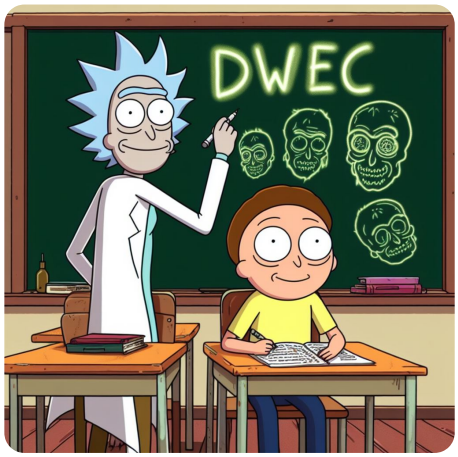
Examen de React 2024



Se pretende evaluar a través de esta prueba práctica los conocimientos adquiridos a lo largo del curso 23_24 en el módulo de DWEC. Para ello, los dos primeros ejercicios se realizarán en una página diferente (PaginaEj1.jsx, PaginaEj2.jsx, ...) y dentro de cada página tantos componentes como consideréis oportunos.

Preparación del examen. (**0,5p**)

Crea una colección en tu base de datos de Firebase llamada "ExamenReact-[tu_nombre]". Dentro de esta colección, cada documento representará una búsqueda realizada y contendrá la siguiente información:

id: Identificador único del documento.

URL: URL de la API utilizada para la búsqueda.

imagenes: Un array de strings que contiene las URLs de las imágenes obtenidas durante la búsqueda.

favoritas: Un array que indica, mediante booleanos por ejemplo, si cada imagen en el array imagenes ha sido marcada como favorita o no. La posición en el array favoritas corresponde a la posición de la imagen correspondiente en el array `imagenes`.

Con esta estructura, cada documento en la colección **"ExamenReact-[tu_nombre]"** representará una búsqueda específica, y podrás almacenar las URLs de las imágenes obtenidas durante esa búsqueda, así como marcar qué imágenes son favoritas o no.

Ejercicio 1 - Inicio Aplicación (**2p**)

- Nuestra aplicación comienza en la ruta `/home` , mostrando una página donde a través de un simple formulario podremos insertar una **URL** y pulsar el botón de **Buscar**.

  > El campo URL es obligatorio Usaremos la librería sweetalert2 para la gestión de información y errores al usuario.

- Seguidamente **si la URL NO aparece** dentro de nuestra colección de Firebase ExamenReact-tunombre, me preguntará si deseo Añadir dicha URL.

  > Si pulsamos en No, limpiará el formulario y volveremos a la RUTA `/home` Si pulsamos en Si continuaremos.

- En caso contrario, es decir, **Si la URL SÍ aparece** dentro de nuestra colección, entonces cargaremos las imágenes con los indicadores de favoritos almacenados en Firebase y asociados a esa URL.

- La ruta `/home` estará protegida hasta que realicemos una búsqueda, momento en el que ya podremos acceder a la ruta del ejercicio2 `/gallery`.

Ejercicio 2 - Galería de Imágenes (**6p**)

Nuestra aplicación continúa con la ruta `/gallery` . Dicha ruta renderiza la *PaginaEj2* con los siguientes componentes:

a) ***Uso del hook personalizado:***

Crear un ***Custom Hook*** llamado `useJsonImageUrl` que tome una URL de una API (que devuelve un JSON con información) y busca en las claves cuyo nombre sea ***"image"***, su contenido, quedándonos con aquellas imágenes con extensión jpg o png.

Se va a suponer y por tanto considerar, que las claves con nombre "image" contienen las urls de imagenes.

Dicho hook permitirá extraer todas las URLs de imágenes con la extensión indicada que haya dentro del fichero JSON devuelto por la API especificada en la URL.

NOTA: *Para quien lo quiera hacer: crear el custom hooks sin saber en qué clave están las imágenes, sólo recorriendo recursivamente y sacando las urls de imágenes que finalizan por jpg o png.*

> **Requerimientos:**
>
> - El hook debe aceptar una URL y una cadena de texto con el tipo de extensión a buscar.
> - El hook debe realizar una solicitud a la URL proporcionada y extraer todas las **URLs de las imágenes** con la extensión indicada del JSON devuelto, gestionando los posibles errores que pudiese encontrar.
> - El hook debe devolver un array que contenga todas las URLs de las imágenes encontradas.
> - El array devuelto no puede contener imágenes repetidas.
> - También debe devolver el número de imágenes obtenidas.
> - El hook debe limpiar los efectos secundarios cuando el componente se desmonte o cuando cambie la URL si fuese necesario.
> - Para simplificar el ejercicio, devolver la data que contiene los datos no todo el JSON.

b) ***Implementación de un contexto global:*** :

Utiliza un **estado global** para gestionar el estado de las imágenes obtenidas de la URL en tu aplicación. Asegúrate de que todas las acciones relacionadas con las imágenes, como la marcación de una imagen como favorita, etc, se realicen a través del estado global.

c) ***Implementación del componente de galería:*** :

Crea un componente llamado `ImageGallery` que renderice una galería de imágenes en tu aplicación.

d) ***Mostrar imágenes en la galería:***

Utiliza las URLs de las imágenes obtenidas del hook y renderiza las imágenes en la galería a través de un componente llamado `imageCard`. Este componente además de contener la imagen ha de contener un **número de imagen.**

e) ***Convertir una imagen en Favorita:***

Cuando pulsemos doble click sobre una imagen colocaremos una **estrella** sobre la imagen, indicando que dicha imagen es favorita. La asignación de favorita o no favorita supone añadir dicha imagen como favorita dentro del array `favImgs` creado para esa labor en firebase. Lógicamente otro doble click quita la estrella asignada a dicha imagen.

f) ***Estilo de la galería:***

> Estiliza tu galería de imágenes para que se muestren **6 imágenes a la vez** (dos filas de tres imágenes).
>
> Además debe de haber dos botones, uno para **Avanzar** o para **Retroceder** hacia atrás dentro del componente ImageGallery para recorrer las imágenes

g) ***Validar la funcionalidad con la API RickandMorty:***

Para probar el funcionamiento de este ejercicio se debe hacer uso de la api de RickandMortyApi a través de la siguiente URL: Rick and Morty Api character

Ejercicio 3 - Componente Header (**1p**)

Crear un componente Header que contenga:

- El Título: **Examen DWEC y tu nombre**.
- En la parte derecha mostraremos la ***URL*** de la API que estamos usando.
- Un ***contador*** con el número de imágenes que hemos encontrado en esa api con la extensión indicada.
- Un ***contador*** con el número de imágenes Favoritas que tenemos.
- Botón para ***Comenzar de nuevo*** que permita enviarnos de nuevo a la ruta de home y comenzar de nuevo el proceso.

> **Restricciones:**

> La gestión y propagación de los datos de la cabecera se realizará a traves de un estado global.

Restricciones OBLIGATORIAS (**0,5p**)

- Se debe de utilizar React Router DOM para la gestión de rutas y errores.
- Las cargas de datos han de mostrar un Spinner.
- El acceso a rutas no permitidas mostrará una pantalla de errores.
- Si accedemos a una API que ya tengamos almacenada, las imágenes favoritas han de tener su estrella que las identifique.
- Para el icono de estrella se usará `fontawesome` cuya url es: https://fontawesome.com/v5/docs/web/use-with/react

---

NOTAS:

- Enlace a los Tipos de datos en Firebase
- No olvides incluir TODAS tus credenciales para poder probar el ejercicio al corregirlo.
- Enlace a la documentación de Firebase https://firebase.google.com/docs/firestore/
- Este examen evalúa todos los RA de este segundo trimestre centrados en el Framework de Javascript **REACT**
- El examen se comprimirá (sin la carpeta node_modules) y se enviará a la entrega correspondiente del Examen.