# TD Todo List - Partie 01

React


**Consignes** ..... **2**
- Réalisation ..... 2
- Rendu ..... 2

**Installations** ..... **3**

**01 - Initialisation du projet** ..... **4**

**02 : Type de données** ..... **5**

**03 : Collection de données** ..... **6**

**04 : App** ..... **7**

**05 : Template** ..... **8**

**06 : Liste dynamique** ..... **9**




Alexandre Leroux (alex@shrp.dev)

# Consignes

## Réalisation

L'acquisition de connaissances et de compétences repose sur l'articulation entre apprentissage théorique et mise en pratique.
L'utilisation d'outils d'intelligence artificielle générative est autorisée à des fins de documentation et de vérification, sous réserve d'un usage modéré, éthique et explicitement signalé.
Tout usage non conforme (ex: production intégrale de contenu, copier-coller, …) sera sanctionné dans l'évaluation.

- **Travail individuel** comptant dans votre **évaluation continue**.
- Réalisation pendant les séances de TD et en autonomie, sur votre temps libre.
- Employez le runtime ***Bun***, le langage ***TypeScript***, le *HMR* ***Vite*** et la bibliothèque ***React*** pour programmer l'application (cf. installations).
- Exploitez autant que possible le typage statique de ***TypeScript*** (type des variables, des paramètres et de la valeur de retour des fonctions, des props de composants…).
- Ce projet est constitué de **plusieurs parties** contenant chacune **plusieurs étapes**. A chaque partie du projet correspond un document PDF.
  **Lorsque vous avez terminé une partie, demandez le support PDF de la partie suivante.**
- **Réalisez le projet en suivant les parties et étapes dans l'ordre,** en prenant le soin de lire l'intégralité des consignes avant de débuter la réalisation.
- **Utilisez *Git*** pour sauvegarder et versionner votre travail (ex: une branche *Git* par partie du projet).
- **L'auto-documentation fait partie de l'exercice** :
  - https://react.dev/
  - https://react.dev/learn/typescript
  - https://bun.sh/
  - https://vite.dev/
  - https://www.typescriptlang.org/

# Rendu

- **Délai de rendu : dimanche 5/10/2025 23h59 au plus tard.**
- Dépôt sur ***Arche*** d'un fichier d'archive (ex: *prenom-nom.zip*) ou d'un lien vers votre projet *Git* privé (*Github*) en ayant pris le soin d'ajouter votre responsable de TD en tant que collaborateur du projet :
    - **Groupe 1** : Alexandre Leroux (Github : ***shrp777*** / alex@shrp.dev)
    - **Groupes 2 et 3** : Qi Qiu (Github : ***Qi-tchi*** / qi.qiu2021@gmail.com)
- Version complète du projet sur la **branche *main*** du dépôt *Git*.
- **Renseignez votre prénom, nom et numéro de groupe de TD dans le fichier *README.md*** placé à la racine de votre projet.

# Critères d'évaluation

- Respect des consignes,
- Qualité du code (absence de bug, structuration, cohérence…),
- *UI / UX*,
- Comportement en cours : autonomie, intérêt, participation, assiduité, ponctualité, respect…

# Installations

- ***VS Code*** (https://code.visualstudio.com/),
- ***Bun*** (https://bun.sh/).
- ***Git*** (https://git-scm.com/).
- En option, ***React Developer Tools*** (https://react.dev/learn/react-developer-tools).

# 01 - Initialisation du projet

- Avec votre terminal de commande, le *CLI* de **Bun** et l'utilitaire ***vite*** (https://vite.dev/), créez un nouveau projet ***React*** :

```
$ bun create vite todolist
 ✔ Select a framework: › React
 ✔ Select a variant: › TypeScript
```

- Placez-vous à la racine du dossier du projet créé :

```
$ cd todolist
```

- installez les dépendances *NPM* :

```
$ bun install
```

- Démarrez votre application *React*

```
$ bun run dev
```

- Consultez l'application dans votre navigateur web à l'adresse http://localhost:5173/ (le port peut différer).
- Pour stopper l'exécution de votre application, placez-vous dans le terminal et effectuez le raccourci clavier **CTRL C** (que ce soit sur *Mac* ou *Windows*).

**Etape 02 page suivante**

# 02 : Type de données

- Importez les modules *NPM* suivants :
    - **uuid** (dépendance de production),
    - **@types/uuid** (dépendance de développement)

- Définissez au choix une *interface* ou un *type* nommé ***Task*** dans un fichier **./src/Task.ts** avec les caractéristiques suivantes :

| **nom de l'attribut** | **type de l'attribut / valeurs autorisées** |
|---|---|
| id | chaîne de caractères (*UUID v4*) |
| content | chaîne de caractères |
| createdAt | date |
| completedAt | date (valeur optionnelle) |
| status | todo, doing, done (*Sum Type*) |

- Exportez ***Task*** afin de le rendre disponible à l'import dans d'autres fichiers du projet.

**Etape 03 page suivante**

# 03 : Collection de données

- Dans un fichier **./src/data.ts**, importez ***Task*** puis définissez une constante nommée ***tasksCollection*** de type liste de ***Task***.
- Exportez la constante ***tasksCollection*** afin de la rendre disponible à l'import dans d'autres fichiers du projet.
- Instanciez 5 objets de type ***Task*** au sein de cette liste :

| id | content | status | createdAt | completedAt |
|---|---|---|---|---|
| 1 | Installer VS Code, Bun et Git | done | aujourd'hui | aujourd'hui |
| 2 | Apprendre TypeScript | doing | aujourd'hui | |
| 3 | Apprendre React | doing | aujourd'hui | |
| 4 | Réaliser mon TD | doing | aujourd'hui | |
| 5 | Penser à saisir prénom et mon nom dans le fichier README.md | todo | aujourd'hui | |

**Etape 04 page suivante**

# 04 : App

- Remplacez le contenu par défaut du fichier **./src/App.tsx** par celui renseigné ci-après.

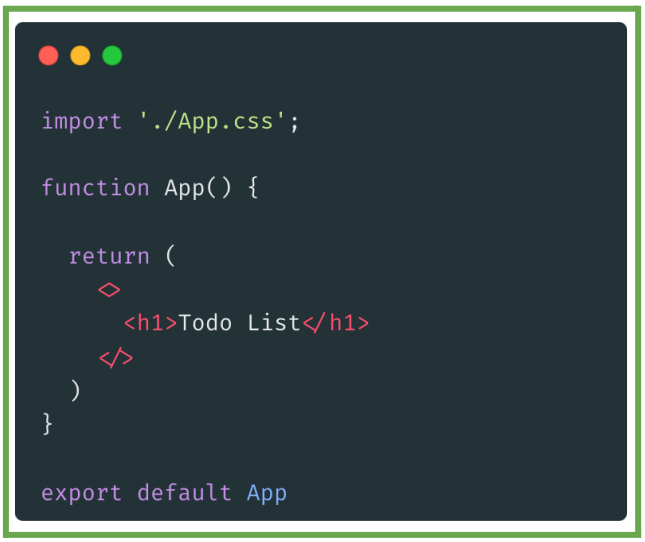
```tsx
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

function App() {
  const [count, setCount] = useState(0)

  return (
    <>
      <div>
        <a href="https://vite.dev" target="_blank">
          <img src={viteLogo} className="logo" alt="Vite logo" />
        </a>
        <a href="https://react.dev" target="_blank">
          <img src={reactLogo} className="logo react" alt="React logo" />
        </a>
      </div>
      <h1>Vite + React</h1>
      <div className="card">
        <button onClick={() => setCount((count) => count + 1)}>
          count is {count}
        </button>
        <p>
          Edit <code>src/App.tsx</code> and save to test HMR
        </p>
      </div>
      <p className="read-the-docs">
        Click on the Vite and React logos to learn more
      </p>
    </>
  )
}

export default App
```

**./src/App.tsx (contenu par défaut)**



```tsx
import './App.css';

function App() {

  return (
    <>
      <h1>Todo List</h1>
    </>
  )
}

export default App
```

**./src/App.tsx (nouveau contenu)**

**Etape 05 page suivante**

# 05 : Template

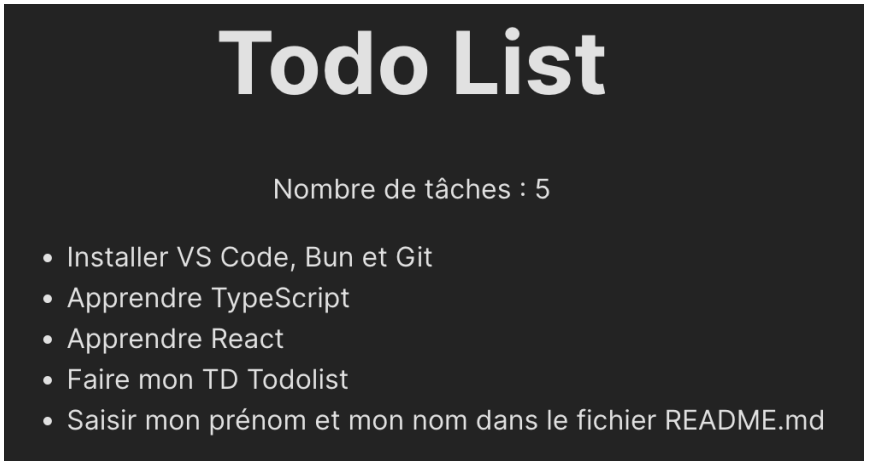
- Importez la constante ***tasksCollection*** dans le fichier **src/App.tsx**,
- Dans la fonction ***App***, déclarez une nouvelle constante nommée ***tasks*** et affectez-lui la valeur de ***tasksCollection***.
- Dans le template retourné par la fonction ***App***, sous la balise de titre de niveau 1, ajoutez une balise *HTML* de type paragraphe et à l'intérieur, affichez la mention *"Nombre de tâches"* suivie du nombre d'items contenus dans la liste ***tasks***, évalué dynamiquement à l'aide de la syntaxe ***TSX*** (la version *TypeScript* de *JSX*).


**Etape 06 page suivante**

# 06 : Liste dynamique

- Dans le composant ***App***, ajoutez la balise *HTML* permettant de représenter une liste d'items non ordonnés.
- Au sein de cette balise *HTML*, à l'aide de la syntaxe ***TSX***, affichez dynamiquement la valeur ***content*** de chaque ***Task*** contenue dans la collection ***tasks***.

  Itérez sur la collection d'items pour afficher dynamiquement la valeur de l'attribut ***content*** de l'objet ***Task*** courant. Utilisez la valeur de l'***id*** de chaque item pour l'affecter à l'attribut ***key*** de chaque balise *HTML* générée dynamiquement.
- Adaptez le style ***CSS*** du fichier **./src/App.css** pour aligner le texte des items sur la gauche.




**Fin de la partie 01**

**Partie 02 dans un autre document**