Front-End Avanzado en React | Bloque 1: Componentes y Estados

Proyecto: Reserva de alojamientos

Resumen del proyecto

Crea el Front-End de una aplicación de búsqueda hotelera. ¡Deberás poner en práctica los fundamentos de React incorporados hasta aquí y su respectiva implementación en el código para lograrlo!

Entregable

Se espera el Front-End de una aplicación web de búsqueda hotelera, con filtros por fechas y características que modifican los resultados a mostrar.

Recursos

Utiliza los siguientes recursos para realizar el proyecto:

- Consulta un video explicativo aquí.
- Descarga las imágenes de hoteles, scripts de utilidad (.jpg, .js) aquí.

A partir de este proyecto empezaremos a trabajar con sintaxis de React y ECMAScript 2015 o superior. Es recomendable que tengas a mano los recursos oficiales de ambas herramientas en cada una de las actividades y ejemplos que vienen a continuación.

- Documentación oficial de React
- Referencia de JavaScript en MDN

La documentación y las referencias oficiales son herramientas muy útiles para aprender y/o desarrollar con distintas tecnologías. A diferencia de los artículos o los fragmentos de código que podamos encontrar en otros sitios web, los recursos oficiales reflejan correctamente las últimas prestaciones de cada herramienta y promueven las prácticas y patrones más recomendadas a la hora de implementarlas.

Guías para realizar el proyecto

a) Crea el entorno

Lo primero que debes hacer a la hora de crear una aplicación en React es preparar el entorno de desarrollo. Existen varias formas de realizar esta tarea pero como **requisito** se pedirá que la configuración del entorno sea "manual" (no puedes usar create-react-app) esto quiere decir que las dependencias deberán ser especificadas por CDN, o Content Delivery Network. Necesitarás 3 librerías para poder arrancar el proyecto: React, ReactDOM y Babel.

Inicialmente deberás crear un archivo `index.html` el cual contendrá todas estas dependencias.

El link con el CDN para React y ReactDom los puedes encontrar en esta página, mientras que el de Babel lo puedes encontrar aquí.

Las dependencias (React, ReactDOM y Babel) las deberás insertar con <script> tags correspondientes antes de cerrar el `<body>` tag.
El orden en el que los scripts son insertados importa, así que si tienes un archivo que será consumido o utilizado por otro, deberás declararlo antes en el `index.html`.
Finalmente crea un `<div>` dentro del `<body>` con el `id="app"`, el cual funcionara como entry point a nuestra aplicación.

Hasta ahora, tu `<body>` de tu `index.html` debería de lucir así:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <!-- React -->
  <script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>
  <!-- ReactDOM -->
  <script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
  <!-- Babel -->
  <script crossorigin src="https://cdnjs.cloudflare.com/ajax/libs/babel-standalone/6.26.0/babel.min.js"></script>
</body>
</html>
```

La estructura de directorios debe ser similar a la siguiente:

- nombre_apellido
  - index.html
  - images
  - components

b) Crea el componente principal `<App />`

Es una buena práctica contar con un componente principal el cual se encargará de ser el contenedor de los subcomponentes que vayamos creando en la aplicación. Por convención, este componente siempre se llama `<App />`. Créalo en un archivo Javascript del mismo nombre y utiliza la función `render` de la librería `ReactDOM` para renderizarlo en nuestro entry point.
Agrega HTML de prueba en este componente, y ejecuta la extensión Live Server para comprobar que todo está bien configurado. Tu `<App />`componente debería verse así:

```
1  function App() {
2    return (
3      <div>
4        <h1>
5          Hemos configurado una aplicacion React a mano, en un entorno local.
6        </h1>
7      </div>
8    );
9  }
10
11 ReactDOM.render(<App />, document.getElementById("app"));
```

entry point de nuestra aplicación en index.html

La configuración manual implica también la carga de todos los archivos Javascript de nuestros componentes en nuestro `index.html` a través de los `<script>`'s tags. Estos tags necesitan tres propiedades:

- `src`: para indicar la ubicación del archivo
- `type`: con el valor "text/babel"
- `data-plugins`: con el valor "transform-es2015-modules-umd"

El atributo `data-plugins` es lo que te permite usar la sintaxis de import/export en tus componentes, la cual es manejada por herramientas como webpack.

```
<script
  src="./components/app.js"
  data-plugins="transform-es2015-modules-umd"
  type="text/babel">
</script>
```

Todos los componentes que se usen en tu proyecto deben estar importados en el index.html, con las sintaxis descrita en la imagen anterior.

Estos `<script>` tags deben estar ubicados después de la declaración de librerías.

El orden de estos `<script>` tags importa: el componente más profundo en nuestra arquitectura debe ser importado de primero, dejando de último el componente raíz, `<App />,` que actúa como contenedor.

c) Desarrolla la aplicación

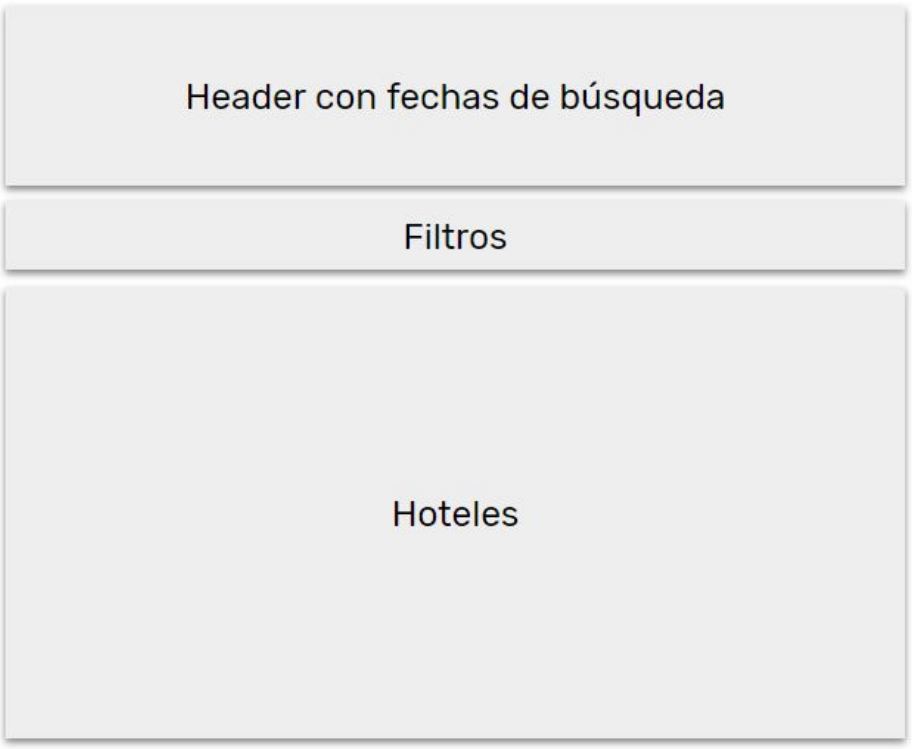
Ahora que has preparado el entorno de desarrollo, el siguiente paso es comenzar a crear los distintos componentes de nuestra aplicación.
Para ello necesitas identificar todos los componentes de la interfaz y comenzar a pensar cómo los mismos interactúan entre sí. Tal como lo hiciste en las actividades anteriores.

1. **Descarga la interfaz** del proyecto desde aquí y abre la misma con tu visor de PDF preferido. *Es recomendable utilizar algún visor como Adobe Acrobat Reader que te permita realizar anotaciones sobre el documento.*
2. **Identifica los componentes** de la interfaz, comenzando por los principales y siguiendo por cada uno de los hijos.
3. **Identifica los estados** y cómo éstos deberán ser consultados y/o modificados por los distintos componentes de la interfaz.

d) Interfaz gráfica

El objetivo del proyecto no consiste en construir una UI pixel perfect según los mocks entregados, estos sirven de referencia a como se debería de ver. Se pueden usar distintos colores, iconos o tipos de letra, así como librerías CSS (bootstrap, bulma, Styled Components) o técnicas (CSS-in-JS, CSS inline) para

alcanzar el estilo deseado. Como **requerimiento** se pedirá que la estructura conserve el layout vertical y extendido a lo ancho del browser: un encabezado con las fechas de búsqueda, seguido por una barra de filtros, y por último un área donde se muestran los hoteles que coinciden con la búsqueda aplicada.



No se requiere que la aplicación sea responsive, pero recomendamos aprovechar la oportunidad del desarrollo para incorporar esta técnica.

Checklist de evaluación

Antes de realizar la entrega del proyecto, revisa que cumplas con los siguientes ítems de evaluación (son los que tendrá en cuenta el/la evaluador/a técnico/a a la hora de corregir tu trabajo):

a. Representa visualmente la estructura vertical de componentes en el explorador siendo fiel al layout de interfaz de la aplicación.

b. No utiliza elementos HTML adicionales al contenedor que necesita React para renderizar los componentes (solo tener el '`<div id="app"></div>`').

c. Permite al usuario/a realizar modificaciones en los valores que se muestran en los campos de la barra de filtros.

d. Las modificaciones en los valores de la barra de filtros impactan en tiempo real sobre la información que muestra el encabezado de la aplicación.

e. El encabezado de la aplicación solo muestra en lenguaje natural los filtros que se encuentran activos, es decir, que tienen un valor asignado. Al menos las fechas.

f. Los hoteles del archivo `data.js` se ven representados a través de su respectiva ficha en la interfaz.

g. Solo se muestran aquellos hoteles que coinciden con los filtros configurados por el/la usuario/a durante la ejecución de la aplicación.

h. Los filtros pueden ser modificados, activados o desactivados por los/as usuarios/as múltiples veces durante la ejecución y la aplicación no genera errores durante este comportamiento.

i. Brindar mensajes explicativos para cuando no exista disponibilidad de hoteles respecto a los filtros aplicados (incluir todos los filtros en esta validación).

Nota: *Recuerda que es posible que necesites un Live Server para poder abrir la aplicación web desde el explorador. En los editores de texto más populares existen plugins que brindan esta funcionalidad.*

El archivo que subas a plataforma tiene que estar comprimido (.zip) con tu nombre y el nombre del proyecto, por ejemplo, julianavelasco_Reserva_Alojamientos.zip.