MyRealFood - Web

En esta prueba se pretende comprobar los conocimientos del candidato con tecnologías web (ReactJS) y otros conocimientos técnicos.

Plazo

El plazo de la realización y entrega de la prueba es de 7 días desde el momento del envío de la misma.

Seleccionar una:

A) Utilizando ReactJS y Firebase (Auth y Cloud Firestore), realizar una web que permita a un usuario registrarse, iniciar sesión y añadir y visualizar su propia lista de tareas con título y descripción. El usuario se tiene que registrar y autenticar utilizando Firebase Auth y los datos de las tareas se tienen que almacenar utilizando Firebase Cloud Firestore.

B) Utilizando ReactJS y la API JSON de OpenFoodFacts, realizar una web que permita buscar la información de un producto mediante su código de barras y se visualice en ella la foto, el nombre, la marca, el listado de ingredientes y la información nutricional. Tiene que contener además una sección de historial con los productos buscados anteriormente por el usuario.

El código de la prueba puede entregarse utilizando uno de los siguientes métodos:

```
> Un repositorio git/svn invitando al usuario develop@myrealfood.app
> Fichero comprimido con todo lo necesario para su ejecución
```

Para la ejecución de la misma, debe de ser suficiente con ejecutar dentro del directorio de trabajo los comandos:

```
$ npm install
$ npm start
```

Responde a las siguientes preguntas

Define con tus palabras los siguientes conceptos y para qué sirven en React: propiedades, estados, webhooks, JSX y cuál es el ciclo de vida de un componente.

Las **propiedades** (o más conocidas como props) almacenan información que fuerzan a los componentes a renderizarse a sí mismos. Con ello React revisa como han cambiado dichos componentes (en cuanto a propiedades, estilos, estados, etc...) mediante el intercambio de información entre ellos.

Los **estados** son también propiedades que, cuando éstos cambian, el componente (y sus componentes hijos) que usan esa propiedad en concreto se re-renderizan también. Los más comunes son: useState, useEffect, useRef, useContext, ...

Hooks es nuevo desde la versión 16.8+ de React y reemplaza los "class components" con componentes funcionales, también pueden tener estado. Gracias a Hooks se hace más fácil leer, usar y entender el concepto de estado (o state). Un Hook puede, por ejemplo, manejar un websocket (una conexión entre cliente y servidor) y usarlo en cada componente.

JSX es una "forma" de escribir código propia de ReactJS, pero que básicamente viene a ser HTML customizado + JavaScript. Originalmente he leído que viene de PHP. Esta forma que tiene React de usar JSX me parece una muy buena idea a la hora de facilitar al programador el escribir su código.

El **ciclo de vida de un componente** de React (con Hooks) depende, y puede ser influenciado y tratado con "useEffect", que es el hook que maneja los side-effects en componentes funcionales cuando se renderizan por primera vez y cada vez que estos se van actualizando. Con ello podremos decidir cuantas veces se renderizará un componente o si por lo contrario sólo queremos que lo haga una única vez.

¿Tienes experiencia creando un proyecto web desde cero (A) utilizando tu propio servidor (B) utilizando servicios en la nube como AWS o Firebase? En caso afirmativo describe brevemente el entorno que has utilizado (tipo de servidor, servicios instalados/utilizados, lenguajes, bases de datos, etc)

Experiencia con React no. He hecho proyectos propios con React desde cero como parte del aprendizage.

No he usado AWS ni Firebase, pero se que son plataformas creadas por Amazon (AWS) y por Google (Firebase) que proporcionan servicios en la nuve y que facilitan la creación de aplicaciones, tanto web como móbil, de una forma fácil y rápida, sobretodo para las empresas.

Te encargan construir una API siguiendo el estándar RESTful, utilizando JSON para el intercambio de datos que permita crear, modificar, eliminar y acceder a los datos de un usuario. Describe las rutas, los métodos (GET, POST, …), los códigos de estado (2xx, 3xx, 4xx, etc) y el JSON que se enviaría y recibiría como respuesta en cada caso.

Usaría el paquete de Express, que es un framework de aplicaciones web de Node.js para trabajar en el lado del servidor.
Este permite el uso de bases de datos no SQL como MongoDB y/o manejar fácilmente rutas de una API. Además de ser uno de los framewords JS más populares hasta la fecha.

Las rutas son las que dan la información de respuesta del servidor mediante la solicitud de la API y estas estan divididas en un método HTTP y una ruta (GET, POST, PUT, DELETE, …).

Estas serían algo así como:

router.get("/ticket, (req, res, next) => {… código para **obtener** data …})
router.post("/ticket, (req, res, next) => {… código para **crear** data …})
router.put("/ticket/:id, (req, res, next) => {… código para **actualizar** data …})
router.delete("/ticket/:id, (req, res, next) => {… código para **eliminar** data …})

Los códigos de estado són respuestas de la API a la petición de obtener la data.
Hay de 5 tipos:
1XX (100-199): Respuestas informativas.
2XX (200-299): Respuestas satisfactorias.
3XX (300-399): Redirecciones.
4XX (400-499): Errores de los clientes.
5XX (500-599): Errores de los servidores.

Un ejemplo puede ser:
res.status(200).json({mensaje: "Has tenido éxito", error: "xyz"})
res.status(404).json({mensaje: "Búsqueda no encontrada", error: "xyz"})

Imagina que tienes que optimizar el tiempo de carga de una web que utiliza ReactJS que tiene cierto contenido estático y cierto contenido dinámico. ¿Qué estrategias de mejora del tiempo de carga y de rendimiento se te ocurren? ¿Dónde irías a mirar o que implementarías para intentar reducir estos tiempos?

El lazy loading sería una forma para manejar lo que tarda la web en ser cargada (o en recibir data), que mediante la carga asíncrona ese componente sólo será cargado cuando sea necesario y no en cada petición de manera que se requiera más tiempo de carga para cargar partes de código innecesarias.

También se podría usar Hooks como useMemo(), que parecido a useEffect() permite memorizar en caché una gran cantidad de información, siempre y cuando no varien sus dependencias.

Otra manera extra de visualizar todo el comportamiento de la app sería instalando en el navegador la extensión React Developer Tools, esta te permite visualizar que actualizaciones se están realizando innecesariamente en los componentes marcandolas en diferentes colores (rojo, amarillo, azul o verde) según la frecuencia de re-renderizado.
De esta manera, el desarrollador podría prevenir y mejorar tiempos de carga excesivos con la ayuda de esta extensión.

También, con la própia herramienta para desarrolladores del navegador que se esté utilizando se pueden manejar los cambios que occuren en la app.
Con esto se pueden detectar problemas de carga viendo su tiempo de ejecución, igual que poder inspeccionar que partes del código se están ejecutando en cada petición de forma innecesaria y así poder optimizar, en este caso, los tiempo de la web.

Imagina que tienes que posicionar una página web en lo más alto de los diferentes motores de búsqueda. ¿Qué acciones llevarías a cabo para conseguirlo?

Usaría SEO con React con Next.js y con el paquete “react-helmet”, ambos utilizan SSR (Server Side Rendering), que lo que hace es actualizar el HTML en el servidor en cada petición de la aplicación. De esta manera podemos controlar los metadatos del <head> e ir actualizandolos constantemente. Sinceramente, he oído hablar de ellos pero no los he utilizado.

Contacto

Si tienes algún problema a lo largo de la realización de la prueba, puedes contactar a *david.vicente@myrealfood.app* y te responderé con la mayor brevedad posible.