

# **Maneja el estado local de tu aplicación con Apollo Client**



Hola :) mi nombre  
es **Sussie Casasola**

- Frontend Developer en  **beek**
- Sígueme en  **@SusCasasola**

**¿Qué es Apollo Client?**

**Apollo Client** te sirve para hacer consultas a API's que están hechas con **GraphQL**.

**¿Porqué Apollo Client?**

- Soporta **varias plataformas** (no sólo React, sino también Angular, Vue, Meteor, Ember, etc..)
- Cuenta con **múltiples librerías** que te ayudan a solucionar varias cosas, como:
  - Hacer peticiones REST (no sólo GraphQL)
  - Sistema de caché (para offline mode)
  - Manejar el estado local de tu aplicación
- Hay una **gran comunidad** trabajando en nuevos updates
- Tienen mucha **presencia** en conferencias, artículos, buena documentación
- La comunidad de developers también está **adoptándolo** para sus proyectos

**¿Porqué debería usar Apollo  
Client para mi estado local?**

- No necesitas instalar librerías aparte ni escribir lógica diferente (Redux, Mobx, Context API de React)
- Puedes acceder a todos los datos de tu aplicación (locales o del servidor) con **Queries** y **Mutaciones**




# ¿Cómo lo hago?

(demo final [aquí](#))



```
yarn add react-apollo apollo-boost graphql graphql-tag
```



```
import ApolloClient from "apollo-boost";
import { ApolloProvider } from "react-apollo";

const client = new ApolloClient({
  uri: "" // URL DE TU SERVICIO
  clientState: {
    defaults,
    resolvers
  }
});

function App() {
  return (
    <ApolloProvider client={client}>
      { /* AQUI VA TU APP */ }
    </ApolloProvider>
  );
}

const rootElement = document.getElementById("root");
ReactDOM.render(<App />, rootElement);
```

# DEFAULTS

**Objeto inicial de tu estado**



```
const defaults = {  
  mainColor: {  
    value: "pink",  
    __typename: "mainColor"  
  }  
};
```

# RESOLVERS

**Obtener información del estado o actualizarlo**



```
const resolvers = {  
  Mutation: {  
    changeMainColor: (_, variables, { cache }) => {  
      const newData = {  
        mainColor: {  
          value: variables.color,  
          __typename: "mainColor"  
        }  
      };  
      cache.writeData({ data: newData });  
    }  
  }  
};
```

**MUTACIÓN**





```
const CHANGE_MAIN_COLOR = gql`  
  mutation changeMainColor($color: String!) {  
    changeMainColor(color: $color) @client  
  }  
`;  
;
```

**QUERY**



```
const GET_MAIN_COLOR = gql`  
  query mainColor {  
    mainColor @client {  
      value  
    }  
  }  
`;  
;
```