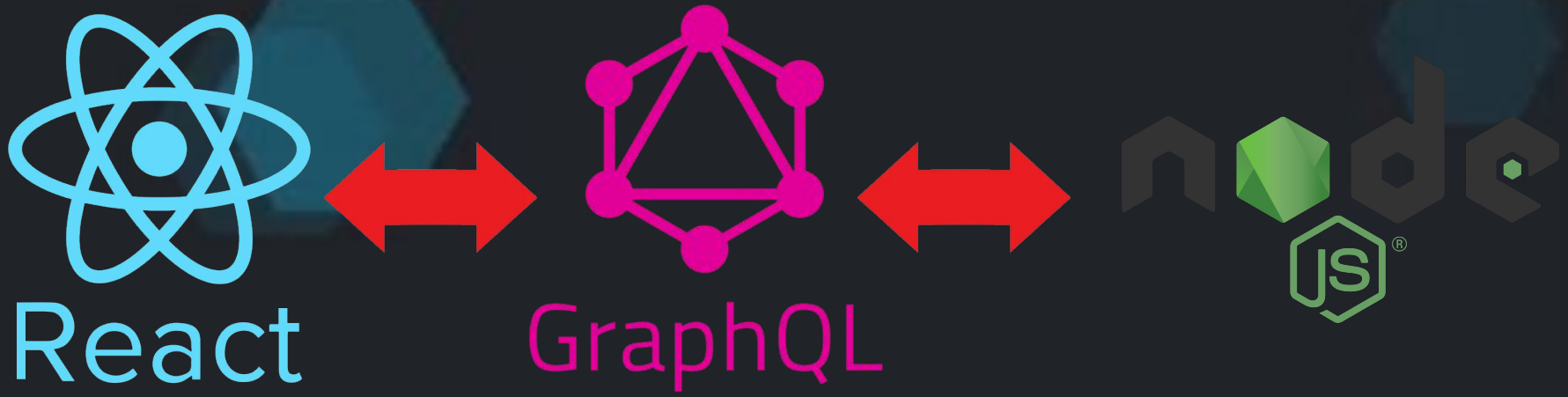

# Integrando consultas entre un Front End de React y un Back End de Node

# SERGIO GARZÓN

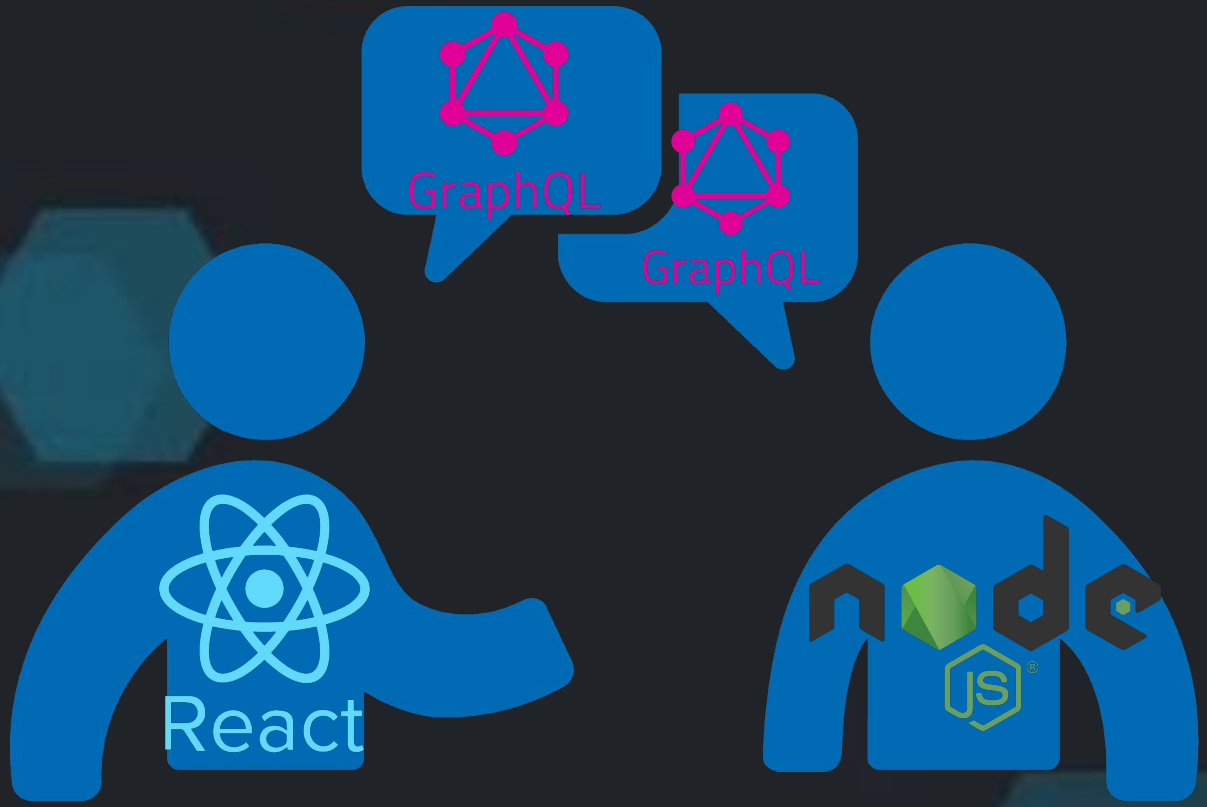

Desarrollador de Software- Desarrollador de Videojuegos - Docente de Programación



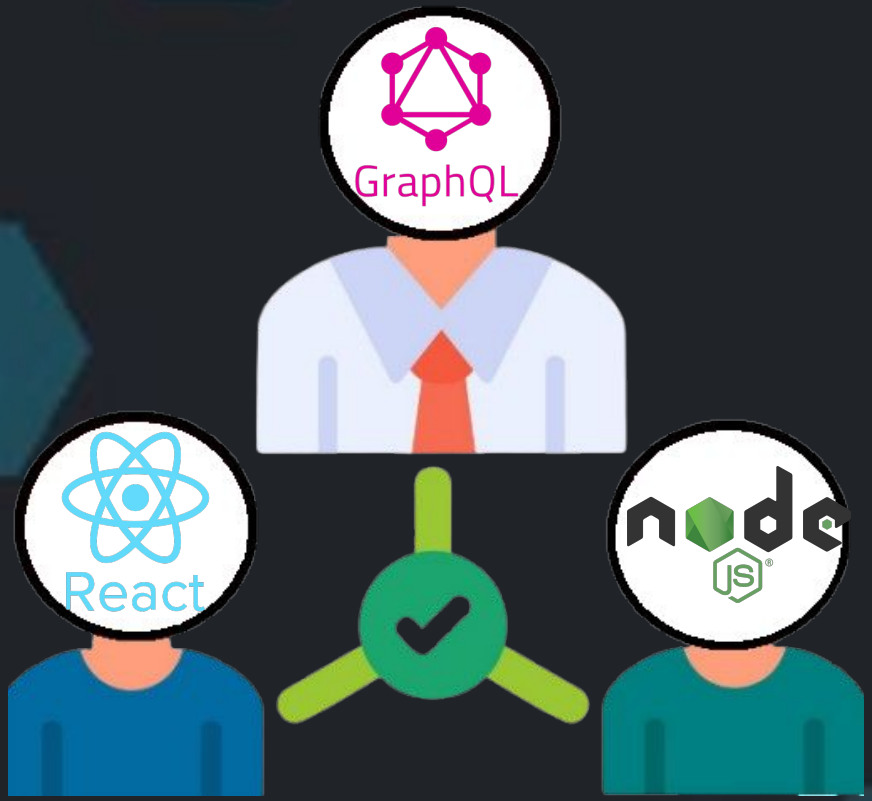

Cómo lograr una integración efectiva entre ambas capas, con GraphQL, para optimizar la comunicación y estructura de nuestras aplicaciones.

## GraphQL:

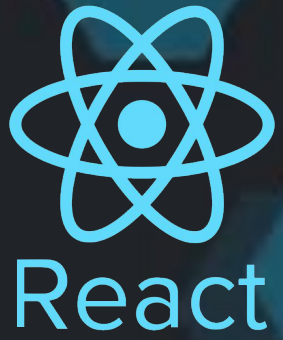
- Desarrollado por  (2015)
- Código abierto (Open source)
- Lenguaje de consultas para lectura y mutación de datos en APIs
- Alternativa de REST API
- Graph (Grafo) QL (Lenguaje de consulta)
- Es declarativo




GraphQL es un lenguaje de comunicación




GraphQL es un intermediario

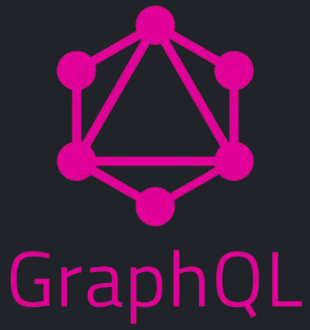




Cliente: Puede ser cualquier framework o librería

**GraphQL:**

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



## GraphQL:

- 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:**

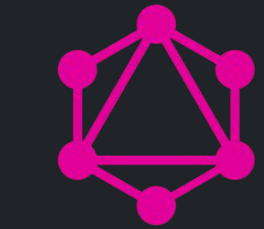
- El cliente hace una solicitud get (por ejemplo a /usuarios) y trae todo.
- Obviamente está más estandarizado, es más fácil de entender.

**Transferencia de datos:**

Ambos utilizan JSON para la transferencia de datos

**Mutaciones para modificación de datos**

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




## Mutaciones en GraphQL

Sin devolución de información

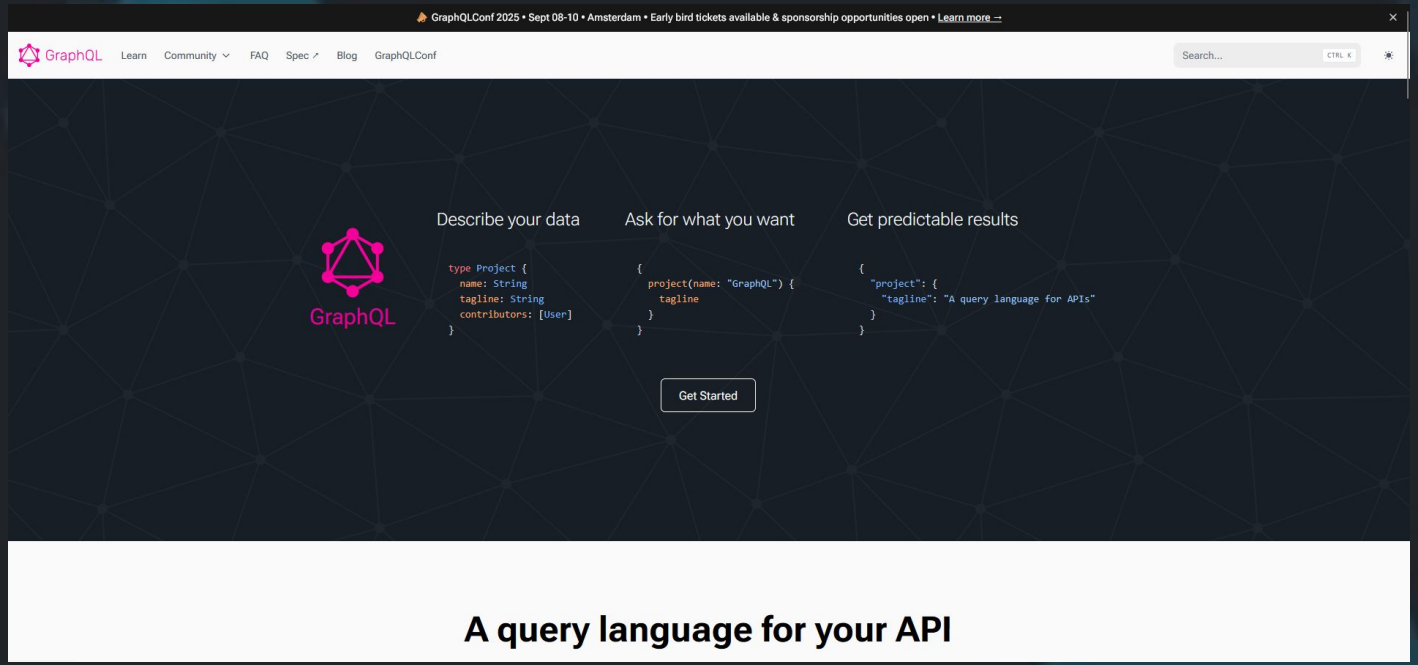
```
mutation {
    crearUsuario ( nombre : "valor" , email : "valor email") {
        id
        nombre
        email
    }
}
```

## Mutaciones en GraphQL

Con devolución de información

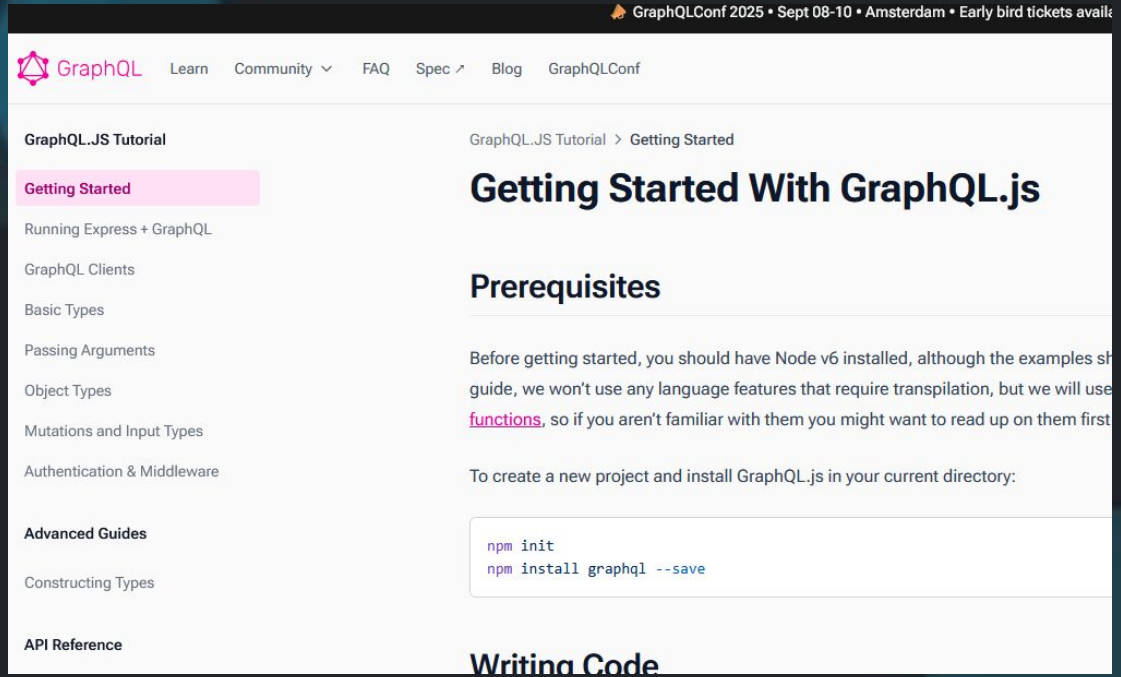
# GraphQL:

Sitio web: https://graphql.org/



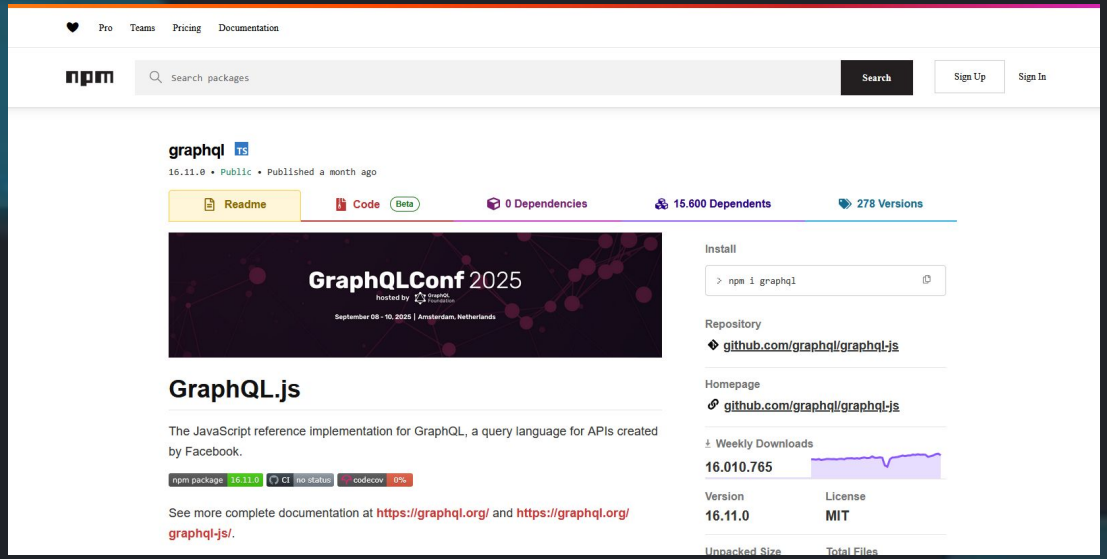
# GraphQL:

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



# GraphQL:

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



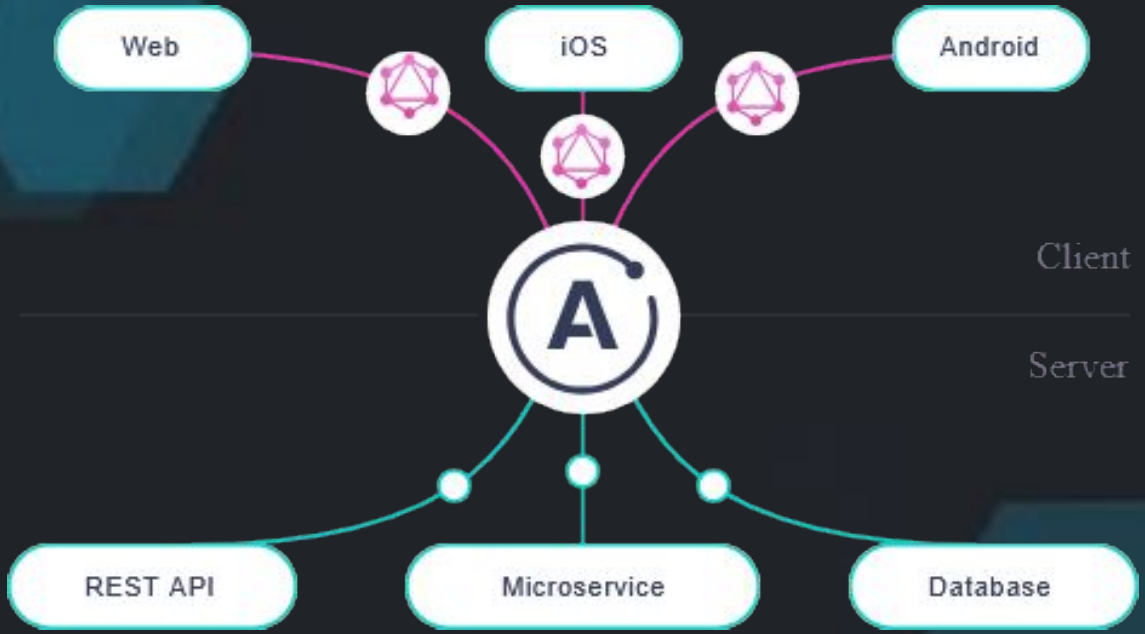

npm install graphql

## Apollo-Client:

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

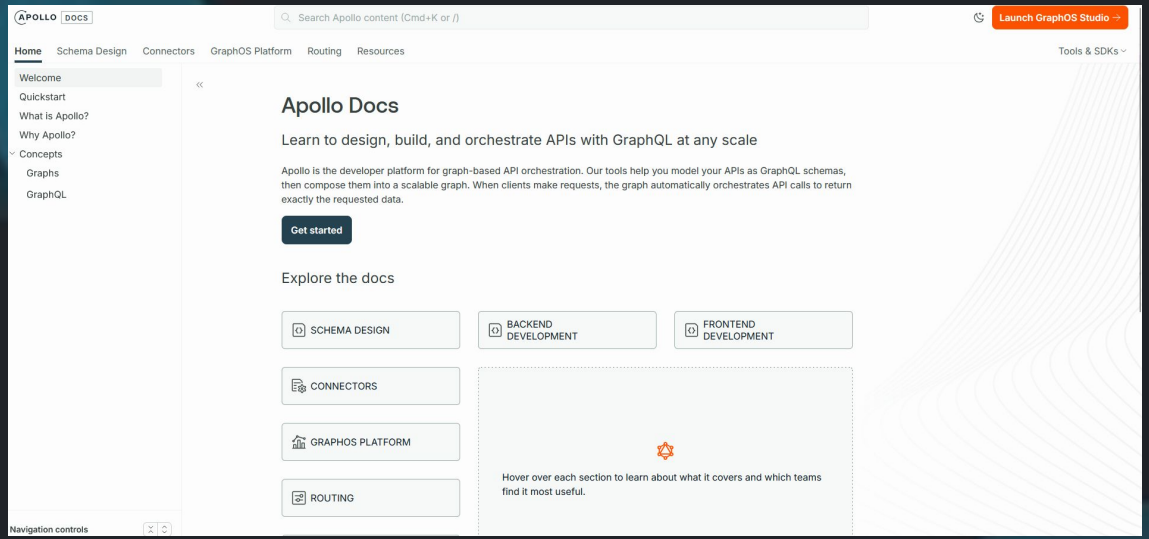
## Apollo-Client:

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



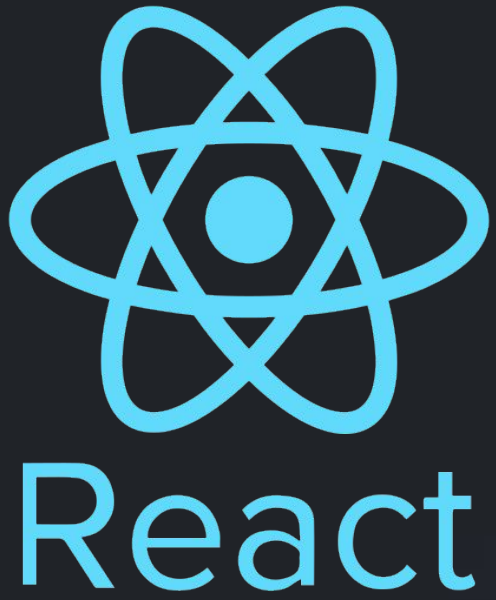
# Apollo-Client:

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



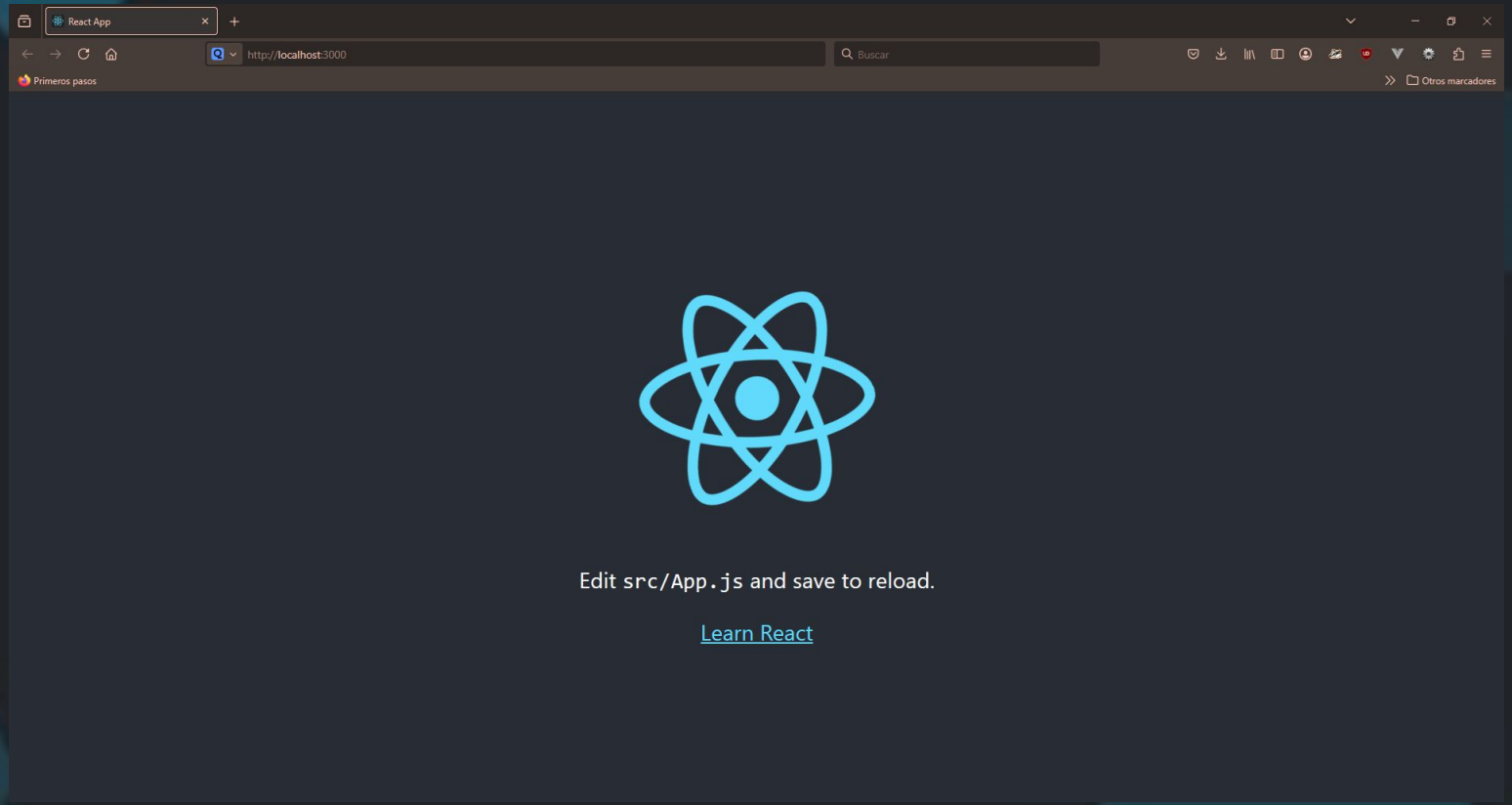

npm install -g apollo-client

# APLICACIÓN FRONT END DE REACT CONVENCIONAL

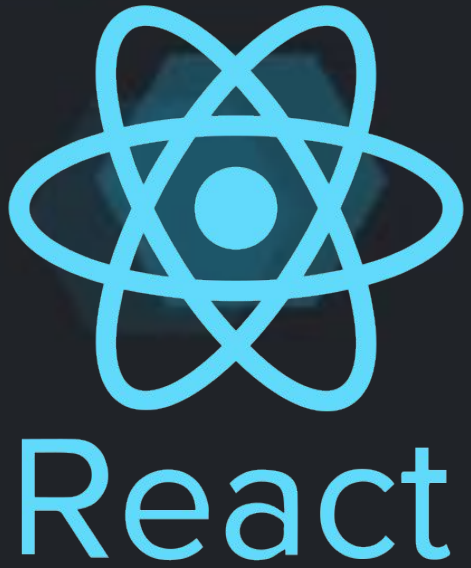



npx create-react-app proyectobeerjs

# APLICACIÓN FRONT END DE REACT CONVENCIONAL



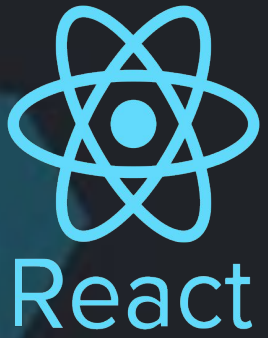
# APLICACIÓN FRONT END DE REACT CON EL TEMPLATE DE VITE



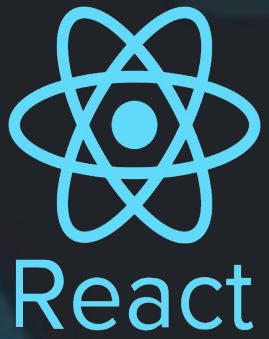



npm create vite@latest proyectobeerjs -- --template react

# APLICACIÓN DE REACT CONVENCIONAL Y CON VITE




Más lento y menor configuración





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

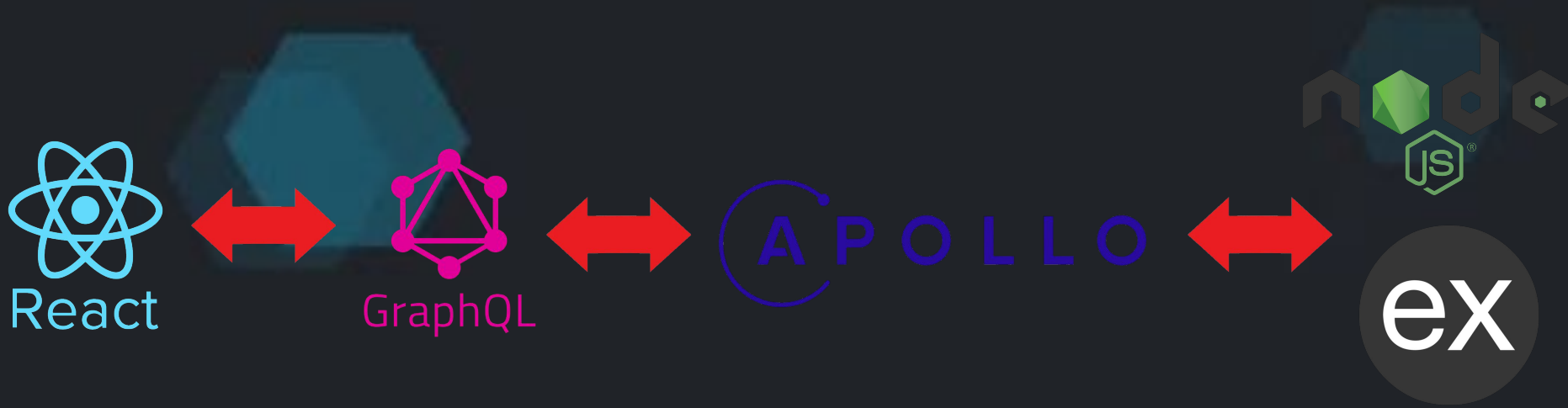
# VIENE OTRA ALTERNATIVA QUE ES REACTQL

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


npm install -g reactql

# VIENE OTRA ALTERNATIVA QUE ES REACTQL

## React + GraphQL + Apollo Client + NodeJS + ExpressJS:



# Ejemplo de código en React con GraphQL

```jsx
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();
```

```jsx
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>Título:</strong> {user.title}
            </li>
          ))}
        </ul>
```

# ¿¿¿Preguntas???

# Muchas Gracias

Sergio Gabriel Garzón

sergiog90arg