

ACÀMICA

Agenda

Daily

Composición en React

Fechas en programación, complejidad y tiempo UNIX.

Manejo de fechas en Javascript

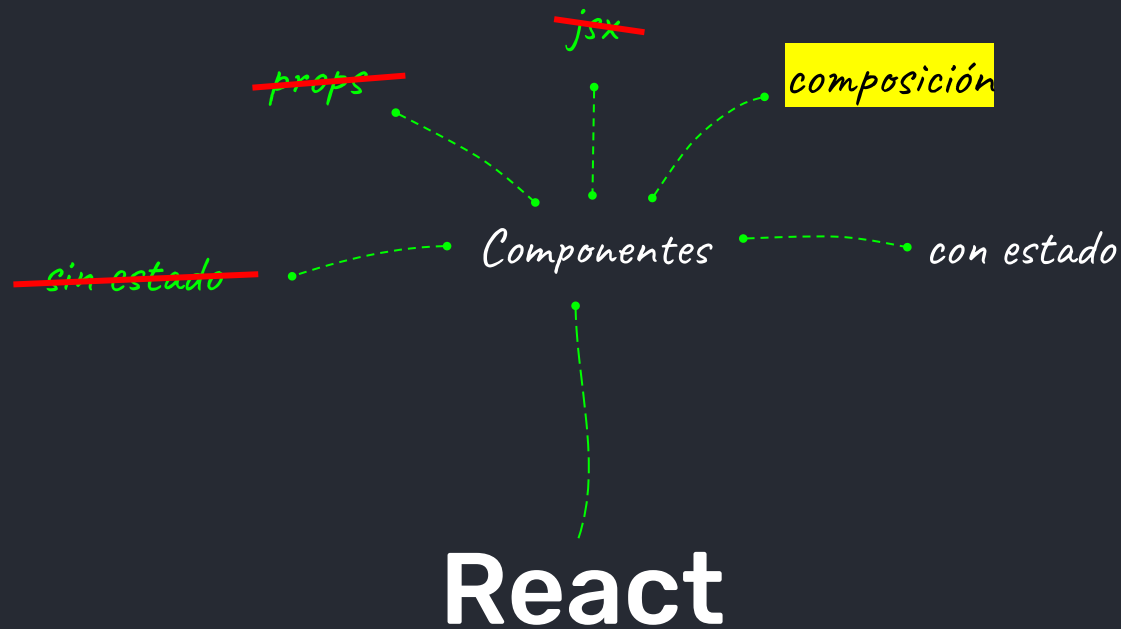
Break

Actividades



Composición

En el meeting de hoy vamos a armar un layout completo con React, pero antes, tendremos que hacer una parada técnica por el manejo de fechas en Javascript.



Daily



Daily



Sincronizando...

Toolbox



¿Cómo te ha ido?
¿Obstáculos?
¿Cómo seguimos?

Challenge



¿Cómo te ha ido?
¿Obstáculos?
¿Cómo seguimos?

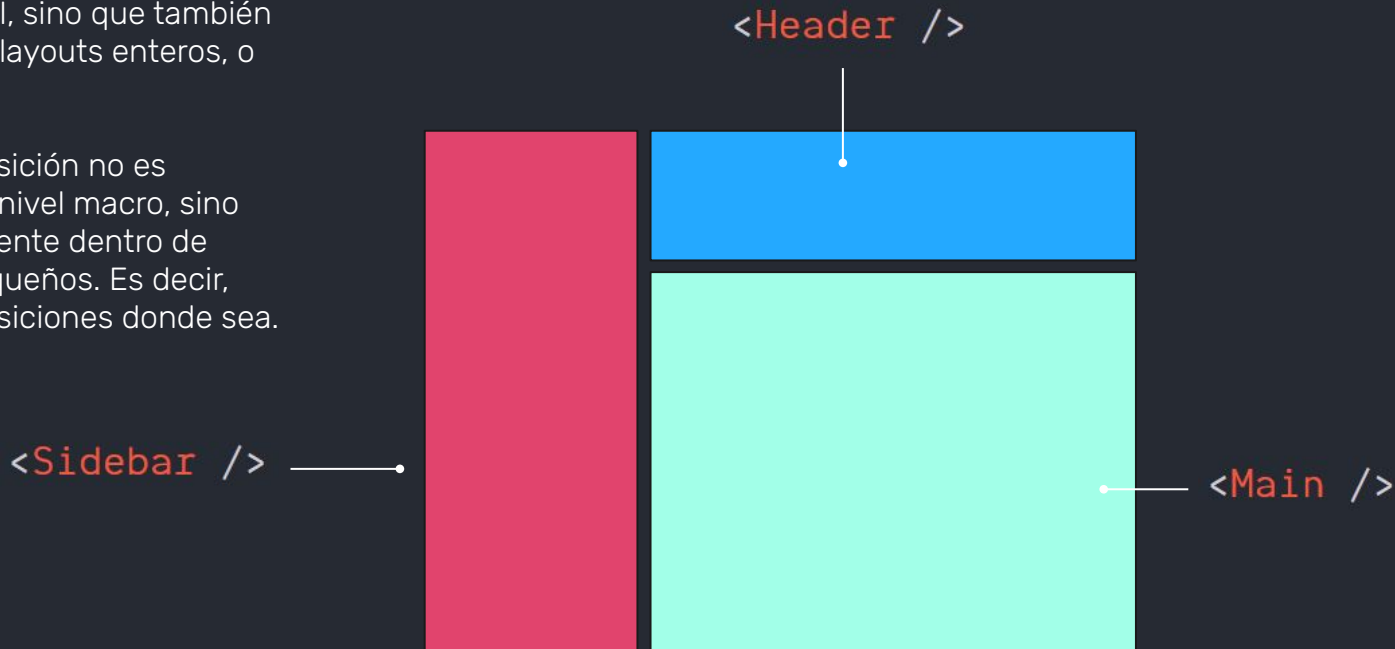
Composición



Composición

Los componentes no son solo piezas aisladas dentro de la UI, sino que también son usados para crear layouts enteros, o composiciones.

El concepto de composición no es exclusivo de layouts a nivel macro, sino que también está presente dentro de componentes más pequeños. Es decir, podemos crear composiciones donde sea.



Esto es una **composición** que solo se encarga de acomodar elementos dentro del componente `<Hotel />`



La Bamba de Areco

La Bamba de Areco está ubicada en San Antonio de Areco, en el corazón de la pampa. Es una de las estancias más antiguas de la Argentina, recientemente restaurada para ofrecer a sus huéspedes todo el confort y esplendor colonial.

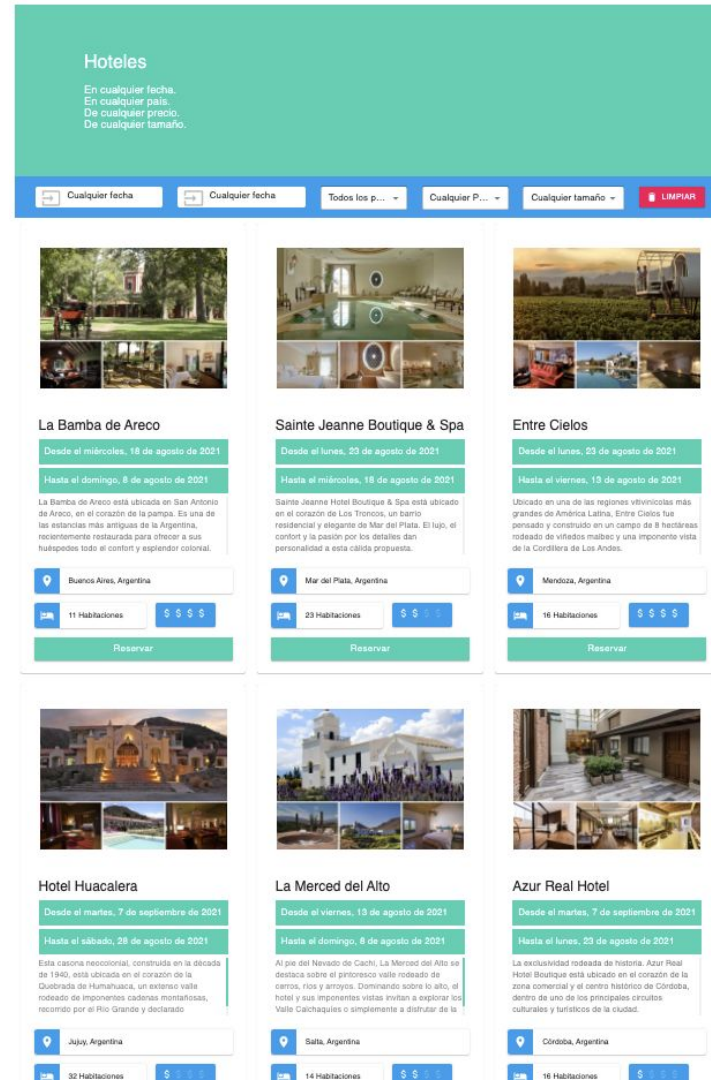
 Buenos Aires, Argentina

 11 Habitaciones 

[Reservar](#)

Esto también es una **composición** que se encarga de distribuir un layout de una aplicación.

Esta composición está conformada por distintos componentes.



Programamos

¡todos y todas!



Paso 1: Configurar y setear el entorno.

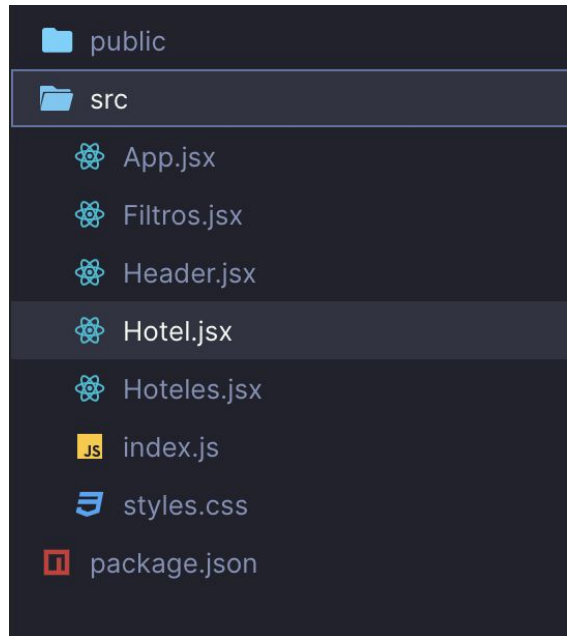
Vamos a empezar a construir el sprint project 2, aplicando los conocimientos adquiridos sobre composición.

Necesitamos a una persona voluntaria que haga las siguientes actividades:

- Cree un codesandbox nuevo, con react.
- En el sandbox, crear 4 nuevos componentes:
 - **Header.jsx**
 - **Filtros.jsx**
 - **Hoteles.jsx**
 - **Hotel.jsx**

Ahora, escoge a una persona para que haga el siguiente paso!

La estructura de directorios de code sandbox se debería ver así:



Idealmente, cada componente se encargará de una sola cosa. Esto se conoce como el “SRI” o **Single Responsibility Principle** (Principio de única responsabilidad)

Header.jsx

Se encargará de renderizar el header, o cabecera de la aplicación.

Filtros.jsx

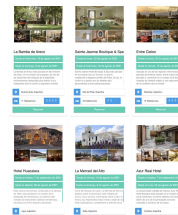
Se encargará de renderizar los filtros de la aplicación.

Hoteles.jsx

Es un contenedor que mostrará una lista de hoteles.

Hotel.jsx

Es un componente que mostrará solo un hotel.



La Bamba de Areco

Desde el miércoles, 18 de agosto de 2021

Hasta el domingo, 8 de agosto de 2021

La Bamba de Areco está ubicada en San Antonio de Areco, en el corazón de la pampa. Es una de las estancias más antiguas de la Argentina, recientemente restaurada para ofrecer a sus huéspedes todo el confort y esplendor colonial.

Paso 2: Conectando componentes

En cada archivo (Filtros, Header, Hoteles y Hotel) crea el componente correspondiente dentro de cada archivo.

Por ahora, cada componente solo deberá retornar un div, con un h1 adentro que tenga el nombre del componente.

También, deberás de importar el archivo de estilos "styles.css", y agregar una clase al div contenedor de cada componente, cuyo nombre será igual al del componente (todo en minúscula, pero es un dato no importante). Más adelante crearemos estas clases.

```
// Hoteles.jsx
import React from "react";
import "../styles.css"
```

```
function Hoteles() {
  return (
    <div className="hoteles">
      <h1>Hoteles</h1>
    </div>
  )
};
```

```
export default Hoteles;
```

```
// Filtros.jsx
import React from "react";
import "../styles.css"
```

```
function Filtros() {
  return (
    <div className="filtros">
      <h1>Filtros</h1>
    </div>
  )
};
```

```
export default Filtros;
```

```
// Hotel.jsx
import React from "react";
import "../styles.css"
```

```
function Hotel() {
  return (
    <div class="hotel">
      <h1>Hotel</h1>
    </div>
  )
};
```

```
export default Hotel;
```

```
// Header.jsx
import React from "react";
import "../styles.css"
```

```
function Header() {
  return (
    <div class="header">
      <h1>Header</h1>
    </div>
  )
};
```

```
export default Header;
```

¡Escoge a alguien que haga el siguiente paso!

Paso 3: Conectando componentes

Luego, implementa estos 4 componentes en App.jsx

Si todo salio bien, deberás ver esto en la UI.

```
import './styles.css';
import Header from './Header';
import Filtros from './Filtros';
import Hoteles from './Hoteles';
import Hotel from './Hotel';

export default function App() {
  return (
    <div className="App">
      <Header />
      <Filtros />
      <Hoteles />
      <Hotel />
    </div>
  );
}
```

Header

Filtros

Hoteles

Hotel

¿Quién debe hacer el siguiente paso?

```
* {  
  · box-sizing: border-box;  
  · padding: 0;  
  · margin: 0;  
}  
  
.App {  
  · font-family: sans-serif;  
  · text-align: center;  
}  
  
.header {  
  · width: 100%;  
  · margin: 0 auto;  
  · background-color: cyan;  
  · height: 200px;  
}
```

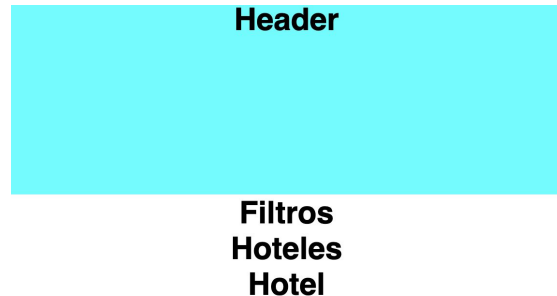
```
.filtros {  
  · width: 100%;  
  · background-color: rgb(71, 111, 243);  
  · height: 70px;  
}  
  
.hotel {  
  · width: 200px;  
  · border: 1px solid;  
  · height: 300px;  
}
```


Paso 4: Estilos básicos para el header

En styles.css, agrega la clase "header" y dale unos estilos básicos. También, resetea todo el css con el selector universal, y opcionalmente, deja la clase "App"

```
* {  
  · box-sizing: border-box;  
  · padding: 0;  
  · margin: 0;  
}  
  
.App {  
  · font-family: sans-serif;  
  · text-align: center;  
}  
  
.header {  
  · width: 100%;  
  · margin: 0 auto;  
  · background-color: cyan;  
  · height: 200px;  
}
```

Este será el resultado del código de estilos.



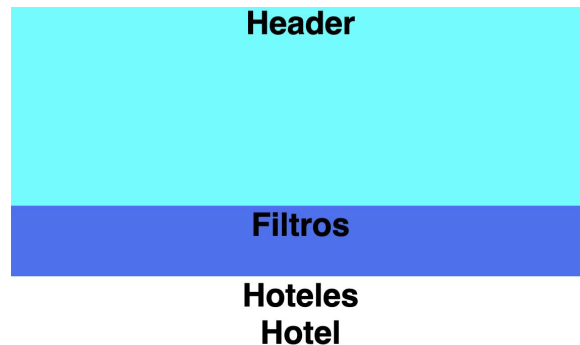
¿Quién debe hacer el siguiente paso?

Paso 5: Estilos básicos para los filtros

En styles.css, agrega la clase "filtros" y dale unos estilos básicos.

```
.filtros {  
  width: 100%;  
  background-color: rgb(71, 111, 243);  
  height: 70px;  
}
```

Este será el resultado del código de estilos.



¿Quién debe hacer el siguiente paso?

A close-up photograph of a white ceramic cup filled with a latte. The surface of the milk is decorated with intricate latte art, featuring a central heart shape surrounded by concentric, wavy lines. The cup is placed on a matching white saucer. In the background, a white napkin and a silver spoon are visible, though they are out of focus. The overall lighting is soft and even.

¡BREAK!

Paso 6: Estilos básicos para Hotel

El componente `<Hotel />` se encargará de mostrar un hotel. En la clase de hoy haremos solo un prototipo, pero luego deberás ajustarlo para que muestre otras propiedades.

Agrega esta clase a "styles.css"

```
.hotel {  
  width: 200px;  
  border: 1px solid;  
  height: 300px;  
}
```

Este será el resultado del código de estilos.



¿Quién debe hacer el siguiente paso?

Paso 7: Refactor entre Hoteles y Hotel

Actualmente, en App.jsx estamos implementando todos los componentes. Pero, la responsabilidad de <Hoteles /> es mostrar una lista del componente Hotel.

Por lo tanto, debemos borrar <Hotel /> de App.jsx e implementarlo en <Hoteles />. Implementa el componente al menos unas 6 veces.

```
import React from "react";
import "../styles.css";
import Hotel from "../Hotel";
```

```
function Hoteles() {
  return (
    <div className="hoteles">
      <h1>Hoteles</h1>
      <Hotel />
      <Hotel />
      <Hotel />
      <Hotel />
      <Hotel />
      <Hotel />
    </div>
  );
}
```

```
export default Hoteles;
```

```
import "../styles.css";
import Header from "../Header";
import Filtros from "../Filtros";
import Hoteles from "../Hoteles";
import Hotel from "../Hotel";
```

```
export default function App() {
  return (
    <div className="App">
      <Header />
      <Filtros />
      <Hoteles />
      <Hotel />
    </div>
  );
}
```

Recuerda borrar de App.jsx la implementación de <Hotel />.

Paso 7: Refactor entre Hoteles y Hotel

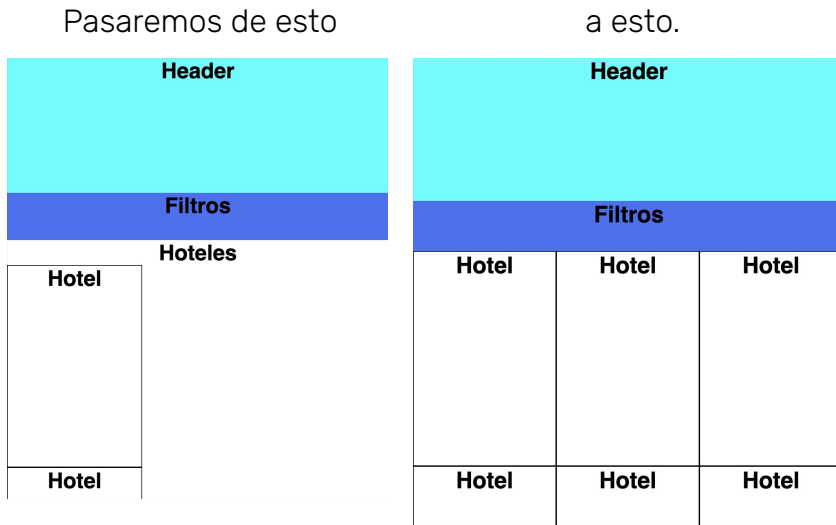
Al hacer esto, vamos a ver que cada hotel se posiciona uno debajo del otro. ¿Por que sucede esto?

Para arreglarlo, deberemos crear la clase `.hoteles` en `styles.css` y hacer que nuestro componente sea un flex container.

```
.hoteles {  
  display: flex;  
  flex-wrap: wrap;  
}
```

También podremos borrar el `h1` que dice "Hoteles" del componente `<Hoteles />`.

Escoge a alguien para el siguiente paso, pero antes, hablemos de instanciación dinámica.



Instanciación dinámica



Instanciación dinámica

Es la técnica que consiste en crear múltiples componentes utilizando iteración.

Usualmente, en web apps, la data vendrá en listas estructuradas como arrays.

En vez de tener 3 variables:

```
const album1 = {
  artista: "Rozes",
  titulo: "Under the Grave",
  lanzamiento: "02-14-2016",
  portada: "url..."
};
const album2 = {
  artista: "The Weeknd",
  titulo: "Starboy",
  lanzamiento: "11-25-2018",
  portada: "url..."
};
const album3 = {
  artista: "Sia",
  titulo: "Cheap Thrills",
  lanzamiento: "4-13-2017",
  portada: "url..."
};
```

Tendremos un array con múltiples objetos:

```
const albums = [
  {
    artista: "Rozes",
    titulo: "Under the Grave",
    lanzamiento: "02-14-2016",
    portada: "url..."
  }, {
    artista: "The Weeknd",
    titulo: "Starboy",
    lanzamiento: "11-25-2018",
    portada: "url..."
  }, {
    artista: "Sia",
    titulo: "Cheap Thrills",
    lanzamiento: "4-13-2017",
    portada: "url..."
  }
];
```


Instanciación dinámica

En React, podemos utilizar funciones de orden superior para iterar sobre un array y así poder crear el JSX que necesitemos.

En vez de implementar 3 componentes:

```
export default function App() {
  return (
    <div className="App">
      <Album
        artista={album1.artista}
        titulo={album1.titulo}
        fecha={album1.lanzamiento}
        portada={album1.portada}
      />
      <Album
        artista={album2.artista}
        titulo={album2.titulo}
        fecha={album2.lanzamiento}
        portada={album2.portada}
      />
      <Album
        artista={album3.artista}
        titulo={album3.titulo}
        fecha={album3.lanzamiento}
        portada={album3.portada}
      />
    </div>
  );
}
```

Podemos iterar sobre un array de álbumes:

```
export default function App() {
  return (
    <div className="App">
      albums.map(function(album) {
        <Album
          artista={album.artista}
          titulo={album.titulo}
          fecha={album.lanzamiento}
          portada={album.portada}
        />
      })
    </div>
  );
}
```

Paso 8: Instanciación dinámica de hoteles.

Para aplicar la técnica de instanciación dinámica en nuestra aplicación, vamos a tener que utilizar nuestra pequeña base de datos de hoteles primero.

En el sandbox, crea un archivo que se llame hotelsData.js, y copia [esta colección](#).

Luego, en App.jsx, importa esta colección y pásala como una prop en el componente <Hoteles />.

Con esto, tendremos disponible la colección de hoteles en el componente de <Hoteles />.

```
// App.jsx debería quedar así
import './styles.css';
import Header from './Header';
import Filtros from './Filtros';
import Hoteles from './Hoteles';
import {hotelsData} from './hotelsData';

export default function App() {
  return (
    <div className="App">
      <Header />
      <Filtros />
      <Hoteles listaHoteles={hotelsData}/>
    </div>
  );
};
```

¿Quién debe hacer el siguiente paso?

Paso 9: Rescatamos las props

Previamente estabamos instanciando a mano 6 hoteles, lo cual no es muy escalable. El objetivo consiste en instanciar tantos hoteles como existan en nuestra colección.

Ahora, deberemos rescatar la prop "listaHoteles" en la definición de este componente y aplicar la tecnica de la instanciación dinámica. Para esto, deberemos de agregar un .map sobre "listaHoteles" y retornar en cada iteración un componente <Hotel />

```
function Hoteles() {  
  return (  
    <div className="hoteles">  
      <Hotel />  
      <Hotel />  
      <Hotel />  
      <Hotel />  
      <Hotel />  
      <Hotel />  
    </div>  
  );  
}
```

```
function Hoteles(props) {  
  return (  
    <div className="hoteles">  
      {  
        props.listaHoteles.map((hotel) => {  
          return <Hotel />  
        })  
      }  
    </div>  
  );  
}
```

¿Quién debe hacer el siguiente paso?

¿Por qué tenemos tantos hoteles ahora?

Filtros		
Hotel	Hotel	Hotel
Hotel	Hotel	Hotel

Paso 10: Agreguemos data de hoteles

Para agregar data a nuestros hoteles debemos especificar algunas props a la implementación del componente `<Hotel />`.

Antes de eso, creemos una carpeta de nombres "images" dentro de la carpeta "public". Codesandbox nos pide que hagamos esto si queremos trabajar con imágenes. Agrega [estas imágenes](#) dentro de la carpeta "images".

Luego, especifica algunas props dentro de la implementación del componente `<Hotel />`, dentro del `.map` en `<Hoteles />`.

```
function Hoteles(props) {  
  return (  
    <div className="hoteles">  
      {props.listaHoteles.map((hotel) => {  
        return (  
          <Hotel  
            nombre={hotel.name}  
            pais={hotel.country}  
            imagen={hotel.photo}  
          />  
        )};  
      })}  
    </div>  
  );  
};
```

¿Quién debe hacer el siguiente paso?

Paso 11: Rescatemos props en <Hotel />

Finalmente, para poder ver las props especificadas en la implementación de <Hotel />, deberemos rescatar estas props en la definición de <Hotel /> para así poder mostrarlas.

```
function Hotel(props) {  
  return (  
    <div className="hotel">  
      <h3>{props.nombre}</h3>  
      <h5>{props.pais}</h5>  
      <img width="100%" src={props.imagen} alt={props.nombre} />  
    </div>  
  );  
}
```









Y con este cambio, **¡BAM!** Podemos ver todos nuestros hoteles, instanciados de forma dinámica.



Y con este cambio, **¡BAM!** Podemos ver todos nuestros hoteles, instanciados de forma dinámica.



[Aquí te dejamos el código de lo construido hoy.](#)

Header		
Filtros		
La Bamba de Areco Argentina	Sainte Jeanne Boutique & Spa Argentina	Entre Cielos Argentina
 	 	 
Hotel Huacalera Argentina	La Merced del Alto Argentina	Azur Real Hotel Argentina
		

Proximos pasos



Estilos para <Hotel />

Ahora que tienes tu aplicación con un prototipo básico para el componente de <Hotel />, trabaja sobre el archivo Hotel.jsx para agregar HTML y CSS, y así mejorar el estilo.

También, agrega el resto de las props, para que se pueda ver la información que el componente pide.



Para la próxima

- 1) Termina el ejercicio de la meeting de hoy y tráelo a la próxima para agregar nuevas funcionalidades.
- 2) Lee la toolbox 28.
- 3) Resuelve el challenge.

ACÀMICA