CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

ACTIVIDAD FINAL
SEGUNDA EVALUACIÓN

Desarrollo Web en entorno cliente
CFGS DAW

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:230207.1258

Licencia

ÍNDICE

ACTIVIDAD FINAL - 2ª EVALUACIÓN

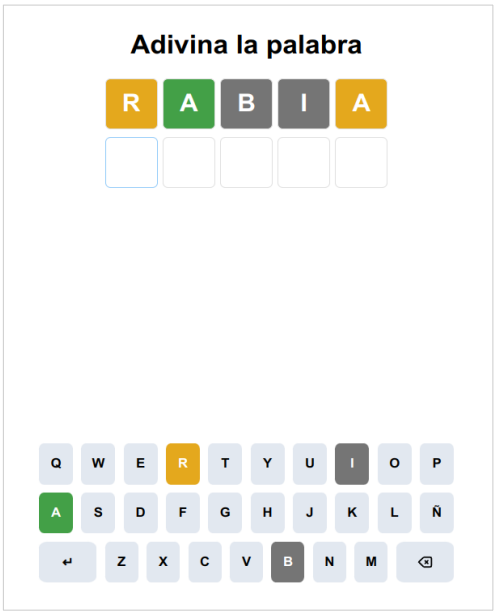
Revisa los criterios de evaluación para saber que es lo que debes tener en cuenta a la hora de realizar el ejercicio: no es suficiente que el programa cumpla la función, debe cumplir además otros criterios para ser considerado un buen código JavaScript.

1. ENUNCIADO

En este ejercicio debes programar un juego para adivinar palabras utilizando React y Redux toolkit.

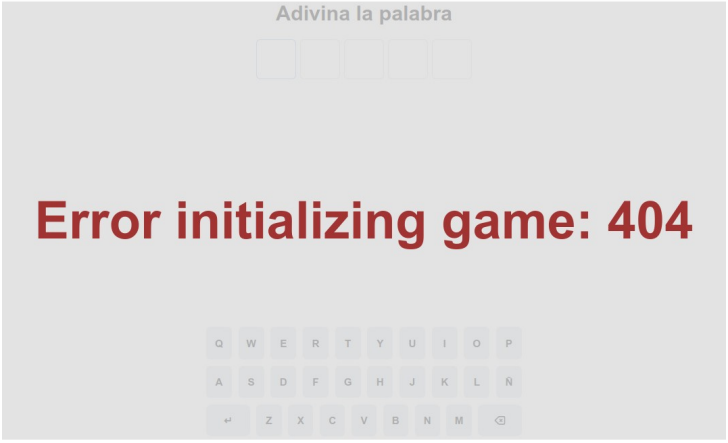
Para generar el HTML debes basarte en los ejemplos que se proporcionan en el repositorio de ejemplos: https://github.com/CEED-2022/ev2-actividad-final-plantillas. El HTML debe ser el mismo para los mismos casos de la aplicación. Debes utilizar los ficheros CSS que se proporcionan en dicho repositorio. No se admitirán para su corrección ejercicios que generen un HTML o CSS diferente.

El juego tiene el siguiente aspecto:



Ha de funcionar de la siguiente forma:

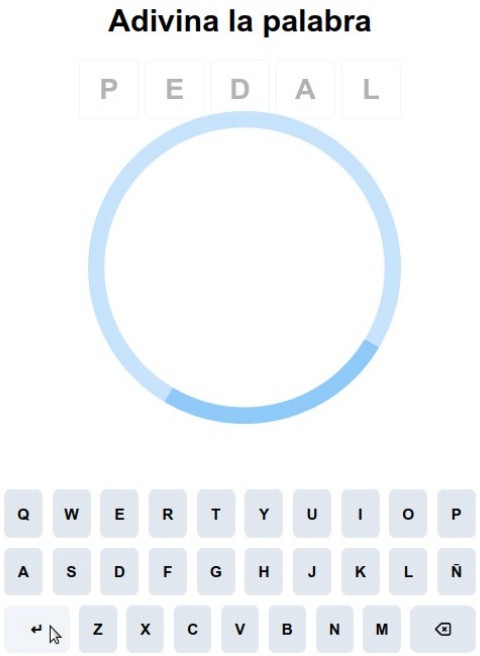
- Al cargar la página se iniciará una nueva partida. Para ello, se enviará una petición `POST /new` al servidor. El servidor devolverá un ID, que será el que se utilice en las peticiones sucesivas para gestionar la partida. Si no se ha podido obtener un ID se mostrará un mensaje de error como el siguiente, similar a los mensajes que se muestran al ganar o perder el juego:



- Al pulsar con el ratón en un recuadro **de la palabra actual**, se seleccionará dicho recuadro. Para seleccionar el recuadro debes asignarle al elemento la clase `selected`.
- Al pulsar una letra en el teclado virtual:
  - Si no hay ningún recuadro seleccionado, no hará nada
  - El recuadro seleccionado se rellenará con la letra pulsada en mayúsculas
  - Después, se seleccionará el primer recuadro vacío de la palabra. Si no hay recuadro vacíos, no habrá ninguno seleccionado,
- Al pulsar borrar en el teclado virtual:
  - Si el recuadro seleccionado tiene contenido, se borrará. Si no:
    - Si es la primera letra, no hará nada
    - El recuadro anterior al seleccionado se borrará
  - Si no hay ningún recuadro seleccionado, se borrará el último recuadro
  - Se seleccionará el primer recuadro vacío de la palabra.
- Al pulsar "enter" en el teclado virtual:
  - Si no están llenos todos los recuadros de la última palabra, mostrará el error bajo el teclado: `'No hay suficientes letras'` y no hará nada más
  - Se lanzará una petición `GET /check/:word` al servidor. Si la palabra no es válida, mostrará el error `'La palabra no está en la lista'` debajo del teclado
  - Se lanzará una petición `POST /guess/:gameId` al servidor para cada una de las letras.
  - **Tras obtener el estado de todas las letras**, se asignarán colores a las letras de la palabra comprobada, según los siguientes criterios en orden de prioridad:
    - Si la letra coincide en la misma posición con la palabra a adivinar, se pondrá en verde añadiendo la clase `green`

        - Si la letra está contenida en la palabra a adivinar, se pondrá en amarillo añadiendo la clase `yellow`
        - Si no, se pondrá en gris añadiendo la clase `gray`
        - Se actualizarán los colores de las letras del teclado. Cada letra tendrá el "mejor" color de las letras comprobadas hasta el momento. El verde tiene mayor prioridad que el amarillo. Es decir:
            - Si la letra no se ha usado en una palabra, tendrá el color por defecto
            - Si la letra se ha usado en alguna palabra y estaba en verde, tendrá color verde
            - Si la letra se ha usado en alguna palabra y estaba en amarillo, y además no estaba en verde en ninguna, tendrá color amarillo.
            - En caso contrario, tendrá color gris oscuro.
    - Si se ha acertado la palabra, se mostrará el overlay sobre el juego con el mensaje `'Has ganado'` Si no:
        - Si hay menos de seis palabras:
            - Se añadirá una nueva palabra debajo de la actual
            - Se seleccionará la primera letra de la nueva palabra
        - Si era la sexta palabra, se mostrará el overlay sobre el juego con el mensaje `Has perdido'`

Mientras se esté esperando alguna petición al servidor se mostrará un icono de carga sobre las palabras, y el teclado virtual no realizará ninguna acción.



No te inventes el funcionamiento del juego. Debe funcionar exáctamente como se indica en las

instrucciones. No añadas funcionalidad adicional.

Puedes ver un ejemplo del juego en funcionamiento en esta dirección:

https://adivina-palabra-example.fly.dev/

2. API

El servidor que has de utilizar para la práctica es el que está en este repositorio: https://github.com/CEED-2022/adivina-palabra-server. Puedes lanzarlo en local con `yarn start`, o bien acceder al mismo servidor desplegado en https://adivina-palabra.fly.dev

El servidor espera JSON en las peticiones que necesiten datos, y devuelve JSON. El servidor puede devolver un campo error en el JSON de respuesta; en ese caso, la petición se considerará fallida con el error devuelto.

Los endpoints del servidor son los siguientes:

2.1.1 POST /new

Crea una nueva partida y devuelve el ID de la partida

Ejemplo:

```
curl -X POST https://adivina-palabra.fly.dev/new
Respuesta:
{"id":"c0c726f2-b591-4155-9a54-7464d1cffee0"}
```

2.1.2 GET /check/:word

Indica si una palabra es válida o no.

Ejemplo:

```
curl -X GET https://adivina-palabra.fly.dev/check/datil
Respuesta:
{"valid":true}


curl -X GET https://adivina-palabra.fly.dev/check/lemon
Respuesta:
{"valid":false}


curl -X GET https://adivina-palabra.fly.dev/check/banana
Respuesta:
{"error":"You must provide a five letters word in \"word\" field"}
```

2.1.3 POST /guess/:id

Chequea una letra de la palabra correspondiente a la partida. Se debe enviar como parámetros la posición de la letra a chequear (empezando por el cero) y la letra. Devolverá en el campo status:

- “in position” si la letra está en la palabra en esa posición

- “in word” si la letra está en la palabra pero no en esa posición
- “wrong” si la letra no está en la palabra

Ejemplos:

```
curl -X POST https://adivina-palabra.fly.dev/guess/$GAME_ID \
  --header 'Content-Type: application/json' \
  --data '{
      "position": 0,
      "letter": "a"
}'
Respuesta:
{"status":"in word"}

curl -X POST https://adivina-palabra.fly.dev/guess/WRONGID \
  --header 'Content-Type: application/json' \
  --data '{
      "position": 0,
      "letter": "a"
}'
Respuesta:
{"error":"Game WRONGID not found"}
```

2.1.4 POST /gess/:id/:word

Este endpoint está pensado para que puedas hacer pruebas con la aplicación. Funciona igual que `/guess/:id` pero tomará `:word` como la palabra a adivinar. No chequeará que el `:id` de partida exista.

Ejemplo:

```
curl -X POST https://adivina-palabra.fly.dev/guess/WHATEVER/adios \
  --header 'Content-Type: application/json' \
  --data '{
      "position": 2,
      "letter": "i"
}'

Respuesta:
{"status":"in position"}
```

3. Entorno de desarrollo

Todo el código HTML, CSS y js estará en el directorio `src`.

Deberás montar un entorno de desarrollo con Webpack. Debe admitir dos scripts:

- `yarn start` lanzara webpack-dev-serve con autoreloading
- `yarn build` generará los ficheros distribuibles de la aplicación en el directorio dist

Se debe poder probar la aplicación en desarrollo con `yarn start`

Se debe poder lanzar la aplicación y funcionar correctamente con el código generado en `dist`. No

incluyas ese código en tu entrega, debe poder generarse automáticamente con `yarn build`

4. Linter

Para que el ejercicio se corrija, el linter no debe devolver ningún error. En caso de que el linter devuelva un error, la máxima nota del trabajo será de 1.

Las reglas del linter son las que están especificadas en el fichero `.eslintrc.json` del repositorio de ejemplo. No puede modificarse este fichero. Asimismo no se admitirá deshabilitar ninguna de las reglas de eslint por ningún medio: en ese caso se procederá como si el programa hubiese fallado el linter.

5. Entrega

Para la entrega debes eliminar los directorios `node_modules` y `dist` y comprimir el directorio de tu proyecto en un único fichero `.zip` o `.gz`.

Debes entregar el ejercicio como un único fichero con el siguiente nombre: `NOMBRE_APELLIDO1.[zip|gz]` (sustituye *NOMBRE* y *APELLIDO1* por tu nombre y tu primer apellido) El fichero se entregará en la tarea del curso habilitada a tal efecto.

No se admitirán entregas pasada la fecha límite. No se admitirán entregas con formato incorrecto.

6. Criterios de evaluación

- El programa es correcto, realiza la función que se solicita en el enunciado
- Se ha utilizado React y Redux Toolkit para desarrollar la aplicación
- Se ha distribuido correctamente la funcionalidad de la aplicación en componentes
- No se ha manipulado el DOM directamente
- Se han utilizado estructuras del lenguaje adecuadas: bucles, condicionales, operadores, etc.
- Se han utilizado variables y constantes de forma adecuada
- Se utilizan correctamente y cuando corresponda los tipos de datos y objetos predefinidos del lenguaje (Arrays, objetos planos, Map, Set, etc.)
- Se han utilizado funciones para estructurar el código, definiendo y utilizando parámetros y valores de respuesta de forma adecuada
- El programa es lo más sencillo posible para realizar su función.
- No existe código repetido: se han extraído los comportamientos comunes a funciones y se ha intentado hacer el código genérico.
- El programa cumple todas las reglas definidas para el linter.
- No se ha modificado el CSS ni el HTML de la plantilla
- Se han utilizado correctamente las funciones de React y Redux Toolkit