### Perspectiva Front-end

Se tiene un archivo con toda la lógica, se presentan algunas malas prácticas en la codificación del mismo.

* Se debe refactorizar en donde se separe los componentes en archivos y se representen una mejor estructura.
* Aplicar las mejores prácticas y buscar el mejor diseño para presentar los datos.

### Perspectiva Back-end

Dentro del back-end no se tiene una base de datos basada en servidor.

* Se debe aplicar un buen diseño de modelo entidad relación y
* aplicar una base de datos como servidor, ejemplo MySQL.
* Representar un objeto de trasporte de datos (DTO) en vez de usar la misma entidad para responder.

### Issues

- Resolver el diseño gráfico

- Separar bien los elementos gráficos como componentes, store, reducer y providers.

- La base de datos debe esta en un servidor como MySQL.

- Aplicar reglas para no guardar elementos vácios.

- Validar carácteres y demás para guardar las entidades de los TO-DO.

- Trabajar con un objeto de trasporte de datos o un objeto plano para representa los datos ante la API.

* Se debe refactorizar en donde se separe los componentes en archivos y se representen una mejor estructura.
* Aplicar las mejores prácticas y buscar el mejor diseño para presentar los datos.
* Se debe aplicar un buen diseño de modelo entidad relación y
* aplicar una base de datos como servidor, ejemplo MySQL.
* Representar un objeto de trasporte de datos (DTO) en vez de usar la misma entidad para responder.
* Hacer un fork en su propio namespace
* Realizar la siguiente representación donde se tiene TO-Do List agripado en listas.

## Reto

Hacer un fork en su propio namespace y presentar la solución más valida para ser discutida, argumentar los aspectos de mejora y aplicar algunas técnica de refactorización. Resolverlo de forma individual, aplicar los commit para cada paso que se realice en la refactorización.

Realizar la siguiente representación donde se tiene TO-Do List agripado en listas.

![alt text]( ./todo-list-kata.gif "Demo funcional del ToDo List")