Módulo 3 – Routing

# ¿Qué es?

Slide 2 - 3

Es la forma en la que el usuario puede desplazarse o visitar diferentes secciones de una aplicación web. Básicamente es una asociación de un URL, con algún componente de ReactJS. Estas URLs pueden ser:

* Estáticas, cuando no se espera que algún path de la URL cambie, por ejemplo “/user/settings”
* Dinámicas, cuando la URL contiene paths que pueden cambiar, generalmente con el objetivo de especificar la carga de algún recurso, por ejemplo: “event/view/:eventId” el cual podría ser “event/view/1” o “event/view/4”

# React Router

Slides 4 - 5

Es una librería de Routing para React, que nos permite definir rutas y los componentes asignados a las mismas.

Además, también podemos features adicionales como:

* Loader, funciones para cargar data al entrar a una ruta
* Actions, funciones que se disparan desde un formulario
* Error boundary, componente para customizar las pantallas en caso de errores, ya sea al renderizar, en la carga de data o al ejecutar un action.
* Nesting routes

# Configurando rutas

Slide 6 -

Definiremos un nuevo componente para que sea nuestra pantalla de inicio. Con un botón “Ver usuarios” que nos dirigirá a la ruta “/users”. En esto usaremos los features:

* Estructura de carpetas con buenas prácticas. Definir carpeta pages, componentes, api.
* Loader, para cargar la lista de usuarios desde una primer.
* Hook de estado de la ruta, para saber si esta o no cargando la data.

Pasos:

1. npm i react-router-dom
2. Crear carpeta “pages” dentro de la carpeta “src”.
3. Crear la carpeta “home” y el archivo index.tsx dentro de ella.
4. Crear el componente Layout.tsx
5. Conectar el nuevo componente con el router para que se muestre en la ruta “/” y poner como layout el nuevo componente.
6. Comprobar que la ruta “home” esté mostrando el componente nuevo.
7. Agregar un botón sencillo con el texto “Ver usuarios”.
8. Crear la carpeta “types” para definir el type de “User”.
9. Crear la carpeta api y crear una función que simular la carga de los usuarios desde el Array, usar “defer” para estado de carga.
10. Crear la carpeta “users” dentro de la carpeta “pages” y el archivo index.tsx con el contenido actual de App.tsx. Tomar en cuenta el uso de Suspense y Await para estado de carga.
11. Crear la ruta “/users” como hijo de la ruta “/” y asignarle el nuevo componente Users.
12. Definir el prop “Loader” para llamar a la función que simula la carga de usuarios.
13. Visualizar la carga de los usuarios.
14. Crear un nuevo componente para ErrorBoundary y definirlo como parte de la ruta “/users”.
15. Visualizar y simular errores para comprobar los cambios.
16. Agegar un nuevo path “/users/:state?” que tome en cuenta si solo mostrar verificados o no.

# Práctica

Consiste en realizar lo siguiente, crear una nueva ruta de la forma “users/view/{id}”, donde {id} es un parámetro obligatorio en la ruta. Este se usará para encontrar el usuario con el index en el Array. En caso de que no exista tal usuario con ese index se deberá mostrar un texto sencillo indicado que el usuario no existe. Se debe agregar un componente <a> con el texto “Ver perfil” en cada card del usuario. El index del usuario debe ser un nuevo prop de tipo numérico y opcional, debido a que solo se enviara desde la vista de lista.

Tomar en cuenta el uso del prop “Loader”, la sintaxis para definir parámetros en React-router y crear una nueva función “loadUserByIndex” para devolver el usuario.

Hint: Crear otro archivo view.tsx dentro de Users que use el hook userLoaderData y que use el componente UserProfile para enviar la data.