Code
* Phpmyadmin
  + Xampp

## Distribución

### Despliegue

Se necesitará una página de hosteo para poder jugar o en su defecto contar con todos los archivos necesarios a nivel local (Xampp + BBDD + Extensión de liveserver de VisualStudioCode).

## Requisitos de los clientes:

* Utilizar cualquiera de los siguientes navegadores y estas versiones o superiores (probablemente con versiones inferiores también se visualice correctamente hasta una versión no muy antigua)
  + **Firefox (Navegador preferente):** 107.0.1 (64 bit)
  + **Opera:** 92.0.4561.34
  + **Microsoft Edge:** Versión 108.0.1462.42 (Compilación oficial) (64 bits)
  + **Internet Explorer:** 
* Crearse un usuario a través de la página web
* Tener los requisitos mínimos de hardware para el posible acceso a la aplicación a través de la web.

## Licenciamiento

La licencia del proyecto será: “Attribution 4.0 International (CC BY 4.0)” que consiste en:

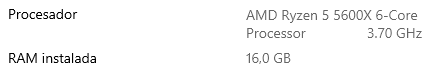
* Compartir - copiar y redistribuir el material en cualquier medio o formato
* Adaptar - remezclar, transformar y construir sobre el material para cualquier propósito, incluso comercial.

# Recursos

## Hardware

### Para su desarrollo, Implementación y distribución

Prestaciones del ordenador utilizado para el desarrollo:



1 TB de Espacio en el disco duro

## Software

### Para su desarrollo, implementación y distribución

* VisualStudio Code [IDE] [[8]](#footnote-8)
* Node.js
* Xampp [[9]](#footnote-9)

## Humanos

El número de horas totales empleadas para el proyecto es de:

* 64h-102h

De las cuales están divididas de la siguiente manera:

|  |  |
| --- | --- |
| **Informarse / Investigación** | 4h |
| **Memoria descriptiva** | 24h-32h |
| **Manual de código** | 8h-14h |
| **Programar / codificar** | 24h-40h |
| **Extras / Otros documentos** | 4h-12h |
| **Total:** | 64h-102h |

## Previsión económica del coste del proyecto

### Personal (pago por mano de obra)

El salario por hora de los empleados es de 8€/h por lo que si una persona realiza el proyecto sola como es en este caso, cobrará entre 512€ hasta 816€.

### Coste de mantenimiento

El hosteo de una página web será de unos 12€ anuales aproximadamente pudiendo ser el primer año gratuito en algunas páginas web. También se puede utilizar el hosting gratuito de herramientas y páginas como github o 000webhost.

En el caso de que el proyecto no necesite hosteo y sea utilizado de forma local, no tendrá coste alguno de mantenimiento.

# Descripción de la aplicación

## Funcionamiento general

La app es una página web para poder jugar a algo similar al Wordle pero contando con un ranking y una base de datos. Al entrar en la página web. Tienes la opción de loguearte, de recuperar la contraseña olvidada y de registrarte (ésta última opción no estará implementada en el código al igual que todo lo rojo en los diagramas).

El registro funcionaría a través de un email, usuario y contraseña, comprobaría los datos si están en la base de datos y dependiendo de si está o no, el registro es exitoso o pide datos nuevos.

La opción de contraseña olvidada estaría de una manera más segura incluyendo el usuario y el email del usuario para obtener la contraseña, pero actualmente sólo contará con la necesidad del nombre de usuario.

Al loggearse correctamente muestra la ventana del juego con opciones para ver el ranking o las estadísticas del usuario. Cuando escribe una palabra, al pulsar ENTER comprueba si existe esa palabra, si la palabra no existe, informa al usuario y debe cambiarla hasta que formule una palabra que sí que existe. Tiene 6 intentos para adivinar la palabra del día.

### Diagrama de casos de uso

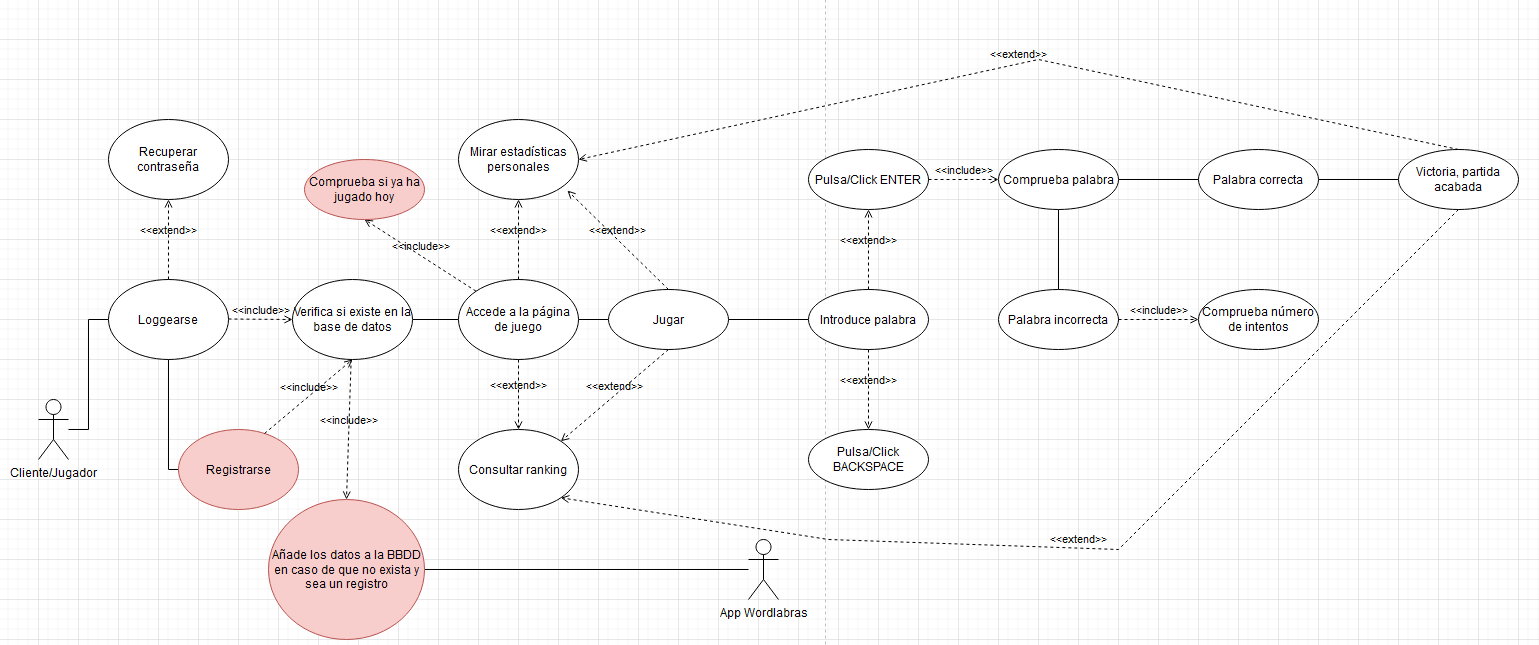


Diagrama de Casos de Uso

### Diagrama de secuencia

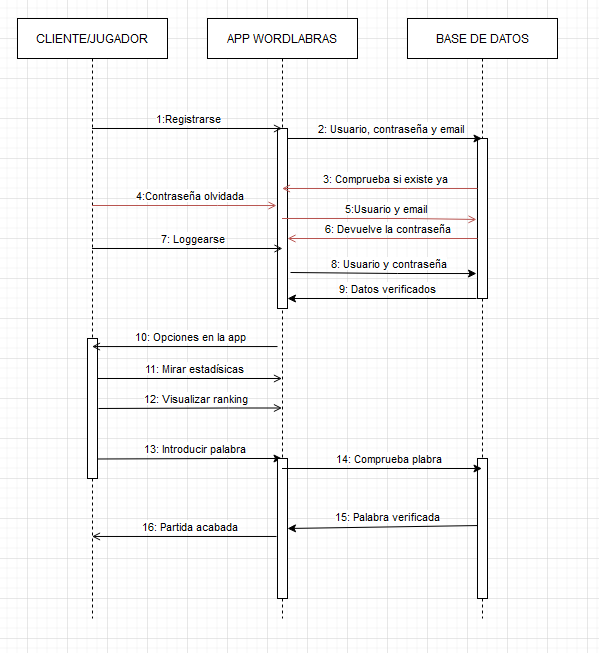


Diagrama de Secuencia

## Arquitectura

### Diseño de las bases de datos

#### Diagrama Relacional

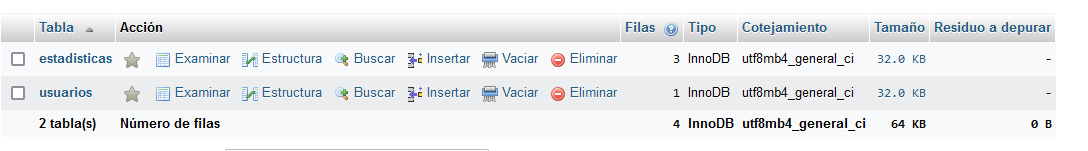
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Diagrama Relacional

#### Base de datos Wordlabras

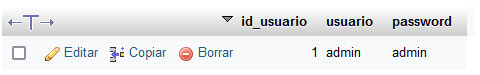
La base de datos se llama como el proyecto; “Wordlabras”, cuenta con una tabla de usuarios, otra de estadísticas.

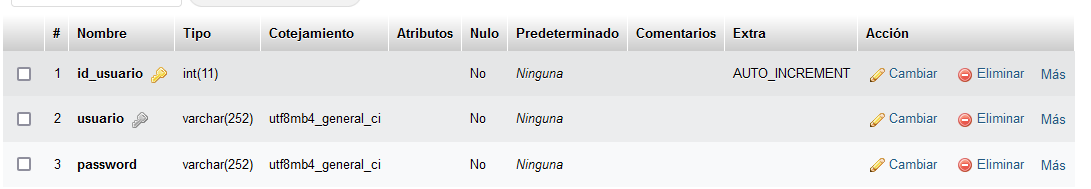


##### Tabla usuarios

La tabla usuarios cuenta con un id\_usuario (inservible, pero aporta información en caso de necesitar una mejora en el futuro), que no viene mal nunca tener un id único ajeno al usuario con el que poder trabajar de forma interna. Un usuario que tendrá un máximo de 252 caracteres y una contraseña que tendrá una cantidad máxima de 252 caracteres. El string de ‘usuario’ será único (si se implementase el registro, el campo único sería probablemente el email o ambos, email y usuario para mayor seguridad). La contraseña puede ser añadida a través del encriptado MD5 para mayor seguridad en la base de datos.

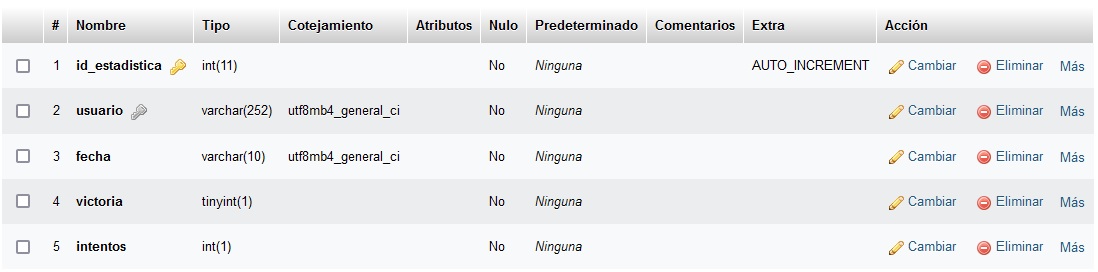
###### Campos de ejemplo





##### Tablas estadísticas

La tabla estadísticas cuenta con un id\_estadistica (inservible, pero aporta información en caso de necesitar una mejora en el futuro), que no viene mal nunca tener un id único ajeno al usuario con el que poder trabajar de forma interna. El usuario es una clave foránea cogida de la tabla usuarios gracias a que es un atributo único en la tabla usuarios. La fecha indicará el día que se jugó y los datos de ese día. La victoria es un tinyint que dirá si es true (1 a 255) o false (0) y dirá si obtuvo ese día la victoria o no. Los intentos es el número de intentos que tuvo ese día a la hora de ganar o perder (con un máximo de 6 intentos).



###### Campos de ejemplo



### Arquitectura del sistema

#### Diagrama de clases

En todo el código realizado sólo contaré con dos clases que serán útiles para el apartado de la base de datos, ya que creo que lo más cómodo para este caso será la utilización de las mismas.

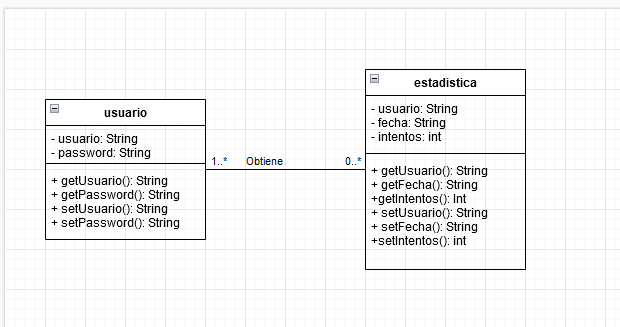


Diagrama de Clases

#### Diagrama de objetos

El diagrama de objetos es bastante similar al de clases al ser una relación sencilla.

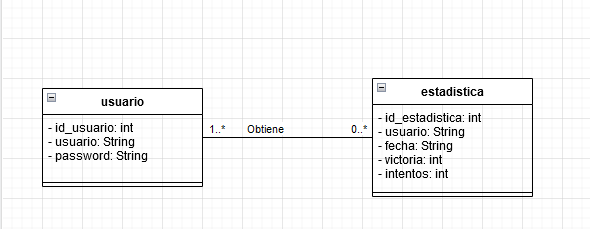
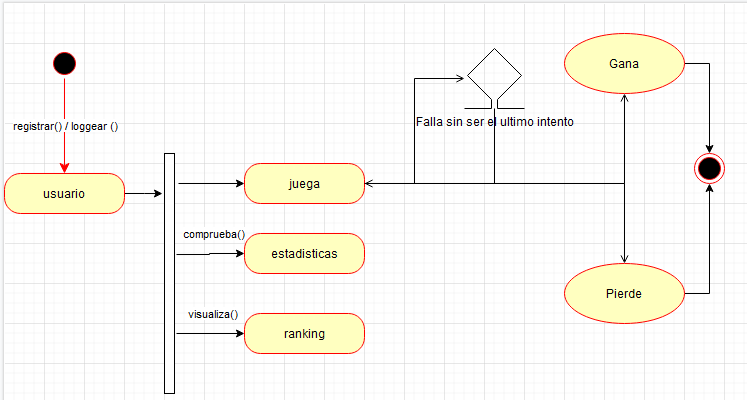


Diagrama de Objetos

#### Diagrama de de estados de las clases principales

Empieza con un usuario que se loggea aunque también se puede registrar (no implementado, por eso no lo he desarrollado más en el diagrama). Una vez loggeado, puede jugar, comprobar estadísticas personales y visualizar el ranking.

Cuando juega, tiene 6 intentos, puede fallar, si falla, vuelve a jugar (que sería escribir otra palabra), así hasta que gane porque la palabra haya sido acertada o porque se han acabado todos los intentos y pierda.



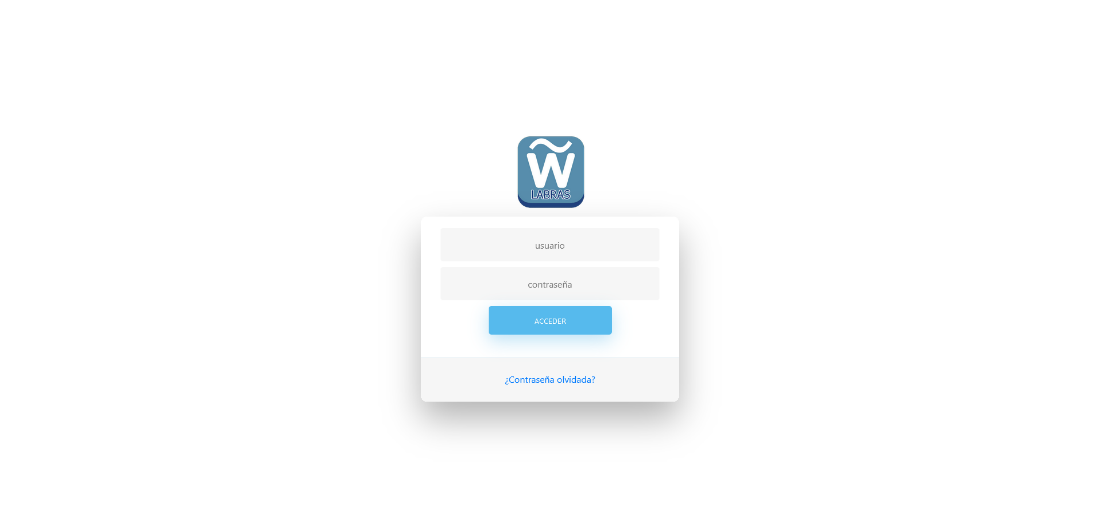
## Interfaz

### Características generales

#### Descripción de la interfaz

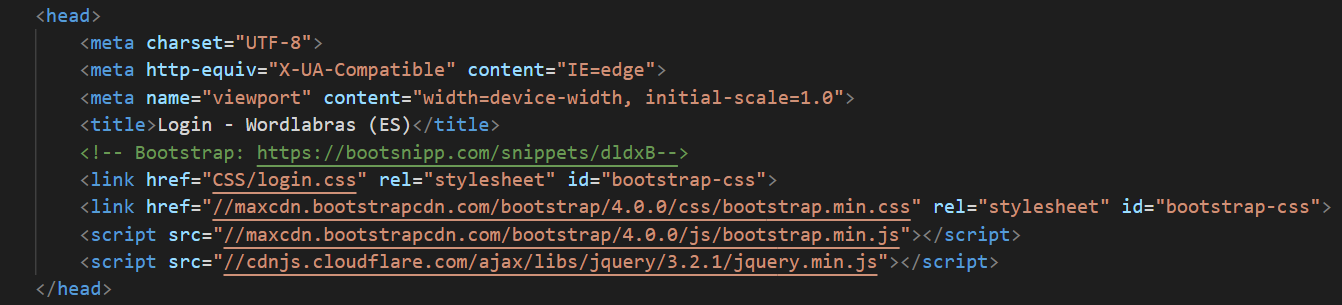
#### Características generales del diseño y la navegación

Pantalla del login creada con ayuda de bootstrap, cuenta con efectos ‘fade-in’ y ‘face-out’ realizados con CSS.



Pantalla Login

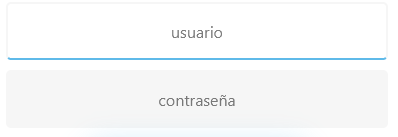
Importe del CSS y el boostrap/jquery utilizado para los efectos y visualización del login.



Imports

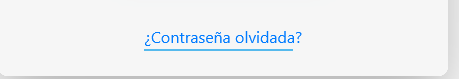
Cuenta con efectos de transiciones in/out para todo el contenido del login además de efectos “hover” para cuando el ratón sea posicionado sin clickar en los campos, véase en “acceder” que oscurecerá el color azul al tener el ratón encima del botón. Los inputs del formulario también contienen efectos hover y por click, haciendo que obtengan una línea de color azul y cambiar un poco el color de fondo y sombreados.

Ejemplo:

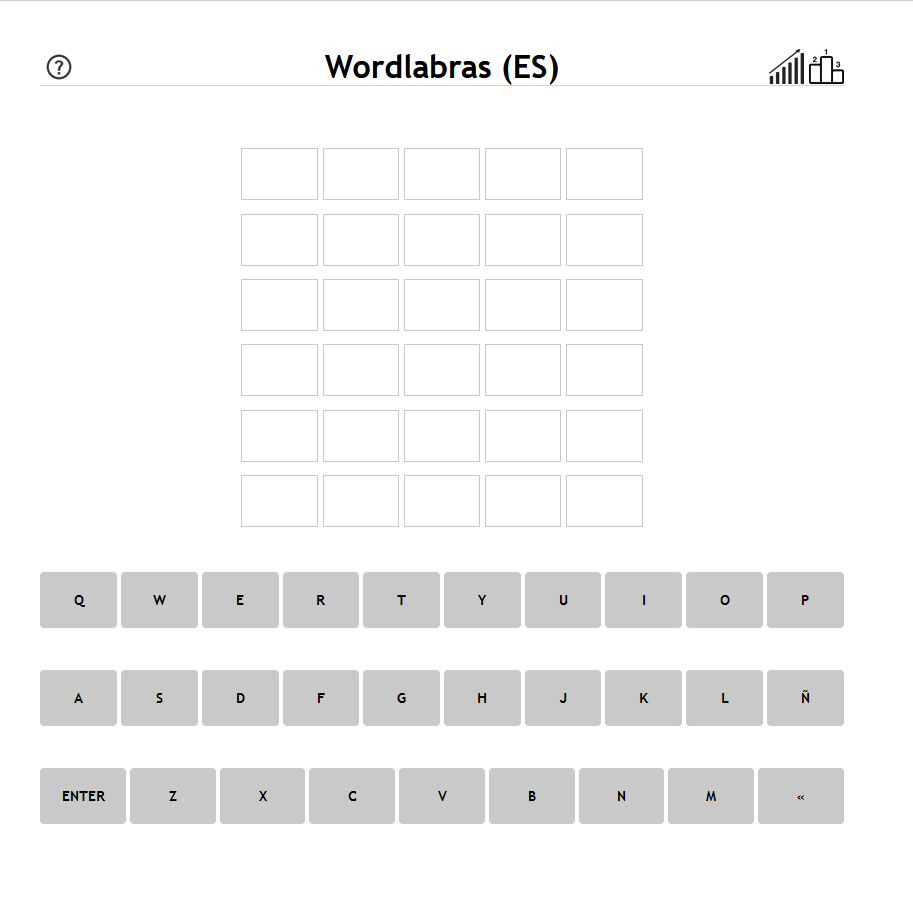


Efecto Hover inputs

Ocurre exactamente lo mismo con “contraseña olvidada”, cuenta con un subrayado azul con efecto (no aparece sin más, si no que se va generando de izquierda a derecha al tener el ratón posado en el botón y cambia el color a azul por ser un hipervínculo).

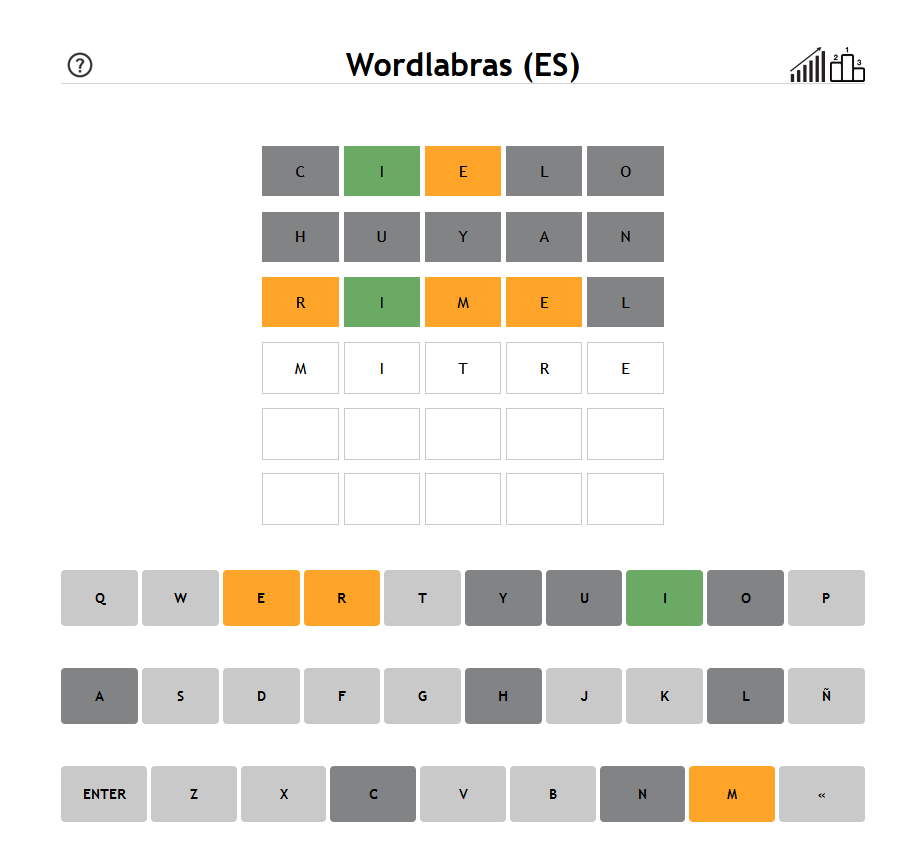


Efecto Hover fuera del form

En la interfaz del juego, podemos ver que tiene una visualización como la siguiente: 

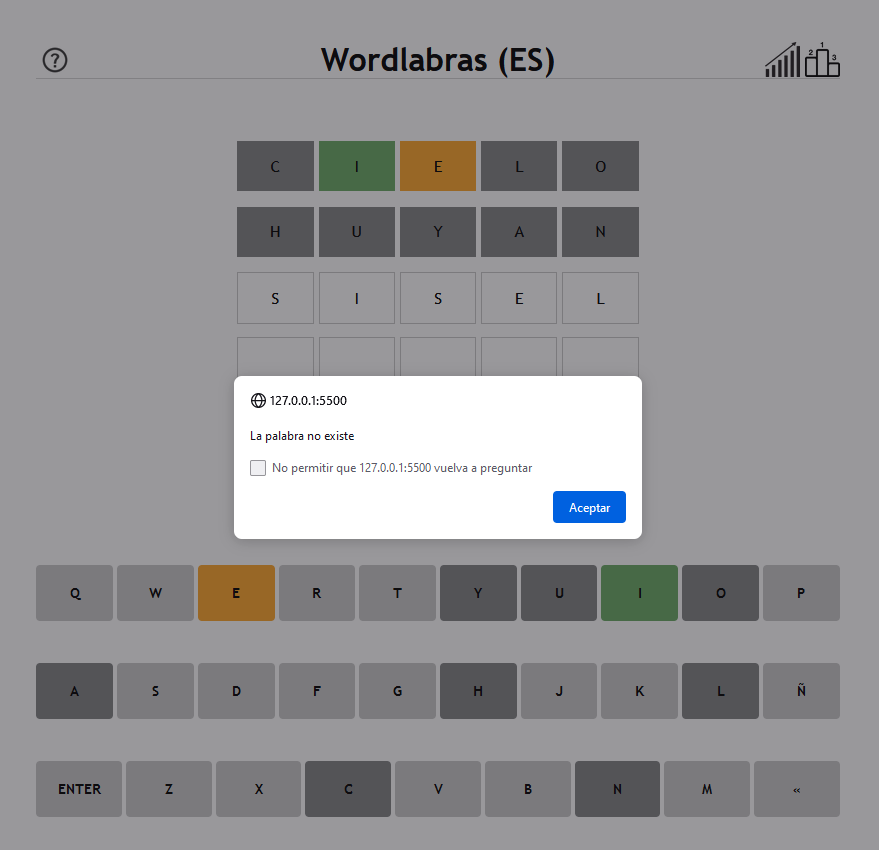
Interfaz Juego Wordlabras

La aplicación al introducir una palabra que existe, mostrará en blanco los intentos y dentro las letras escritas si aún no introdujo la palabra pulsando ENTER. Una vez pulse ENTER se pondrán en color gris oscuro los cuadrados de las letras que no existen en la palabra a adivinar, en naranja las letras que existen en la palabra a adivinar pero que no están en la posición correcta y de color verde cuando la letra introducida y su posición coinciden con la de la palabra a adivinar.



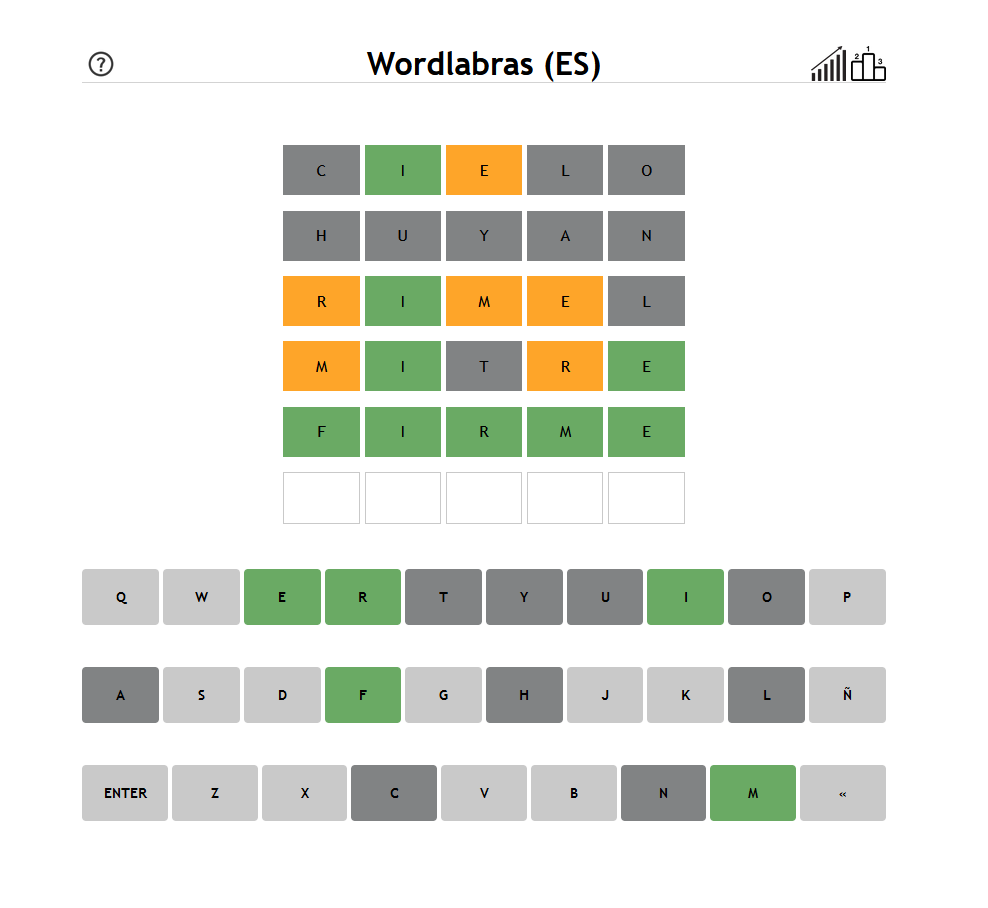
Interfaz Juego Wordlabras

Cuando escribe una palabra que no existe, la aplicación mostrará una alerta advirtiendo que la palabra escrita no existe.



Interfaz Juego Wordlabras

Una vez haya ganado, la palabra a adivinar se pondrá en verde y mostrará una alerta de que has ganado además de la opción de visualizar las estadístidas o el ranking.



Interfaz Juego Wordlabras

Al visualizar las estadísticas, nos saldrá una tabla con todas nuestras partidas, el número de victorias y el número de intentos gastados.

En la página del ranking se mostrará una tabla con todos los usuarios, el que más victorias tenga estará primero, es decir, el número de victorias es el factor determinante para el sistema de puntuaje, en caso de empate, se contarán el número de intentos, y en caso de empate por ambas cosas, será por orden alfabético.

#### Lenguajes empleados y validación

Los lenguajes empleados han sido:

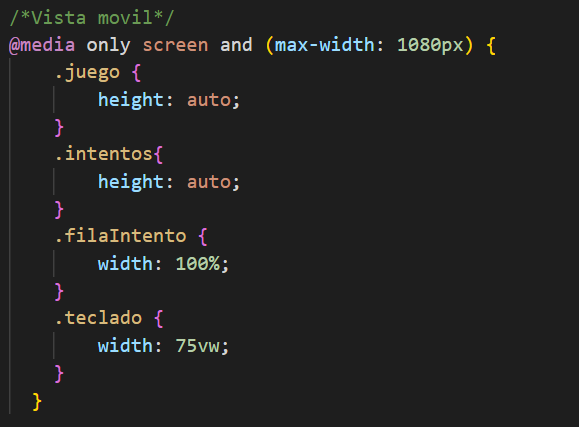
* HTML
  + Se ha utilizado para la visualización y creación de nodos base/iniciales para el juego. También es el apartado principal de las páginas web.
* CSS
  + Utilizado bastante en todo el proyecto para su correcta visualización y su estilo. Hace que todo se vea tan bien como se espera.
  + Utilizando Bootstrap en el login.
* Javascript
  + Se ha utilizado en todo el proyecto para cualquier tipo de interacción, desde dejar el rato encima de la barra para introducir el usuario, como la creación del teclado virtual así como de las cajas de los intentos de palabra así como recoger las teclas pulsadas correctas y el envío de datos a la base de datos.
  + Node.js
  + Axios.js
* SQL
  + Utilizado para la base de datos relacional y para la inyección de datos por medio de JavaScript gracias al uso de algunas librerías que posee (Node.js, Express.js, etc..).

Las validaciones que tendrán serán para los usuarios, palabras y letras introducidas en el juego, registros de usuario (no-realizado), y los datos metidos y sacados por pantalla de la base de datos (estadísticas y ranking).

### Adaptación a dispositivos móviles

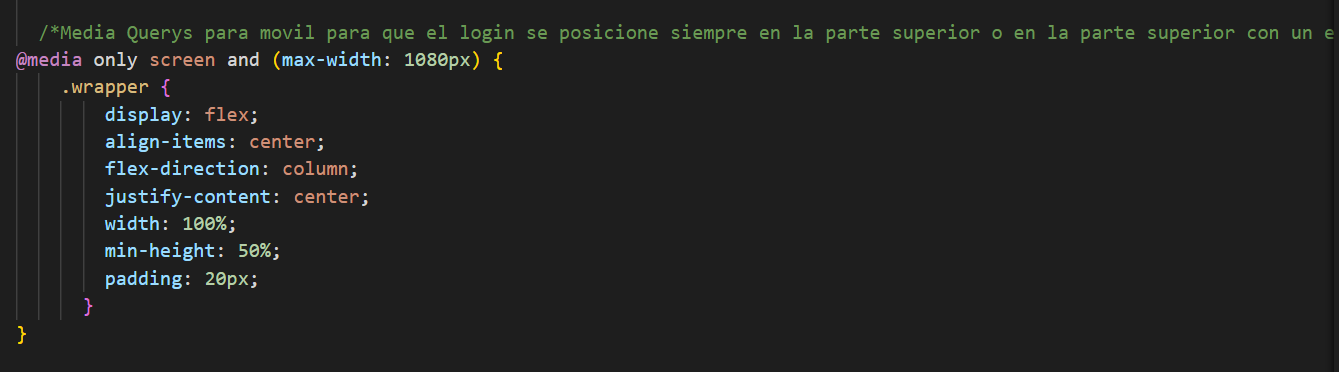
La adaptación se ha realizado a través del uso de “mediaquerys” en el código CSS.

#### Mediaquery para la interfaz del juego



Codificacion MediaQuery Juego

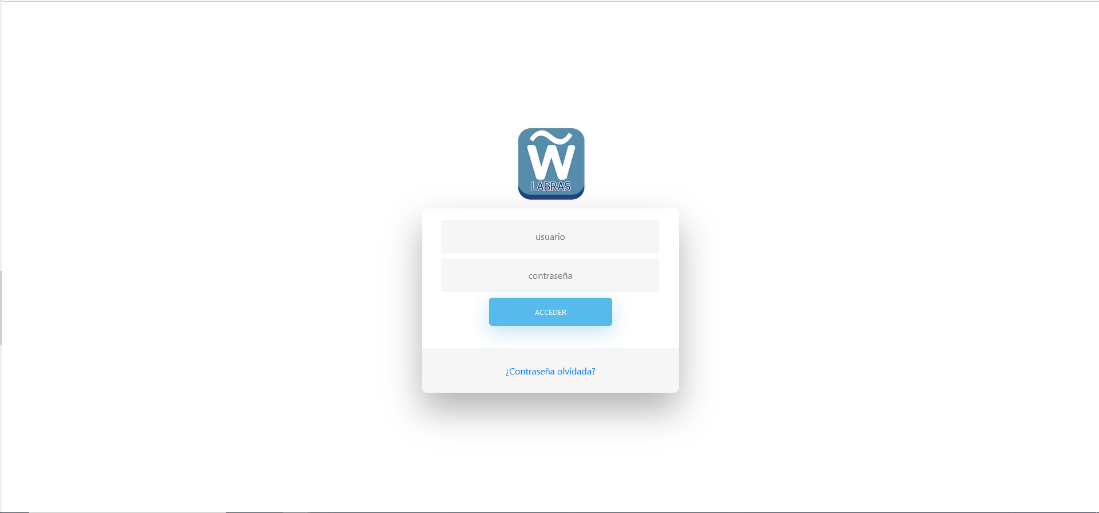
#### MediaQuery para la interfazl del Login



Codificación MediaQuery Login

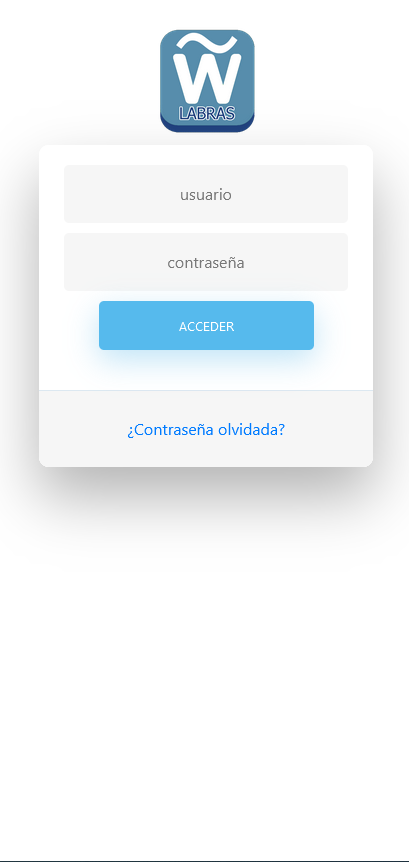
Las mediaquerys para las estadísticas y para el ranking serán más o menos similares, y se visualizarán similarmente a la interfaz del juego ya que será una página donde se visualice una tabla en ambos casos.

#### Vista ordenador del login



Vista Login PC

#### Vista móvil del login



Vista Login Móvil Login

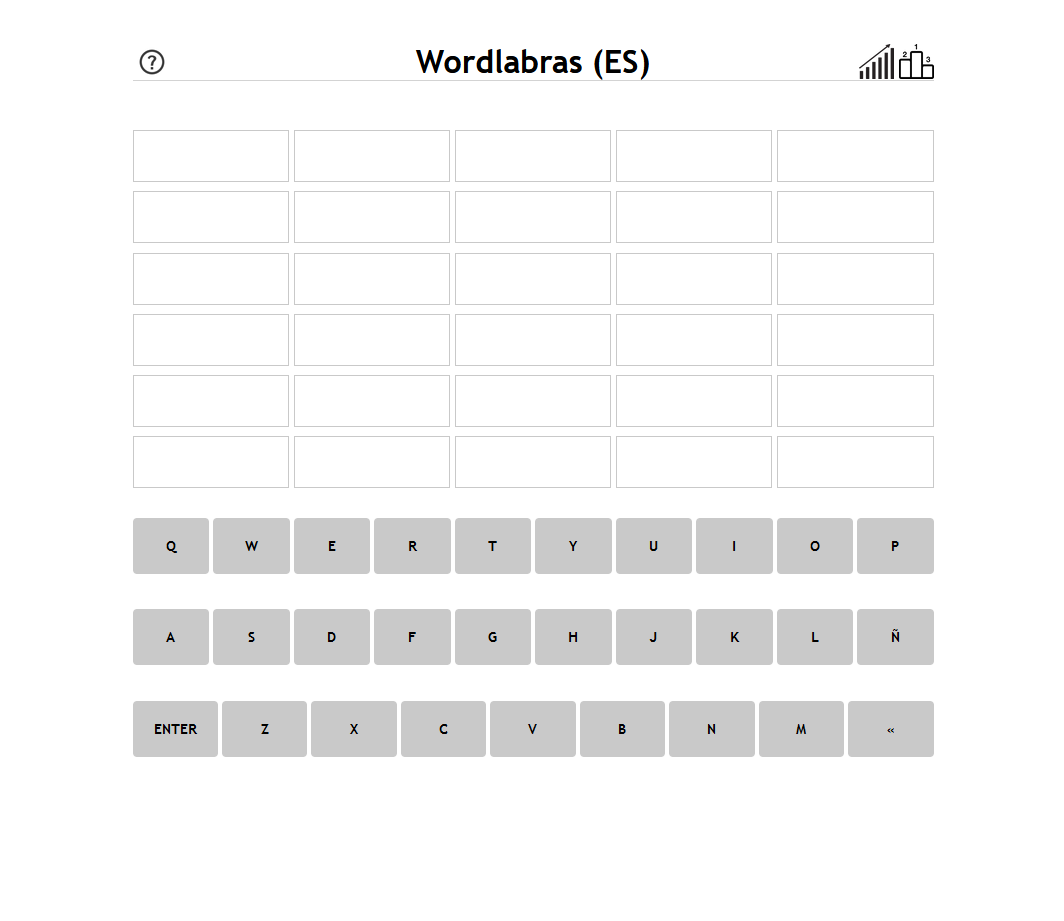
#### Vista ordenador del juego



Vista Ordenador Juego

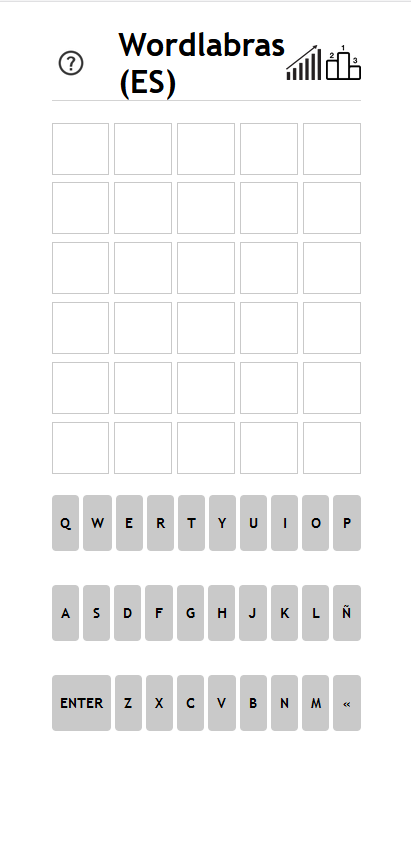
#### Vista móvil/tablet del juego en distintas resoluciones

##### Vista tablet del juego



Vista Tablet Juego

##### Vista móvil del juego



Vista Móvil Juego

#### Vista de las estadísticas (WIP)

Imagen que contiene Tabla

Descripción generada automáticamente

Página de Estadísticas

#### Vista ordenador de la ayuda

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Página de Ayuda

### Usabilidad/accesibilidad

#### Usabilidad

La usabilidad ha sido implementada para que el juego pueda jugarse tanto desde teclado como desde ratón. Puedes utilizar sólo el teclado para teclear las palabras y meter la palabra o borrar letras como el ratón para usar el teclado virtual. Para navegar por las ventanas es cierto que se deberá hacer uso exclusivo del ratón.

#### Accesibilidad

La accesibilidad será global ya que estará disponible para ser visualizado por cualquier dispositivo genérico de una forma correcta, ya sea un ordenador (nativo), como un móvil (adaptado) o una tablet (adaptado). No contará con soporte interno hacia otros tipos de dispositivo al menos nativos, pero eso no quiere decir que desde los propios distintos dispositivos no se pueda adaptar para la correcta accesibilidad y visualización, ya que estará colgado de forma púbica en una página web y en principio la accesibilidad será global.

# Autoevaluación y conclusiones

## Valoración del trabajo y dificultades encontradas

Mi trabajo actual lo valoro de tal manera que la mayor parte del trabajo está en el javascript y en el CSS así como en la BBDD, las conexiones y el enlace entre el juego y la base de datos por medio de javascript. Haciendo uso de Node.js y Axios ha sido posible. Ésta es la parte innovadora del proyecto.

La parte del login si bien es cierta que es “floja” debido a que ha sido realizada prácticamente con boostrap, creo que no era una parte importante del proyecto a visualizar y el tiempo escaseaba. No creo tampoco que sea una parte complicada la realización de un HTML/CSS para una página de login por lo que no veo que tenga mucho peso. He dedicado bastante tiempo las dos últimas semanas de entrega del proyecto a la realización del mismo para que pueda ser presentable. Creo que aún le quedan muchas cosas que me hubiese gustado implementar y muchas otras por mejorar, creo que este proyecto podría haber sido mucho más vistoso y bonito con más tiempo dedicado.

Por ejemplo; la implementación de que el usuario decida con cuántas letras tiene la palabra que quiere adivinar y que se auto genere el juego en base a esa opción que escoja. Tenía preparado el TXT y la API de la RAE para hacerlo de esa manera, pero me vi obligado a quitarla por la falta de tiempo y el tiempo que suponía adaptar esa idea con el código a medio realizar. Veía otras partes más importantes como es la base de datos y decidí centrarme mejor en eso.

También me hubiese gustado implementar la API que encontré para meter el diccionario de la RAE pero sólo vi opciones a través de Python y no tuve tiempo para pensar en cómo poder implementarlo desde Javascript u otra alternativa. Creo que pude hacer mucho más si hubiese dedicado más tiempo, pero simplemente no se dio porque estaba con las prácticas y a otras cosas personales de interés.

**Las dificultades encontradas han sido:**

* Por supuesto el tiempo invertido. Creo que no he invertido lo suficiente para lo que me gustaría y quizás incluso después de toda la entrega, mejore el proyecto e implemente poco a poco lo que se me ocurrió por practicar la codificación y programación.
* Entendimiento de la memoria, hay partes que ha sido difícil de hacer y entender, hay algún diagrama que no recuerdo haberlo visto en clase (diagrama secuencial y diagrama de objetos).
* Aprendizaje de node.js y axios (que utiliza node.js). No fue como tal dificultad, pero fue algo nuevo que aprender.
* La lógica ha supuesto algo de reto, sobretodo para que funcione correctamente un poco a base de ver posibles errores e ir corrigiéndolos.
* CSS vistoso, costó recordar un poco las ocpiones que tenía.

## Valoración de la herramienta o aplicación desarrollada

La herramienta en sí me parece muy buena y estable, sobretodo sin el uso de la base de datos. Tiene varias mejoras por implementar para su mejor diseño y funcionalidad con la base de datos, como pueda ser la seguridad y la estructura. El código me parece limpio, aunque se ha realizado casi desde 0 sin hacer uso de frameworks que hubiesen ahorrado tiempo. En sí la idea es muy divertida tanto de realizar como de probar, la herramienta como tal tiene ya un predecesor que ha sido valorado muy positivamente y del que se han diversificado diferentes versiones, esta aplicación podría decirse que no es más que otra diversificación. El implementar un ranking creo que es una mejora positiva para la aplicación.

## Conclusiones finales

### Del diseño

El diseño tenía pensado diferenciarlo un poco más respecto al de Wordle pero no me ha dado tiempo a centrarme demasiado en ese aspecto y al tener la referencia de Wordle pues intenté que quedase bien basándome en esa referencia. Los colores escogidos además para el acierto o no de las letras en la palabra a adivinar me parecían un acierto y una pena cambiarlos (gris, verde y naranja).

El diseño del Login está hecho con bootstrap y la verdad es que no tengo mucho que decir sobre el diseño, creo que está perfecto, le faltaría por implementar el registro y con ello los cambios necesarios para una mayor seguridad (añadir la página correspondiente de registro y añadir el apartado de email para el usuario) pero en general está bien.

### De la aplicación

La aplicación me gusta mucho, me hubiese gustado haber filtrado un poco más las palabras escogidas para jugar, pero al ser demasiadas porque están cogidas de la RAE y ya toqué bastante pues no quise tocar más ni dedicarle más tiempo, pero hay verbos en diferentes tiempos como “pasan” y “pasar” que no me gustaría que estuvieran.

### Del tiempo empleado

El tiempo que he empleado ha sido aproximadamente entre 64h-120h. No estimo muy bien cuántas horas pudieron ser ya que no hice un seguimiento y ha sido cuanto menos irregular, pero podría decirse que fueron más o menos rondando esas horas.

### De los conocimientos adquiridos

Los conocimientos adquiridos han sido sobre todo node.js, axios.js y la manera de conectar con la base de datos desde JavaScript sumado a poder hacer sentencias a la base de datos desde Javascript. También me ha venido muy bien para recordar lo aprendido durante el curso y lo agradezco bastante ya que creo que ha sido bastante útil. También me ha motivado a seguir programando y haciendo proyectos personales para mi futuro desarrollo así como porque me gusta.

Los conocimientos de conectar el frontend con el backend también han sido inmensos y laboriosos

# Bibliografía

1. HTML: <https://developer.mozilla.org/es/docs/Web/HTML>
2. CSS: <https://developer.mozilla.org/es/docs/Web/CSS>
3. JavaScript: <https://developer.mozilla.org/es/docs/Web/JavaScript>
4. Node.js: <https://es.wikipedia.org/wiki/Node.js>
5. Descarga: <https://nodejs.org/en/download/>
6. Axios: <https://axios-http.com/es/docs/intro>
7. SQL: <https://es.wikipedia.org/wiki/SQL>
8. MySQL: [Enlace](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjon97U5uz7AhWux4UKHf5ICIAQmhN6BAhhEAI&url=https%3A%2F%2Fes.wikipedia.org%2Fwiki%2FMySQL&usg=AOvVaw2AcYhwh8LjZo-SUx7BRaVP)
9. VisualStudio Code: <https://code.visualstudio.com/>
10. Enlace: <https://www.apachefriends.org/es/index.html>

Manual de código

# HTML

## Index.html

En el index.html tendremos el formulario de login con acceso al formulario de registro y olvidar contraseña. En dicho HTML lo primero que veremos serán enlaces de ayuda por medio de “link” para enlazar el CSS y bootstrap y “script” de Bootstrap además de JQuery.

Texto

Descripción generada automáticamente

La estructura que seguirá será de un div que bajará con efecto donde guardaremos todo el contenido, luego dentro de ese div padre de todos, tendremos un div con el logo de la aplicación y otro div con el formulario dentro. Antes del formulario dejamos un div en caso de querer meter o implementar una imagen o algo externo al formulario pero en éste caso estará en desuso. Una vez entramos al formulario tendremos 3 tipos de inputs, el del usuario, contraseña y el que envía los datos que será para acceder al juego. Debajo del formulario tendremos otro Div padre de dos enlaces, uno para registrarse como usuario en caso de no tener cuenta y otro por si olvidaste la contraseña (queda por desarrollar a si que lo hice sencillo con un alert para simular un caso real aunque poco práctico diciendo que envíes un correo solicitando la contraseña).

Texto

Descripción generada automáticamente

Al final tengo implementado el script de axios aunque no sé si al final utilizaré axios o node para la BBDD.

## Wordlabras.html

Éste html y los siguientes que voy a explicar están en la carpeta “HTML”. Éste HTML es el HTML del juego, pese a estar muy vacío para todo lo que tiene es porque la mayor parte de las cajas del juego se generan a través de JavaScript de forma interna (esto hace que no sea estático y que se puedan implementar cambios a futuro como que puedas escoger el número de letras que tenga la palabra a adivinar por ejemplo y generar una caja en función de ese número escogido, aunque para hacer esto habría que cambiar parte del código de generado de intentos)

Texto

Descripción generada automáticamente

En el juego tenemos un div entero del juego que tendrá unas medidas para no ocupar por completa la pantalla o sí, dependiendo de la vista, con eso jugaremos para la visualización. En él encontraremos el header que es la cabecera del juego, ésta es la que contiene lo botones para visualizar las opciones, en éste caso serían: Ayuda (completamente inútil actualmente, se haría un html con la explicación del juego sobre cómo jugar), el título del juego, y a la derecha cuenta con un botón para consultar las estadísticas en el cual tengo una función porque pensé en hacerla como un iframe o una ventana que está invisible pero al darle click se visualizará, pero no está realizada, la otra opción es que te redirija a la página de estadísticas. La siguiente es el ranking que visualizará la página con un ranking sacado de la base de datos. Cada botón tiene un emoticono/imagen para visualizar qué es cada cosa, si bien es cierto que pienso que tanto el de estadísticas y ranking quizás a primera vista y sin saber nada puede que no sean tan claros como me gustaría (saber qué son sólo con ver la imagen, es decir, que sean intuitivos).

Tras el header tenemos el “main” que es donde tenemos el juego básicamente, ahí se generarán las cajas de los intentos por medio de JavaScript y el teclado en el div de teclado.

Debajo tengo un div oculto de “stats” que como digo, mi idea inicial era tener una ventanita oculta con las estadísticas para luego consultar BBDD y visualizarla.

Tenemos importado al final el javascript que hará todo el proceso para la creación tanto de los intentos como del teclado entre otras cosas.

## Registro.html

La página de registro es muy similar a la del index.html (login y página principal e inicial) a si que no diré mucho más. Viene a ser lo mismo cambiando un par de cosas, como son los inputs de los datos que consultaremos en BBDD y el input submit con una función en “onclick” que hará todo el proceso de comprobación en BBDD y si es correcto el registro del usuario, si no es correcto, lanzará alertas con los posibles fallos.

Tengo importado axios también por si lo utilizo.

Texto

Descripción generada automáticamente

## Ayuda.html

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

## Stats.html / rank.html

WIP (Work In Progress). Previsto su uso, contendrán más o menos lo comentado en los anteriores HTML’s.

Captura de pantalla de computadora

Descripción generada automáticamente

# CSS

## Login.css

Dicho login está sacado de Bootstrap, lo que hace es poner un fondo del html azul cyan (para nosotros invisible ya que lo que se verá será el fondo del body y sus hijos, se hace para diferenciar las capas mejor).

De lo importante a comentar, la estructura, básicamente hacemos un flex en columna y que esté alineado y justificado por el centro (posición centro) ocupando toda la pantalla, así distribuiremos los divs y todo el contenido el formulario en vertical y alineado en el centro de todo lo que ocupa el formulario (que es el 90% de la ventana). Esto creará esa ventana de login siempre alineada en el centro, dispuesta y ordenada de forma vertical.

Texto

Descripción generada automáticamente

El FormFooter serán los botones de “Registrarse” (añadido por mí) y el de “Olvidé la contraseña”, que aparecerán de forma diferente, con otros colores y efectos, por ejemplo tienen borde pero realmente será un color casi invisible para notar que tiene borde poruqe será también un efecto visual. El formulario posee nodos (los cuales me han molestado a la hora de trabajar con el formulario al recoger los datos) vacíos e inactivos para que queden bien vistosos, les deja en un color gris para que quede bonito el espacio entre los inputs del formulario por medio de etiquetas H2 inactivasm también dando efectos de activas por medio de clases.

Texto

Descripción generada automáticamente

Después tendremos lo visual sobre los botones de etiqueta “input” con los tipos correspondientes. Tendrán los valores dados, con efectos proporcionados con los “-webkit-box-X”, los “-X-transition” y el transition. Esto creará un efecto de aparecer y desaparecer con un tiempo estipulado, en este caso el efecto de aparición y desaparición será de 0.3 segundos. También tienen un efecto de cuando dejas el ratón encima de los inputs el color de fondo cambiará a azul claro. El efecto de “Scale” lo que hace es disminuir 0.05 en escala X e Y el botón.

Texto

Descripción generada automáticamente

Lo demás es un poco de lo mismo con los demás inputs (type text) y demás efectos/animaciones con duraciones, formas y modos hasta que llevamos a los “@KeyFrames”. Éstos KeyFrames son los que permiten controlar los pasos intermedios en una secuencia de animación CSS, en el caso de nuestro código sería para la calase “fadeInDown”, realiza una opacidad al 0% inicialmente hasta un 1 en el 100% del efecto (acabado) por ejemplo.

Texto

Descripción generada automáticamente

Podremos ver más efectos

Texto

Descripción generada automáticamente

Otros displays

Texto

Descripción generada automáticamente

Y por último unas líneas que añadí a mayores para que se viese el logo de la forma que yo quería estando donde lo puse (fuera del div del formulario), una mejor visibilidad del margen superior ya que el Bootstrap estaba hecho para contener una imagen superior encima del formulario pero dentro del mismo, y en éste caso la quité y la saqué fuera porque quedaba mejor y era otro propósito al inicial. Y por último una MediaQuery que hace que se visualice correctamente en vista móvil/Tablet donde el ancho máximo sea de 1000px. Lo que haré será hacer que ocupe toda la pantalla básicamente, lo demás lo vuelvo a declarar por si no se coge correctamente en algunos navegadores.

Texto

Descripción generada automáticamente

## Main.css

Éste CSS será para la página del juego. Sé que es mala práctica (sobretodo si se hace sin saber) el dar valor a todo con \* pero como lo he hecho a sabiendas, he puesto el color a negro y una fuente parecida al Wordle junto con otras parecidas en caso de que no las detecte el navegador a toda la página web (es decir, ha sido a sabiendas).

En la página he creado variables que utilizaré en el futuro de éste CSS. Los botones tendrán fondo blanco sin borde, las imágenes utilizarán un valor fijo dado por variable de medidas, junto con un margen y padding de 0 (que estén bien juntitas las imágenes sobretodo del header que serán a las que afecte).

Texto

Descripción generada automáticamente

Tendremos una clase “juego” que dejará el height en 90vh con una variable y creará la estructura del div del juego con un display flex, se distribuirá de forma vertical y crearemos un grid de 3 estructuras dentro del juego, éstas estructuras serán: Header (33fr), intentos (5fr) y teclado (33fr) con espacios entre ellos de 1px + 1vw para darles espacios dinámicos.

El header se distribuirá con una sección de 4 columnas (una columna ocupando como 2 columnas por lo que tendremos visualmente 3) y 1 fila. Todos los hijos del header ocuparán el 100% (todo lo que puedan) y se distribuirán con display flex y en el centro (es decir, que las 3 columnas ocuparán los “fr” asignados al 100% y se posicionarán en el centro por defecto). Después asigno por encima un grid área para distribuir los espacios de la misma manera que hice pero asignándolo de forma específica por si algún navegador no aceptase display flex pero sí grid (lo cual sería raro pero nunca está de más prevenir).

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

El teclado ocupará el 100%, estará centrado y se alineará de forma centrada horizontal y verticalmente. Tendrá una separación por cada hijo de 4px verticalmente. Cada cajita del teclado tendrá también su propio espacio y tendrá una separación de 4px también que hará la separación “horizontalmente” (y hará el efecto de teclado virtual gracias a esto y los 4px del padre de gap). Los divs tendrán botones, les daremos forma, así como al texto para que se vea bien la letra asignada del teclado.

Texto

Descripción generada automáticamente

Tenemos estética de las cajas de intentos, y unos efectos CSS para lo que se supone que iba a hacer inicialmente de unos stats que aparezcan habiendo estado invisibles hasta que acabe la partida como en el wordle original pero actualmente está sin uso ya que la parte de estadísticas no está hecha y no estará hecha así (lo dejo para que se vea cómo se podría hacer)

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Aquí tenemos los estilos para cuando introduzca letras (El “important” es para que sobrepase cualquier valor inicial y tenga prioridad sobre los otros valores dados), los efectos de “Flip” que son los giros de las cajas de los intentos y los MediaQuerys para la Tablet y vista móvil.

Texto

Descripción generada automáticamente

## Ayuda.css

Donde todo este CSS se centrará de forma rápida con ayuda del CSS “main” de hacer que se visualice correctamente la página de ayuda

Texto

Descripción generada automáticamente

Ayuda.css

Texto

Descripción generada automáticamente

Ayuda.css

# JavaScript

## Main.js

Main.js será el JavaScript que se encargue del juego (que es la aplicación en sí, por lo tanto es el main). Primero cogeremos los divs de intentos y de teclado para poder generarlos. Creamos un array con los espacios necesarios de los intentos, en éste caso 5x6 (5 letras \* 6 intentos), que serán 6 arrays de 5 caracteres.

Inicializo el intento a 0 y la columna del intento a 0, éstas variables nos servirá para saber en qué intento estamos tanto de letra como de palabra (numIntento será el intento o fila y la columna la letra actual). La función “numeroRandom” será para generar un número aleatorio en función del número que le pasemos por parámetro (en éste caso será el número de letras que contiene el .txt que utilizaremos, pero realmente podría ser un .json como hice inicialmente o con una API utilizar sus funciones). Después tenemos el fetch para generar la palabra aleatoria y tenerla siempre accesible

Texto

Descripción generada automáticamente

Después he dejado en comentarios la forma inicial que tenía que era con un archivo .json pero tenía pocas palabras y no me acabó de convencer, también tengo puesta la API de la RAE y un repositorio donde hacen uso de ésta API pero en Python y de donde saqué el txt y la propia carpeta para futuras implementaciones

Texto

Descripción generada automáticamente

Creé una variable de juego acabado y en los comentarios explico el porqué de esa variable, aunque actualmente en las capturas no se verá su uso. Luego he creado un Array con todas las letras del teclado y en el orden que está en el teclado para luego generarlo.

Texto

Descripción generada automáticamente

Dicho array también lo usaré para comprobar qué letras leerá el evento “onclick” y el del teclado para ver si coincide con las posibles letras del teclado virtual, ya que si no, con el teclado detectará teclas como F12, F1, 2, 3, 4, etc… Y no queremos que pinte en las cajas de intentos esas pulsaciones porque no forman ninguna palabra.

Después tendremos la creación de la estructura de los intentos y su visualización, lo crearemos dándoles atributos como id para luego poder manejarlos. Se han creado con un bucle for-each utilizando el arrayIntentos pero éste podría ser variable como una mejora y que el usuario escogiese el número de letras de la palabra y que se auto-generase éste cuadro de intentos en base a esa elección y no sería muy difícil de implementar (sería hacer un par de cambios).

Después tenemos el contador de botones creados, para poder simular un teclado real y que no ponga todos los botones en la misma fila (en éste caso, en el mismo div). Mientras los creamos también les damos los eventos de click, no lo haremos con el teclado porque si lo hacemos así, el usuario deberá dar click en el botón para tener el “focus” de la tecla virtual y poder detectar dicha pulsación del teclado real. Tendremos que dejar la lectura de teclado en “document” para que lo lea siempre la pulsación del teclado y que sea más amena la experiencia del usuario.

Texto

Descripción generada automáticamente

Posteriormente tendremos la función utilizada en el evento de ‘keydown’ que sería la pulsación, donde la función será “recogerTecla”, que lo que hará será comparar la tecla pulsada con las de ENTER o borrado para hacer una función y otra (enviar la palabra o borrar letra) o compararla con las letras del array del teclado como comenté en dicho apartado, para que no detecte pulsaciones o teclas como “F12” por ejemplo.

Texto

Descripción generada automáticamente

Después tendremos una función muy parecida pero con el evento click en cada letra del teclado virtual generado, no necesitamos recorrer aquí el array con las letras del teclado virtual porque el usuario va a tener que hacer click en cada letra directamente entonces no puede haber fallo con letras no gestionadas (o que no estén en dicho teclado). Cabe decir que el teclado no está preparado para poner tildes por ello mismo. Tampoco son necesarias porque eso podría detectarse a la hora de comparar la palabra sin tilde perfectamente.

Texto

Descripción generada automáticamente

La función “addLetra(param)” será para añadir la letra al div correspondiente (dependiendo de la columna y fila en la que nos encontremos). Usaremos el contador “columnaIntento” para manejar la posición en la fila o intento que nos encontramos. Le he añadido un atributo de ‘datos’ con la letra para un futuro uso con los efectos.

La función “borrarLetra()” borrará la letra de la posición actual y nos posicionará en la columna anterior en caso de que no estemos en la columna inicial.­­

Texto

Descripción generada automáticamente

El siguiente método a ver es el de “comprobarPalabra()”, este método se encarga de comparar la palabra introducida cuando se pulsa “ENTER”, primero comprueba si la columna actual es la última, si no lo es, no hace nada, así no hace todos los siguientes pasos en caso de que no sea una palabra de 5 letras. Una vez la palabra es de 5 letras, comprueba en el .txt por medio de una petición fetch, entre todas las palabras del archivo .txt separadas por saltos de línea si coincide con alguna palabra, si no coincide, diremos que la palabra no existe y deberá volver a escribir la palabra, pero no borraremos toda la fila por si sólo se equivocó en la última letra aunque podría haber sido una opción borrar todas las letras de la fila (sólo que no me gustaba y el Wordle original además no funciona de esa manera). En vez de utilizar un boolean para saber si la palabra existe o no, he utilizado un contador (0 = no existe, +1 = existe), porque inicialmente el .txt estaba mal y tenía las palabras repetidas aunque luego eso lo solucioné con un comando y una expresión regular para modificar el archivo. Usará también la función efectos que básicamente serán todos los efectos que hemos visto por CSS también de “flip” y los colores que asignaremos (lo veremos en más profundidad cuando vayamos a la función). Lo primero que comprueba es si la palabra intento coincide con la palabra aleatoria, así no miramos más, en ese caso el juego habrá acabado, actualmente no hace nada más pero debería guardar los datos (número de intentos y si es victoria o no) y enviárselos a la BBDD.

En caso de que no sea así, comprobaremos si el intento es el último, si lo es, es que ha perdido y sacaremos por pantalla un alert sencillo (mejorable con un mensaje más currado, ya sea con un div oculto o un iframe). En caso de que no sea su último intento, aumentamos el número de fila (intento) y reseteamos la columna intento a 0 (posición 1 en columna).

Texto

Descripción generada automáticamente

La función “addColoATecla(letra, color)” es una función que añade a todos los nodos (esto incluye tanto la tabla de intentos como la del teclado, por eso está hecho de ésta manera) de la letra que hayamos pasado por parámetros el parámetro que pasemos (verde, naranja o gris oscuro).

El método efectos es un método donde comprobamos letra por letra si coincide con la palabra y sus posiciones, primero comprobamos si no existe la letra, en cuyo caso la pintamos de color gris oscuro, en caso de que exista sin conocer aún la posición, obtendrá el color naranja, y si existe y en la misma posición, se pintará de verde (esto hará que sobrescriba el color naranja pero realmente se hace tan rápido que no será posible apreciarlo). Después en toda la fila (que hemos llamado “cuadro” la cual está pasada por parámetro cogida de “filaIntento” y la posición que será el “index”. Lo hacemos con un “setTimeout” para que no sea inmediato y se pueda apreciar el efecto de flip (giro), así como añadir los colores). Con esto conseguiremos el efecto tanto de los colores en ambas partes (teclado e intentos) como de giro de los cuadros de intentos.

Texto

Descripción generada automáticamente

## Mysql.js (WIP)

Actualmente usando node.js para realizar la conexión con la BBDD, comprobamos la conexión con una función “connect” sencilla y un mensaje por consola, tendremos funciones con posibilidad de exportar/importar (gracias a node.js y la declaración “export” en éste js y la declaración de “import” en el js que queramos importarlas) las funciones necesarias para hablar con la BBDD. Una será comprobarEmail que como dice la función comprobará que el email tiene formato email por medio de una expresión regular, después deberá comprobar dicho email en la BBDD para comprobar que no exista ya (ya que el campo de email sólo estará en el registro y no será para login). La función de comprobarUsuario que comprobará si existe el usuario, devolverá un booleano o un int de 0 o 1 (aún por hacer), en caso falso o 0 será que el usuario no existe, en caso verdadero o 1 será que existe, y dependiendo de ese return, en función de si estamos en el registro o en el login, haremos una cosa u otra (en registro dará error si existe porque ya está creado el usuario, y en login no hará nada y comprobará luego el siguiente dato, que será la contraseña). La función registrarUsuario() será para el registro, necesitará los parámetros que son los 3 datos para registrar un usuario (añadí el email pese a que en principio no debería estar ni ser necesario así como hacer el registro). Quedan por hacer además de éstas funciones la de login, la de sacar con un SELECT las estadísticas y un ranking, en el cual el SELECT para obtener el ranking sería:

|  |
| --- |
| (SELECT usuario FROM `usuarios` WHERE usuario ='admin'), (SELECT SUM(victoria) FROM `estadisticas` WHERE estadisticas.victoria = 1), (SELECT SUM(intentos) FROM `estadisticas`)) |

Texto

Descripción generada automáticamente

## App.js (WIP)

Éste JS será para todo el funcionamiento y conexión/intercambio con la BBDD pero manteniendo toda la base en el js de mysql, es decir, aquí tendrá todas las funcionalidades más cercanas a la app (recogida de datos, uso de las funciones de mysql.js, etc…).

Texto

Descripción generada automáticamente

Pantalla negra con letras blancas

Descripción generada automáticamente

# Manual de Juego

Para emplear la herramienta se necesitará el software y hardware indicado en la memoria descriptiva. Al estar puesta en local (localhost) se tendrá en cuenta que toda la aplicación ha sido probada y usada sólo en local, pero probablemente se podría subir a una página de hosting y hacer que funcionase (sobretodo funcionalidades básicas como es el juego de forma aislada sin conexión al back-end y con ello a la base de datos).

* VisualStudioCode
* Se necesitará el liveserver de VisualStudioCode que se utilizará para inicializar el proyecto desde la carpeta raíz
  + 
* Instalar las librerías necesarias
* Activar el backend previamente y que escuche por el puerto
* Xampp
* La BBDD

Pasando a el manual del juego, en la página de ayuda tengo escrito e indicado muy bien cómo jugar, a si que dejaré una captura de dicho apartado del propio proyecto para que sirva como referencia al manual de juego

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Los pasos que seguiremos para poder usar la aplicación serán los siguientes:

### Registrarse

El inicio será en la página de login, pero no tendremos usuario para registrarnos, por lo que deberemos darle al botón de “Registrarse”.



## Login

Esto nos redirigirá a la página de registro donde pondremos nuestros datos para luego hacer login. Al dar a “Registrarse”, se hará de forma automática si los datos eran correctos (si no, saltará un aviso) y nos redirigirá a la página de login donde pondremos nuestros datos y le daremos a “Acceder”

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

## Jugar

Con ayuda de la imagen mencionada anteriormente sobre “cómo jugar”, entenderemos mejor cómo funciona el juego, pero de forma resumida; se trata de adivinar la palabra aleatoria y el propio juego te va dando pistas, color gris es que la letra no está, color naranja es que la letra está pero no en la posición exacta de la palabra que has indicado y color verde es que la letra está en la palabra aleatoria y en la posición de la palabra escrita.

Tabla

Descripción generada automáticamente

Gráfico, Gráfico en cascada

Descripción generada automáticamente

## Ganar

Escribes palabras hasta que des con la palabra correcta

Gráfico, Tabla, Gráfico de rectángulos

Descripción generada automáticamente

## Perder

En el caso de que hayas gastado todos los intentos (6), habrás perdido y no podrás escribir e intentar más palabras.

Tabla

Descripción generada automáticamente

# Pruebas del sistema

## Pruebas a realizar

### Pruebas del Registro

#### Todos los campos vacíos del formulario de Registro

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

#### Email vacío en los campos

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

#### Email inválido en la página de registro

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

#### Email inválido por parámetros propios de un email ([ejemplo@algo.otro](mailto:ejemplo@algo.otro))

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

#### Todos los parámetros correctos

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Pulsamos “Registrarse” y no da ningún error.

(WIP) – Después te redirigirá a la página de login.

### Pruebas del juego

#### Pulsar una tecla que no esté en el teclado virtual

No la pinta por pantalla la pulsación de “F12” por ejemplo.

Tabla

Descripción generada automáticamente

#### Pulsar “ENTER” sin haber escrito nada antes

No envía datos ni ocurre nada.

Tabla

Descripción generada automáticamente

#### Escribir una palabra inexistente y pulsar o hacer click en “ENTER”

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

#### Efectos flip

Introducimos una palabra y deberían de girar las letras como un efecto tipo “pasapalabra”.

Tabla

Descripción generada automáticamente

#### Letras repetidas

Deberá indicar con el color si la palabra aunque esté repetida existe y si está o no en la posición correcta como es en el caso de la siguiente imagen, donde la primera “A” existe en la palabra y está bien posicionada y la segunda indica que existe otra “A” pero está en otra posición (La palabra era Rabea).

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente con confianza baja

En la siguiente imagen podemos observar que la palabra es “Japón” y no contiene dos “A”, por lo que una de ellas deberá pintarla en gris como es en el caso de la siguiente imagen, por lo que funciona correctamente.

Tabla

Descripción generada automáticamente

#### Notificar al usuario cuando gana

Tabla

Descripción generada automáticamente con confianza media

#### Notificar al usuario cuando pierde e informarle de la palabra que era

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

#### Intentar borrar letra sin tener letras

Tabla

Descripción generada automáticamente

No ocurre nada pero realmente por dentro da un error esperado de que no se puede borrar porque está vacío:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. HTML: <https://developer.mozilla.org/es/docs/Web/HTML> [↑](#footnote-ref-1)
2. CSS: <https://developer.mozilla.org/es/docs/Web/CSS> [↑](#footnote-ref-2)
3. JavaScript: <https://developer.mozilla.org/es/docs/Web/JavaScript> [↑](#footnote-ref-3)
4. Node.js: <https://es.wikipedia.org/wiki/Node.js>

   .1 Descarga: <https://nodejs.org/en/download/> [↑](#footnote-ref-4)
5. Axios: <https://axios-http.com/es/docs/intro> [↑](#footnote-ref-5)
6. SQL: <https://es.wikipedia.org/wiki/SQL> [↑](#footnote-ref-6)
7. MySQL: [Enlace](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjon97U5uz7AhWux4UKHf5ICIAQmhN6BAhhEAI&url=https%3A%2F%2Fes.wikipedia.org%2Fwiki%2FMySQL&usg=AOvVaw2AcYhwh8LjZo-SUx7BRaVP) [↑](#footnote-ref-7)
8. VisualStudio Code: <https://code.visualstudio.com/> [↑](#footnote-ref-8)
9. Enlace: <https://www.apachefriends.org/es/index.html> [↑](#footnote-ref-9)