

Module : M202 – Développement front-end

Filière : Développement digital – option web full stack

TP7

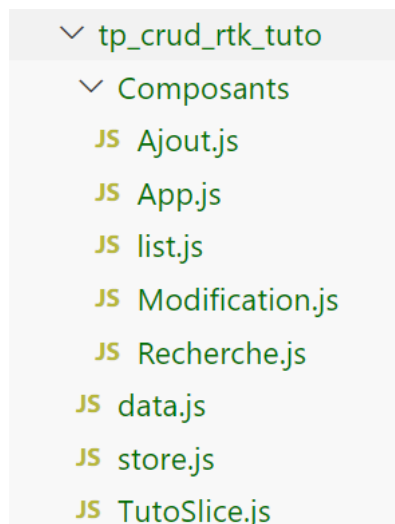
Objectifs :

- Créer une application CRUD en Redux Toolkit

Enoncé

On souhaite créer une application web gérant les tutoriels et offrant à l'utilisateur les possibilités suivantes : Chercher, ajouter, modifier et supprimer des tutoriels.

On propose la structure suivante :

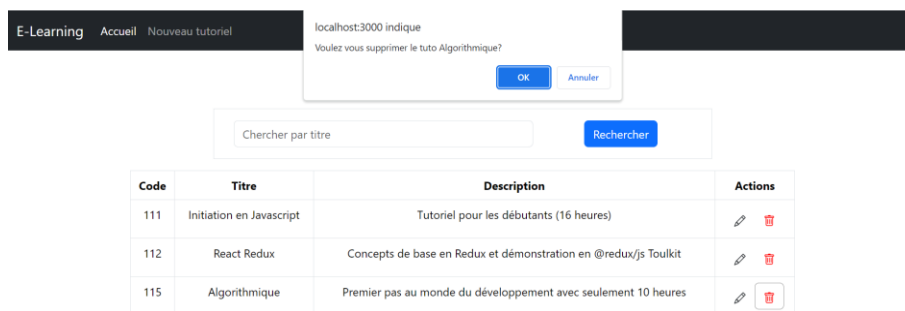


1. Configurer le slice « TutoSlice » pour définir le **state**, **reducer** et les **actions** associées permettant de gérer les tutoriels.
 2. Configurer le store principal sous « store.js »
 3. Sous le composant « App », ajouter :
 - Un **navbar** permettant la navigation entre les composants de l'application
 - La configuration des différentes routes
 4. Ajouter le code nécessaire au reste des composants pour correspondre aux captures ci-après :
- Le composant « **list** » comprend le composant « **Recherche** » et affiche en bas la liste de tous les tutoriels.

Liste des tutoriels

| Code | Titre | Description | Actions |
|------|--------------------------|---|---------|
| 111 | Initiation en Javascript | Tutoriel pour les débutants (16 heures) | |
| 112 | React Redux | Concepts de base en Redux et démonstration en @redux/js Toulkit | |
| 115 | Algorithmique | Premier pas au monde du développement avec seulement 10 heures | |

- Le clic sur l'icône de suppression , sur l'une des lignes affichant les tutoriels, permet d'afficher un message de confirmation avant de procéder à la suppression définitive du tutorial.



- Le clic sur l'icône de modification , sur l'une des lignes affichant les tutoriels, permet d'afficher le composant « **Modification** » prérempli avec les données du tutorial correspondant.

Mise à jour du Tutorial 112

Titre

Description

- Sur le menu de navigation, si l'utilisateur choisit l'option « Nouveau tutorial », l'application doit afficher le composant « **Ajout** » donnant la possibilité d'ajouter un nouveau tutorial à la liste existante.

Nouveau Tutoriel

Code

Titre

Description









Ajouter

- Voici la liste initiale des tutoriels :

```
export const listeTuto = [
  {
    code: 111,
    titre: "Initiation en Javascript",
    description: "Tutoriel pour les débutants (16 heures)",
  },
  {
    code: 112,
    titre: "React Redux",
    description: "Concepts de base en Redux et démonstration en @redux/js
Toulkit",
  },
  {
    code: 115,
    titre: "Algorithmique",
    description:
      "Premier pas au monde du développement avec seulement 10 heures",
  },
];
```

AMÉLIORATION :

Vous pouvez ajouter des alertes, à afficher sur la page d'accueil, après chaque opération permettant d'indiquer sa réussite :

| E-Learning Accueil Nouveau tutorial | | | |
|-------------------------------------|-----------------------------|---|---|
| Liste des tutoriels | | | |
| Tutoriel bien ajouté | | | |
| Chercher par titre | | Rechercher | |
| Code | Titre | Description | Actions |
| 111 | Initiation en Javascript | Tutoriel pour les débutants (16 heures) |   |
| 112 | React Redux | Concepts de base en Redux et démonstration en @redux/js Toulkit |   |
| 115 | Algorithmique | Premier pas au monde du développement avec seulement 10 heures |   |
| 200 | Introduction à UI/UX Design | une formation intéressante pour les débutants en Design |   |

Indications :

Vous pouvez utiliser les hook « `useNavigate` » et « `useLocation` » afin d'assurer une navigation par programme :

`useNavigate` et `useLocation` sont des hooks fournis par la bibliothèque React Router , largement utilisée pour la gestion de la navigation dans les applications React.

- **`useNavigate` :**

Description : **`useNavigate`** est un hook qui renvoie une fonction de navigation. Il est utilisé pour déclencher des changements d'URL programmiquement.

Exemple :

```
import { useNavigate } from "react-router-dom";
function MyComponent() {
  const navigate = useNavigate();
  const handleClick = () => {
    // Naviguer vers "/nouvelle-page" lorsque le bouton est cliqué
    navigate("/nouvelle-page");
  };
  return <button onClick={handleClick}>Aller à la nouvelle page </button>;
}
```

- **`useLocation` :**

Description : **`useLocation`** est un hook qui renvoie l'objet de localisation actuel, représentant l'URL courante de l'application.

Exemple :

```
import { useLocation } from "react-router-dom";
function CurrentPath() {
  const location = useLocation();
  return (
    <div>
      <p>Le chemin actuel est :{location.pathname}</p>{" "}
    </div>
  );
}
```

Ces hooks sont souvent utilisés ensemble pour construire des composants qui réagissent aux changements d'URL et déclenchent des navigations en fonction de certaines conditions ou interactions utilisateur.

Exemple :

Voici un exemple où **`useNavigate`** est utilisé avec un deuxième argument contenant un message, et **`useLocation`** est utilisé pour récupérer et afficher ce message :

```
import { useNavigate, useLocation } from 'react-router-dom';

// Composant de la page principale
function HomePage() {
  const navigate = useNavigate();
```

```

const handleClick = () => {
  // Naviguer vers la page d'accueil avec un message
  navigate('/', { state: { message: 'Bienvenue sur notre site !' } });
};

return (
  <div>
    <h2>Page d'Accueil</h2>
    <button onClick={handleClick}>Afficher un Message</button>
  </div>
);
}

// Composant de la page de détails
function DetailsPage() {
  const location = useLocation();
  const message = location.state && location.state.message;

  return (
    <div>
      <h2>Détails de la Page</h2>
      {message && <p>{message}</p>}
    </div>
  );
}

// Composant principal
function App() {
  return (
    <div>
      <HomePage />
      <hr />
      <DetailsPage />
    </div>
  );
}

```

Dans cet exemple, lorsque l'utilisateur clique sur le bouton sur la page d'accueil, **useNavigate** est utilisé pour naviguer vers la page d'accueil avec un objet **state** contenant le **message**. Ensuite, sur la page de détails (**DetailsPage**), **useLocation** est utilisé pour récupérer le **message** de l'objet **state** de l'URL et l'afficher.