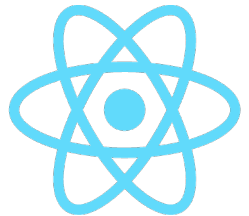


# Desarrollo y programación de Aplicaciones con React

## 7. Paginación y routing

**CORE**  
*networks*



# Paginación y routing

## Single Page Application en React

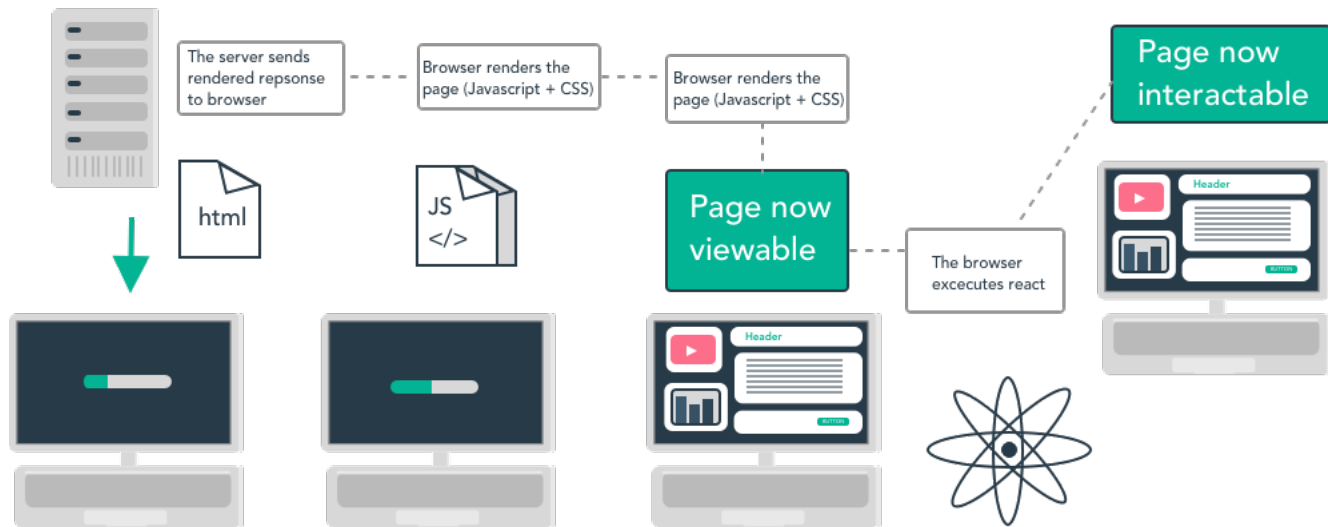
Una Single-Page Application (SPA) es un tipo de aplicación web que ejecuta todo su contenido en una sola página.

El navegador carga todo el contenido HTML, CSS y JavaScript tras la respuesta a la primera petición al servidor y de acuerdo al flujo de navegación del usuario, va renderizando dinámicamente cada componente de IU de una manera ágil y rápida, favoreciendo así la UX.

Este mecanismo de construcción de vistas en el navegador se engloba dentro del llamado *client-side rendering* frente al tradicional *service-side rendering*.

# Paginación y routing

## Single Page Application en React



# Paginación y routing

Single Page Application en React

La implementación de SPA en React se lleva a cabo mediante la librería React Router.

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

# Paginación y routing

El elemento fundamental de la librería se implementa con el componente `BrowserRouter` en `index.js`

```
ReactDOM.render(  
  <BrowserRouter>  
    <React.StrictMode>  
      <App />  
    </React.StrictMode>  
  </BrowserRouter>,  
  document.getElementById('root')  
)
```

# Paginación y routing

Y en el componente raíz se establece un componente Routes que incluye los componente Route con cada ruta de la url que especifica qué componente será cargado en cada momento.

```
<Routes>
```

```
  <Route path="/" element={<Inicio />} />
```

```
...
```

# Paginación y routing

Los eventos para navegar modificando la ruta de la url se definen con el componente Link.

```
<Routes>  
  <Route path="/" element={<Inicio />} />  
  ...
```

```
<Link to="/soporte">  
  <button type="button">Soporte</button>  
</Link>
```

# Paginación y routing

Podemos recoger las rutas erróneas con la máscara asterisco situándola al final de todas las rutas.

```
<Route path="*" element={  
  <div className="container">  
    <h1>Lo sentimos la página no existe</h1>  
    <Link to="/"><button>Volver a inicio</button></Link>  
  </div>  
} />
```



# Paginación y routing

Es posible definir componentes anidados para modular la carga de la aplicación mediante lazy loading.

...

```
<React.Suspense fallback={<>...</>}>  
  <InicioVentas /> // Componente-módulo  
</React.Suspense>
```

# Paginación y routing

Y el componente con carga Lazy Loading tendrá a su vez otro componente Routes con los componentes de ese componente-módulo.

```
<Routes>
```

```
  <Route path="/" element={<Outlet />}>
```

```
    <Route index element={<InicioVentasLayout />} />
```

```
    <Route path="tabla-clientes" element={<TablaClientes />} />
```

```
  ...
```

# Paginación y routing

Las rutas pueden definir parámetros:

```
<Route path=":cif" element={<VisualizarCliente />} />
```

...

```
<Link to={`/ventas/visualizar-cliente/${cliente.cif}`}>Visualizar</Link>
```

# Paginación y routing

Se puede implementar navegación programática con el método `navigate()`.

```
const handleSubmit = (e) => {  
  ...  
  navigate("/ventas/tabla-clientes");  
}
```