

Wealth Health

Développeur React chez WealthHealth.

Faire passer une librairie jQuery vers React pour améliorer les performances et la stabilité.



CONTEXTE

- ✓ Je travaille pour le département technologique d'une grande Sté financière WealthHealth
- ✓ Sté qui utilise une appli interne web nommée HrNet pour la gestion des dossiers des employés
- ✓ Application trop ancienne , qui utilise JQuery coté Front-end, ce qui entraine des Bugs et de plus en plus de plaintes en interne

OBJECTIFS

- ✓ Convertir l'appli HrNet en appli React plus moderne et plus performante
 - ✓ Créer ses propres composants React à la place des plugins jQuery
 - ✓ Choisir et convertir un des 4 plugin jQuery en un composant React.
 - ✓ Améliorer l'expérience utilisateur et la satisfaction des employés

CONTRAINTES

- ✓ Choisir 1 seul plugin sur les 4 à convertir
- ✓ Déposer un repo sur Github séparé pour le code converti
 - ✓ Convertir l'appli principale sans utiliser jQuery
- ✓ Faire des tests de performance et comparer les 2 versions

Projet à convertir

Fichier App.js Projet JQuery

Gère la création d'interface pour la saisie d'info sur les employés, la sélection d'états, la configuration des menus déroulants et des sélecteurs de dates

Fichier employees-list.js

Responsable de la récupération des données des employés à partir du local storage local et de l'affichage des données dans un tableau en utilisant le plugin data table de JQuery . Ce qui permet aux utilisateurs de visualiser et agir avec les infos

Fichier employes-list.html

Ce fichier html affiche la liste des employés à l'aide du plugin DataTable

Fichier index.html

Ce fichier permet de créer un nouvel employé en saisissant les informations dans le formulaire

COMPARAISON

Aspect	Projet jQuery	Projet React avec Redux
Technologie	jQuery	React avec Redux
Architecture	MVC	Composants et Redux
Gestion de l'état	Manipulation du DOM	Gestion via Redux
Composants	Moins modulaires	Composants modulaires
Réutilisabilité	Moins	Plus grande
Persistence des données	LocalStorage	Redux Persist
Bibliothèques	jQuery UI, DataTables	React, Redux
Réactivité	Moins réactive	Réactive avec React
Performance	Moins performant	Mise à jour efficace

Choix du plugin jQuery

Point de comparaison	Code source de la modal pour le projet jQuery	Code source du composant React
Type de fonction	Fonction anonyme qui s'exécute immédiatement	Fonction nommée qui exporte le composant
Bibliothèque utilisée	jQuery	React
Passage des options	Attributs data	Props
Affichage ou masquage de la modal	Méthodes .modal(show) ou .modal(hide)	Condition ternaire selon la valeur de isOpen
Définition des propriétés et des méthodes	Objet \$.modal	PropTypes

Dans le projet React :
A la soumission du formulaire

(component Form.js)

Soumettre

Would you like to register **Pascaline CHRISTOPHE**

Register

Cancel

```
<Modal
  isOpen={modalOpen}
  onClose={handleCloseModal}
  modalClassName="my-custom-modal-class"
  firstName={firstName}
  lastName={lastName}
  onCancel={handleCancel}
  onSave={handleSave}
  showButtons={true}
  actionLabel={actionLabel}
```

MODAL 1

Plugin : [banby-modal-customize-react](#)

Exemple utilisé dans le projet React

```
<Modal
  isOpen={modalOpen}
  onClose={handleCloseModal}
  // fonctions optionnelles
  firstName="john" // votre choix
  lastName="Doe" // votre choix
  modalTitle="Titre de la Modal"
  showButtons={true}
>
</Modal>
```

Would you like to register **John Doe** or start again

Register

Cancel

MODAL2

Dans le projet React :
Après la soumission :
Register
(component Form.js)

Plugin : [banby-modal-customize-react](#)


Exemple utilisé dans le projet React



Employé enregistré

```
<Modal
  isOpen={secondModalOpen}
  onClose={() => setSecondModalOpen(false)}
  modalClassName="my-custom-modal-class"
  modalTitle={modalTitle}
>
  <h2>{modalTitle}</h2>
</Modal>
```

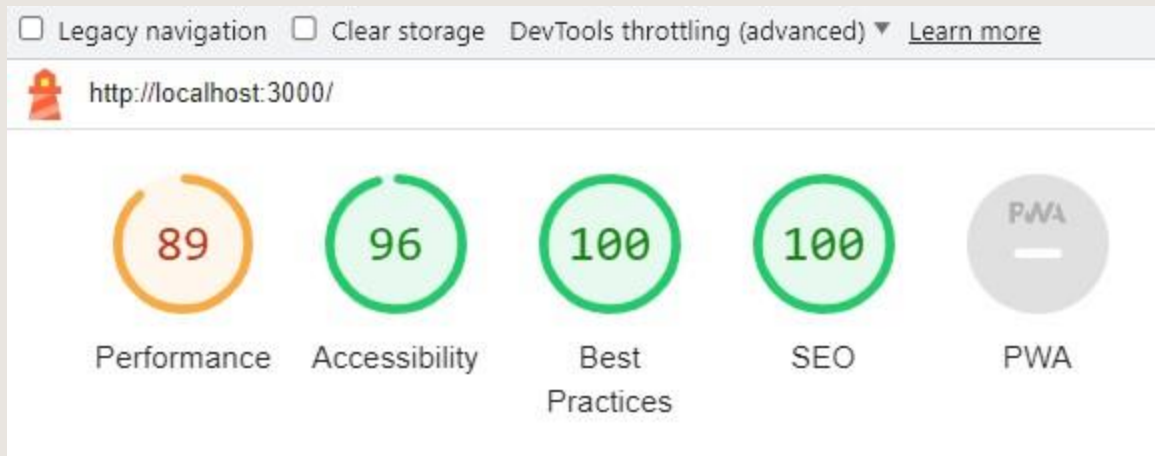
```
<Modal
  isOpen={modalOpen}
  onClose={handleCloseModal}
  // fonctions optionnelles
  modalTitle="Titre de la Modal"
  showButtons={false}
>
</Modal>
```



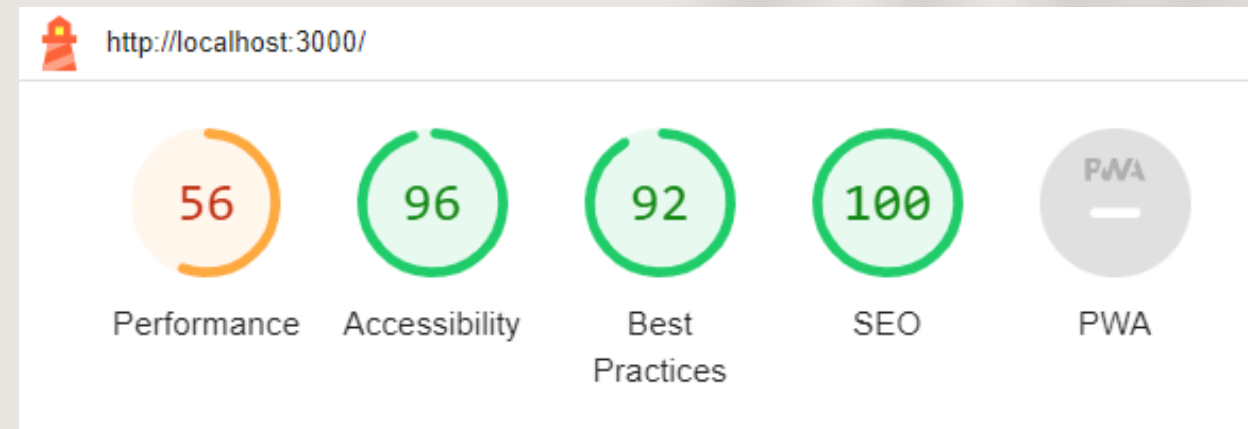
I'm a modal to customize !

Comparison Test Lighthouse Home.js

Desktop

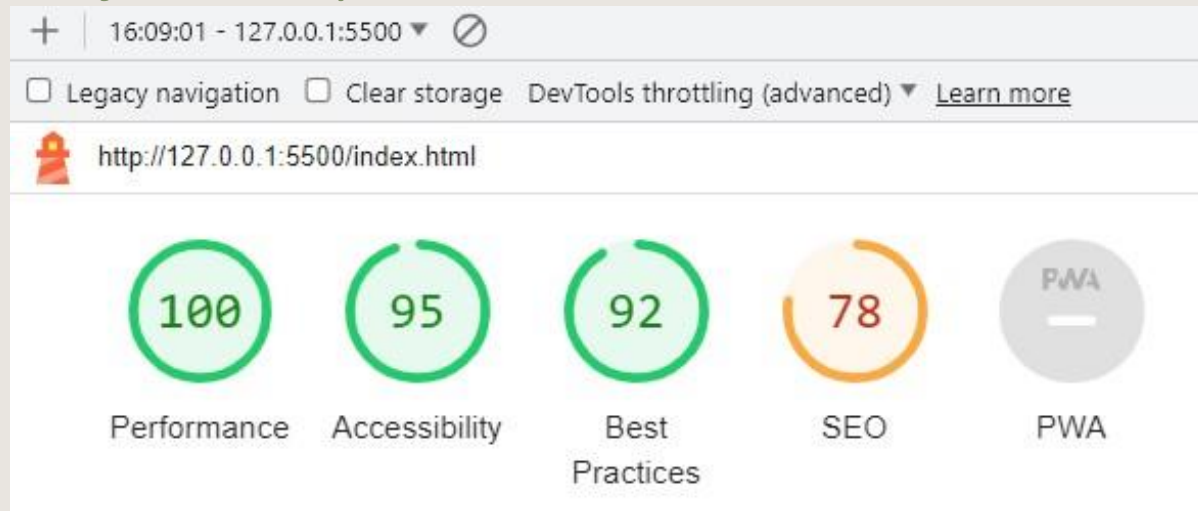


Smartphone

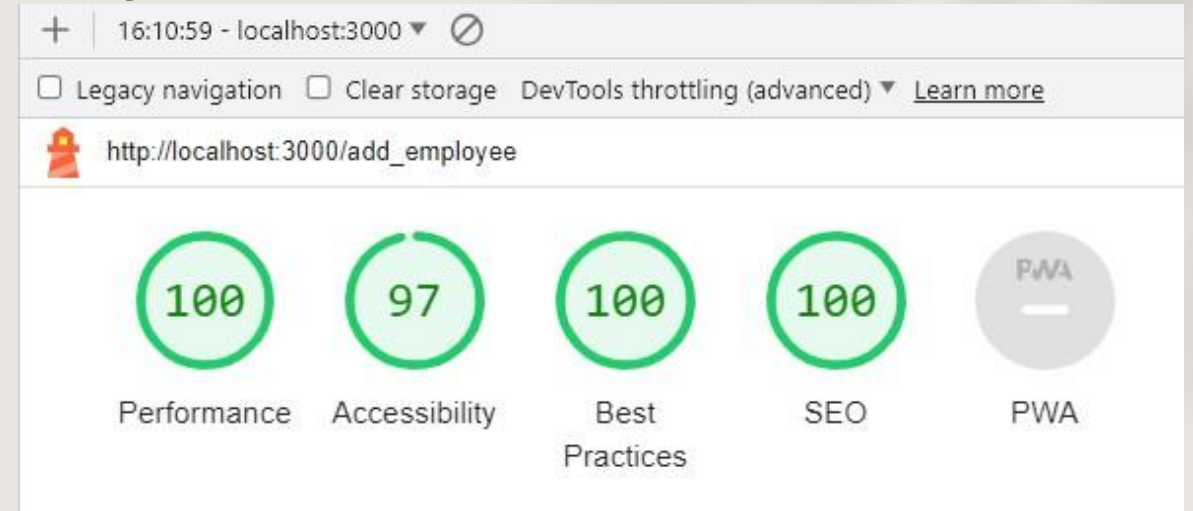


Comparaison Test Lighthouse Form.js DeskTop

Projet JQuery



Projet React

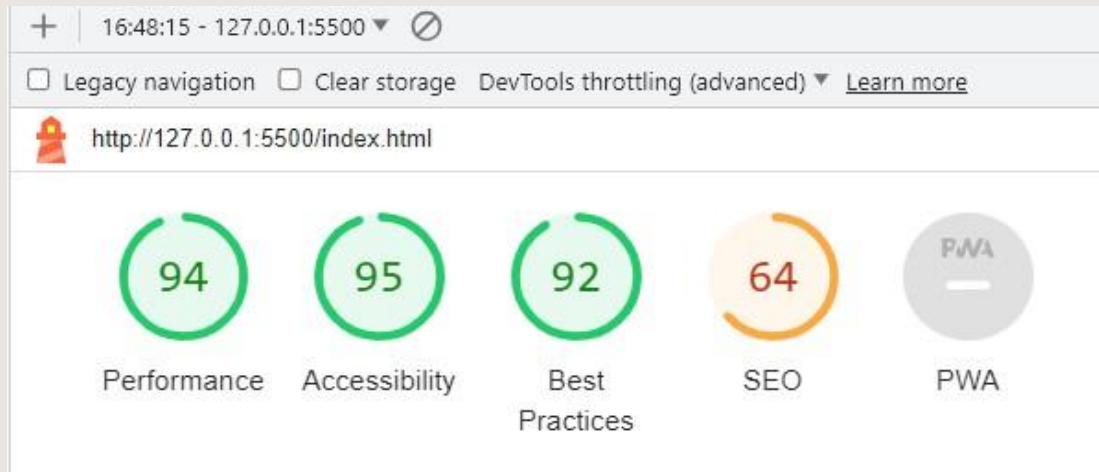


(seo) il faudrait ajouter un texte descriptif aux liens et créer une balise meta description.

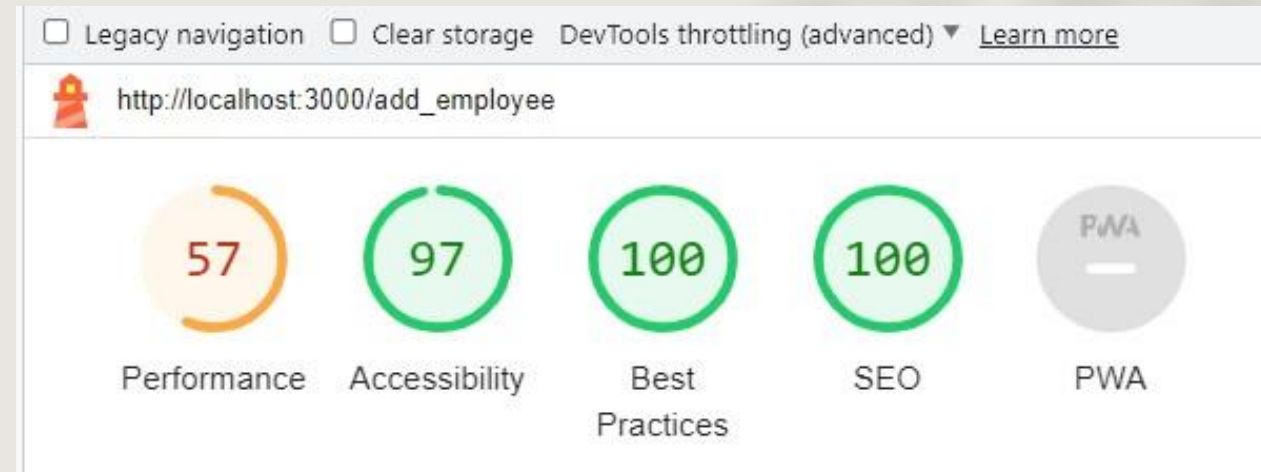
l'analyse Lighthouse montre que le projet jQuery est performant, mais React offre des avantages en termes de maintenabilité et de réutilisabilité du code.

Comparaison Test Lighthouse Form.js Smartphone

Projet JQuery

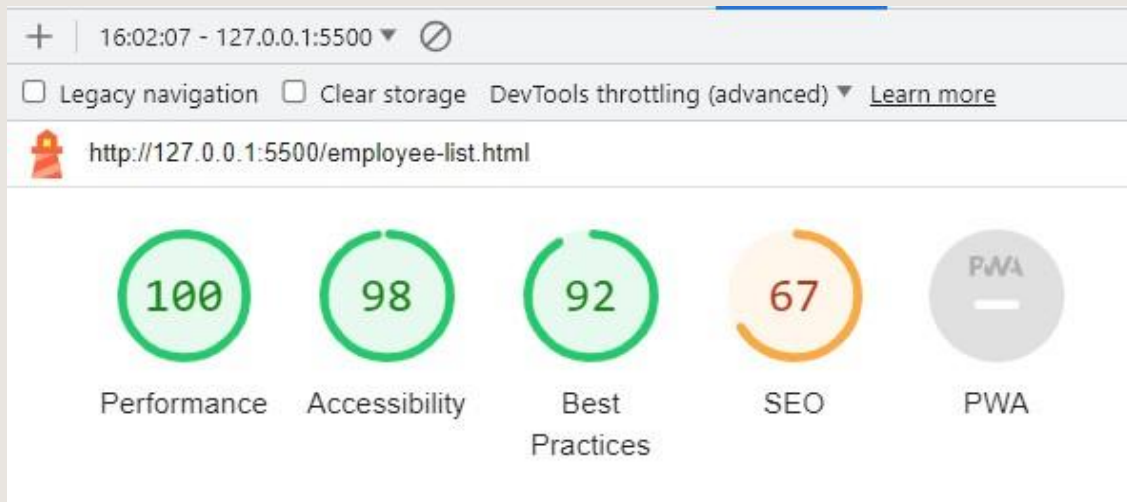


Projet React

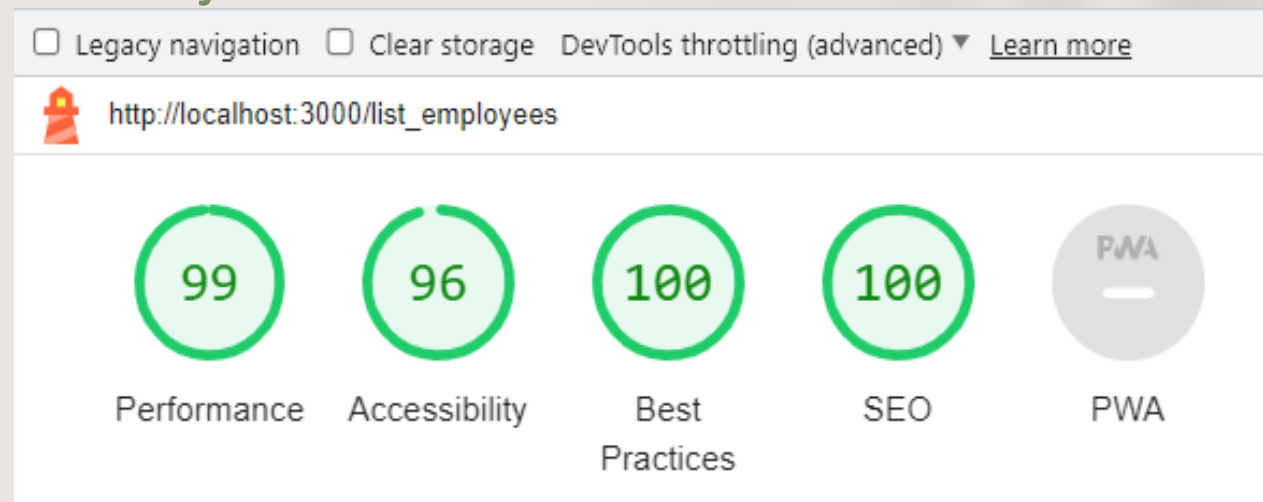


Comparaison Test Lighthouse `listEmployees.js` Desktop

Projet JQuery



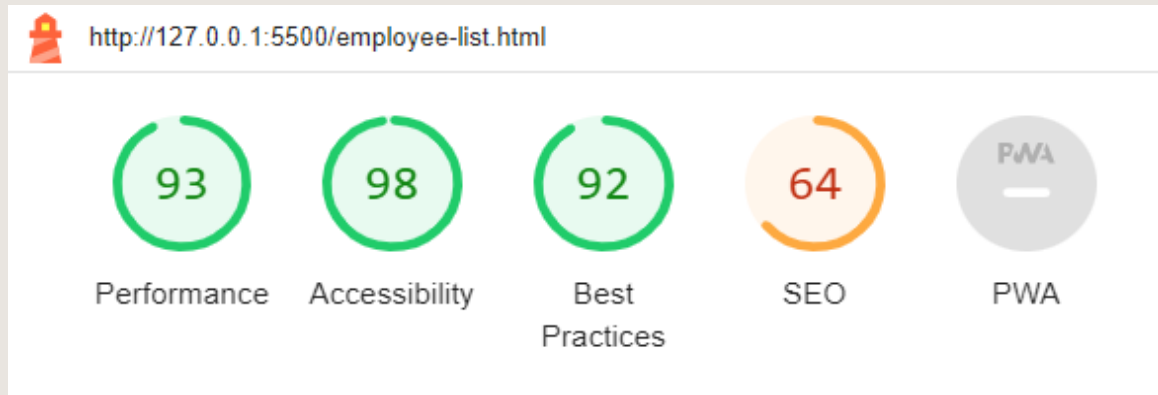
Projet React



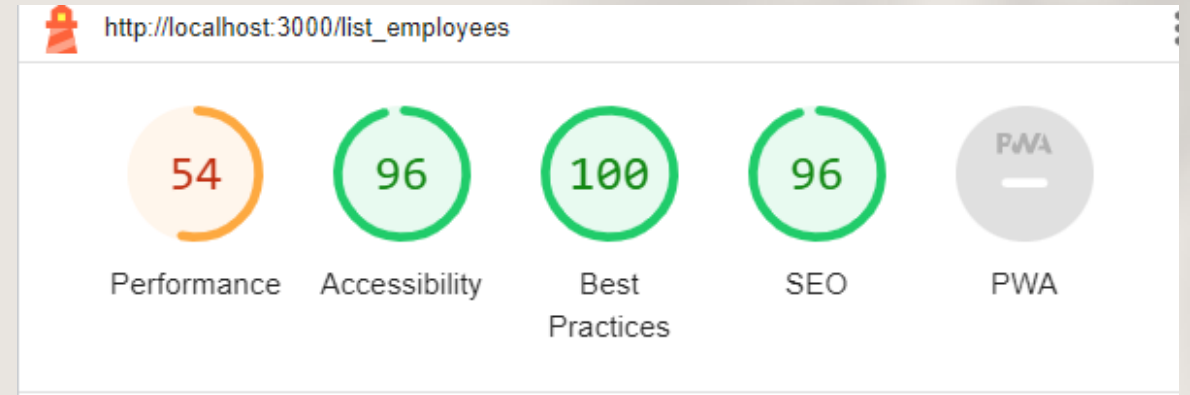
L'analyse Lighthouse montre que le projet jQuery est performant en raison de sa simplicité et de sa légèreté, mais React offre des avantages en termes de maintenabilité et de réutilisabilité du code.

Comparaison Test Lighthouse `listEmployees.js` Smartphone

Projet JQuery



Projet React



L'analyse Lighthouse montre que le projet jQuery est performant en raison de sa simplicité et de sa légèreté, mais React offre des avantages en termes de maintenabilité et de réutilisabilité du code.

Tests unitaires fichier `pagination.test.js`

```
describe("Pagination Component", () => {  
  /**  
   * Test pour vérifier si le composant Pagination est rendu correctement.  
   */  
  it("devrait rendre correctement le composant Pagination", () => {  
    // Définit les valeurs simulées pour le nombre total de pages, la page actuelle et la fonction onPageChange  
    const pageCount = 10;  
    const currentPage = 0;  
    const onPageChange = jest.fn();  
  
    // Rend le composant Pagination avec les valeurs simulées  
    const { getByText } = render(  
      <Pagination  
        pageCount={pageCount}  
        currentPage={currentPage}  
        onPageChange={onPageChange}  
      />  
    );  
  
    // Vérifie la présence de l'élément de pagination pour la première page (page 1)  
    const paginationElement = getByText("1");  
    expect(paginationElement).toBeInTheDocument();  
  });  
});
```

1er : Vérifie que le composant `Pagination` est rendu correctement.

```
it("devrait appeler la fonction onPageChange lorsque la page est changée", () => {  
  // Définit les valeurs simulées pour le nombre total de pages, la page actuelle et la fonction onPageChange  
  const pageCount = 10;  
  const currentPage = 0;  
  const onPageChange = jest.fn();  
  
  // Rend le composant Pagination avec les valeurs simulées  
  const { getByText } = render(  
    <Pagination  
      pageCount={pageCount}  
      currentPage={currentPage}  
      onPageChange={onPageChange}  
    />  
  );  
  
  // Récupère le bouton de la troisième page et simule un clic dessus  
  const thirdPageButton = getByText("3");  
  fireEvent.click(thirdPageButton);  
  
  // Vérifie que la fonction onPageChange a été appelée avec le numéro de page 2 (sélection - 1)  
  expect(onPageChange).toHaveBeenCalledTimes(2);  
});
```

2eme : Vérifie si la fonction `onPageChange` est correctement appelée lorsque la page est changée.

PASS `src/tests/pagination.test.js`

Pagination Component

✓ devrait rendre correctement le composant `Pagination` (55 ms)

✓ devrait appeler la fonction `onPageChange` lorsque la page est changée (24 ms)

Tests unitaires fichier employeesSearch.test.js

```
describe("Component EmployeeSearch", () => {  
  // vérifier si le composant EmployeeSearch est rendu correctement.  
  it("devrait rendre correctement le composant EmployeeSearch", () => {  
    // Crée une fonction simulée pour onSearch  
    const onSearch = jest.fn();  
  
    // Rend le composant EmployeeSearch avec la fonction simulée onSearch  
    const { getByPlaceholderText } = render(  
      <EmployeeSearch onSearch={onSearch} />  
    );  
  
    // Récupère l'élément d'entrée de recherche en utilisant son texte de placeholder  
    const searchInput = getByPlaceholderText("Entrez les lettres du nom...");  
  
    // S'attend à ce que l'élément d'entrée de recherche soit présent dans le document  
    expect(searchInput).toBeInTheDocument();  
  });  
});
```

1er : Vérifie que le composant EmployeeSearch est rendu correctement.

```
it("devrait appeler la fonction onSearch lorsque la valeur de recherche change", () => {  
  // Crée une fonction simulée pour onSearch  
  const onSearch = jest.fn();  
  
  // Rend le composant EmployeeSearch avec la fonction simulée onSearch  
  const { getByPlaceholderText } = render(  
    <EmployeeSearch onSearch={onSearch} />  
  );  
  
  // Récupère l'élément en utilisant son texte de placeholder  
  const searchInput = getByPlaceholderText("Entrez les lettres du nom...");  
  
  // Simule un changement de valeur de recherche  
  fireEvent.change(searchInput, { target: { value: "John" } });  
  
  // exemple avec la fonction onSearch qui appellera "John"  
  expect(onSearch).toHaveBeenCalledWith("John");  
});
```

2eme : Ce test vise à vérifier si la fonction onSearch est correctement appelée lorsque la valeur de recherche change.

```
PASS src/tests/employeeSearch.test.js  
Component EmployeeSearch  
  ✓ devrait rendre correctement le composant EmployeeSearch (52 ms)  
  ✓ devrait appeler la fonction onSearch lorsque la valeur de recherche change (37 ms)
```

```
Tests:      4 passed, 4 total  
Snapshots: 0 total  
Time:       2.914 s  
Ran all test suites related to changed files.
```