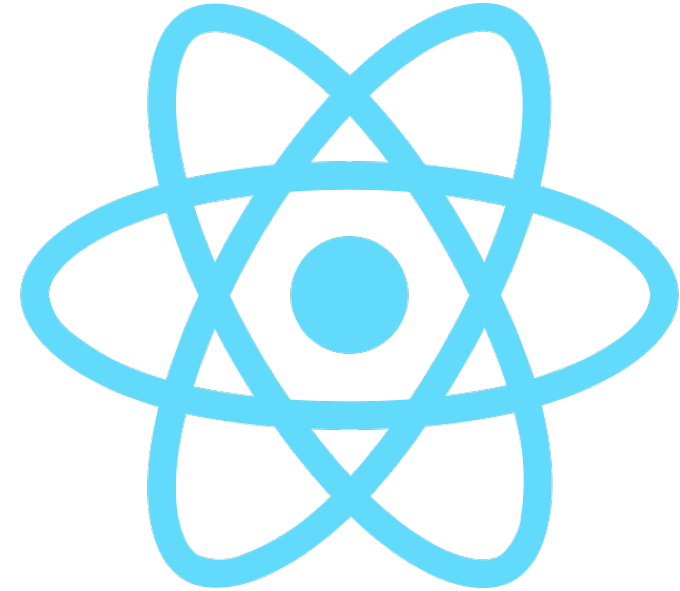


# Crea tu primera app con React



**Mariona Roca**

Lead Teacher in Ironhack





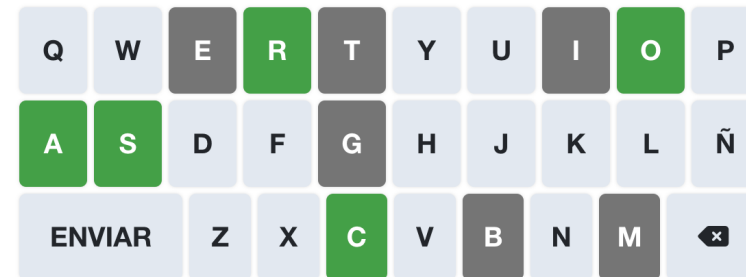
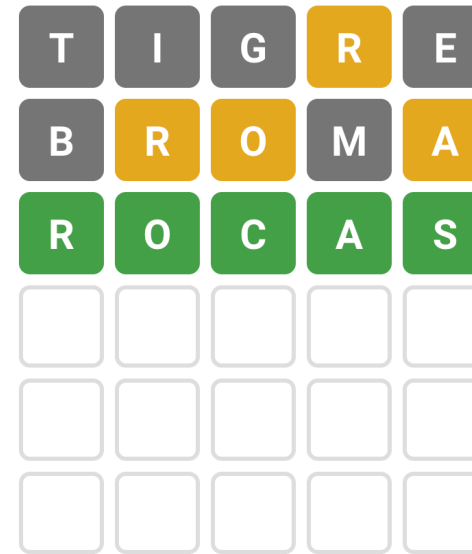
Código y slides  
disponibles en  
**GITHUB:**

<https://github.com/grannysimons/primer-app-react>.git

# WORDLE

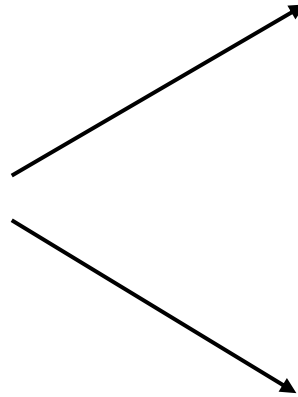
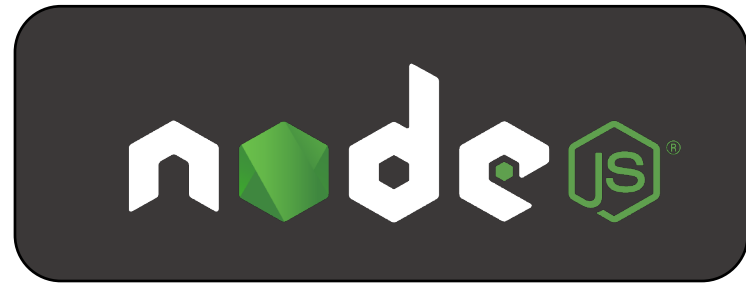
<https://lapalabradeldia.com/>

LA PALABRA  
DEL DÍA



# **SET UP DEL PROYECTO**

# SET UP DEL PROYECTO



# SET UP DEL PROYECTO



<http://getbootstrap.com>



**LAYOUT**

**Animate.css**

Just-add-water CSS animations

<http://animate.style>



**ANIMACIONES  
CSS**

**Font Awesome**

<https://fontawesome.com/>



**ICONOS  
TIPOGRÁFICOS**

# SET UP DEL PROYECTO

1. Abrimos terminal apuntando al directorio
2. Ejecutamos la ultima versión de ***create-react-app***\*:

```
npx create-react-app@latest Wordle
```

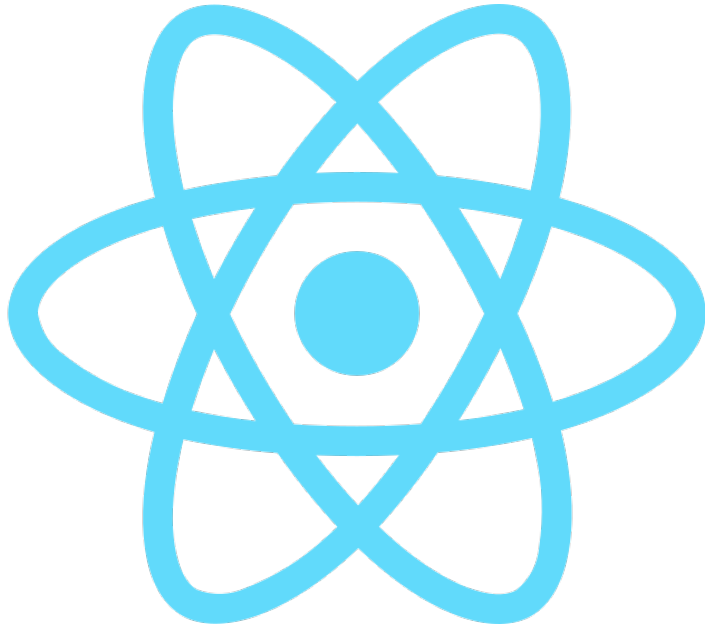
3. Limpiar ficheros que no vamos a utilizar

\* En producción mejor utilizar un framework tipo Vite, Next o Gatsby

# **QUE ES REACT JS**

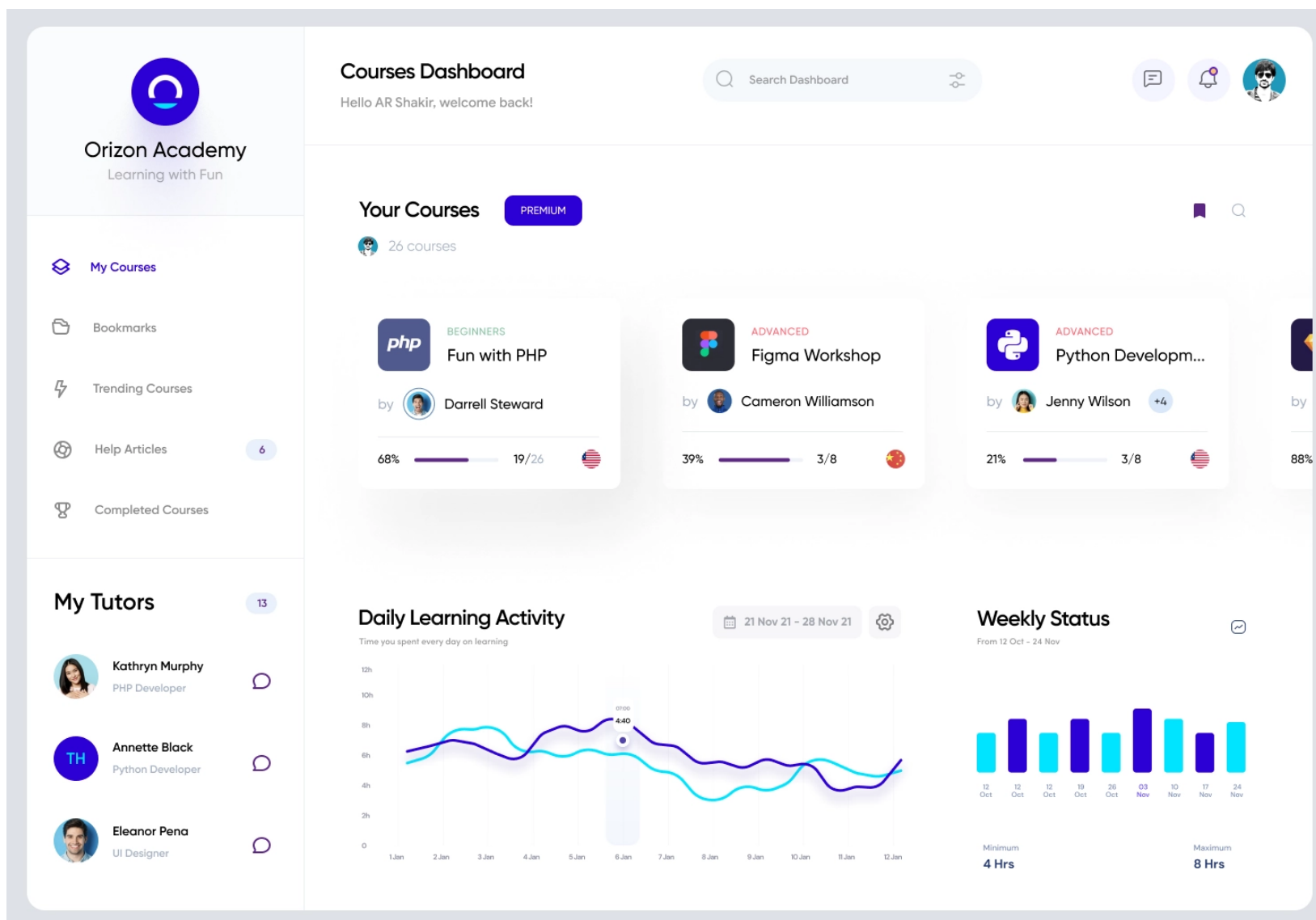


# QUÉ ES REACT?



- Para construir **SPA** (un solo fichero html)
- Estructurada de **COMPONENTES**
- Cada componente tiene:
  - **props** (parámetros de entrada)
  - **estado** (parámetro interno)
- Componentes se **re-renderizan individualmente** con cambios en props o estado

# QUÉ ES REACT?



# COMPONENTES

# COMPONENTES COMO SE DEFINEN?



```
// fichero App.jsx o App.js  
// FUNCTIONAL COMPONENT
```

```
import './App.css';  
import Wordle from './components/Wordle';
```

```
function App() {  
  return (  
    <div className="App">  
      <Wordle />  
    </div>  
  );  
}
```

```
export default App;
```

# COMPONENTES COMO SE DEFINEN?



```
// fichero App.jsx o App.js  
// FUNCTIONAL COMPONENT
```

```
import './App.css';  
import Wordle from './components/Wordle';
```

```
function App() {  
  return (  
    <div className="App">  
      <Wordle />  
    </div>  
  );  
}
```

```
export default App;
```

# COMPONENTES COMO SE DEFINEN?



```
// fichero App.jsx o App.js  
// FUNCTIONAL COMPONENT
```

```
import './App.css';  
import Wordle from './components/Wordle';
```

```
function App() {  
  return (  
    <div className="App">  
      <Wordle />  
    </div>  
  );  
}
```

```
export default App;
```

# COMPONENTES COMO SE DEFINEN?



```
// fichero App.jsx o App.js  
// FUNCTIONAL COMPONENT
```

```
import './App.css';  
import Wordle from './components/Wordle';
```

```
function App() {  
  return (  
    <div className="App">  
      <Wordle />  
    </div>  
  );  
}
```

```
export default App;
```

JSX:

- **className** en lugar de class
- Elementos sin cierre necesitan barra
- Expresiones de js entre { }

# COMPONENTES COMO SE DEFINEN?



```
// fichero App.jsx o App.js  
// FUNCTIONAL COMPONENT
```

```
import './App.css';  
import Wordle from './components/Wordle';
```

```
function App() {  
  return (  
    <div className="App">  
      <Wordle />  
    </div>  
  );  
}
```

```
export default App;
```

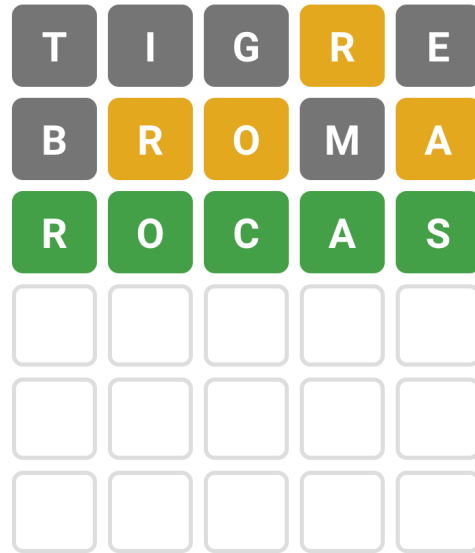


# COMPONENTES

Palabra correcta! ×

LA PALABRA

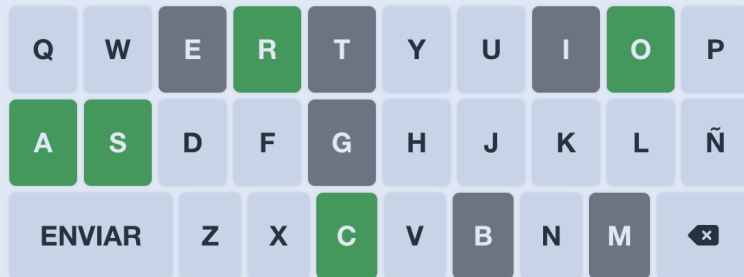
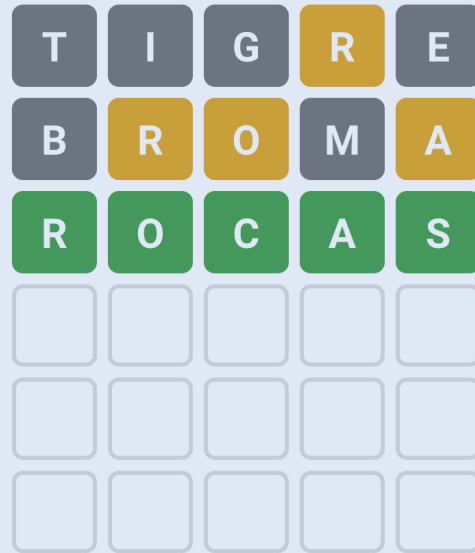
DEL DÍA



# COMPONENTES

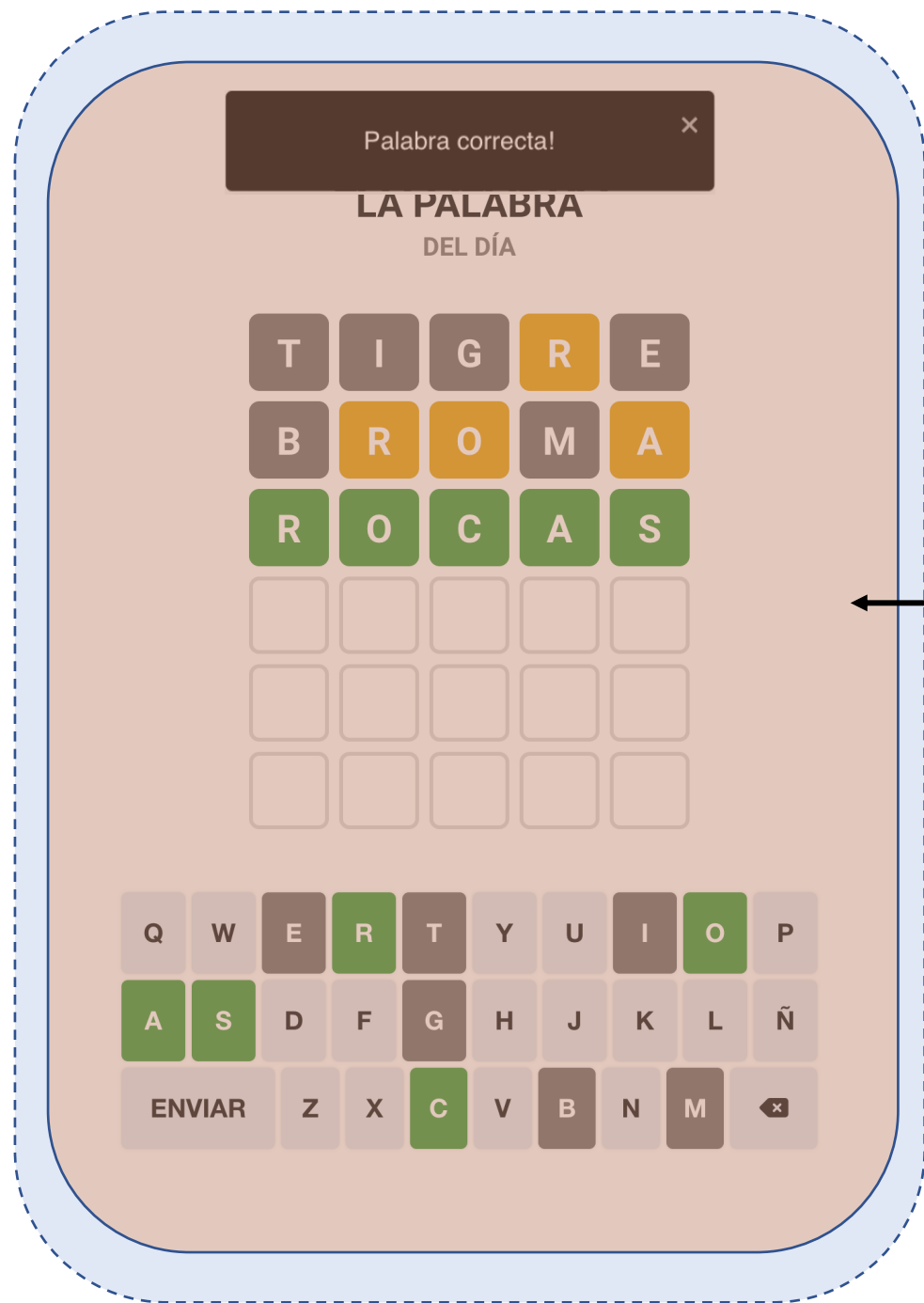
Palabra correcta! x

LA PALABRA  
DEL DÍA



← <App />

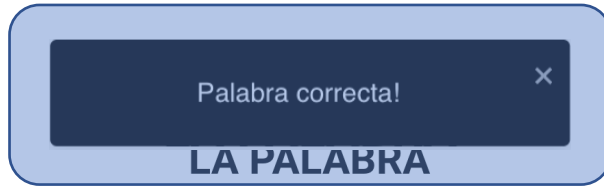
# COMPONENTES



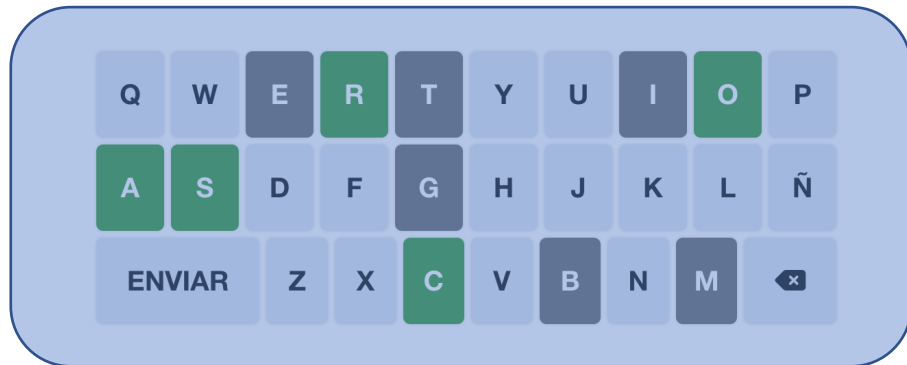
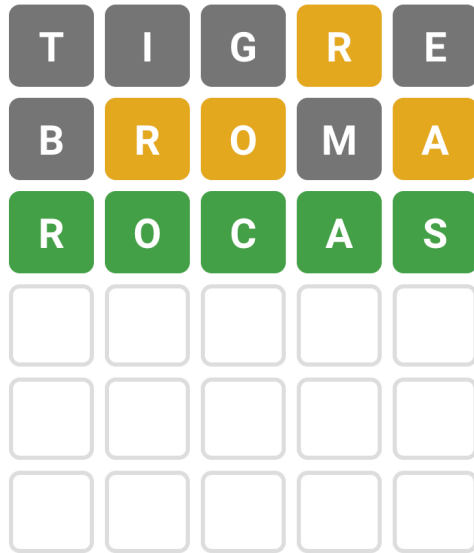
<App />

<Wordle />

# COMPONENTES



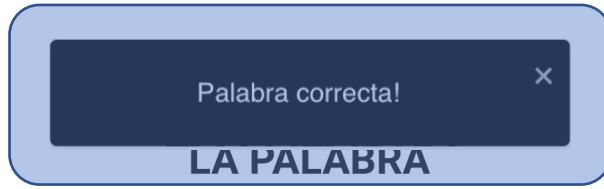
DEL DÍA



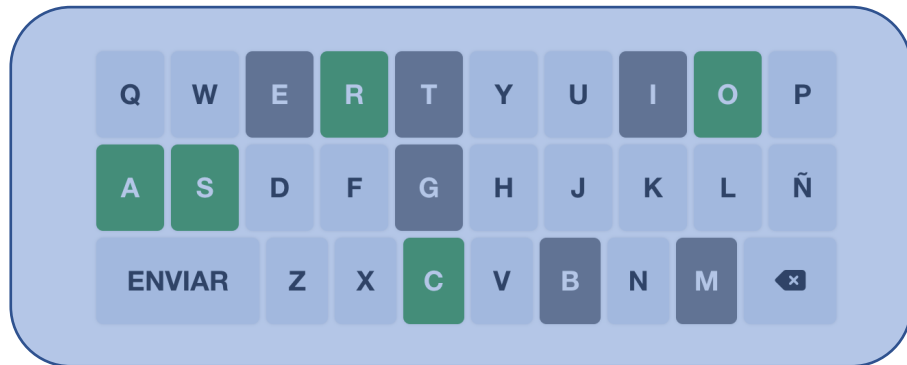
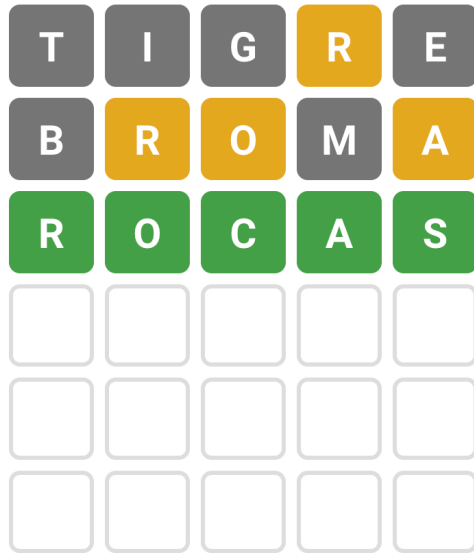
`<react-toastify />`

`<react-simple-keyboard />`

# COMPONENTES

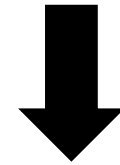


DEL DÍA



`<react-toastify />`

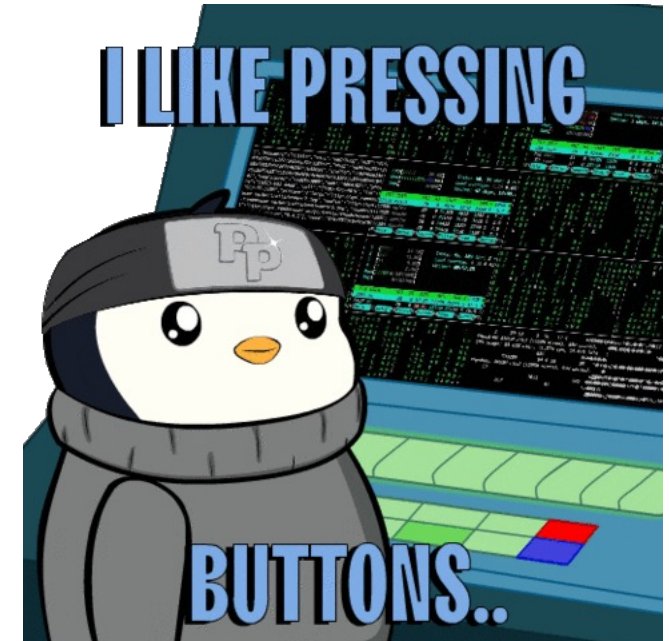
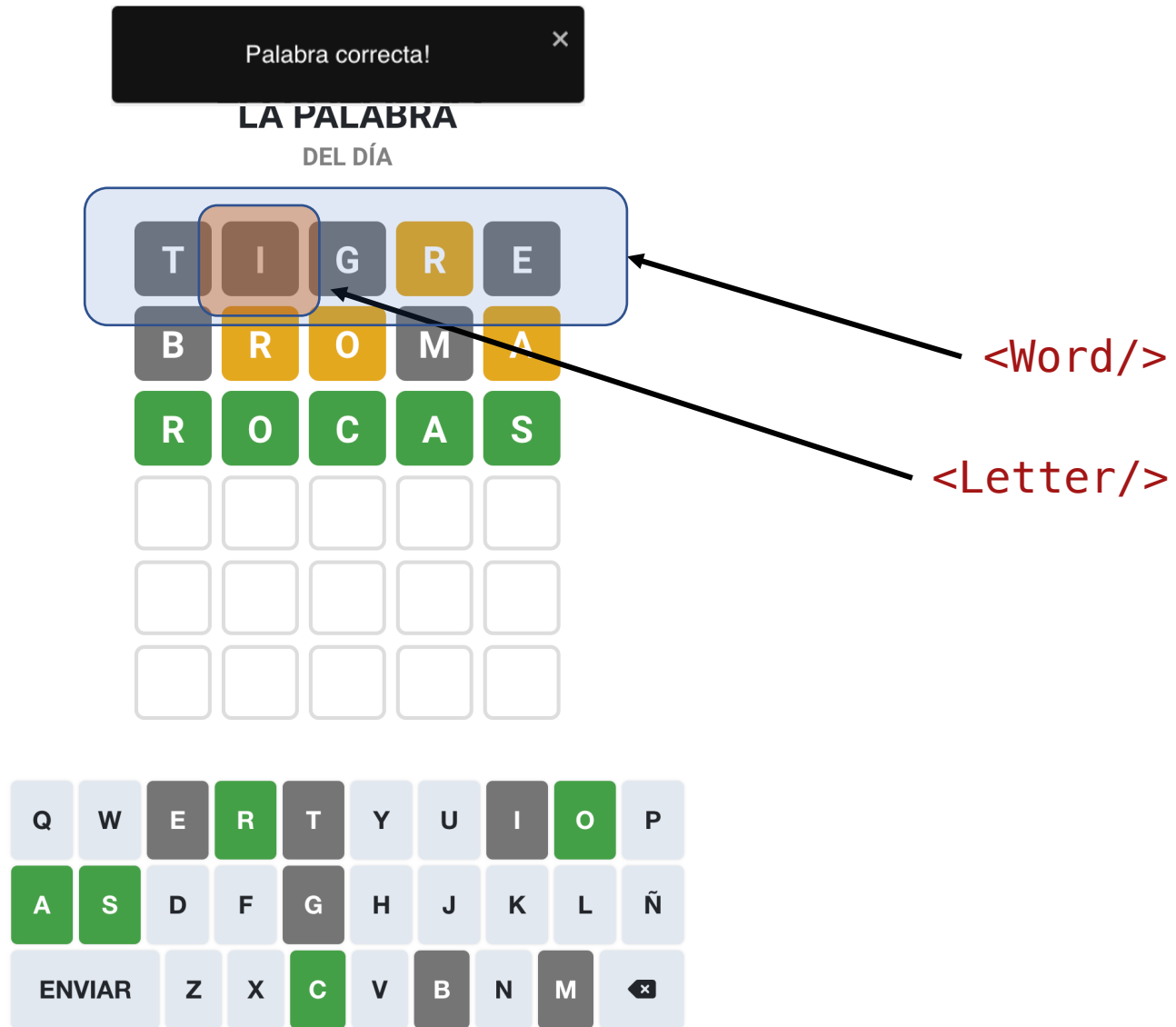
`<react-simple-keyboard />`



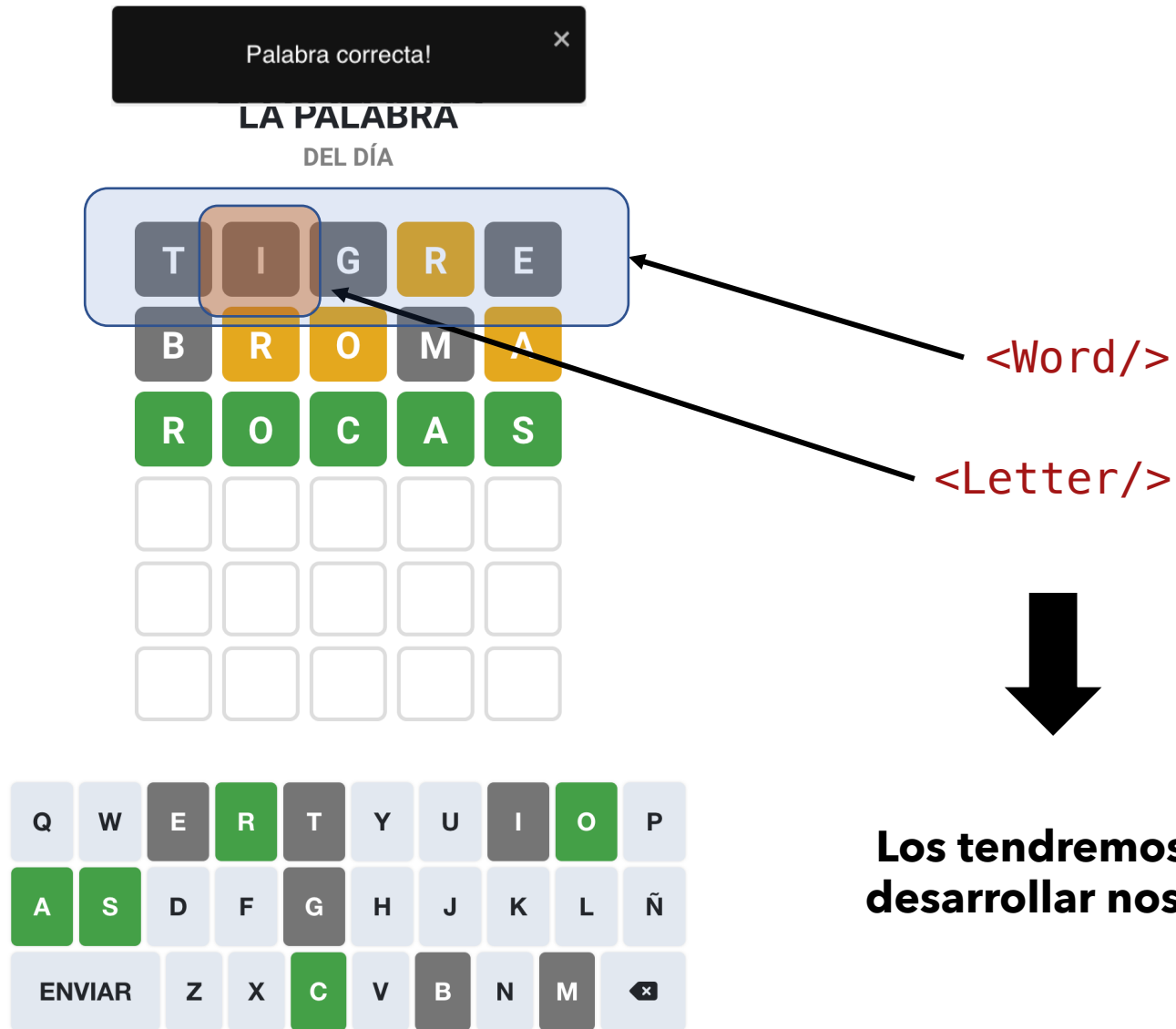
Disponibles en



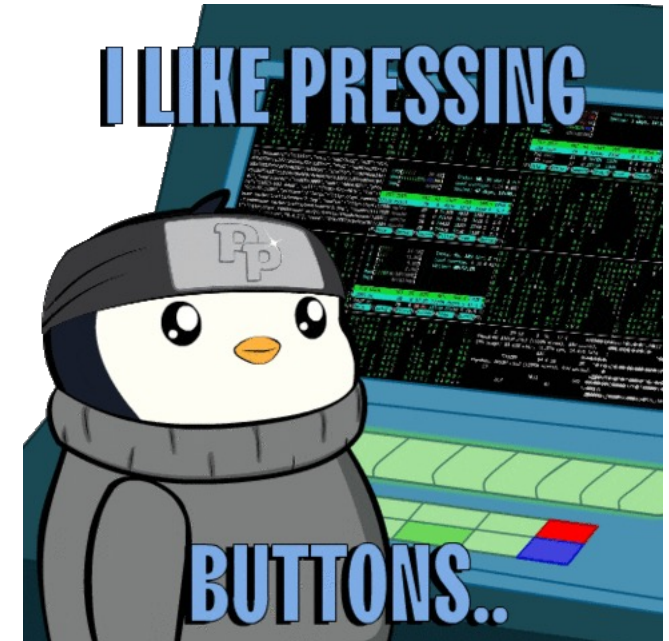
# COMPONENTES



# COMPONENTES



Los tendremos que  
desarrollar nosotros



# COMPONENTES

```
import Keyboard from 'react-simple-keyboard';  
import 'react-simple-keyboard/build/css/index.css';
```

**<react-simple-keyboard />**

```
<Keyboard  
  onPress={keyPress}  
  layout={{  
    default: [  
      "Q W E R T Y U I O P",  
      'A S D F G H J K L Ñ',  
      "{enter} Z X C V B N M {bksp}"  
    ]  
  }}  
  display={{  
    '{bksp}': '<i class="fa-solid fa-delete-left"></i>',  
    '{enter}': 'ENVIAR'  
  }}  
  theme={"hg-theme-default hg-layout-default myTheme"}  
  layoutName="default"  
  buttonTheme={buttonTheme}  
/>
```

```
let buttonTheme = [{  
  class: "hg-red",  
  buttons: letters0k  
},  
{  
  class: "hg-highlight",  
  buttons: lettersNot0k  
}  
];
```



# COMPONENTES

`<react-simple-keyboard />`

## QUIEN ES keyPress()?

El usuario ha pulsado una tecla.  
Y ahora qué?

```
const keyPress = (letter) => {}
```

# COMPONENTES

`< react-toastify />`

```
import { ToastContainer, toast } from 'react-toastify';  
import 'react-toastify/dist/ReactToastify.css';
```

```
const toastNoExiste = () =>  
  toast("La palabra no existe!", {  
    position: "top-center",  
    autoClose: 5000,  
    hideProgressBar: true,  
    closeOnClick: true,  
    pauseOnHover: true,  
    draggable: true,  
    progress: false,  
    theme: "dark",  
  });
```

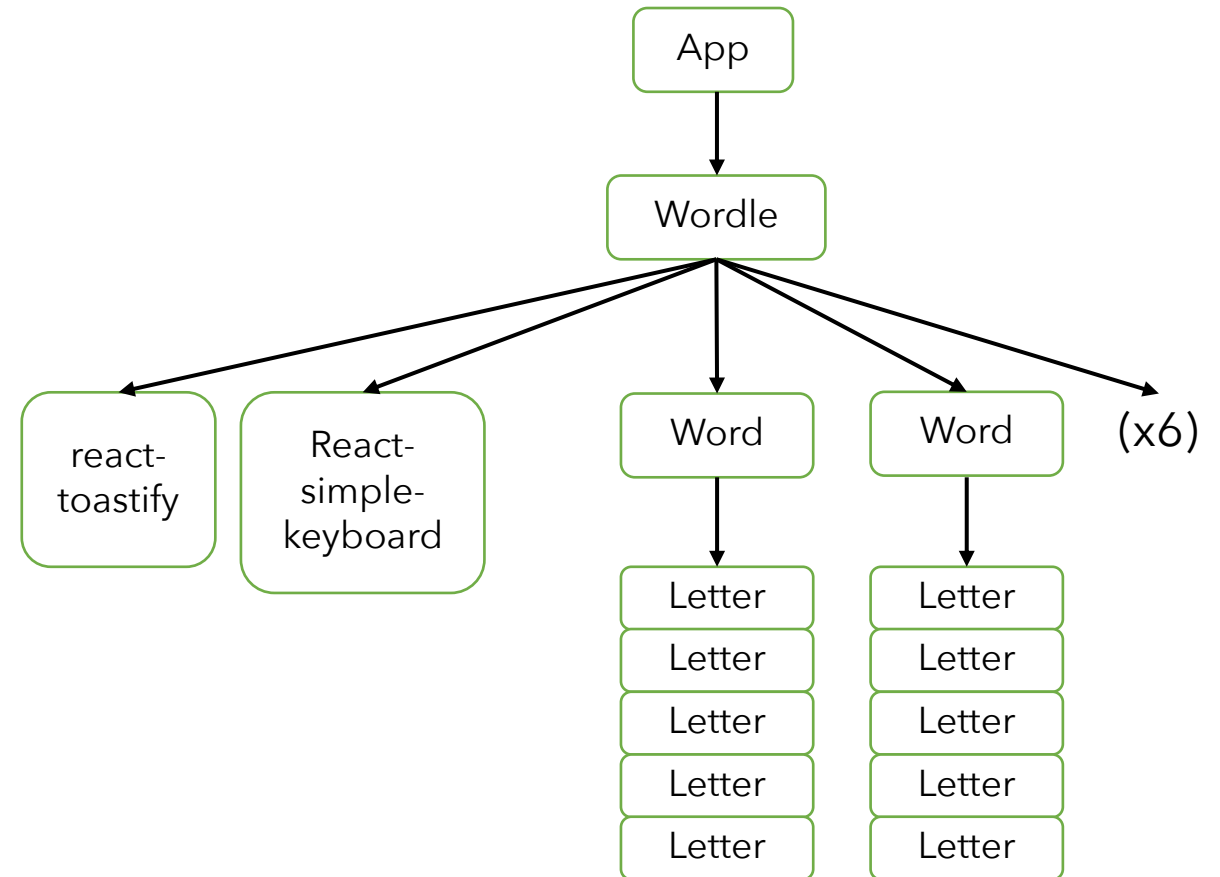
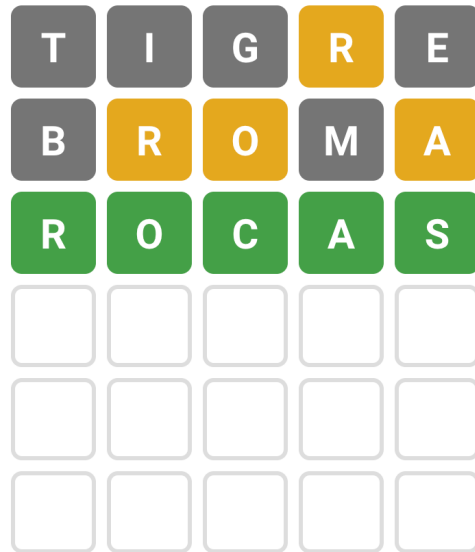
Definiremos más funciones para las notificaciones de  
"Palabra correcta!"  
"has perdido! La solución era \${solucion}"

```
<ToastContainer />
```

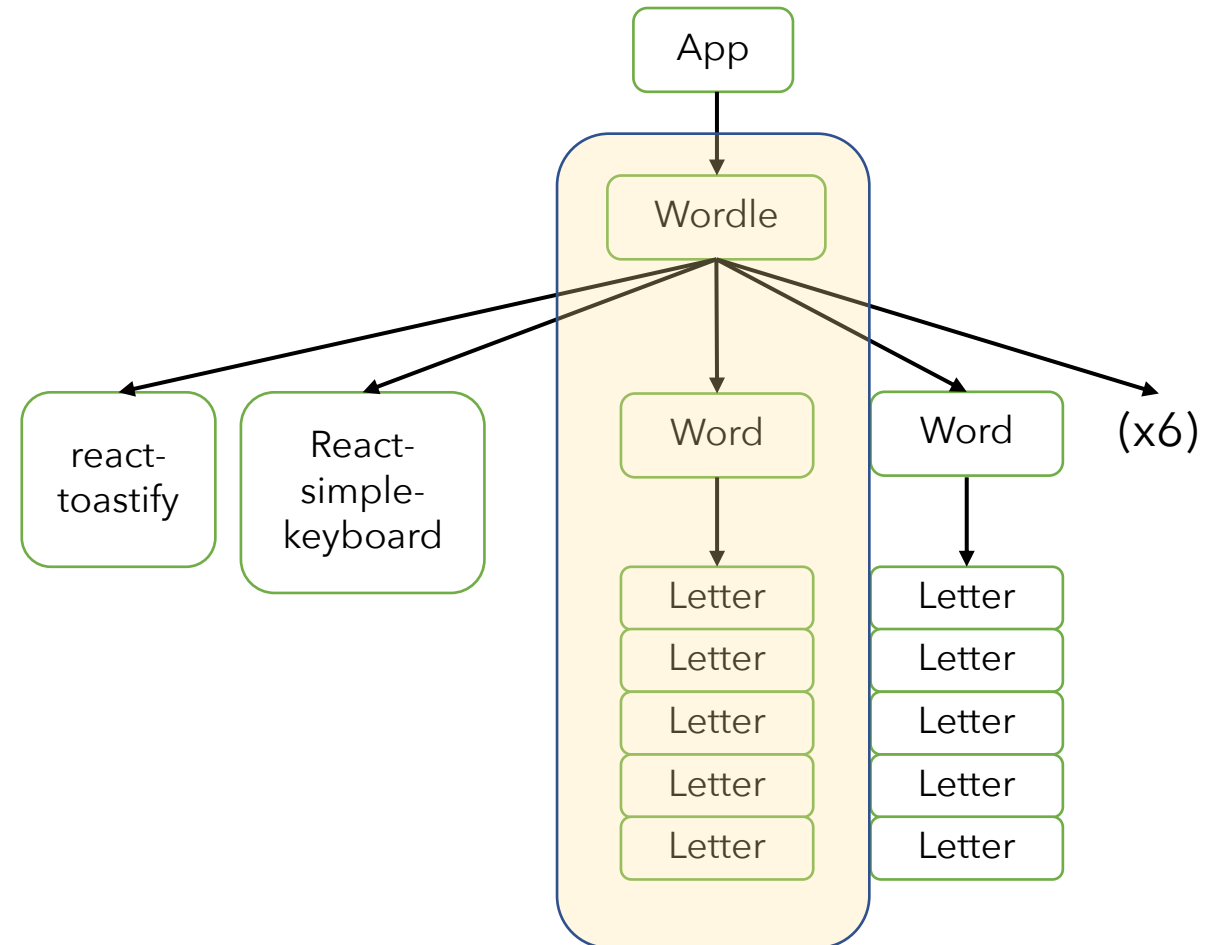
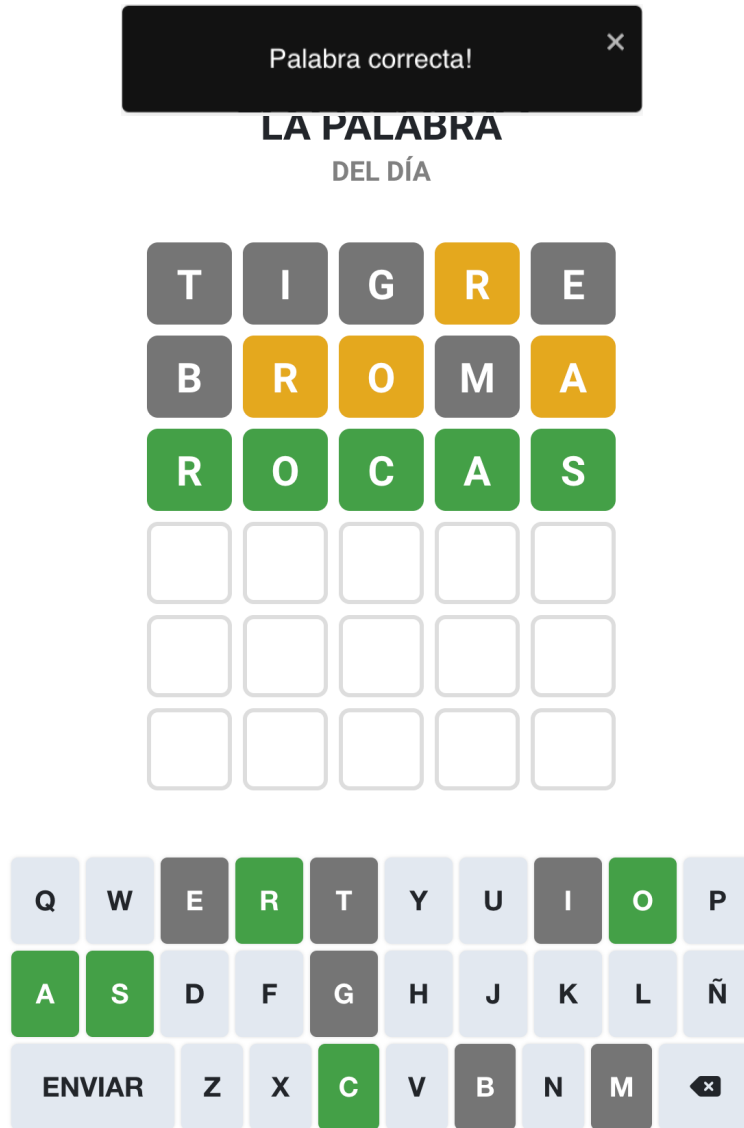
# COMPONENTES

Palabra correcta! ×

**LA PALABRA**  
DEL DÍA



# COMPONENTES



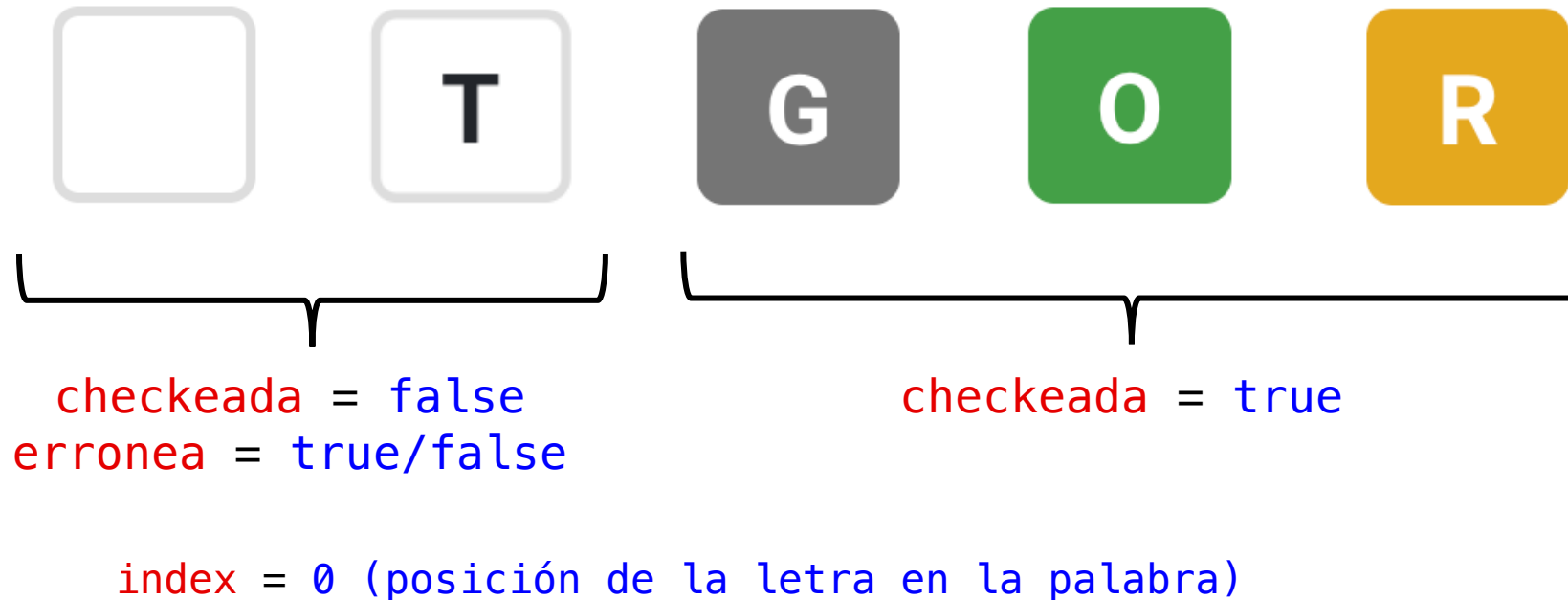
# COMPONENTES

**Letter**

# COMPONENTES

**PROPS** (Parametros de entrada)

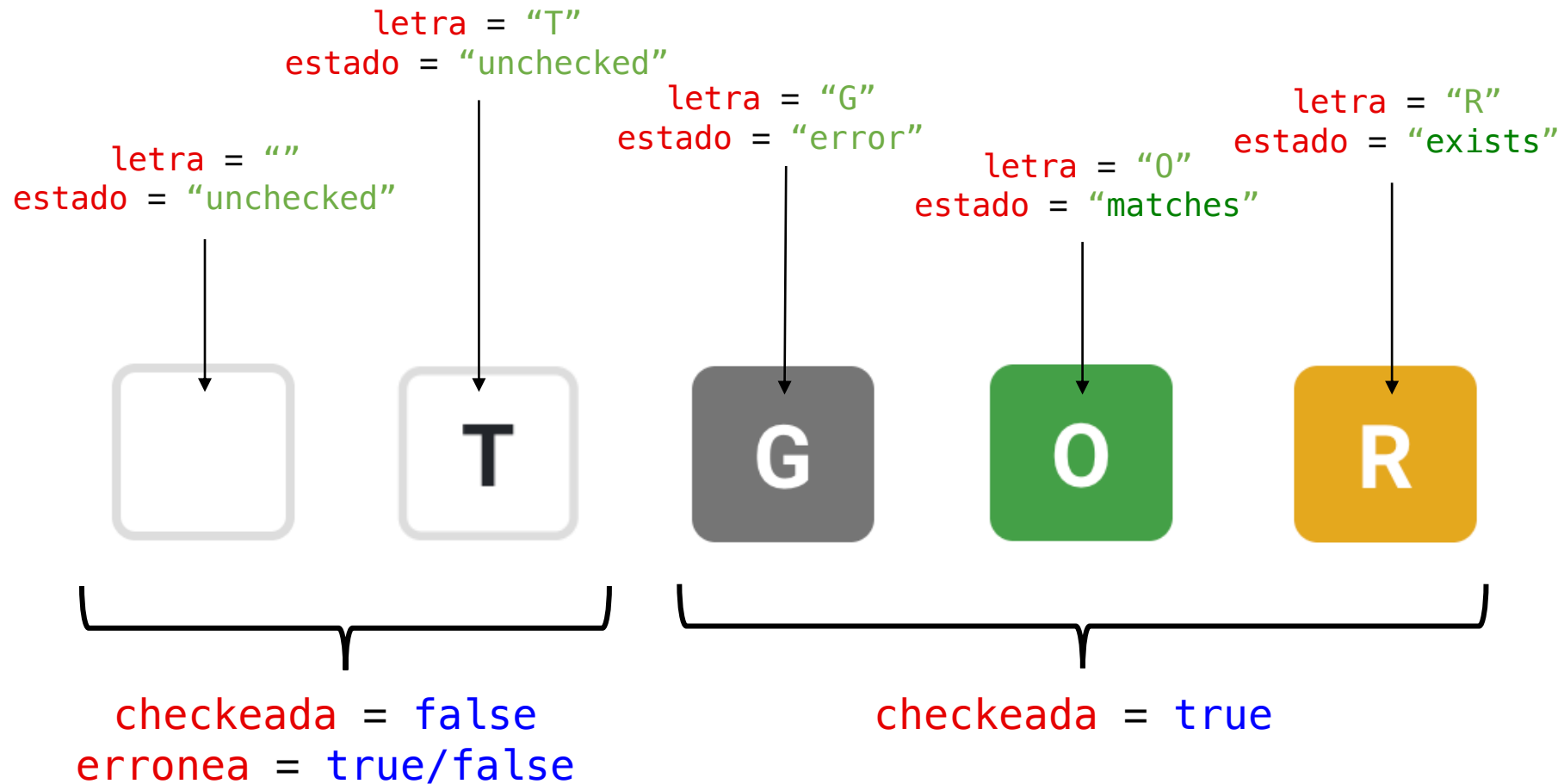
**Letter**



# COMPONENTES

**PROPS** (Parametros de entrada)

**Letter**

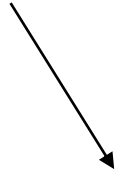


`index = 0` (posición de la letra en la palabra)

# COMPONENTES

# Letter

## PROPS



```
function Letter ({checkeada, erroneea, letra, estado, index}) {  
  
}  
  
export default Letter;
```



# COMPONENTES

Clases definidas en  
Animate.css

**Letter**

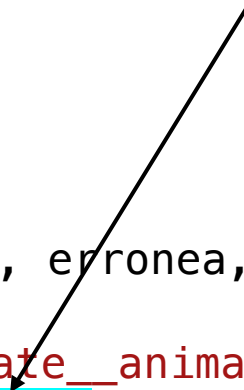
```
function Letter ({checkeada, erronea, letra, estado, index}) {  
  let classesError = "animate__animated animate__shakeY";  
  let classesCheck = `animate__animated animate__flipInX animate__delay-${index}s`;   
  let classesNoCheck = "non-animate"  
  
  return(<div className={`col letter`} >  
    {letra}  
  </div>);  
}  
  
export default Letter;
```

Clase definida  
en Bootstrap

# COMPONENTES

## Letter

unchecked  
error  
exists  
matches



```
function Letter ({checkeada, erronea, letra, estado, index}) {  
  
  let classesError = "animate__animated animate__shakeY";  
  let classesCheck = `${estado} animate__animated animate__flipInX animate__delay-${index}s`;  
  let classesNoCheck = "non-animate"  
  
  return(<div className={`col letter ${checkeada} && classesCheck} ${erronea} && classesError}  
    ${!checkeada} && !erronea && classesNoCheck`} >  
    {letra}  
  </div>);  
}  
  
export default Letter;
```



# COMPONENTES

**Letter**

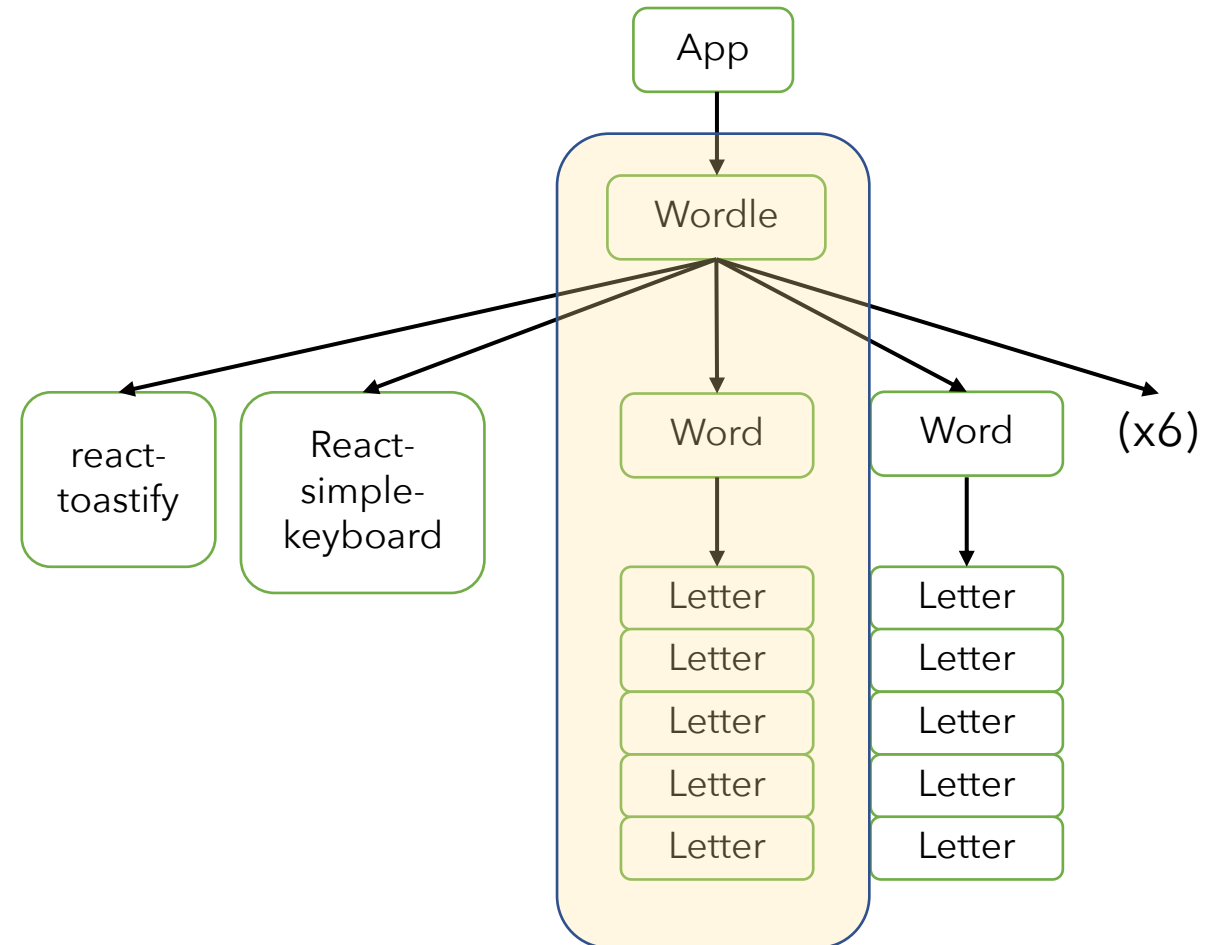
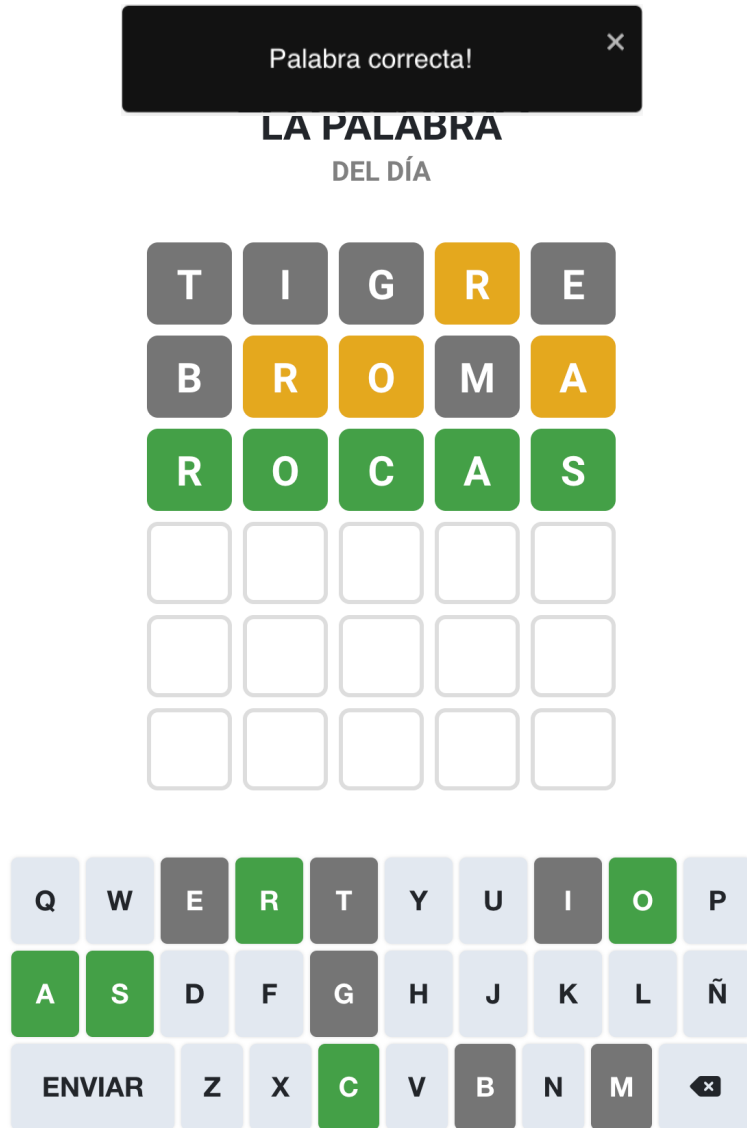
```
import Letter from "./Letter";
```

```
<Letter checkada={true} erronea={false} letra="T" estado="exists" index={0}/>
```

componente **Letter**

Las **props** se pasan al componente  
como si fueran atributos de la etiqueta html

# COMPONENTES



# COMPONENTES

**Word**



# COMPONENTES

**Word**

**PROPS** (Parametros de entrada)

```
palabra = {  
  contenido: "BROMA",  
  estados: ["error", "exists", "exists", "error", "exists"],  
  checkeada: true,  
  erronea: false  
}
```



# COMPONENTES

**Word**

```
import Letter from "../Letter";

export default function Word({palabra}) {

  let contenido = palabra.contenido;
  for(let i=contenido.length; i<5; i++) {
    contenido+=" ";
  }

  return (
    <div className="word row align-content-center">

      </div>

  )}
```



# COMPONENTES

**Word**

```
import Letter from "../Letter";

export default function Word({palabra}) {

  let contenido = palabra.contenido;
  for(let i=contenido.length; i<5; i++) {
    contenido+=" ";
  }

  return
    <div className="word row align-content-center">
      {Array.from(contenido).map((letter, k) =>

        )}
    </div>
  )}
```





# COMPONENTES

**Word**

```
import Letter from "../Letter";

export default function Word({palabra}) {

  let contenido = palabra.contenido;
  for(let i=contenido.length; i<5; i++) {
    contenido+=" ";
  }

  return
    <div className="word row align-content-center">
      {Array.from(contenido).map((letter, k) => <Letter key={k}
        checkeada={palabra.checkeada} erronea={palabra.erronea} letra={letter}
        estado={palabra.estados[k]} index={k}/>
      )}
    </div>
  )}
}
```



# COMPONENTES

**Word**

```
import Word from "./Word";

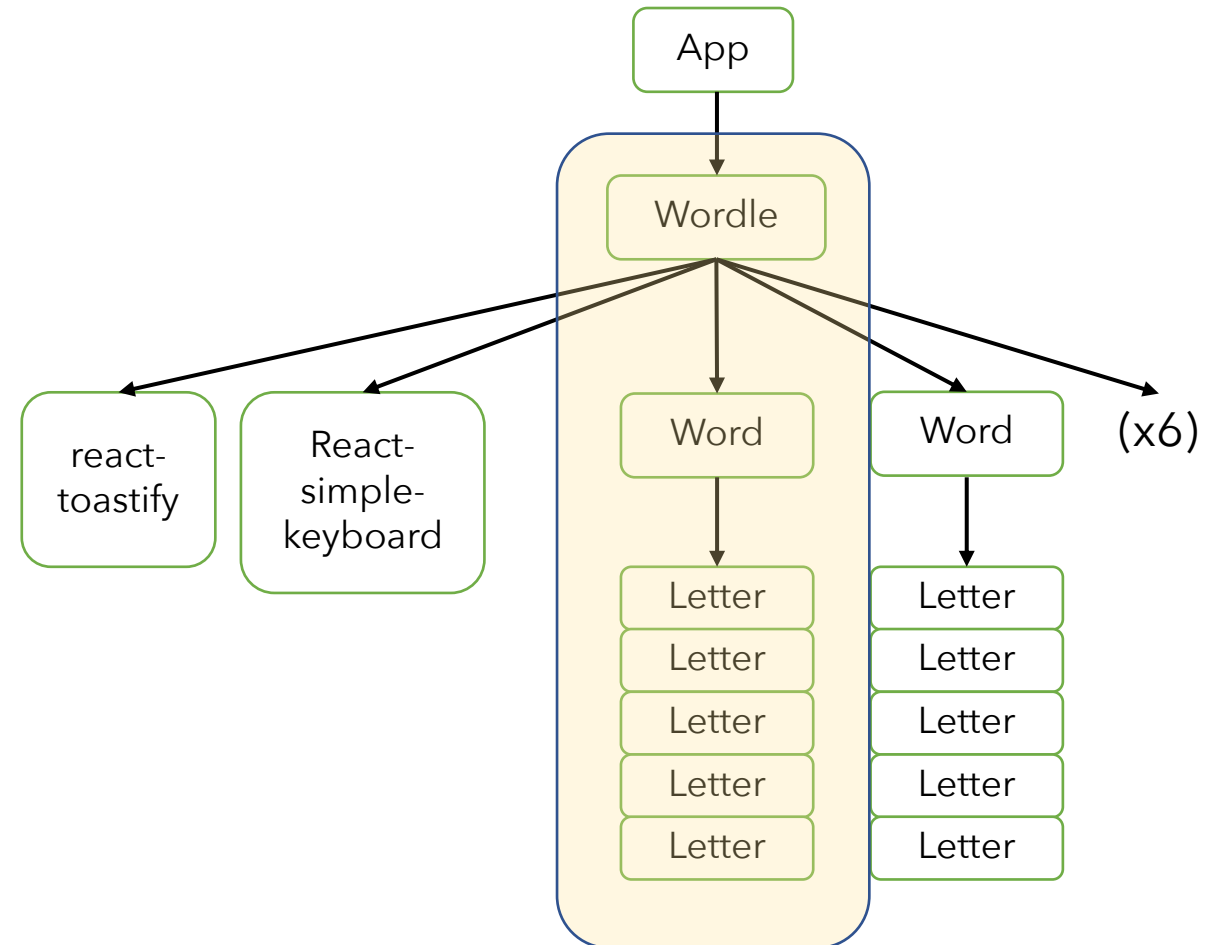
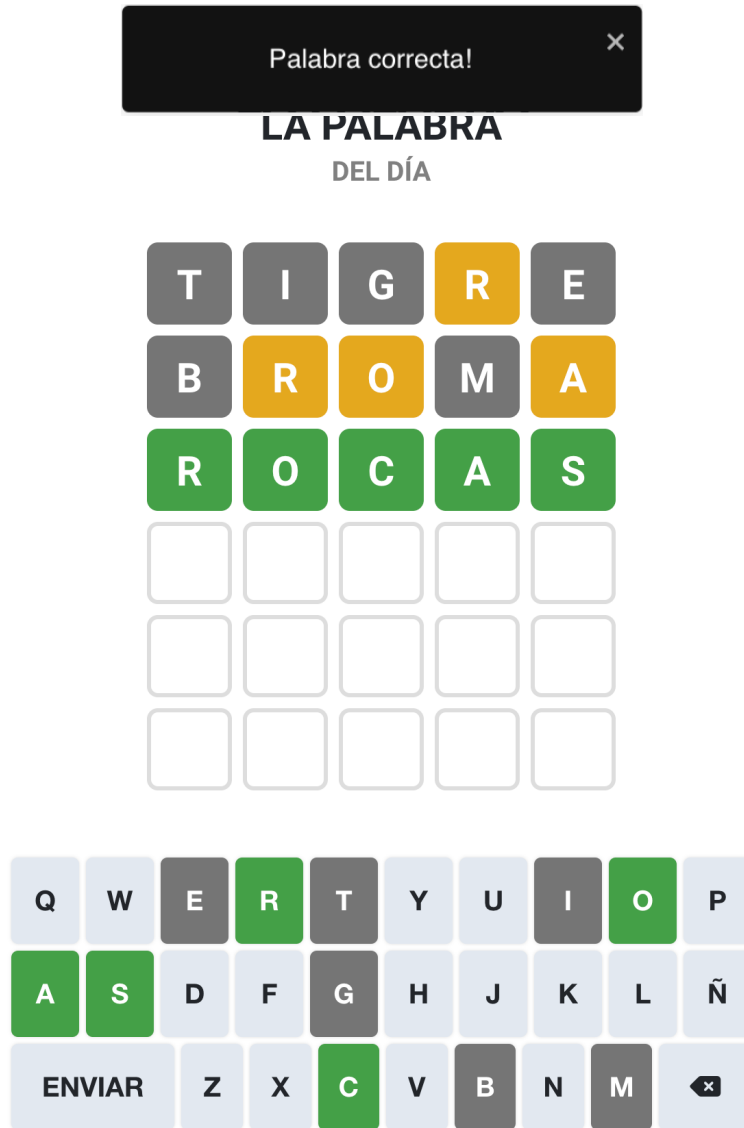
let objetoPalabra = {
  contenido: "BROMA",
  estados: ["error", "exists", "exists", "error", "exists"],
  checkeada: true,
  erronea: false
}
```

```
<Word palabra={objetoPalabra} />
```

componente **Word**

Las **props** se pasan al componente  
como si fueran atributos de la etiqueta html

# COMPONENTES



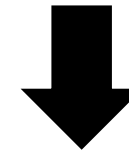
# COMPONENTES

**Wordle**

# COMPONENTES

**Wordle**

Componente Wordle manda!

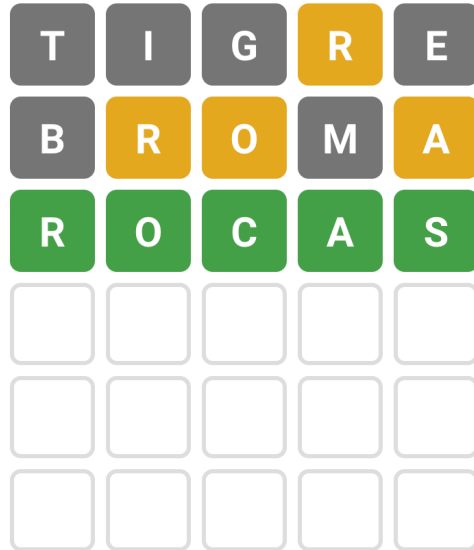


- **Stateful** component
- No necesita información externa: **NO TIENE PROPS**
- **Pasa props a cada Word** para decirle lo que tiene que pintar

(a su vez, Word pasará props a cada Letter para decirle como se tienen que pintar)

Palabra correcta! ✕

LA PALABRA  
DEL DÍA

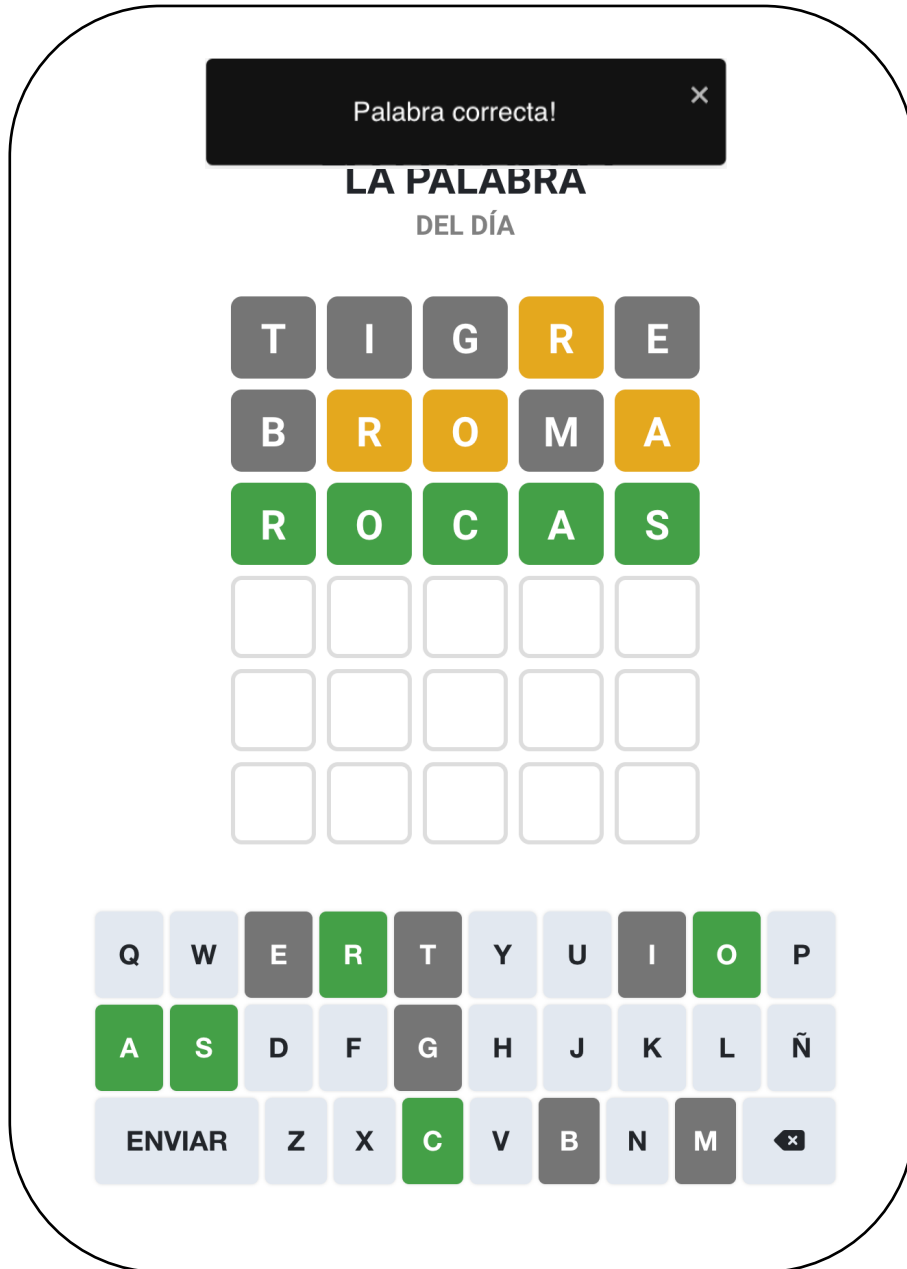


# COMPONENTES

## Wordle

Qué **estados** necesita Wordle?

- `palabras` → Array de las palabras que ha escrito y comprobado el usuario
- `solucion` → La solución a la partida
- `fin` → True/false que indica si ha terminado
- `lettersOk` → Las letras en verde del teclado
- `lettersNotOk` → Las letras en gris del teclado
- `todasPalabras*` → Un array con todas las palabras disponibles para comprobar si una palabra introducida existe



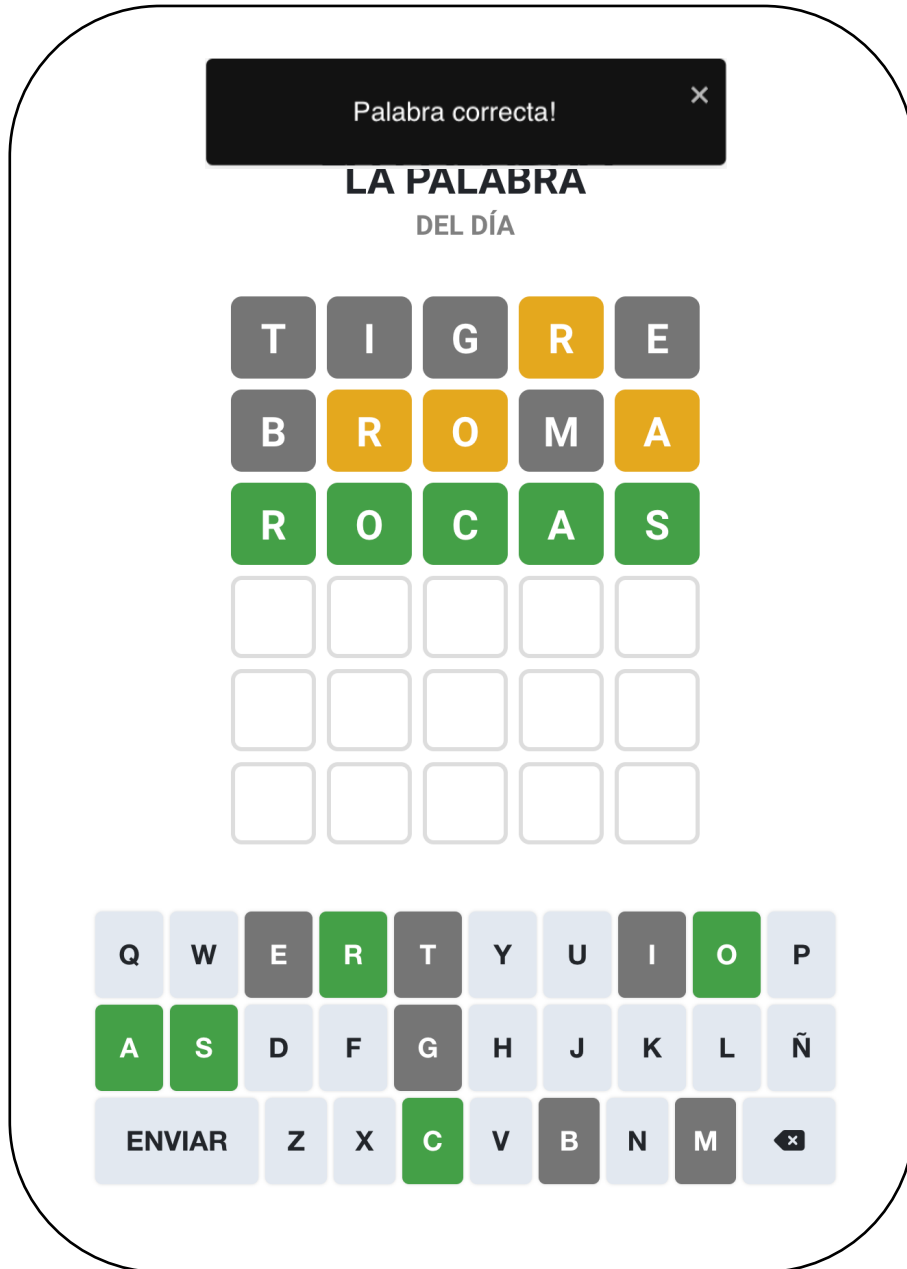
# COMPONENTES

## Wordle

`useState` es un  
hook de React para  
crear estados de  
componentes

Definición del **estado**  
interno de Wordle:

```
const [palabras, setPalabras] = useState(palabrasInicial);  
const [lettersOk, setLettersOk] = useState('');  
const [lettersNotOk, setLettersNotOk] = useState('');  
const [solucion, setSolucion] = useState('');  
const [fin, setFin] = useState(false);
```



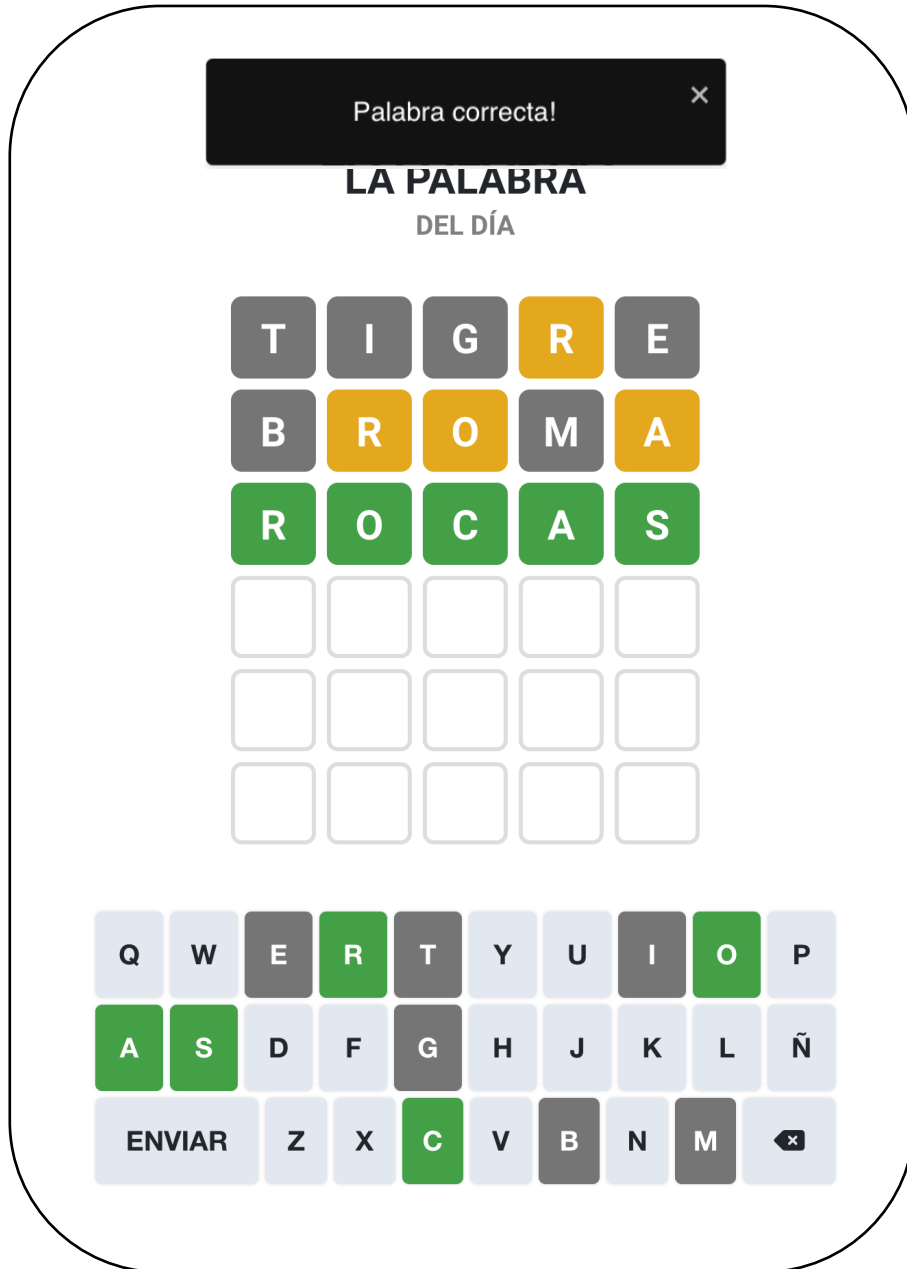
# COMPONENTES

## Wordle

`useState` es un  
hook de React para  
crear estados de  
componentes

Definición del **estado**  
interno de Wordle:

```
const [palabras, setPalabras] = useState(palabrasInicial);  
const [lettersOk, setLettersOk] = useState('');  
const [lettersNotOk, setLettersNotOk] = useState('');  
const [solucion, setSolucion] = useState('');  
const [fin, setFin] = useState(false);
```





# COMPONENTES

```
const [palabras, setPalabras] = useState(palabrasInicial);
```

<pre>{   contenido: "TIGRE", estados: ["error", "error", "error", "exists", "error"],   chequeada: true, erronea: false }</pre>
<pre>{   contenido: "BROMA", estados: ["error", "exists", "exists", "error", "exists"],   chequeada: true, erronea: false }</pre>
<pre>{   contenido: "ROCAS", estados: ["matches", "matches", "matches", "matches", "matches"],   chequeada: true, erronea: false }</pre>
<pre>{   contenido: "", estados: ["unchecked", "unchecked", "unchecked", "unchecked", "unchecked"],   chequeada: false, erronea: false }</pre>
<pre>{   contenido: "", estados: ["unchecked", "unchecked", "unchecked", "unchecked", "unchecked"],   chequeada: false, erronea: false }</pre>
<pre>{   contenido: "", estados: ["unchecked", "unchecked", "unchecked", "unchecked", "unchecked"],   chequeada: false, erronea: false }</pre>

Palabra correcta! ×

LA PALABRA  
DEL DÍA

TIGRE

BROMA

ROCAS

QWERTYUIOP

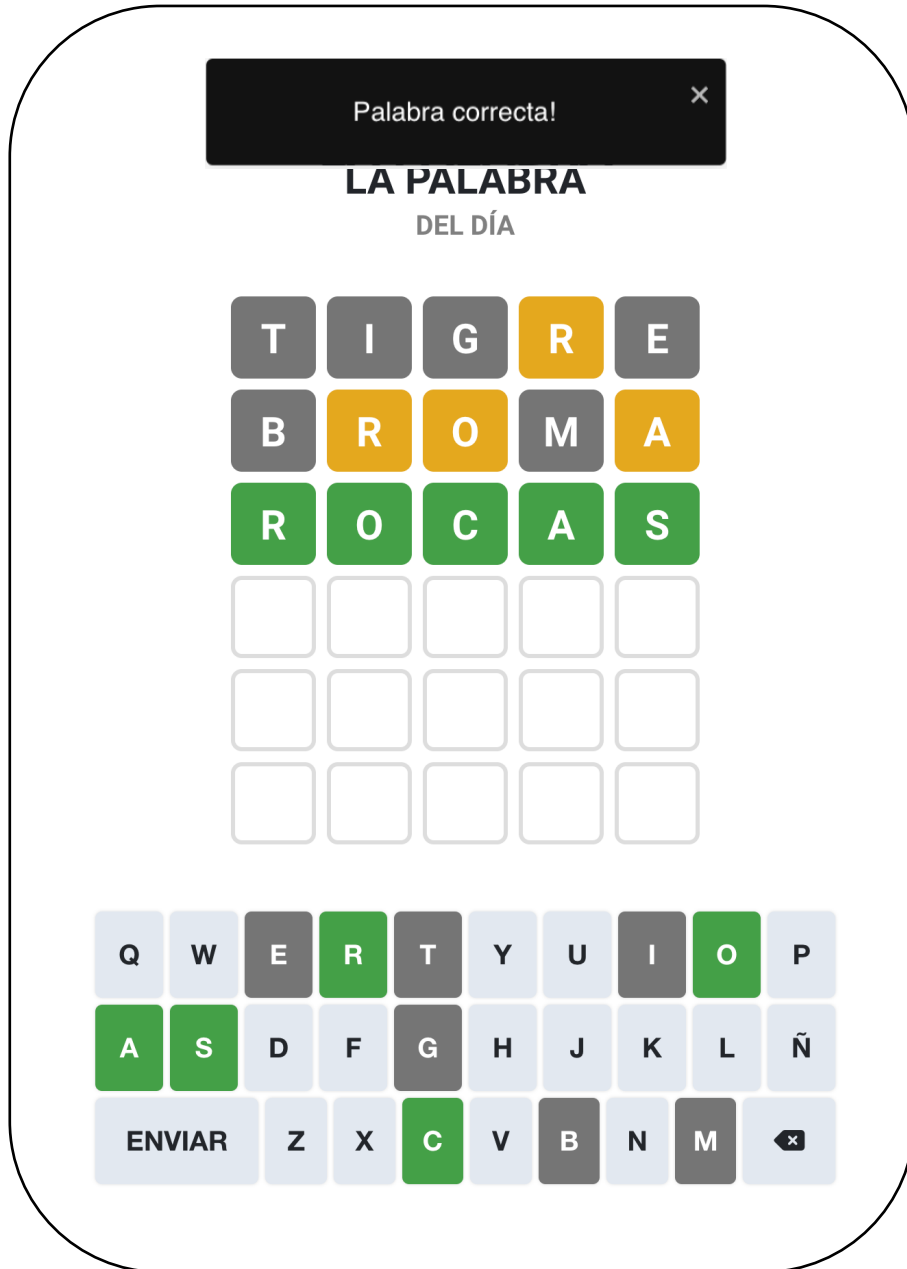
ASDFGHJKLÑ

ENVIARZXCVBNM

# COMPONENTES

```
const [palabras, setPalabras] = useState(palabrasInicial);
```

```
{palabras.map((palabra, k) => <Word  
  palabra={palabra} key={k}/>)}
```



# COMPONENTES

```
const [solucion, setSolucion] = useState('');
```

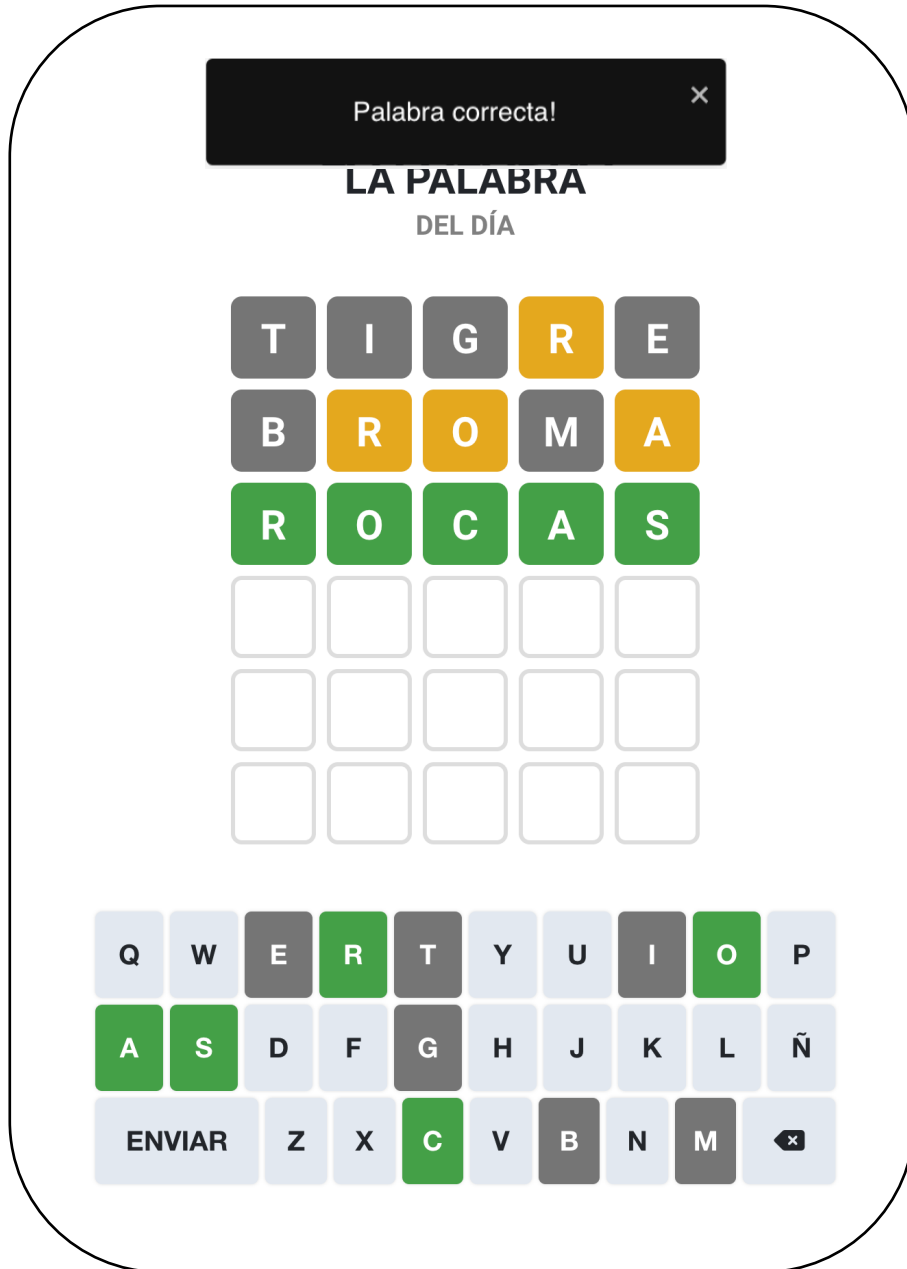
La solución se calcula e inicializa al momento de renderizar el componente.

Para comprobar si una palabra introducida es correcta.

```
const [fin, setFin] = useState(false);
```

Inicialmente es *false*.

Si se detecta una solución correcta o se han comprobado las 6 palabras posibles pasa a true.



# COMPONENTES

```
const [lettersOk, setLettersOk] = useState('');  
const [lettersNotOk, setLettersNotOk] = useState('');
```

Cuando se pulsa **ENVIAR**, chequea la letra pulsada:

**Si existe en la palabra**, la guarda en **lettersOk**.

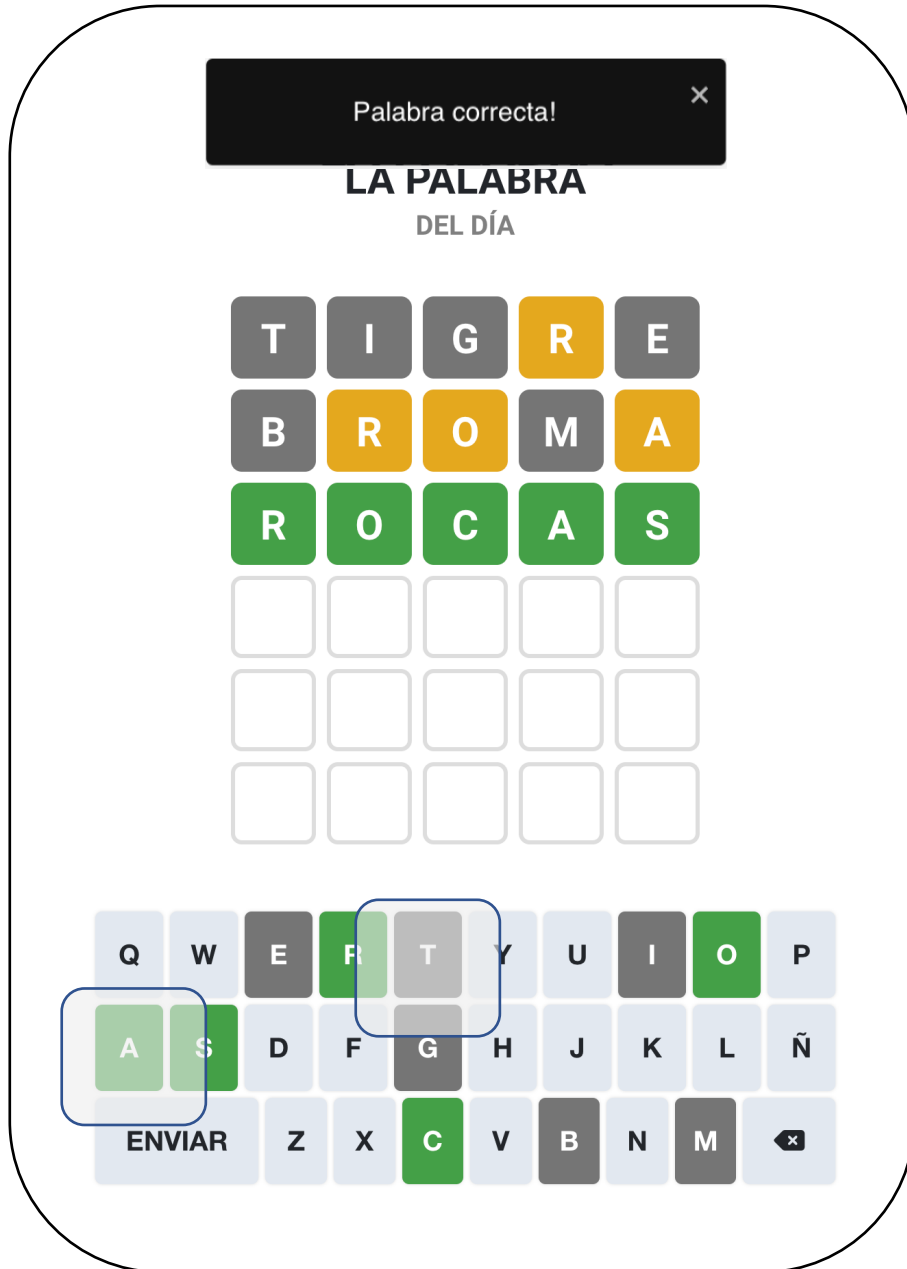
**Si no existe**, la guarda en **lettersNotOk**.

En el ejemplo:

lettersOk = "R O A S C"

lettersNotOk = "E T I G B M"

El componente teclado aplicará unos estilos diferentes a las dos listas de letras



# KEYPRESS()

```
const keyPress = (letter) => {
```

```
  if(letter === "{bksp}") {
```

```
  } else if (letter === "{enter}") {
```

```
  } else {  
    //letra
```

```
  }
```

```
}
```

# KEYPRESS

```
const keyPress = (letter) => {  
  //si se ha terminado la partida: return;  
  
  //busco el índice de la palabra actual  
  
  if(letter === "{bksp}") {  
    //actualizamos la palabra actual eliminando la ultima letra  
  } else if (letter === "{enter}") {
```

```
    } else {  
      //letra
```

```
    }
```

```
  }
```

# KEYPRESS

# KEYPRESS

```
const keyPress = (letter) => {
  //si se ha terminado la partida: return;

  //busco el índice de la palabra actual

  if(letter === "{bksp}") {
    //actualizamos la palabra actual eliminando la ultima letra
  } else if (letter === "{enter}") {

  } else {
    //letra

    //si ya hay 6 palabras checkeadas: return;
    //añadimos la letra pulsada a la palabra actual
  }
}
```



# KEYPRESS

```
const keyPress = (letter) => {
  //si se ha terminado la partida: return;

  //busco el índice de la palabra actual

  if(letter === "{bksp}") {
    //actualizamos la palabra actual eliminando la ultima letra
  } else if (letter === "{enter}") {
    //si la palabra actual tiene 5 letras:
    //si la palabra coincide con la solución:
    //ganado!
    //toastPalabraCorrecta();
    //si no coincide, seteamos:
    //contenido = ""
    //checkeada = false
    //erronia = true
    //toastNoExiste();

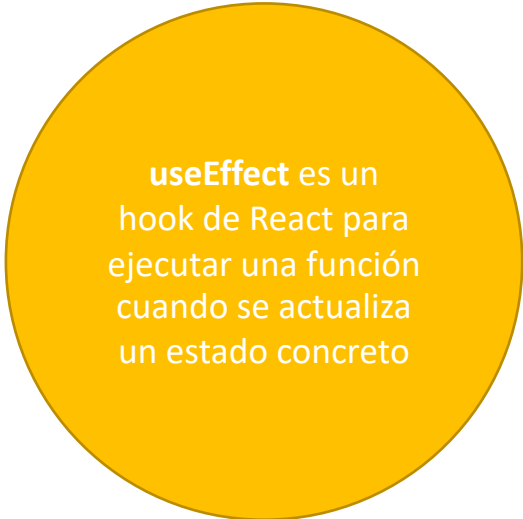
    //para cada letra de la palabra actual:
    //si letra coincide con la letra de la misma posicion en la solucion:
    //actualizamos letrasOk
    //actualizamos estados con "matches"
    //si no coincide pero está incluida
    //actualizamos letrasOk
    //actualizamos estados con "exists"
    //sino:
    //actualizamos letrasNoOk
    //actualizamos estados con "error"
  } else {
    //letra

    //si ya hay 6 palabras checkeadas: return;
    //añadimos la letra pulsada a la palabra actual
  }
}
```

**DETECTAR SI  
HA PERDIDO?**

# DETECTAR SI HA PERDIDO?

```
useEffect(()=>{  
    if(fin) return;  
  
    if(palabras.filter(palabra => palabra.checked).length === MAX_NUM_WORDS) {  
        //palabra incorrecta!  
        toastHasPerdido();  
        setFin(true);  
    }  
}, [palabras])
```



**useEffect** es un  
hook de React para  
ejecutar una función  
cuando se actualiza  
un estado concreto

**OBTENER  
PALABRAS  
DE LA API**

**URL:**

`https://random-word-api.herokuapp.com/all?lang=es`

```
const obtenerPalabras = () => {  
  
  fetch("https://random-word-api.herokuapp.com/all?lang=es")  
    .then(result => result.json())  
    .then((todasPalabras)=>{  
  
      let palabrasArr = todasPalabras.filter(palabra => {  
        return palabra.length == 5 &&      !palabra.toLowerCase().includes("á") &&  
        !palabra.toLowerCase().includes("é") &&      !palabra.toLowerCase().includes("í") &&  
        !palabra.toLowerCase().includes("ó") &&      !palabra.toLowerCase().includes("ú") &&  
        palabra.toLowerCase() == palabra  
      })  
  
      setSolucion(palabrasArr[Math.floor(Math.random()*palabrasArr.length)].toUpperCase());  
      setTodasPalabras(palabrasArr);  
  
    })  
}
```

```
useEffect(() => {  
  obtenerPalabras();  
}, []);
```

Ejecuta el acceso a la api  
solo en el primer renderizado  
del componente!

**MUCHAS  
GRACIAS**