



**¡Les damos la
bienvenida!**

¿Comenzamos?

Esta clase va a ser

- grabada

Clase 09. REACT JS

Routing y navegación

Temario

08

Workshop: Hooks, Children y Patrones

- ✓ Custom Hooks
- ✓ Patrones

09

Routing y navegación

- ✓ [Organicemos
nuestra app](#)
- ✓ [React router](#)

10

Eventos

- ✓ Eventos
- ✓ Componentes
basados en
eventos

Objetivos de la clase

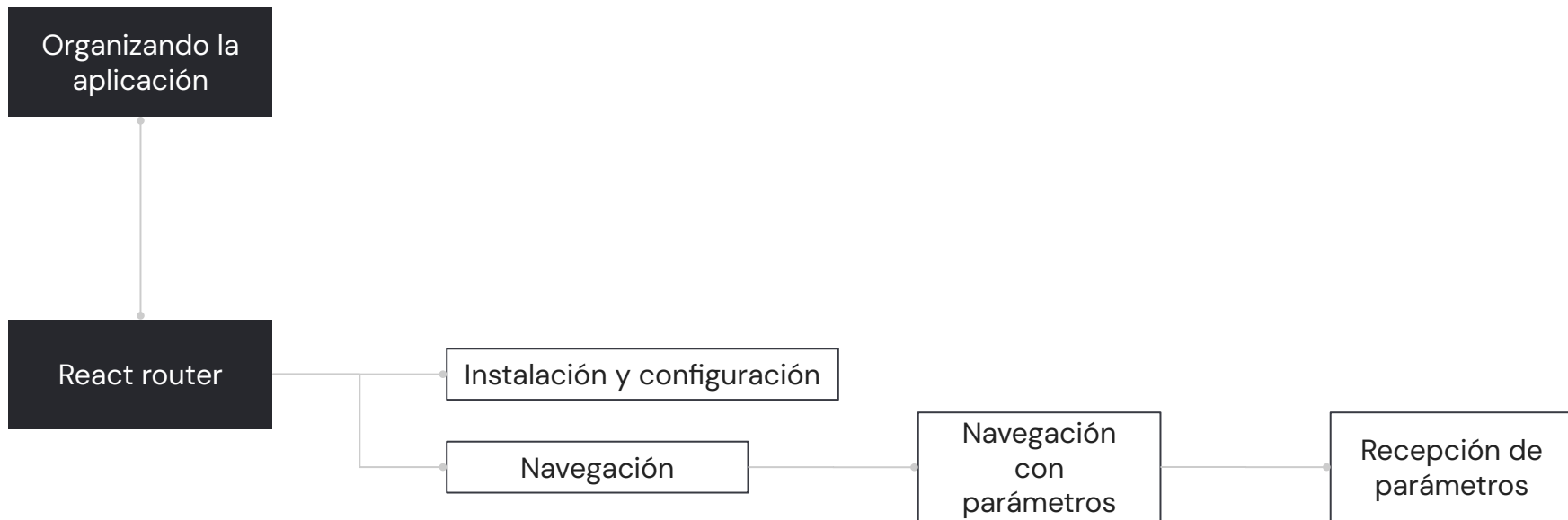


Organizar nuestra aplicación.



Configurar la navegabilidad entre componentes.

MAPA DE CONCEPTOS



Organicemos nuestra app

La facilidad con la que nuestra aplicación permite agregar funcionalidades y navegarlas es un factor clave en términos de experiencia y escalabilidad.



La buena navegabilidad permite a...

- ✓ **Users:** entender dónde están parados y guardar favs/marcadores a secciones en las que tienen interés
- ✓ **Navegadores:** Permitir controlar las acciones de ir adelante y volver, y conocer el nav history
- ✓ **Crawlers:** Entender la estructura de la app y proveer acceso optimizado/visibilizado a las distintas secciones

Organizando la app

Preguntémonos

¿Cuál es el punto de inicio de nuestra app?

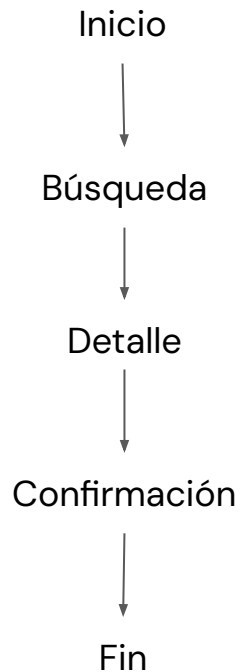
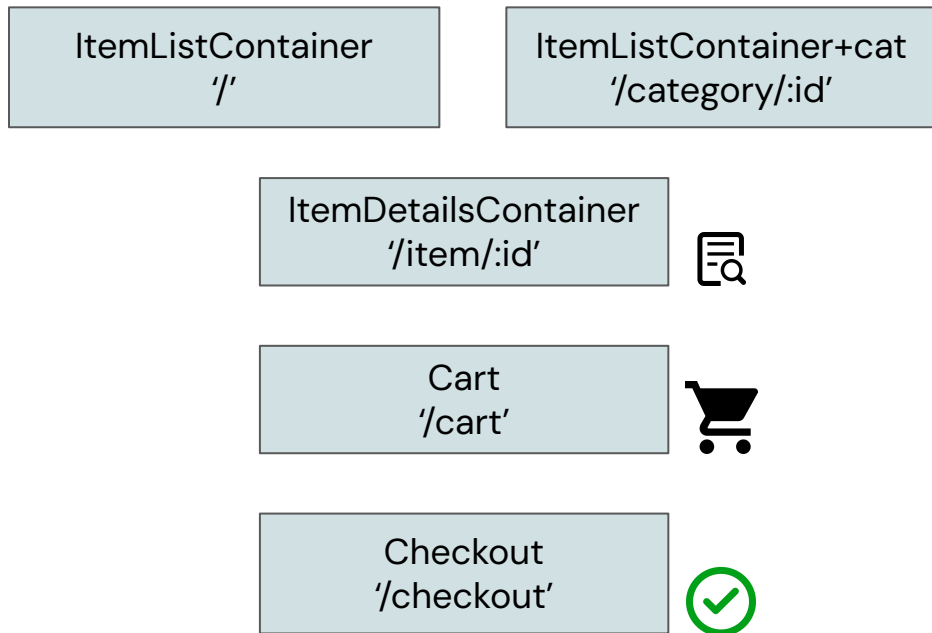


Paso a paso



Routing

Routing



React router

Instalación

Instalación

```
npm install react-router-dom
```

Por defecto, React no viene con un mecanismo integrado de navegación.

Esto es para mantener sus dependencias al mínimo y dado que no todo proyecto necesita routing, se maneja como una dependencia aparte.

Hay varias soluciones, pero hoy instalaremos:
react-router-dom

Versión react router

Si bien instalaremos react-router-dom publicado en NPM, veremos varias versiones:

- ✓ react-router => librería core (no instalar).
- ✓ react-router-dom => para routing en el browser.
- ✓ react-router-native => para routing en react-native.

Y algunas otras...

Configurar nuestra app con el router

Importar

Una vez instalado, importaremos el módulo desde react-router-dom:



```
import { BrowserRouter, Routes, Route } from "react-router-dom";
```

Agregar la funcionalidad a toda la app

Una vez realizado el import necesitamos configurar dos cosas:

1. Wrappear (envolver) la aplicación en un **BrowserRouter**
2. Crear un **Routes** (donde proyectaremos las vistas navegadas)
3. Crear los **Route's** de las distintas navegaciones con sus componentes asociados

```
import { BrowserRouter, Routes, Route } from "react-router-dom";

export default function App() {
  return (
    <BrowserRouter>
      <Navbar />
      <Routes>
        <Route exact path="/" element={<Home />} />
        <Route exact path="/pokemons" element={<Pokemons />} />
        <Route exact path="/pokemon/:pokemonId" element={<Pokemon />} />
        <Route exact path="/category/:categoryId" element={<Category />} />
        <Route path="/pokemons/*" element={<h1>Wildcard</h1>} />
      </Routes>
    </BrowserRouter>
  );
}
```

Especificidad de match

Por defecto se matchean (coinciden) únicamente partes de la url, por lo tanto '/' va a matchear '/cart' o '/checkout', a no ser que le digamos que use la propiedad exact:

```
<Routes>
  <Route exact path="/" element={<Home />} />
  <Route exact path="/pokemons" element={<Pokemons />} />
  <Route exact path="/pokemon/:pokemonId" element={<Pokemon />} />
  <Route exact path="/category/:categoryId" element={<Category />} />
  <Route path="/pokemons/*" element={<h1>Wildcard</h1>} />
</Routes>
```



Ejemplo en vivo

Vamos al código



Agregar un router a tu app

En tu aplicación, instalá react-router-dom.

Duración: **15 minutos**



ACTIVIDAD EN CLASE

Agregar un router a tu app

Descripción de la actividad.

En tu aplicación, instala react-router-dom, agrégala al root de tu app, y configura tus rutas apuntando a tu Home. Si tienes otro nombre, o la organizaste distinta, no hay problema.



Break

¡10 minutos y volvemos!

Navegar a una ruta

Navegar a una ruta

Ahora que tenemos todo configurado, podemos importar un `link` perteneciente a `react-router-dom`, en cualquier componente del sub-árbol del `<BrowserRouter>`,

Usarlo para que al clickear, el `BrowserRouter` renderice ese `Route` que habíamos declarado dentro del `Switch`.

```
components > Navbar > JS Navbar.js > ...  
import React from 'react';  
import { Link } from 'react-router-dom';
```

```
<li><Link to={`/cart`} >Carrito</Link></li>
```

**Navegar a una ruta (con
parámetros)**

Navegar a una ruta (con parámetros)

Si hacemos la ruta dinámica (con parámetros), podremos navegarla idénticamente, pero de manera dinámica.

```
<Routes>
  <Route exact path="/" element={<Home />} />
  <Route exact path="/pokemons" element={<Pokemons />} />
  <Route exact path="/pokemon/:pokemonId" element={<Pokemon />} />
  <Route exact path="/category/:categoryId" element={<Category />} />
  <Route path="/pokemons/*" element={<h1>Wildcard</h1>} />
</Routes>
```

Navegar a una ruta – NavLink

¿Notaron que cambiamos el Link por un NavLink?

```
t => <li><NavLink to={` /categories/${cat.id}`} activeClassName="currentCategory" className="text-white">{cat.name}</NavLink>
```

Un NavLink es un link con un estilo, está siempre detectando la ruta actual, y si coincide con la suya nos activa la clase que le demos para que el user sepa qué item de la lista corresponde con la vista actual.

Recibir parámetros por ruta

Recibir parámetros por ruta

React router provee integración con Hooks.

useParams:

- ✓ Lo podemos utilizar para leer en js los parámetros de la ruta.
- ✓ En combinación con un useEffect, nos sirve para obtener actualizaciones sobre los parámetros.

```
function ComponentWithRouteParams() {  
  const { userId } = useParams();  
  useEffect(() => {  
    console.log('Received userId to: ', userId);  
    return () => {  
      console.log('Will change userId:', userId);  
    }  
  }, [userId]);  
}
```



Ejemplo en vivo

Vamos al código



Segunda pre-entrega de tu Proyecto final

Durante esta entrega, se hará una revisión integral del estado actual de avance de tu proyecto.



Segunda pre-entrega

Formato: Link a último commit de git donde se cumplan los objetivos + GIF mostrando la navegabilidad. Debe tener el nombre "PreEntrega2+Apellido".

Consigna.

Configura en App.js el routing usando un BrowserRouter de tu aplicación con react-router-dom

Componentes:

1. Navbar con cart
2. Catálogo
3. Detalle de producto

Objetivos.



Deberás desarrollar la navegabilidad básica de la aplicación, demostrando que la app permite ver el catálogo, y navegar a un detalle.



Segunda pre-entrega

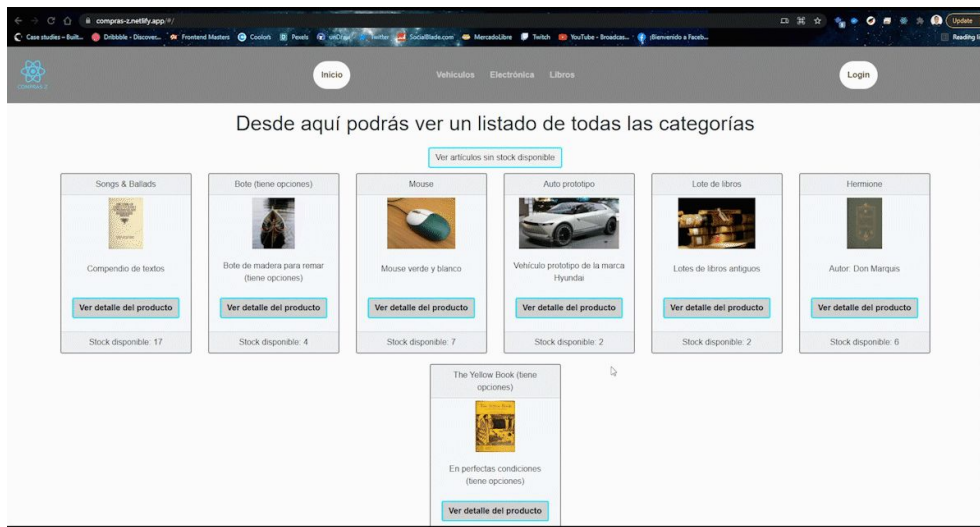
Se debe entregar.

- ✓ **Rutas a configurar**
'/' navega a <ItemListContainer />
'/category/:id' <ItemListContainer />
'/item/:id' navega a <ItemDetailContainer />
- ✓ **Links a configurar**
Clickear en el brand debe navegar a '/'
Clickear un Item.js debe navegar a '/item/:id'
Clickear en una categoría del navbar debe navegar a '/category/:categoryId'
- ✓ Para finalizar deberá integrar los parámetros de tus async-mocks para reaccionar a :itemId y :categoryId utilizando efectos y los hooks de parámetros que vimos en clase! Si te encuentras en una categoría deberías poder detectar la navegación a otra categoría y volver a cargar los productos que correspondan a dicha categoría.



Segunda pre-entrega

Ejemplo GIF:



Notas:

- ✓ No usar HashRouter como en el ejemplo del gif (usar BrowserRouter)
- ✓ Utilizar el id de la categoría como nombre en la URL param en vez de números (vehículos, por ej)
- ✓ Utilizar el id del item como URL param

¿Preguntas?

Resumen de la clase hoy

- ✓ Organización de navegabilidad
- ✓ Routing estático
- ✓ react-router-dom

Opina y valora
esta clase

Muchas gracias.

#DemocratizandoLaEducación