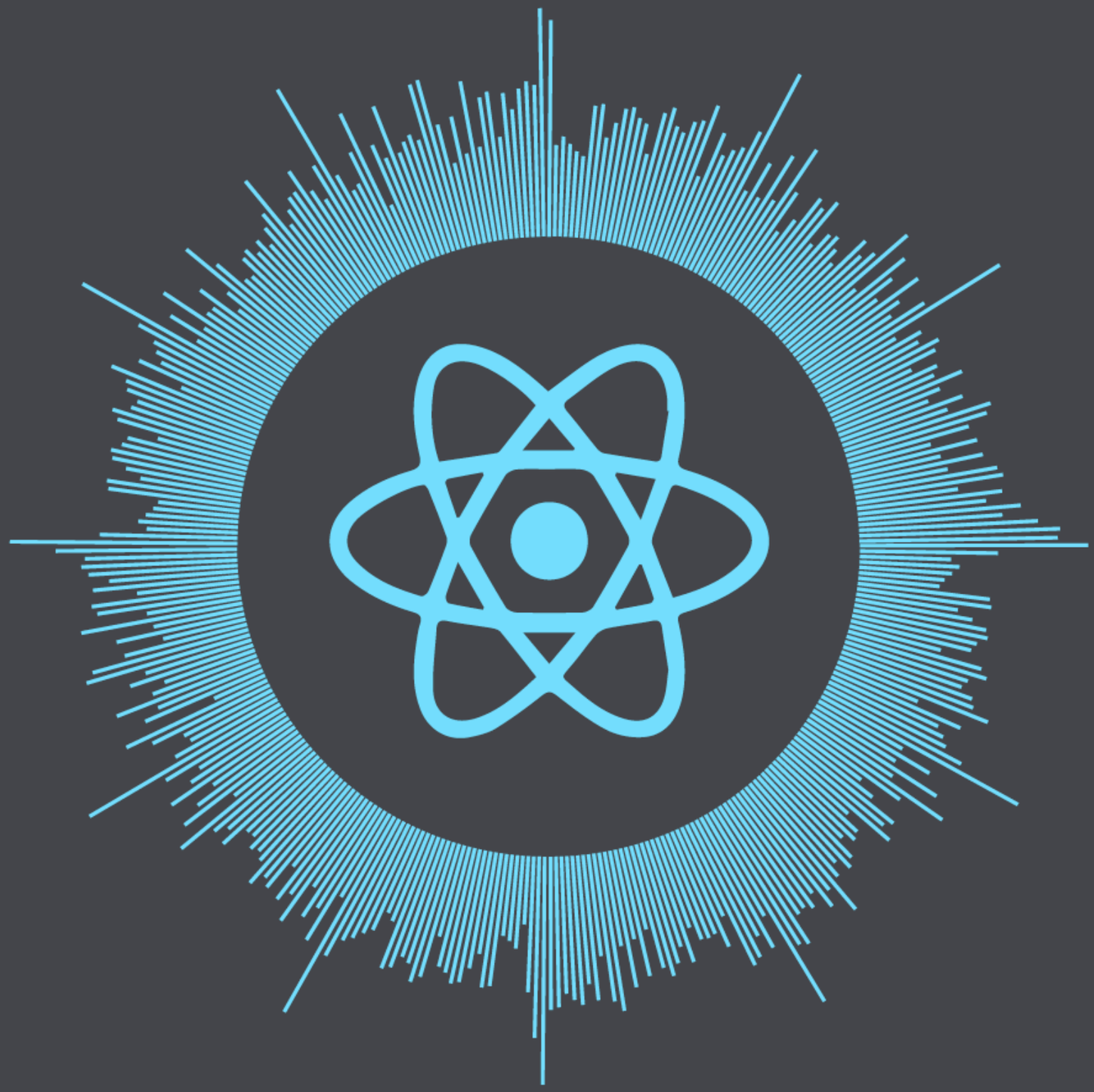


React

Presentando **JSX**

Considera la declaración de esta variable

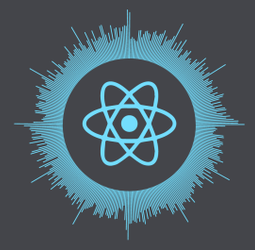
```
const element = <h1>Hello, world!</h1>;
```

Esto no es un string ni tampoco es HTML

Es JSX y es una extension de la sintaxis de JS y es recomendable usarlo con REACT para construir las interfaces de usuario

Expresiones **JSX**

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```



JSX como expresión

Esto significa que puedes usar JSX dentro de control de JS asignarlo a variables, aceptarlo como argumentos y retornarlo desde dentro de funciones.

```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
  return <h1>Hello, Stranger.</h1>;
}
```

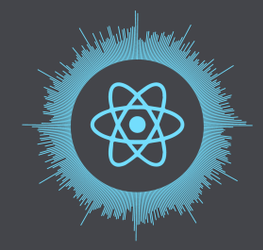
Especificando atributos

Se puede usar comillas para especificar strings literales como atributos:

```
const element = <div tabIndex="0"></div>;
```

Pero también puedes insertar expresiones JS en un atributo usando {…}

```
const element = <img src={user.avatarUrl}></img>;
```



Nomenclatura camelCase

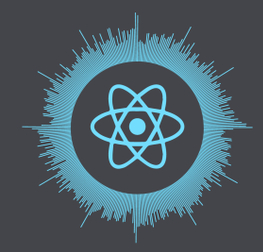
Dado de JSX es mas cercano a JS que a HTML, React utiliza la convención de nomenclatura camelCase en vez de nombres de atributos HTML

Por ejemplo, **class** se vuelve en **className** JS, y así envisto otros ejemplos.

tabindex —> tabIndex

z-index —> zIndex

background-color —> backgroundColor



Renderizando elementos

Digamos que hay un elemento ** ** en alguna parte de tu archivo html:

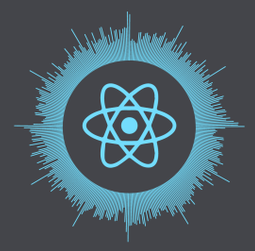
```
<div id="root"></div>
```

Es llamado nodo “raiz” por que todo lo que este dentro de el será manejado por REACT DOM

Las apps construidas solo con REACT usualmente tienen un único nodo “root” en el DOM. Dado el caso de que estes integrando REACT en una app existente, puedes tener tantos nodos raiz del DOM aislados como quieras.

Para renderizar un elemento de REACT en un nodo “root”, ambos se tienen que pasar a `ReactDOM.render()`

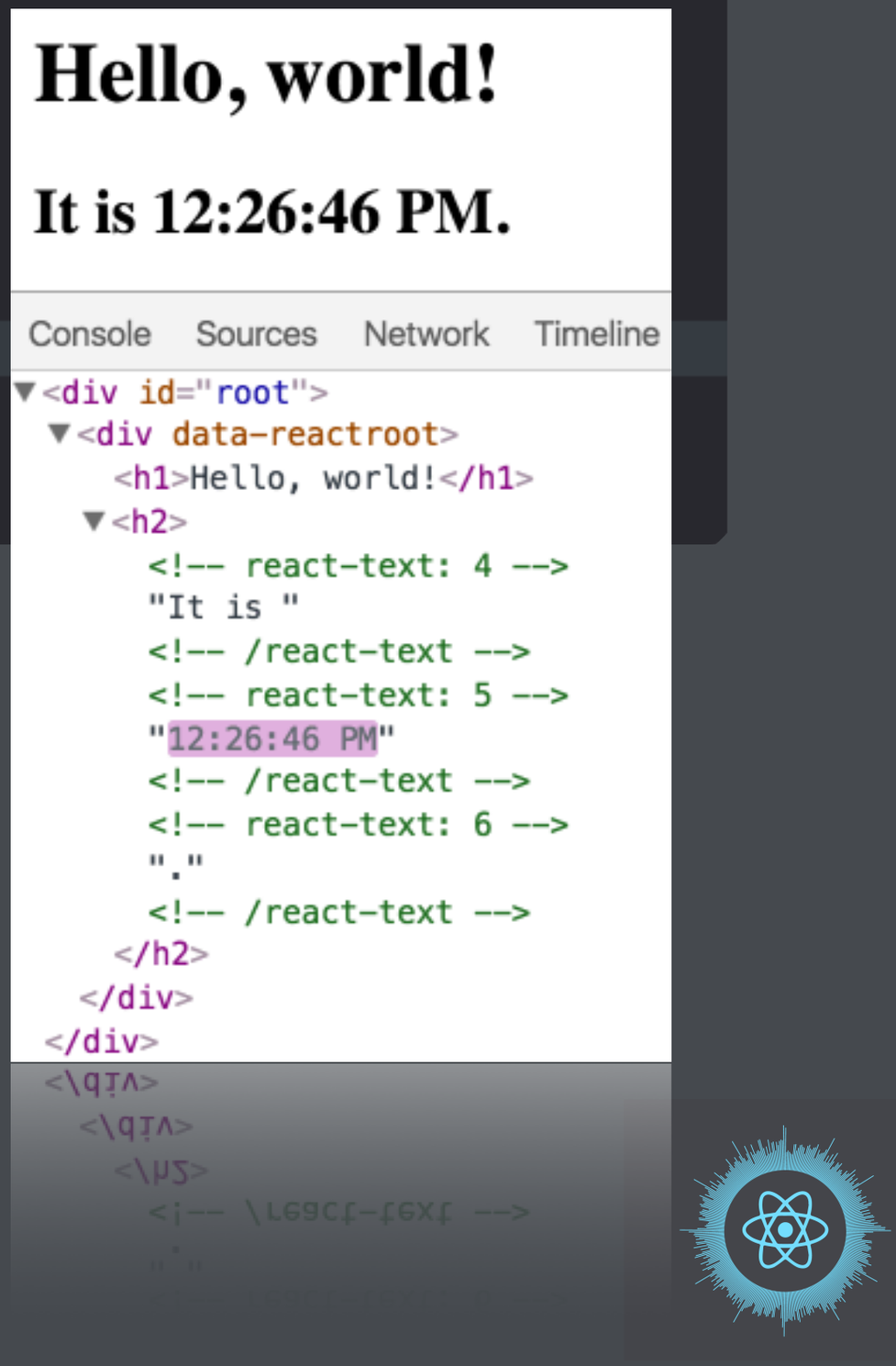
```
const element = <h1>Hello, world</h1>;
ReactDOM.render(element, document.getElementById('root'));
```



Actualizando elementos renderizados

Los elementos de REACT son inmutables. Una vez creado, no puedes cambiar a sus hijos o atributos, Un elemento es como un fotograba solitario en una película: este representa la interfaz de usuario en cierto punto en el tiempo.

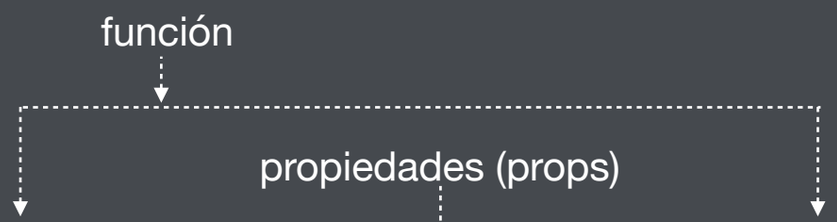
```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(element, document.getElementById('root'));
}

setInterval(tick, 1000);
```



Componentes y propiedades

Conceptualmente, los componentes son funciones de JS. Aceptan entradas arbitrarias (props “de solo lectura”) y devuelven a REACT Elementos que describen lo que debe aparecer en pantalla.

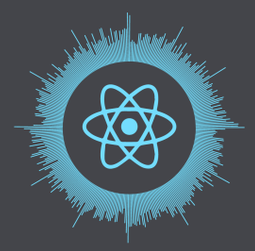
La forma más sencilla es escribir una función.




```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Renderizando el componente

```
const element = <Welcome name="Sara" />;
```



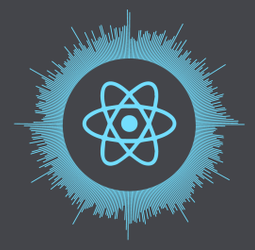
Ejemplo completo

Conceptualmente, los componentes son funciones de JS. Aceptan entradas arbitrarias (props) y devuelven a REACT Elementos que describen lo que debe aparecer en pantalla.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Resultado

Hello, Sara



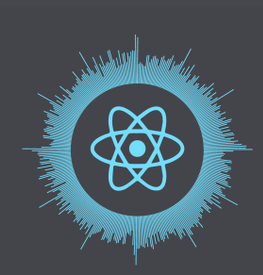
Documentación

Conceptualmente, los componentes son funciones de JS. Aceptan entradas arbitrarias (props) y devuelven a REACT Elementos que describen lo que debe aparecer en pantalla.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Resultado

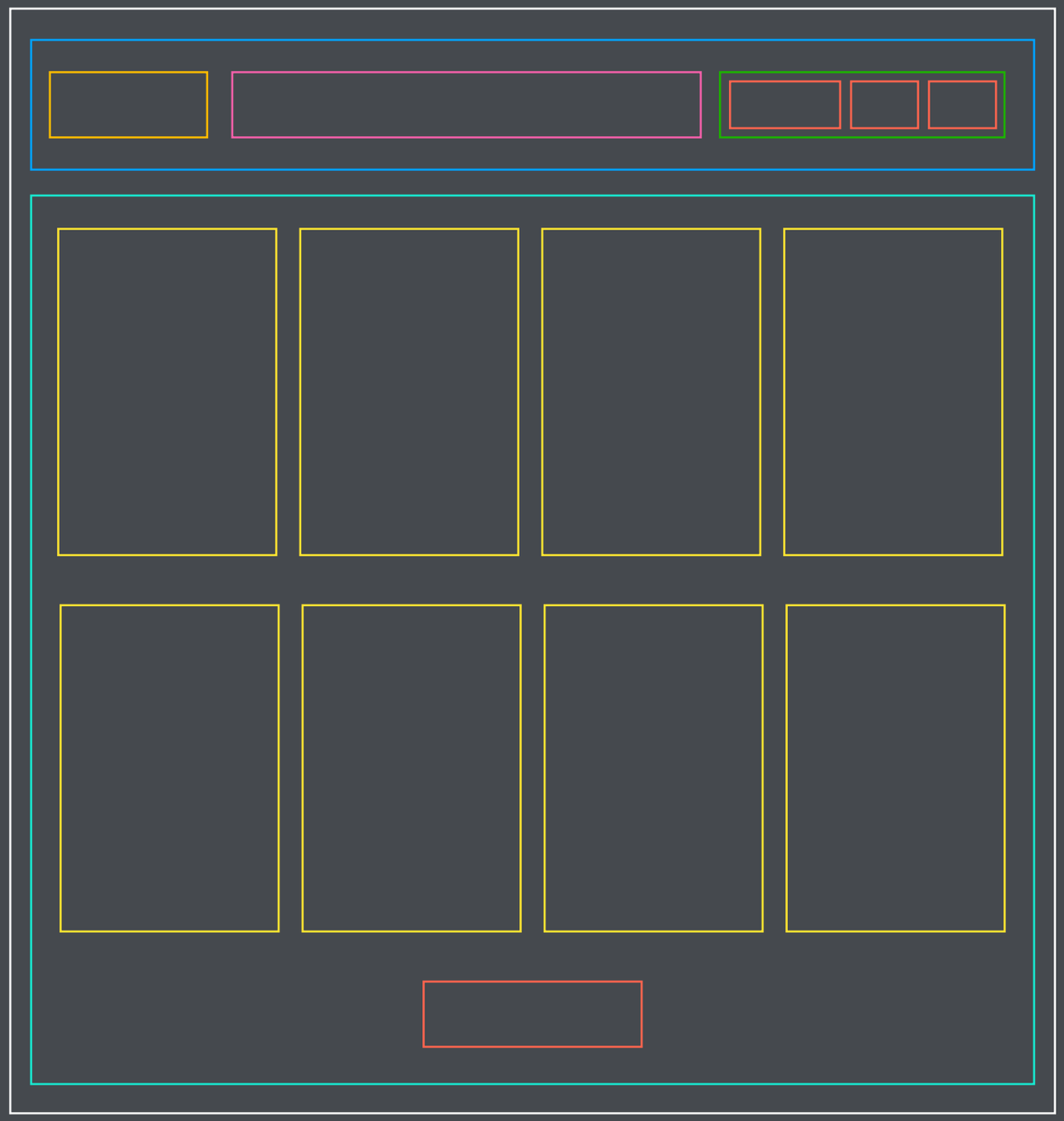
Hello, Sara

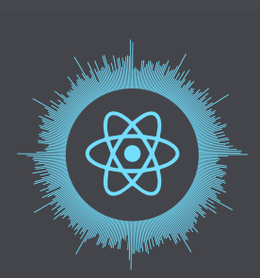


Extracción de componentes “Divide y vencerás”

App

- Header
  - Logo
  - Search
  - Nav
- Product List
  - Product Shelf
- Button





Practica

App

Header

Logo

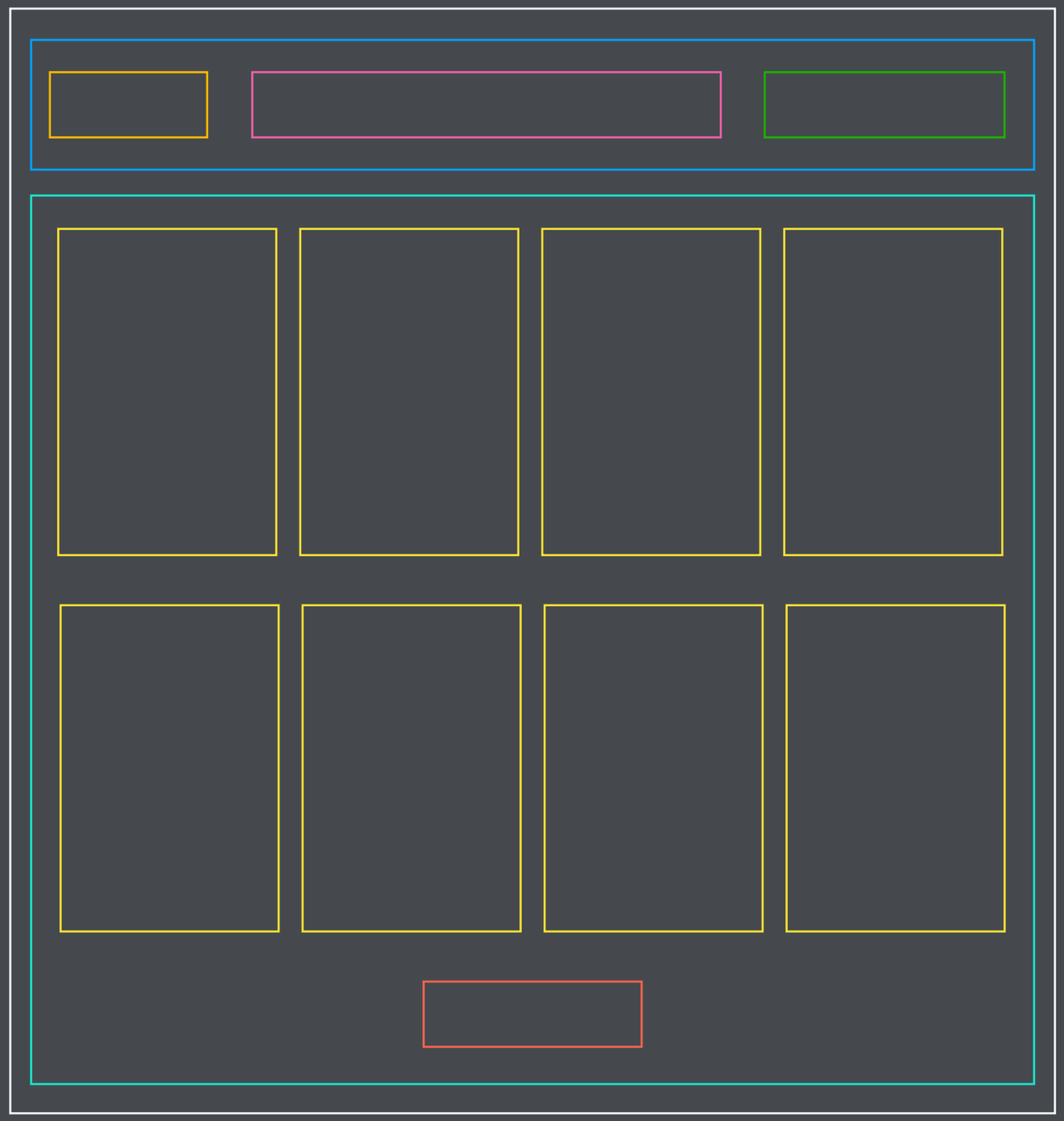
Search

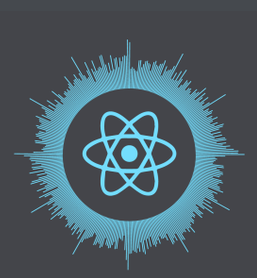
Nav

Product List

Product Shelf

Button





Estado y ciclo de vida

Continuara...