Curso de React.js AVANZADO

# ¿Qué es React.js?

Es un sistema de plantillas para gestionar las vistas que utiliza un lenguaje de marcado llamado JSX, similar a HTML  
.  
**Características básicas de react**  
.

* Está basado en **componentes**: Toda la UI se divide en elementos más pequeños llamados componentes, en react, todo es un componente,
* Es [**declarativo**](https://dev.to/itsjzt/declarative-programming--react-3bh2): **expresa la lógica de un cálculo sin describir su flujo de control**, es un estilo de programación en el que el que se define “qué” es la solución sin importar cómo se llegó a ella.

# Proyecto y tecnologías que usaremos

En este curso realizaremos una aplicación muy parecida a Instagram, llamada petgram. Tendremos nuestras rutas, gestión de usuarios y likes.

Utilizaremos como empaquetador y transpilador:

* [Webpack](https://platzi.com/cursos/webpack/)
* [Babel](https://babeljs.io/)

Estilado con CSS en JS con:

* [styled-components](https://www.styled-components.com/)

Como linter utilizaremos:

* [Standard JS](https://standardjs.com/)

Para fetching (obtención) de datos:

* [GraphQL](https://platzi.com/cursos/graphql/)
* [React Apollo](https://www.apollographql.com/docs/react/)

Para el enrutado de la SPA utilizaremos:

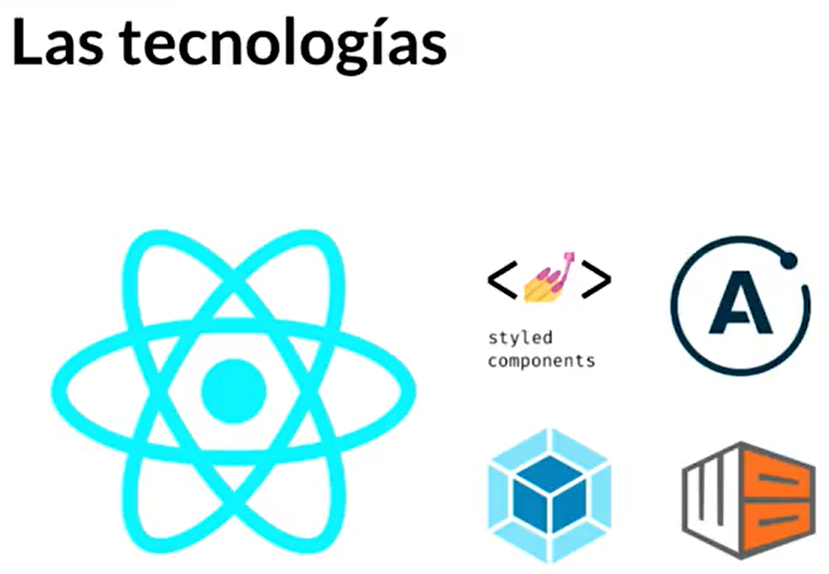
* [Reach Router](https://reach.tech/router)

Para las buenas prácticas utilizaremos:

* [Lighthouse](https://developers.google.com/web/tools/lighthouse/?hl=es)
* [Cypress](https://www.cypress.io/)

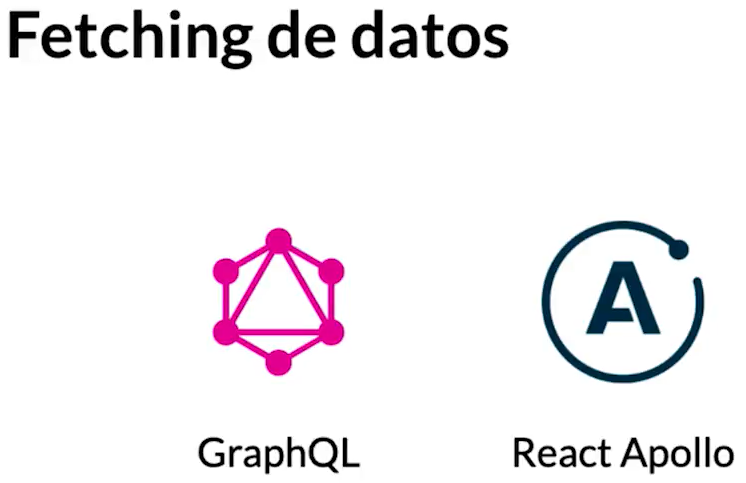
Por último haremos SEO, PWA y Deploy con:

* [React Helmet](https://github.com/nfl/react-helmet)
* [Workbox](https://developers.google.com/web/tools/workbox/)
* [Progressive Web App](https://platzi.com/cursos/pwa-react-js/)
* [Deply con Now](https://platzi.com/cursos/deploy-now/)



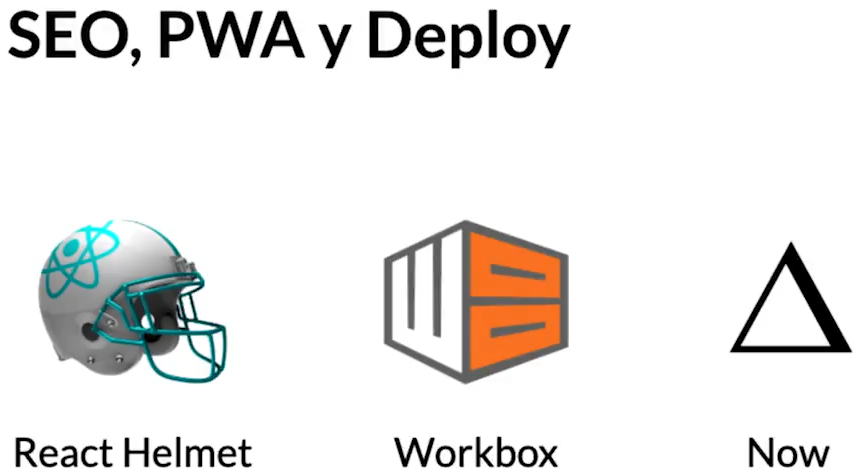












# Clonando el repositorio e instalando Webpack

Pasos para iniciar nuestro proyecto:

* **Paso 1:** Vamos a clonar el repositorio desde [github.com/midudev/curso-platzi-react-avanzado](https://github.com/midudev/curso-platzi-react-avanzado) usando git clone URL\_DEL\_REPO en nuestra consola.
* **Paso 2:** Vamos a instalar webpack y webpack-cli como dependencias de desarrollo con: npm i webpack wepack-cli --save-dev.
* **Paso 3:** Crearemos una carpeta llama src y dentro de ella un archivo index.js en el cual colocaremos solo un console.log('Empezamos el curso!').
* **Paso 4:** Crearemos en el root de nuestro proyecto un archivo webpack.config.js el cual tendrá toda la configuración de webpack
* **Paso 5:** Instalaremos html-webpack-plugin con: npm i html-webpack-plugin --save-dev.
* **Paso 6:** Instalaremos webpack-dev-server con: npm i webpack-dev-server --save-dev.
* **Paso 7:** Añadiremos un nuevo script llamado dev: "dev": "webpack-dev-server".

// webpack.config.js

**const** HtmlWebpackPlugin = require("html-webpack-plugin")

module.exports = {

output: {

filename: 'app.bundle.js'

},

plugins: [

**new** HtmlWebpackPlugin()

]

}

Nota:

Comando que corregir

npx webpack --entry ./src/index.js

"dev": "webpack serve --open",

# Instalación de React y Babel

En esta clase vamos a configurar React instalando las dependencias npm i react react-dom y Babel para poder transpilar código *jsx* a JavaScript Vanilla con: npm i @babel/core @babel/preset-env babel-loader @babel/preset-react --save-dev.

Ahora añadiremos en nuestro webpack.config.js lo siguiente:

// webpack.config.js

{/\*...\*/}

module.exports = {

{/\*...\*/}

module: {

rules: [

{

test: /\.js$/,

exclude: /node\_modules/,

use: {

loader: 'babel-loader',

options: {

presets: [

'@babel/preset-env',

'@babel/preset-react'

]

}

}

}

]

}

}