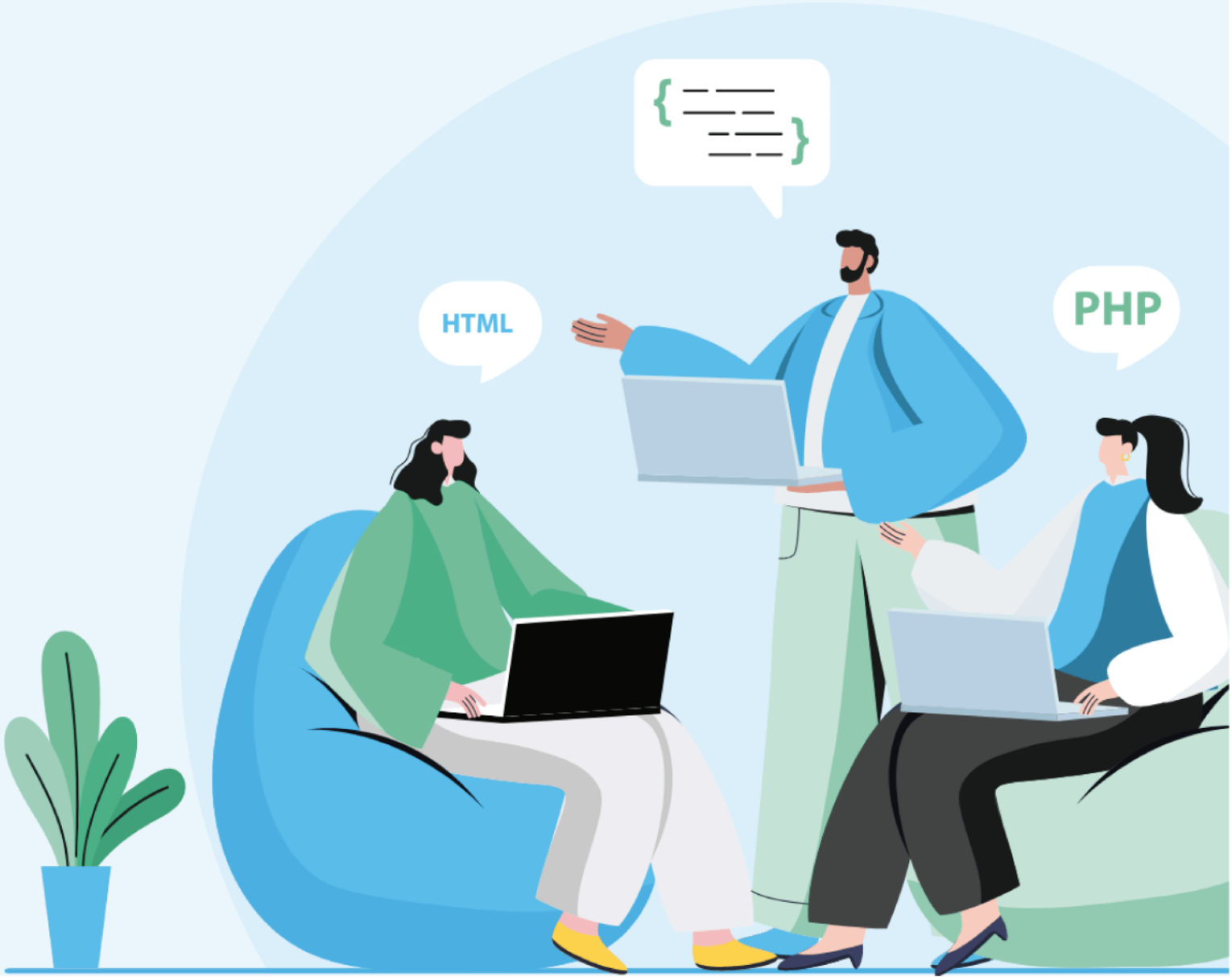
alkemy

Skill Up Lab - React Unidad 4



Unidad 4: Armado de la vista "Listado"

Antes de adentrarse en esta unidad

Para que puedas aprovechar mucho más el contenido presente en esta unidad, te recomendamos que antes de avanzar tengas muy en claro lo siguiente:

- cómo trabajar con componentes genéricos presentes a lo largo de toda la aplicación (como el header y el footer)
- los distintos mecanismos de implementación de CSS existentes dentro de React
- cuándo conviene y cuándo no conviene usar algún tipo de librería de CSS
- cuál es el mejor lugar dentro de la aplicación para almacenar nuestros archivos de CSS
- cómo implementar la librería de CSS Bootstrap a un nivel intermedio

Objetivos de la unidad

Con el estudio de esta unidad buscamos que:

- Armes el componente "Listado", para que puedas mostrar allí todos los resultados obtenidos desde la API
- Armes un componente "Detalle" que permita visualizar allí, la información de solamente un ítem en particular
- Entiendas cómo se renderiza un bloque de información (array) dentro del componente
- Identifiques cómo realizar un renderizado condicional, para que mientras la información llega desde la API la persona que usa la aplicación, pueda esperar cómodamente (mensaje: Cargando....)

Clase 1: Armado de la vista "Listado"

¿Para qué nos servirá este componente?

Este componente tiene como finalidad mostrar todos los datos recibidos desde la API. Aquí se deberán listar cada uno de los ítems recibidos, mostrando brevemente alguna parte de su información.

¿Cualquier aplicación hecha con React cuenta con este tipo de componente?

Generalmente sí, pues siempre que se quiera mostrar un gran conjunto de información (listado de productos, servicios, platos de comida, cartelera de cine, etc) se necesita este tipo de lógica a nivel de frontend que permita, no solo tomar los datos de la API, sino que también esté en la posibilidad de presentar esta información lo mejor posible de cara al visitante de la aplicación.

¿Cómo se presenta la información dentro de este componente?

Justamente eso es lo que aprenderemos en esta clase. Pero para hacerte un breve resumen, lo primero será hacer el llamado a la API y una vez tengas su respuesta, almacenar los datos en el estado local del componente, para posteriormente a través de una iteración, ir renderizando uno a uno los elementos obtenidos de dicha respuesta.

Clase 2: Protección Ruta "/listado"

¿Por qué es necesario proteger esta ruta?

Puntualmente en este challenge, es un requerimiento para que puedas demostrar tus habilidades. Pero está bueno que sepas que este proceso de "protección" de rutas es algo muy común ya que el mismo te permite evitar que personas que no estén autenticadas en la aplicación, puedan ver información sensible de la misma.

¿Cuál es el flujo exacto que se genera dentro de la aplicación al proteger esta ruta?

Como punto de partida, a esta ruta no se podrá ingresar si no se cuenta con la correspondiente autenticación de parte de la aplicación. Por eso, en dicho escenario, cualquier persona que carezca de credenciales válidas, será

redireccionada al formulario de login y su ingreso a esta ruta no será posible a menos que su usuario y contraseña sean los correctos.

¿Qué se necesita para poder proteger una ruta?

Una de las principales cosas que son necesarias para poder llevar a cabo esto, es una respuesta de parte del servidor que nos certifique en la parte del Front que la persona que está usando la aplicación se ha autenticado correctamente. Generalmente a esto se le llama "token". Y es gracias a la existencia del mismo, que vamos a poder implementar toda la lógica que nos permita dar acceso o no a esta ruta.

¿Por qué el token tiene que ser generado del lado del servidor?

Esto se debe a que es el servidor quien almacena las credenciales (usuario y contraseña) de los usuarios de la aplicación. Pues tener esta información del lado del frontend, no solamente es casi imposible, sino que a su vez es altamente peligroso, pues cualquier persona con un poco de conocimiento, gracias a las herramientas de desarrollo del navegador, podría corromper esta información.

Clase 3: Obtención de datos de la API

Listo, ya estamos logueados ¿y ahora?

Una vez logueados de manera satisfactoria en la aplicación. Lo que vamos a hacer ahora es disparar un pedido a la API para que la misma nos responda con la información esperada (listado de películas) y así poder renderizar dicha información correctamente en nuestra interfaz.

¿El uso de esta API demanda algún tipo de información especial?

No pero sí. Generalmente para consultar una API, la misma demanda algo que tradicionalmente se conoce como una "API KEY". La API KEY es algo que debemos obtener previamente de la API para, a través de esta, poder poder identificar que somos nosotros quienes haremos consultas al servidor.

Básicamente la API KEY es la que nos permite que nuestro llamado sea atendido. Y si bien no todas las APIS usan esta característica, la gran mayoría sí lo hace.

Pero... ¿Entonces la API KEY es lo mismo que el TOKEN?

No. El token es la confirmación de que la persona que se ha logueado en la aplicación lo hizo con las credenciales correctas, mientras que la API KEY es la "llave" que nos identifica frente a la API, para que la misma pueda atender nuestros llamados.

Listo. La API nos respondió con la información ¿y ahora?

Ahora viene la parte divertida, pues después de recibir la información, tendrás que poner en práctica tus habilidades de HTML, CSS y JS para mapear esta información y poder presentarla de una manera linda y agradable.

En un proyecto real ¿que tan común es trabajar con APIs?

Es lo más común de todo, pues la información que presentamos dentro de nuestro frontend, generalmente viene desde una API. Por ello es tan importante procesar estos conocimientos e implementarlos como algo habitual.

Clase 4: Renderización de los datos en el componente

Cuando ya tenemos los datos ¿cómo mostramos los mismos en el componente?

Para esta labor vamos a necesitar trabajar con mucho JSX y con algunas herramientas como el método MAP o FILTER de JavaScript, pues son estos métodos los que comúnmente se utilizan para llevar a cabo dicho proceso.

Pero ¿se hace necesario usar SI o SI un map o un filter?

Sí, pues dentro del esquema JSX que trabaja React, no es posible usar un iterador como un for o un forEach para renderizar la información, por tal motivo se hace necesario entender cómo funciona el map o el filter y reconocer cuál es su estructura.

¿Qué tan habitualmente se usa esto en un proyecto real?

Demasiado. Prácticamente cuando desarrollas dentro del entorno de React, te acostumbras a usar tanto el map o el filter, que se terminan volviendo tu mejores aliados de batalla en el proceso de desarrollo.

Clase 5: Manejo de errores (Sweet alert)

Realmente ¿es indispensable el manejo de errores en mi aplicación?

Como ya lo hablamos en momentos anteriores, El manejo de errores es la técnica a través de la cual como desarrolladores podemos manejar las distintas respuestas obtenidas después de procesar la información obtenida desde la API, por tal motivo debemos servir de mecanismos funcionales a la persona que visita nuestra aplicación, para que ésta sepa en qué momentos alguna petición generó un error como respuesta y sepa cómo actuar en consecuencia.

Entendido, pero... ¿necesito instalar otra librería para esto?

No, pues si anteriormente ya instalaste Sweet Alert, con ella es más que suficiente, aquí tu objetivo será entonces, identificar en qué momento tu

aplicación pudiera generar un mensaje de error e implementar justo ahí un elemento visual informativo con dicha librería.

En un proyecto real ¿es obligatorio usar Sweet Alert para el manejo de errores?

No. El manejo de errores es algo crucial que debes implementar sí o sí, pero la manera en cómo lo lleves a cabo y la implementación que decidas, corre 100% por tu cuenta en función de lo que te sea más cómodo y más útil de implementar.

Clase 6: Ingreso al detalle de cada ítem

¿De qué trata el "Detalle" de cada ítem?

Aquí vas a poder mostrar la información detallada de cada uno de los elementos recibidos como respuesta de la API. Como es común que en el listado completo de ítems solamente se muestre una pequeña porción de información, vamos a necesitar de un lugar determinado en donde mostrar el completo de la información. Y ese lugar, será el componente Detalle.

Pero si tengo muchos ítems ¿tengo que hacer muchos componentes "Detalle"?

No, ya que la idea fundamental es que este mismo componente te pueda servir para mostrar el detalle de cualquier ítem que el usuario haya seleccionado.

Y eso ¿cómo se hace?

Bueno, para lograr esto vamos a tener que hacer uso de las "rutas dinámicas" que nos provee React Router.

Y ¿qué es una ruta dinámica?

Básicamente es una ruta como cualquier otra, pero que va a recibir como parámetro un bloque de información crucial que permitirá que el componente que se renderiza sea dinámico. Por ejemplo en la siguiente ruta: http://localhost:3000/productos/45gb32F, la última parte (45gb32F) sería la parte dinámica, la misma vendría a representar el ID de un producto y dependiendo del ID:

http://localhost:3000/productos/56bhF

http://localhost:3000/productos/67rfG

Vamos a poder renderizar el detalle del producto que coincida con ese ID en particular. Pero, ¿qué mejor que el video para que estos conceptos queden mucho más claros?


alkemy