**React Redux**

1. **Introduction**

* Redux permet de gérer l'état de tous vos composants et sous-composants de votre application.
* Redux utilise un 'store', de sorte que tous les 'state' de l'application sont gérés par un objet central.
* Avec Redux, votre 'state' est désormais centralisé en un seul endroit et chaque component peut y accéder directement
* Lorsque vous utilisez Redux, le 'STORE' centralisé est désormais le seul endroit où le 'state' sera modifié dans votre application
* On n’a Pas besoin d'envoyer des 'props' aux components enfants Redux centralise tout notre state en un seul endroit.
* Cet endroit centralisé s'appelle le STORE.

1. **Redux Classique**

A partir de l’interface utilisateur (UI) :

Si l'utilisateur veut supprimer un produit par exemple

* On a une action qui se déclenche

L'action a un double rôle :

* Il a le rôle d'aller vers la base de données pour la suppression du produit
* Il a aussi le rôle d'aller vers le « reducer » pour la suppression du produit dans le Store

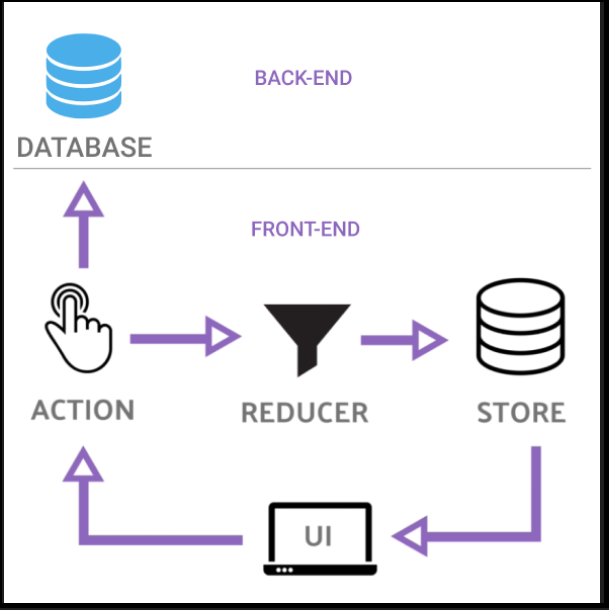
Le « reducer » est le seul élément qui est capable de supprimer les données dans le store

Le store contient toutes les données de l’application, donc on n’a pas besoin d’aller les éléments dans la base de données

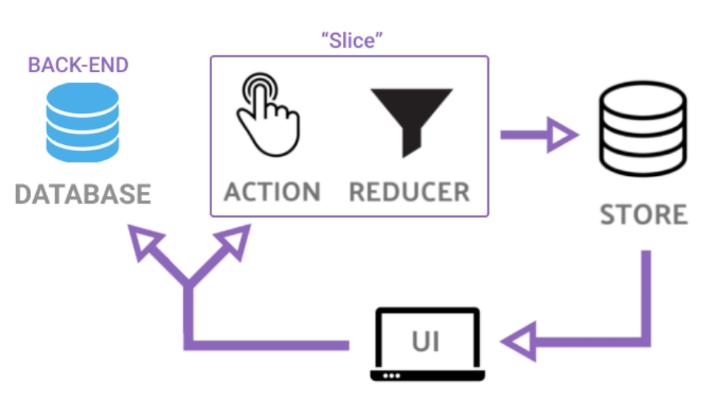
Si l’interface a besoin des données il va les récupérer dans le « store », il n’a pas besoin d’interroger en permanence la base de données

Donc Des composants qui ne se connaissent pas peuvent récupérer leur donnée dans le store (au même endroit)

Ainsi quand il y a une donnée qui évolue dans le « store », tous les composants qui sont liées à cette donnée, qui puisent leur donnée dans le store, vont aussi changer leur donnée en conséquence

****

1. **Redux-toolkit**

****

* On part toujours de notre interface
* On a toujours notre « store » dont l’interface puise tous ces données
* A partir de notre interface si l’utilisateur clique sur un bouton pour supprimer un produit :
* Il y a une action qui se déclenche :
* Dans cette architecture l’ « action » et le « reducer » ont été mis ensemble
* De ce fait l’action ignore ce qu’on peut envoyer vers la base de données
* L’ « action » et le « reducer » sont regroupés dans ce qu’on appelle un « slice »
* Dans ce Slice, on va dire au reducer d'effectuer telle action et d'actualiser le Store

**Redux Toolkit**

Les étapes :

*1 - npm i @reduxjs/toolkit react-redux*

*2 - Créer et configurer le store*

*3 - Créer slice (reducers)*

*4 – Le Hook "useDispatch" pour lancer les reducers, permettant de déclencher les actions*

*5 – Le Hook "useSelector" pour puiser la data dans le store*