



# 420-710 : Laboratoire 02

## Objectif:

Développer une application To-Do avec React

Dans ce laboratoire, vous développerez une application de gestion de tâches (To-Do) avec React.

Vous apprendrez à :

- Gérer les états dans React
- Manipuler des événements
- Structurer vos composants
- Utiliser des modules CSS pour styliser votre application

De plus, vous ajouterez une fonctionnalité de recherche pour filtrer les tâches et une option pour inclure ou exclure les tâches archivées.

## Références

(🔥 Spruce Emmanuel [How to Build a TODO App from Scratch with React.js](#)

# Livrables du projet

Pour ce laboratoire, vous devez soumettre les éléments suivants :

1. **Code source complet** : Tous les fichiers de votre projet React, y compris les composants, les modules CSS et le fichier principal App.jsx. **SUPPRIMEZ LE DOSSIER `./node_modules/`.**
2. **Capture d'écran de l'application** : Une image montrant l'interface utilisateur finale de votre application To-Do.
3. **Fichier README.md** : Un bref rapport (environ 1 page) décrivant :
  - Les défis rencontrés lors du développement
  - Les solutions trouvées pour surmonter ces défis
  - Les apprentissages clés tirés de ce projet
  - Les améliorations potentielles que vous pourriez apporter à l'application

Assurez-vous que votre code est bien commenté et suit les bonnes pratiques de développement React.

## Étapes

### Étape 1 : Initialisation du projet React

1. **Créer un nouveau projet React**
  - Créez un projet en React en JS
  - Nommez votre projet de manière appropriée et organisez vos fichiers de façon cohérente.
2. **Installation des dépendances**
  - Aucune installation supplémentaire n'est nécessaire, car vous utiliserez uniquement les outils fournis et les modules CSS.

### Étape 2 : Structurer vos composants

## 1. Créer les composants de base

- Supprimez le contenu du composant principal `App.jsx` et `App.css` qui contiendra la logique globale de votre application.
- Décomposez votre application en plusieurs composants :
  - Un composant `Header` pour afficher le titre de l'application.
  - Un composant `TodoForm` contenant un champ de saisie pour ajouter de nouvelles tâches.
  - Un composant `TodoList` pour afficher la liste des tâches.
  - Un composant `TodoItem` pour représenter chaque tâche individuelle.

## 2. Exemple de structure initiale (sans les détails d'implémentation)

- Dans votre fichier `App.jsx` :

```
import React from 'react'; import Header from './components/Header'; import TodoForm from './components/TodoForm'; import TodoList from './components/TodoList'; function App() { return ( <div> <Header /> <TodoForm /> <TodoList /> </div> ); } export default App;
```

## Étape 3 : Gestion de l'état et ajout de tâches

## 1. Utiliser `useState` pour gérer l'état des tâches

- Dans `App.jsx`, utilisez le hook `useState` pour créer une liste de tâches.
- Voici un exemple d'utilisation de `useState` pour gérer une liste de tâches dans `App.jsx` :

```
const [tasks, setTasks] = useState([ { id: 1, text: 'Faire les courses', isCompleted: false, isArchived: false }, { id: 2, text: 'Réviser pour l'examen', isCompleted: false, isArchived: false }, { id: 3, text: 'Faire du sport', isCompleted: true, isArchived: false }, ]); // ... Reste du composant App }
```

- Le tableau des tâches doit contenir des objets avec les propriétés suivantes :
  - `id` : Identifiant unique de la tâche.
  - `text` : Description de la tâche.
  - `isCompleted` : Statut de la tâche (complétée ou non).
  - `isArchived` : Statut de la tâche (archivée ou non).

## 2. Ajouter des tâches

- Dans `TodoForm.jsx`, créez un formulaire avec un champ de saisie pour ajouter de nouvelles tâches.
- Lorsque l'utilisateur soumet une nouvelle tâche, ajoutez-la à l'état géré dans `App.jsx`.

# Étape 4 : Afficher la liste des tâches

## 1. Afficher la liste des tâches dans `TodoList.jsx`

- Récupérez les tâches depuis `App.jsx` et affichez-les dans une liste dans le composant `TodoList`.
- Pour chaque tâche, utilisez le composant `TodoItem.jsx` pour afficher le texte de la tâche et un bouton pour la marquer comme complétée.

## 2. Exemple de rendu des tâches dans `TodoList.jsx`

```
import React from 'react'; import TodoItem from './TodoItem'; function TodoList({ tasks }) { return ( <ul> {tasks.map(task => ( <TodoItem key={task.id} task={task} /> ))} </ul> ); } export default TodoList;
```

## Étape 5 : Manipuler les événements pour marquer les tâches comme complétées ou archivées

### 1. Marquer une tâche comme complétée

- Dans `TodoItem.jsx`, ajoutez un bouton "Compléter" qui, lorsqu'il est cliqué, met à jour l'état de la tâche pour indiquer qu'elle est complétée.
- Utilisez `onClick` pour déclencher cette action.

### 2. Archiver une tâche

- Ajoutez un bouton "Archiver" pour déplacer la tâche dans une section d'archives.
- Lorsqu'une tâche est archivée, déplacez-la hors de la liste principale sans la supprimer.

## Étape 6 : Styliser l'application avec des modules CSS

### 1. Créer des fichiers de style CSS modulaires

- Utilisez les modules CSS pour styliser vos composants individuellement.
- Par exemple, créez un fichier `TodoItem.module.css` pour styliser chaque tâche.

### 2. Exemple de fichier CSS pour un composant `TodoItem.module.css`

```
.task { background-color: #f4f4f4; margin: 10px 0; padding: 10px; border-radius: 5px; display: flex; justify-content: space-between; } .completed { text-decoration: line-through; color: grey; } .archived { opacity: 0.5; }
```

### 3. Appliquer les styles dans `TodoItem.jsx`

- Importez et appliquez les styles définis dans vos modules CSS :

```
import styles from './TodoItem.module.css'; function TodoItem({task}) { return ( <li className={`${styles.task} ${task.isCompleted ? styles.completed : ''}`}> <span>{task.text}</span> <button>Compléter</button> <button>Archiver</button> </li> ); } export default TodoItem;
```

## Étape 7 : Ajouter une barre de recherche et une option pour inclure/exclure les tâches archivées

### 1. Ajouter une barre de recherche

- Dans `App.jsx`, créez une barre de recherche permettant à l'utilisateur de filtrer les tâches par texte.
- Utilisez l'état pour gérer la valeur de la recherche et filtrez les tâches dans