



React Router I





**Activen las cámaras los que puedan y
pasemos asistencia**

Crear una Single Page Application con React utilizando React Router.

- Unidad 1: Introducción a React.
- Unidad 2: Estados de los componentes y eventos.
- Unidad 3: Renderización dinámica de componentes.
- Unidad 4: Consumo de APIs con React.
- Unidad 5: React Router I
- Unidad 6: Context
- Unidad 7: React Router II
- Unidad 8: JWT



Te encuentras aquí





Inicio

{desafío}
latam_



/ Crear un sistema de navegación basado en rutas. */*

/ Crear diferentes rutas que utilicen componentes como vistas en la aplicación. */*

/ Crear una ruta por defecto enlazada a la vista Not Found. */*

A vertical decorative line on the right side of the slide, featuring a large white '@' symbol and several curly braces '{' and '}' in white and light gray.

Objetivos

Activación de conceptos

Contesta la pregunta correctamente y gana un punto

Instrucciones:

- Se realizará una pregunta, el primero en escribir “YO” por el chat, dará su respuesta al resto de la clase.
- El docente validará la respuesta.
- En caso de que no sea correcta, dará la oportunidad a la segunda persona que dijo “Yo”.
- Cada estudiante podrá participar un máximo de 2 veces.
- Al final, el/la docente indicará el 1º, 2º y 3º lugar.
- Esta actividad no es calificada, es solo una dinámica para recordar los conceptos clave para abordar esta sesión.





Activación de conceptos



Recordando el módulo anterior

¿En qué situaciones necesitamos ocupar
useEffect?



Activación de conceptos



Recordando el módulo anterior

¿Cuáles son las fases de un componente?



Activación de conceptos



Recordando el módulo anterior

¿Para qué sirve el callback que le pasamos a
useEffect?



Activación de conceptos



Recordando el módulo anterior

¿Para qué sirve el arreglo de dependencias que recibe useEffect?



Activación de conceptos



Primer lugar:



Segundo lugar:



Tercer lugar:

/* Mi primera Single Page Application */

React Router

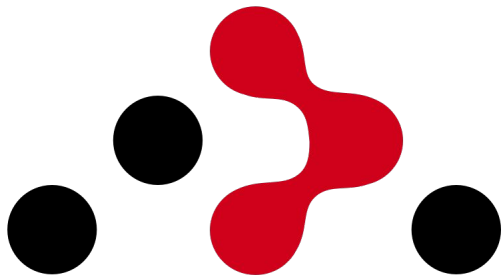
¿Qué es una SPA?

- Una Single Page Application(SPA) es una aplicación que ofrece la navegación de su contenido basada en rutas dentro de una misma página web sin que ésta se tenga que recargar.
- Este tipo de aplicaciones simula el comportamiento y fluidez de una aplicación de escritorio al renderizar parcialmente los componentes interpretados como vistas en función a la ruta que el usuario consulte.
- Algunos ejemplos son: Gmail, Twitter, Trello, Netflix, Google Maps.



React Router

¿Qué es React Router?

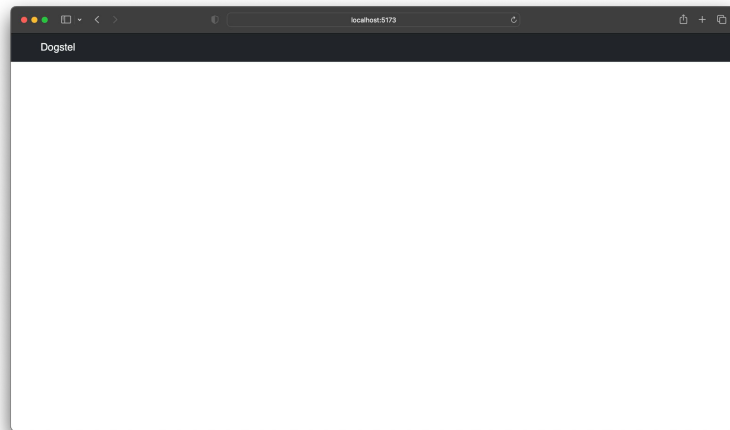


- React Router es un paquete disponible en NPM que nos permite dividir fácilmente una aplicación de React en distintas URLs.
- Con React Router podemos gestionar diferentes rutas enlazadas a diferentes vistas, mejorando de esta manera la usabilidad de nuestra aplicación y la experiencia de nuestros usuarios y podemos construir una SPA.

React Router

Setup del proyecto

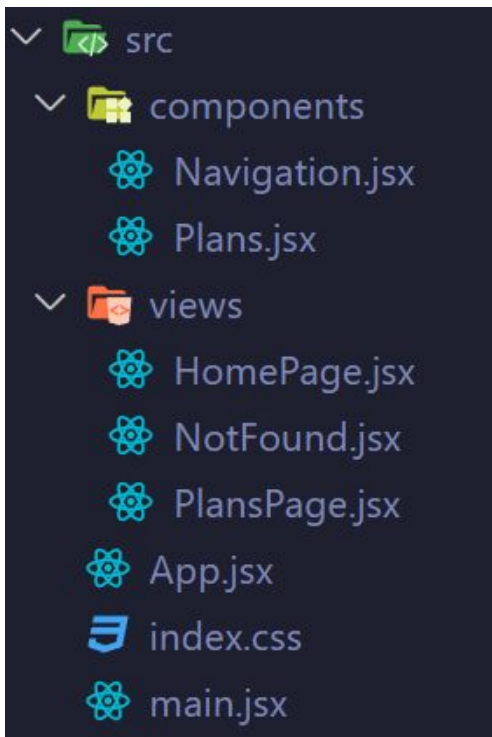
- Descarga el proyecto Dogstel desde el **Material de apoyo clase - React Router I.**
- Entra al proyecto desde el terminal e instala los paquetes con `npm install`.
- Levanta el proyecto con `npm run dev`.
- Entra con el navegador a `localhost:5173`.
- También te puedes apoyar del `README.md`



La aplicación de apoyo se trata de la página de un hotel de Perros

React Router

Revisemos el código



La carpeta views también contiene componentes, pero estos los entenderemos como Vistas en nuestro proyecto.

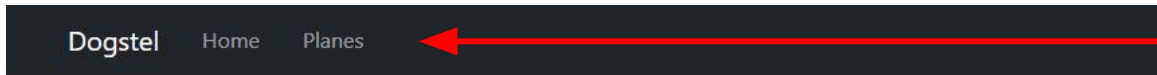
React Router

Mi primer enrutador

Una vez descargado el proyecto, instala por terminal el paquete de *react-router-dom*

```
npm install react-router-dom@6
```

El objetivo será crear una SPA (Single Page Application) con los componentes que ya tenemos a disposición.



Bienvenido a **Dogstel** 

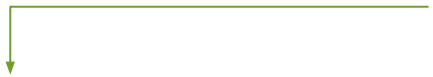
El mejor hotel para perros de la región

Construiremos un Navbar que
funcione con React Router

React Router

Mi primer enrutador

Levanta la aplicación y modifica el código del archivo main.jsx para importar y agregar el componente **BrowserRouter** del paquete react-router-dom y el componente Navbar.



BrowserRouter es el componente encargado de mantener actualizado el UI en función de la URL, este debe envolver a toda la aplicación. Todos los componentes se agregan dentro de BrowserRouter.

{desafío}
latam_

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";

import App from "../App.jsx";

import "bootstrap/dist/css/bootstrap.min.css";
import "../index.css";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

```

import { Container, Navbar } from "react-bootstrap";
import { Link } from "react-router-dom";

const Navigation = () => {
  return (
    <Navbar
      bg="dark"
      variant="dark"
    >
      <Container className="justify-content-start">
        <Navbar.Brand>Dogstel</Navbar.Brand>
        <Link
          to="/"
          className="text-white ms-3 text-decoration-none"
        >
          Home
        </Link>
        <Link
          to="/planes"
          className="text-white ms-3 text-decoration-none"
        >
          Planes
        </Link>
      </Container>
    </Navbar>
  );
};

export default Navigation;

```

React Router

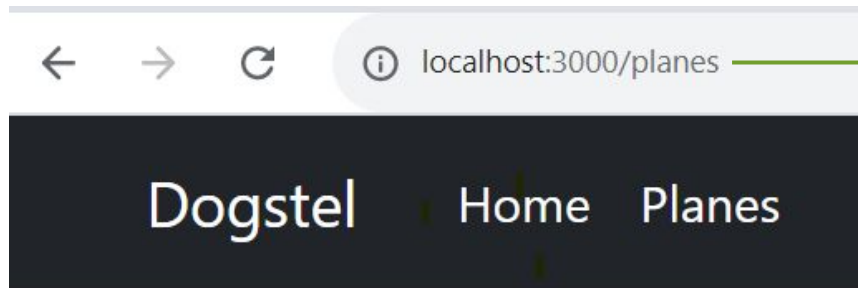
Mi primer enrutador

- Ahora, abramos el componente *Navigation* para agregar 2 Links: **Home** y **Planes**
- El componente **Link** funciona similar a la etiqueta de hipervínculo(a) de HTML.
- Una diferencia importante radica en que este componente produce el redireccionamiento con React Router y evita la recarga del navegador.

React Router

Mi primer enrutador

Luego, levantemos nuestra aplicación y probemos el Navbar presionando en la opción “Planes”.



Como se puede observar, la URL de la aplicación cambia en función a la ruta que accedemos.

Con este redireccionamiento evitamos tener que volver a consumir todos los recursos de nuestra aplicación al evitar que se recargue el navegador.

React Router se encargará de actualizar parcialmente el DOM en función a la ruta que consultemos.

Ejercicio

Crea una nueva aplicación de React y:

1. Modifica la estructura de carpetas y archivos según el ejemplo anterior.
2. Instala el paquete *react-router-dom*
3. Crea un componente Navbar con 2 componentes Links que dirijan a Registro y Login.
4. En el componente App, importa e incluye en el template el BrowserRouter y el Navbar .
5. Prueba tu enrutador confirmando que cambia la URL en función a la ruta que consultas.

Ejercicio ¡Manos al teclado!



*/** Crear un sistema de navegación basado en rutas.**/* 

*/** Crear diferentes rutas que utilicen componentes como vistas en la aplicación. **/*

*/** Crear una ruta por defecto enlazada a la vista Not Found. **/*

Objetivos

```
import { Route, Routes } from "react-router-dom";
import Navigation from "../components/Navigation";
```

```
import HomePage from "../views/HomePage";
import PlansPage from "../views/PlansPage";
```

```
const App = () => {
  return (
    <div>
      <Navigation />
      <Routes>
        <Route
          path="/"
          element={<HomePage />}
        />
        <Route
          path="/planes"
          element={<PlansPage />}
        />
      </Routes>
    </div>
  );
};

export default App;
```

React Router

Creación de vistas

- Ahora que nuestra aplicación puede redireccionar al usuario a diferentes rutas, es momento de definir en nuestro enrutador cuáles son los componentes que utilizará como vistas.
- Para esto, modifica el archivo App nuevamente agregando los componentes **Routes**, **Route** y las 2 vistas incluidas en el proyecto.

React Router

Creación de vistas

El componente **Routes** funciona como un envoltorio de rutas.

El componente **Route** define una ruta dentro de nuestro enrutador y tiene la siguiente estructura:

```
<Route path="/<ruta>" element={< Componente(vista) />} />
```

En donde el atributo **path** define la ruta y el atributo **element** el componente que se mostrará cuando coincida el **path** con la URL del navegador.

Aunque se escriben varios componentes **Route**, solo se renderiza 1.

React Router

Creación de vistas

Ahora probemos nuevamente nuestra aplicación y comprobemos que se muestran ambas vistas en ambas rutas:

Home

Dogstel Home Planes

Bienvenido a **Dogstel** 🐕

El mejor hotel para perros de la región

Planes

Dogstel Home Planes

Tenemos planes especiales para ti y su mascota

Plan - Básico



Plan - Estándar



Plan - Guauf



*/** Crear un sistema de navegación basado en rutas. **/* 

*/** Crear diferentes rutas que utilicen componentes como vistas en la aplicación. **/* 

*/** Crear una ruta por defecto enlazada a la vista Not Found. **/*

Objetivos

React Router

La ruta por defecto

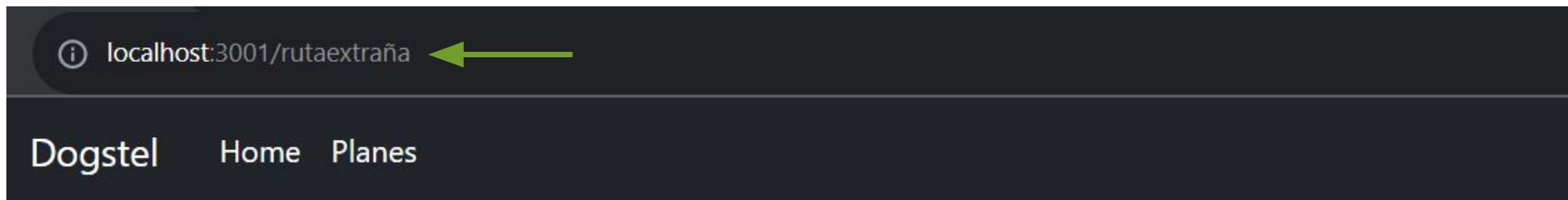
- La ruta por defecto nos ayudará a poder personalizar y devolver una vista en caso de que el usuario consulte nuestra página web con una URL que no coincida con ninguna de nuestras rutas.
- Para agregar la ruta por defecto, incluye la vista **NotFound** para que sea utilizado por un **Route** nuevo que tenga como path un asterisco: *

```
<Routes>
  <Route
    path="/"
    element={<HomePage />}
  />
  <Route
    path="/planes"
    element={<PlansPage />}
  />
  <Route
    path="*"
    element={<NotFound />}
  />
</Routes>
```

React Router

La ruta por defecto

Si consultamos nuestra aplicación con una ruta extraña o que simplemente no coincida con ningún componente **Route**, obtenemos la vista **NotFound**



La ruta que intentas consultar no existe :/

Ejercicio

En la aplicación creada en el primer ejercicio:

1. Crea una vista NotFound.
2. Agrega la ruta por defecto.
3. Comprueba en el navegador que la ruta por defecto funciona al consultar la aplicación con una URL diferente que no coincida con ninguna de las rutas creadas.

Ejercicio ¡Manos al teclado!



`/* Crear un sistema de navegación basado en rutas. */` ✓

`/* Crear diferentes rutas que utilicen componentes como vistas en la aplicación. */` ✓

`/* Crear una ruta por defecto enlazada a la vista Not Found. */` ✓

Objetivos



Cierre



¿Existe algún concepto que no
hayas comprendido?

Reflexionemos

- Revisemos el material de estudio asincrónico en donde repasamos los conceptos de esta clase.
- Revisemos el desafío de la unidad leyendo la descripción y requerimientos.

A vertical line separates the white left side from the blue right side. Along this line, there are several large, stylized symbols: a closing curly brace '}', an '@' symbol, an opening curly brace '{', and a smiley face ':D'.

¿Qué sigue?

¿Tienen alguna duda respecto al Desafío?



*Academia de
talentos digitales*

www.desafiolatam.com



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam