

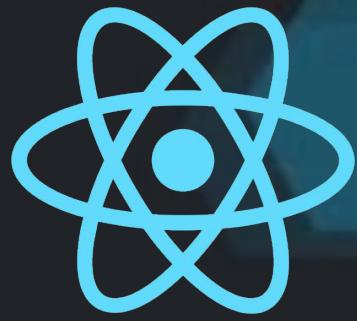


Integrando consultas entre un
Front End de React y un Back End
de Node

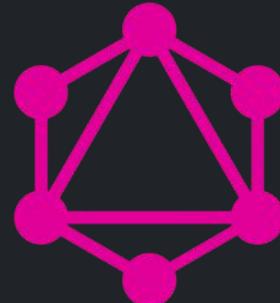
SERGIO GARZÓN



Desarrollador de Software- Desarrollador de Videojuegos - Docente de Programación



React



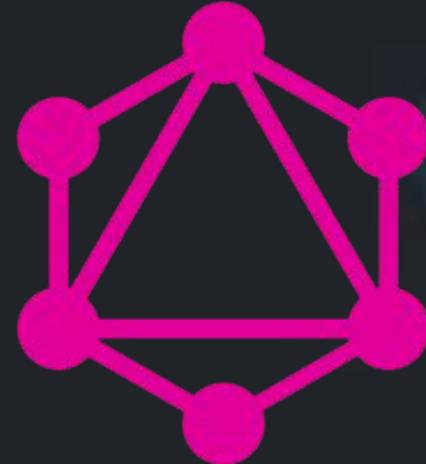
GraphQL



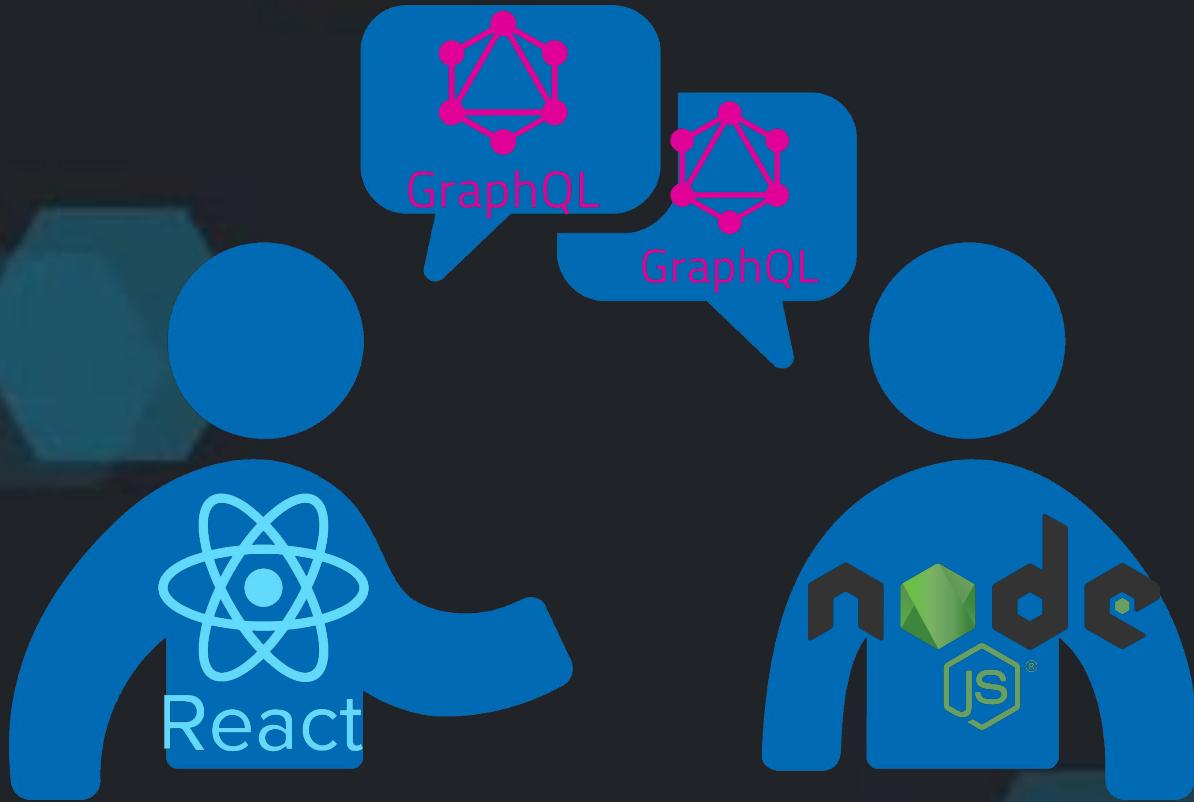
Cómo lograr una integración efectiva entre ambas capas, con GraphQL, para optimizar la comunicación y estructura de nuestras aplicaciones.

GraphQL:

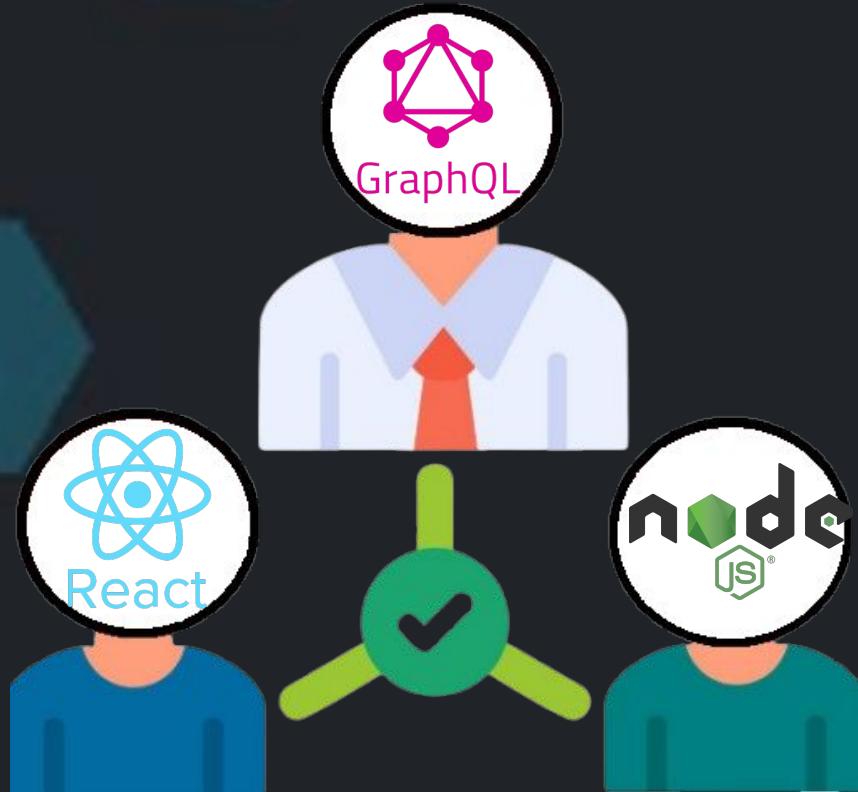
- Desarrollado por  (2015)
- Código abierto (Open source)
- Lenguaje de consultas para lectura y mutación de datos en APIs
- Alternativa de REST API
- Graph (Grafo) QL (Lenguaje de consulta)
- Es declarativo



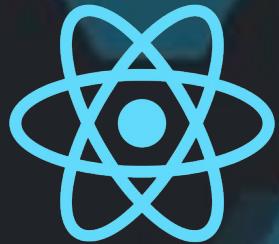
GraphQL



GraphQL es un lenguaje de comunicación



GraphQL es un intermediario



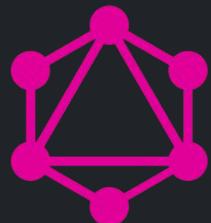
React



Cliente: Puede ser cualquier framework o librería

GraphQL:

- El cliente especifica que que el servidor devuelva lo que necesite.
- Reduce over-fetching (muchos datos) y under-fetching (pocos datos)



GraphQL

GraphQL:

- Especificamos y nos devuelve de manera correcta lo que queremos
- Una aplicación puede devolver id, título, fecha de finalización, pero podemos decirle que no devuelva la fecha de finalización.

```
{  
  aplicacion {  
    id,  
    titulo  
  }  
}
```

```
"datos": {  
  "aplicacion": [  
    {"id": 1,  
     "titulo" : "BeerJS"},  
    {"id": 1,  
     "titulo" : "BeerJS 100"}  
  ]  
}
```

REST API:

- El cliente hace una solicitud get (por ejemplo a /usuarios) y trae todo.
- Obviamente está más estandarizado, es más fácil de entender.

{REST API}

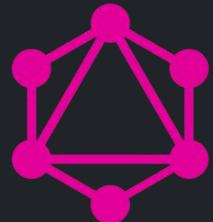
Transferencia de datos:

Ambos utilizan JSON para la transferencia de datos

{JSON}

Mutaciones para modificación de datos

- Es mucho mejor en GraphQL (más flexibles, más eficientes que REST API)
- Único endpoint



GraphQL

{REST API}

Mutaciones en GraphQL

Sin devolución de información

```
mutation {
  crearUsuario ( nombre : "valor" , email : "valor email") {
    id
    nombre
    email
  }
}
```

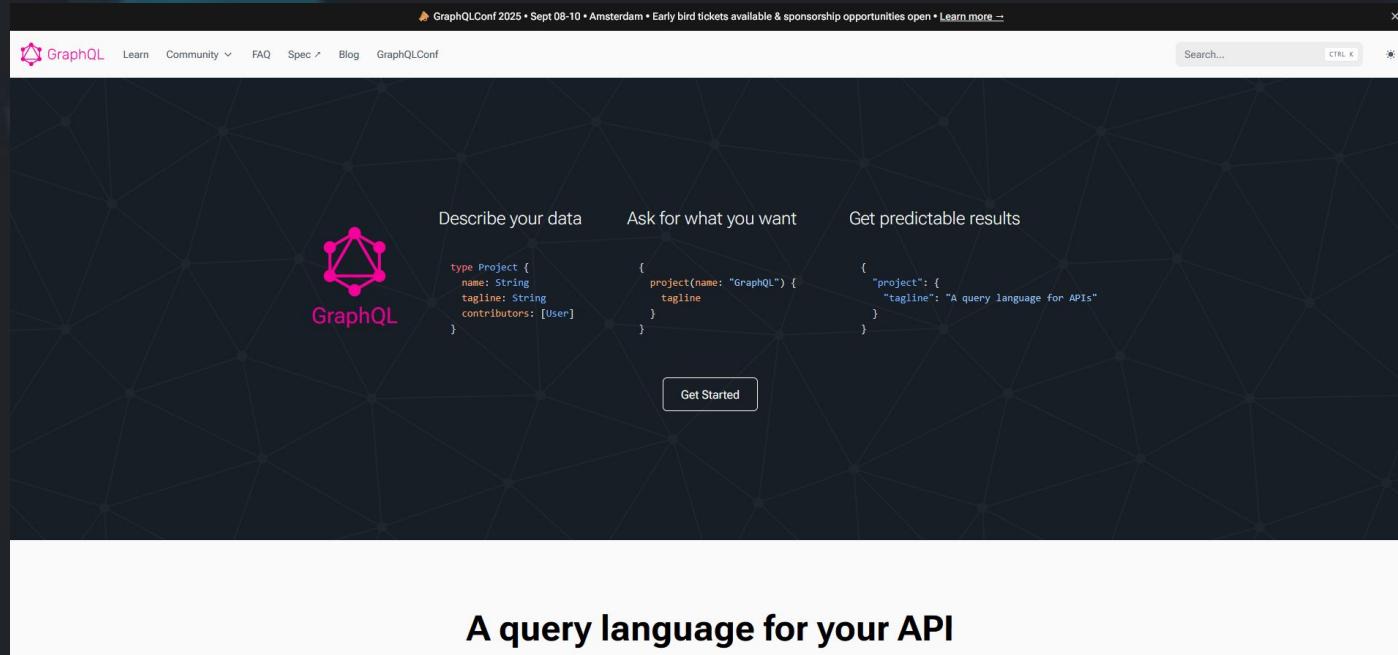
Mutaciones en GraphQL

Con devolución de información

```
{  
  "datos" : {  
    "crearUsuario" : {  
      "id" : 1,  
      "nombre" : "valor",  
      "email" : "valor email"  
    }  
  }  
}
```

GraphQL:

Sitio web: <https://graphql.org/>



The screenshot shows the official GraphQL website homepage. The header includes the GraphQL logo, navigation links for Learn, Community, FAQ, Spec, Blog, and GraphQLConf, and a search bar. A banner at the top right promotes the GraphQLConf 2025 event. The main visual features a dark background with a network graph pattern. On the left, there's a large GraphQL logo icon and the word "GraphQL". Three sections are displayed: "Describe your data" (with a schema snippet), "Ask for what you want" (with a query snippet), and "Get predictable results" (with a response snippet). A "Get Started" button is located below the first two sections. At the bottom, a white callout box contains the text "A query language for your API".

GraphQLConf 2025 • Sept 08-10 • Amsterdam • Early bird tickets available & sponsorship opportunities open • [Learn more →](#)

GraphQL Learn Community ▾ FAQ Spec ▾ Blog GraphQLConf

Search... CTRL K

Describe your data

Ask for what you want

Get predictable results

GraphQL

```
type Project {  
  name: String  
  tagline: String  
  contributors: [User]  
}  
  
{  
  project(name: "GraphQL") {  
    tagline  
  }  
}  
  
{  
  "project": {  
    "tagline": "A query language for APIs"  
  }  
}
```

Get Started

A query language for your API

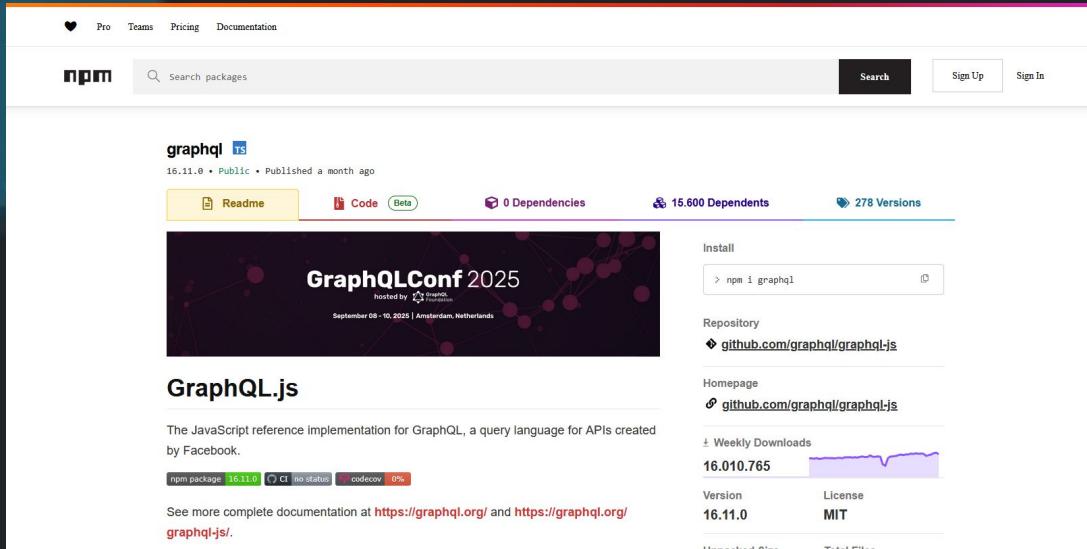
GraphQL:

Para instalar Graphql, en la página tenemos el comando “npm install graphql –save”

The screenshot shows the GraphQL.js Tutorial website. At the top, there's a navigation bar with links for Learn, Community, FAQ, Spec, Blog, and GraphQLConf. A banner at the top right mentions "GraphQLConf 2025 • Sept 08-10 • Amsterdam • Early bird tickets available". The main content area has a sidebar on the left with a pink header titled "Getting Started". The sidebar contains links to "Running Express + GraphQL", "GraphQL Clients", "Basic Types", "Passing Arguments", "Object Types", "Mutations and Input Types", and "Authentication & Middleware". Below this, there's a section titled "Advanced Guides" with a link to "Constructing Types". At the bottom of the sidebar is a link to "API Reference". The main content area has a breadcrumb trail "GraphQL.JS Tutorial > Getting Started" and a large title "Getting Started With GraphQL.js". Underneath, there's a section titled "Prerequisites" with text about Node.js requirements and a link to "functions". Further down, there's a section with instructions for creating a new project and installing GraphQL.js, followed by a code block showing the command "npm init" and "npm install graphql --save". At the very bottom, there's a section titled "Writing Code".

GraphQL:

También podemos encontrarlo en la página de npm (<https://www.npmjs.com/>)



The screenshot shows the npm package page for `graphql`. At the top, there's a navigation bar with links for `Pro`, `Teams`, `Pricing`, and `Documentation`. Below that is the npm logo and a search bar with the placeholder "Search packages". To the right of the search bar are "Sign Up" and "Sign In" buttons.

The main content area features the package name `graphql` in bold, with a `ts` badge indicating it's typed. It shows the version `16.11.0`, status as `Public`, and a publish date of "Published a month ago". Below this are tabs for `Readme` (selected), `Code` (Beta), `0 Dependencies`, `15.600 Dependents`, and `278 Versions`.

A large banner image for `GraphQLConf 2025` is displayed, featuring a dark background with a network of nodes and lines. The text "GraphQLConf 2025" is in white, along with "hosted by GraphQL.org" and "September 09 - 10, 2025 | Amsterdam, Netherlands".

The package description is titled "GraphQL.js". It states: "The JavaScript reference implementation for GraphQL, a query language for APIs created by Facebook." Below this is a row of badges: "npm package 16.11.0", "CI no status", and "codecov 0%".

Links for the repository (`github.com/graphql/graphql.js`) and homepage (`github.com/graphql/graphql.js`) are provided. A chart shows "Weekly Downloads" at 16,010,765. The package details section includes the version `16.11.0`, license `MIT`, and metrics for "Unpacked Size" and "Total Files".



`npm install graphql`

Apollo-Client:

Podemos instalar varias librerías para que nos ayuden a consumir la API GraphQL, la más popular es Apollo-Client



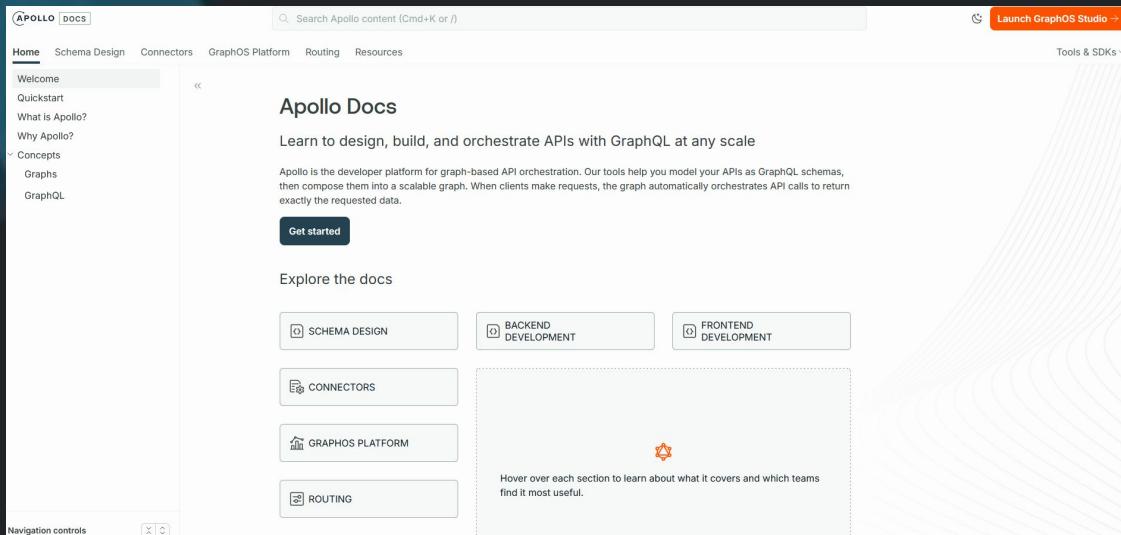
Apollo-Client:

GraphQL realiza la petición, apollo la recibe y gestiona esa solicitud con el backend



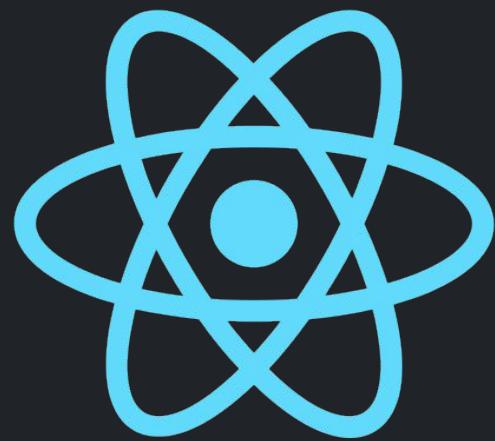
Apollo-Client:

Sitio web: <https://www.apollographql.com/docs>



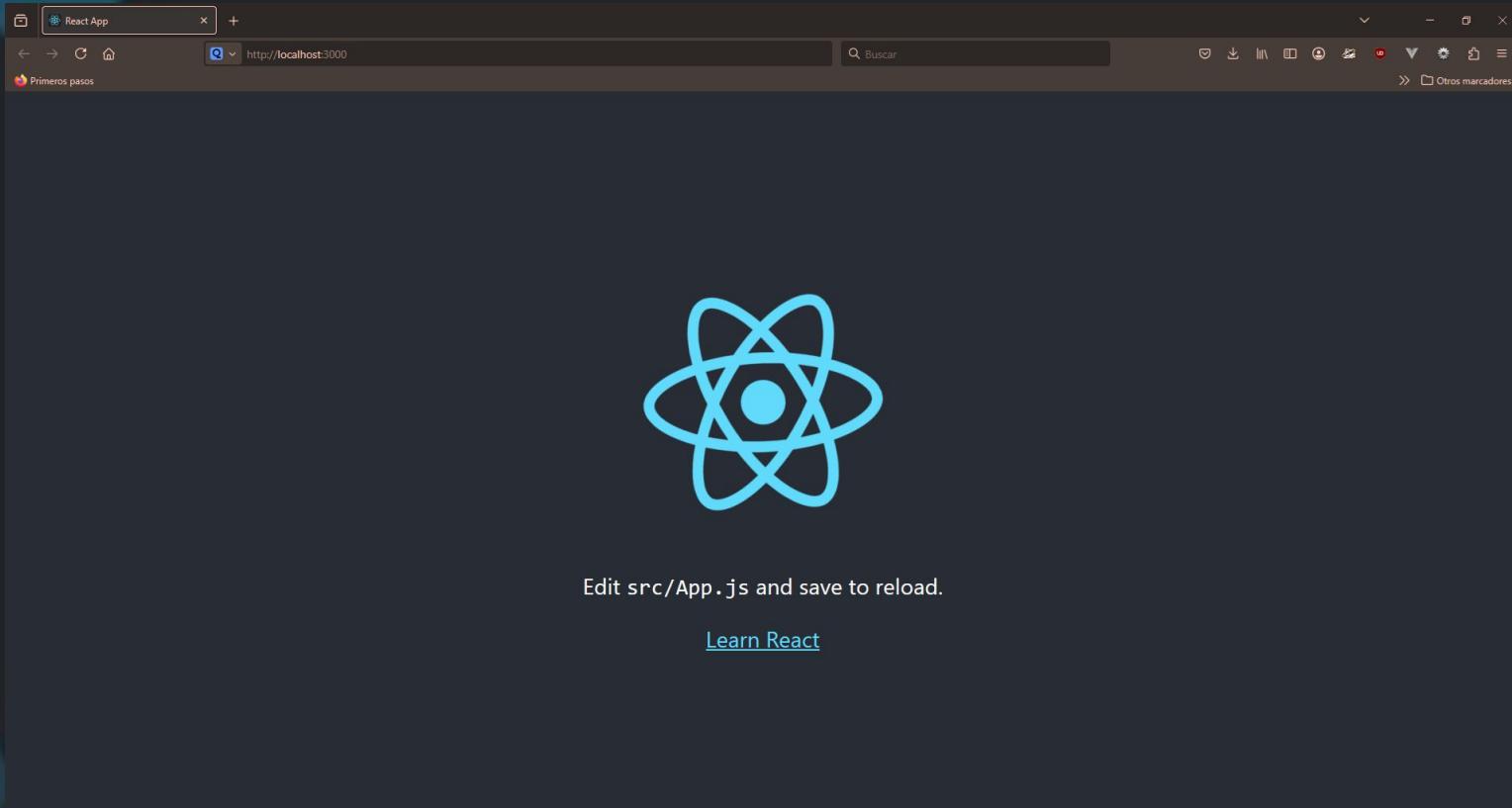
npm install -g apollo-client

APLICACIÓN FRONT END DE REACT CONVENCIONAL

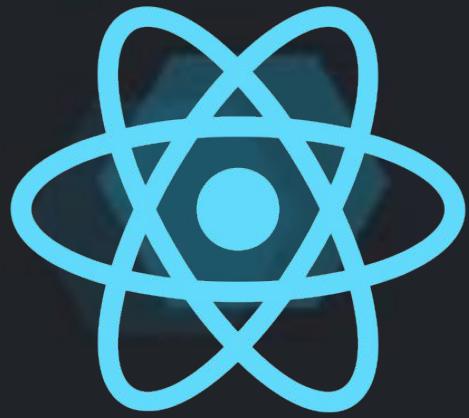


npx create-react-app proyectobeerjs

APLICACIÓN FRONT END DE REACT CONVENCIONAL



APLICACIÓN FRONT END DE REACT CON EL TEMPLATE DE VITE



React

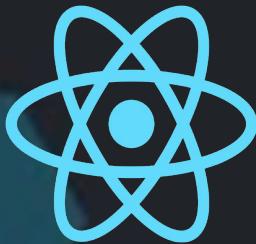


VITE



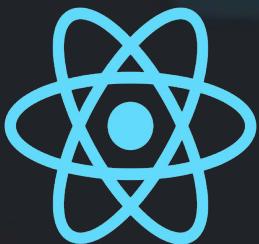
```
npm create vite@latest proyectobeerjs -- --template react
```

APLICACIÓN DE REACT CONVENCIONAL Y CON VITE



React

Más lento y menor configuración



React

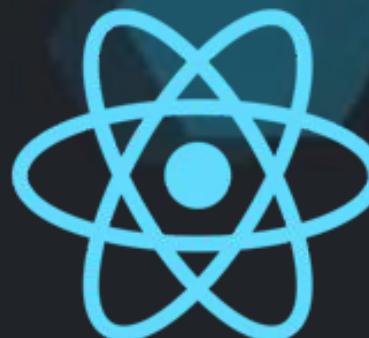


VITE

Más rápido y un poquito de configuración

OTRA ALTERNATIVA QUE VIENE ES REACTQL

Ya viene incorporado GraphQL en React, y también incorpora TypeScript



+



+



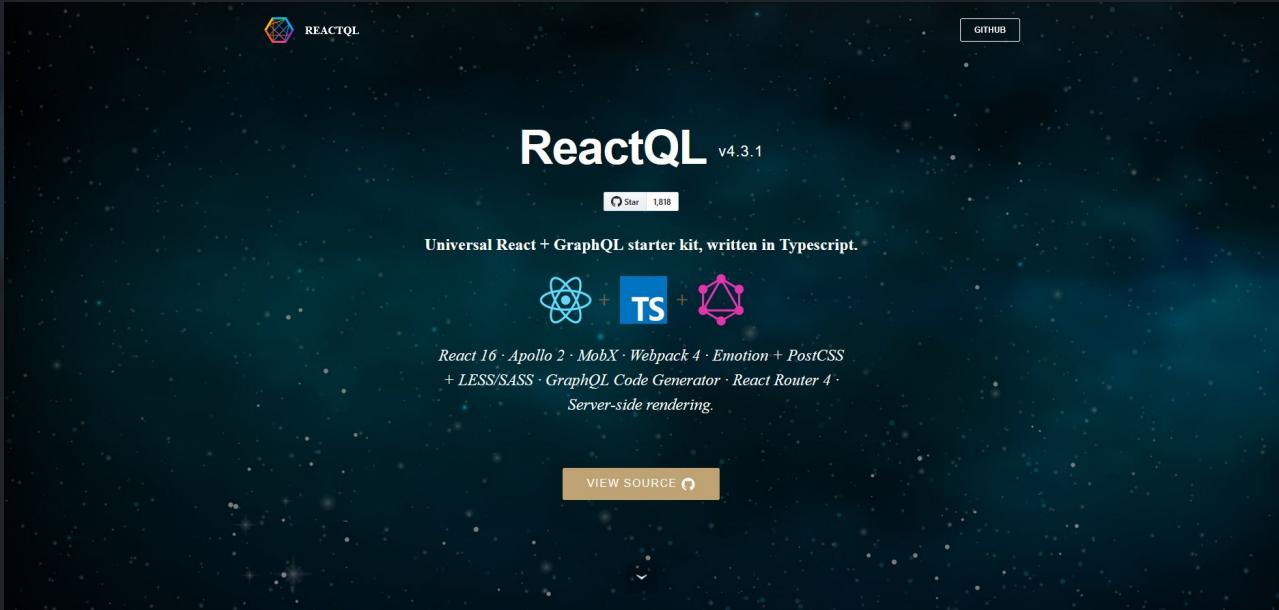
VIENE OTRA ALTERNATIVA QUE ES REACTQL

Y también viene Apollo Client



VIENE OTRA ALTERNATIVA QUE ES REACTQL

Sitio web: <https://reactql.js.org/>



npm install -g reactql

VIENE OTRA ALTERNATIVA QUE ES REACTQL



REACTQL

Message from GraphQL server: *Hello from graph.cool!*

Currently loading?: nope

- [Home](#)
 - [About](#)
 - [Contact](#)
-

Changed route: about

Runtime info:

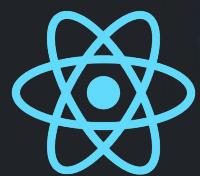
- Environment: *development*
 - Running: *In the browser*
-

Stylesheet examples:

Styled by CSS

Styled by SASS

React + GraphQL + Apollo Client + NodeJS + ExpressJS:



React



GraphQL



APOLLO



ex

Ejemplo de código en React con GraphQL

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { ApolloClient, InMemoryCache } from '@apollo/client';

const client = new ApolloClient({
  uri: 'http://localhost:3000',
  cache: new InMemoryCache(),
});

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <ApolloProvider client={client}>
      <App />
    </ApolloProvider>
  </React.StrictMode>
);

reportWebVitals();
```

```
import React from 'react';
import { gql, useQuery } from '@apollo/client';

const GET_USERS = gql`query GetUsers {
  users {
    id
    title
  }
}`;

function UserList() {
  const { loading, error, data } = useQuery(GET_USERS);

  if (loading) return <p>Cargando usuarios...</p>;
  if (error) return <p>Error al cargar usuarios: {error.message}</p>;

  return (
    <div>
      <h2>Lista de Usuarios</h2>
      {data.users && data.users.length > 0 ? (
        <ul>
          {data.users.map((user) => (
            <li key={user.id}>
              <strong>ID:</strong> {user.id} - <strong>Titulo:</strong> {user.title}
            </li>
          ))}
        </ul>
      ) : (
        <p>No se han encontrado usuarios.</p>
      )}
    </div>
  );
}
```



¿ ¿ ¿ Preguntas ???

Muchas Gracias



[Sergio Gabriel Garzón](#)



[sergiog90arg](#)