CHECKPOINT 19

1 - ¿Qué es Heroku? ¿Existen otras alternativas? Haz una tabla si existen similares con los beneficios y desventajas.

Heroku es una plataforma de servicios en la nube que permite manejar los servidores y sus configuraciones, escalamiento y administración.

Heroku te permite desplegar tus aplicaciones en la web de forma sencilla. Entre las ventajas de usar Heroku se encuentran:

- Su facilidad de uso, simplemente tienes que decirle que lenguaje de backend o base de datos vas a utilizar y solo te tienes que preocupar por el desarrollo de tu app
- Su capacidad de soportar múltiples lenguajes de programación como Node.js, Ruby, Java, PHP, Python, Go, Scala y Clojure.

Al haberse vuelto una opción de pago, hace que otras alternativas sean una mejor opción para todas aquellas personas que se estén iniciando en el mundo de la programación y no quieran tener que invertir dinero para el desarrollo de sus aplicaciones al inicio. Entre las mejores alternativas a Heroku para el despliegue de aplicaciones web podemos encontrar Render, Vercel y Netlify. Todas ellas comparten una serie de ventajas, como las que se muestran a continuación, que hacen que las tres opciones se presenten como una alternativa a Heroku:

- Facilidad de uso.
- Admisión de múltiples lenguajes de programación.
- El pricing empieza en 0€, permitiendo desplegar servicios web de forma gratuita.
- Permite actualizaciones automáticas desde el repo de GitHub que contiene tu web.

Algo a considerar en cualquiera de las tres opciones es que los planes gratuitos que ofrecen limitan el uso que ofrecen, haciendo de pago los servicios más avanzados, pero como una forma de iniciarse en el mundo de la programación las tres son alternativas a considerar.

2 - ¿Qué es Redux? Qué beneficios tiene, cuando lo utilizarás, problemas comunes (riesgos).

Redux es una librería de JavaScript para el manejo del estado de las aplicaciones. Normalmente se utiliza junto con otras librerías como React o Angular para la construcción de interfaces de usuario.

En el desarrollo de aplicaciones sencillas, el manejo del estado no suele ser un problema puesto que se puede almacenar en las propiedades (props) de los componentes. La gestión de estado se vuelve un problema cuando las aplicaciones empiezan a crecer.

Utilizando **Redux**, el **estado** de tu aplicación está almacenado en un **árbol** dentro de un único **store**. La forma de poder cambiar el estado del árbol es emitiendo una **acción**. Una acción es un objeto en el que se describe que es lo que ha ocurrido. Para indicar la forma en la que una acción transforma el árbol se utilizan **reducers**.

Un **reducer** es una función con el siguiente formato **(estado, acción) => nuevoEstado**

Es importante aclarar que utilizando Redux no se modifica el estado de la aplicación directamente, sino que se especifican los cambios que queremos que ocurran a través de la emisión de acciones. Utilizando reducers, se decide la forma en la que cada acción transforma el estado de la aplicación.

A continuación, se muestra un ejemplo que explica lo que acabamos de ver:

1) Importamos createStore de Redux:

```
import { createStore } from “redux”;
```

2) Creamos un store de Redux para almacenar el estado de la app:

```
let store = createStore(sum);
```

3) Especificamos con un reducer la forma en la que la acción va a transformar el estado de la aplicación:

```
function sum(state = 0, action) {
        switch (action.type) {
        case “INCREMENT”:
                return state +1;
        case “DECREMENT”;
                return state -1;
        default:
                return state;
               }
        }
```

4) Despachamos una acción especificando la forma en la que se va a modificar el estado:

```
store.dispatch( { type: “INCREMENT” } );
```

3 - ¿Qué es un HOC? Explicar qué es, qué beneficios tiene, cuando lo utilizarás, por qué lo utilizarás, problemas comunes (riesgos).

HOC son las iniciales que corresponden a Higher-Order Component. A diferencia de un componente normal, que transforma props en una interfaz de usuario, un HOC transforma un componente en otro componente.

La misión del HOC es transformar el componente que se toma como argumento, normalmente a través de las props, en un nuevo componente.

El principal beneficio de los HOC es que permiten reutilizar una misma lógica entre componentes. Cuando tienes diferentes componentes que comparten varias funcionalidades comunes, en vez de duplicar esa lógica en cada componente, puedes crear un HOC que recoja esa lógica y exportarlo junto con los componentes que quieres que tengan acceso a sus propiedades.

El uso de HOC evita la duplicidad de código en tus aplicaciones y favorece su mantenimiento al tener la lógica separada en una función y solo tenga una razón para cambiar, si bien, podemos tender a incrementar demasiado nuestro árbol de componentes a medida que la complejidad de nuestra aplicación va creciendo ya que estamos utilizando composición. Derivado de lo anterior, hay que saber identificar los casos en los que el uso de HOC es apropiado para que no se convierta en un inconveniente.