



WEALTH HEALTH

HRnet

Faire passer une librairie jQuery vers React

Projet 14

Soutenance du lundi 24 juillet 2023

Table des matières:

Objectif du projet et Spécifications fonctionnelles et Techniques

Partie 1 - Démonstration du prototype

Partie 2 - Phase de développement :

Environnement de développement

- Création du plugin Table
- Création de l'application HRnet avec l'importation du plugin Table

Partie 3 - Phase de Tests

Conclusion et piste de réflexion

Objectif du projet

et Spécifications fonctionnelles et Techniques :

La société financière Wealth Health utilise HRnet, une application web interne en JQuery, pour gérer les dossiers des employés.

objectif: convertir l'application en une application 100% React en partant de la conversion des 4 plugins de l'application (sélecteurs de date, les fenêtres modales, les menus déroulants et le tableau) en composants React.

Spécifications:

- => convertir un plugin parmi les 4 (**Choix du plugin Table**) dans un repository git séparé ,
 - => le documenter
 - => le publier sur npm sous forme de package
 - => Importer ce plugin dans la nouvelle application,
 - => convertir les 3 autres plugins directement dans l'application
 - => tester (utiliser lighthouse pour la performance)

Partie 1

Démonstration du prototype

Partie 2

Phases de développement

Environnement de développement

Environnement de Développement Intégré : Visual Studio Code (VS Code), enrichi par un ensemble d'extensions et de packages pour optimiser le flux de travail.

- **Dépendances essentielles pour l'application React** pour **construire, interagir et naviguer** dans notre application React (React, react-dom, react-icons, react-redux, react-router-dom)
- **Prettier** pour le **formatage de code** => cohérence du style du code.
- **ESLint pour le linting** => qualité du code par des règles de codage permet la détection d'erreurs potentielles.
- **Babel** pour transpiler le code JSX et typescript en JavaScript.
- Extensions : ES7 et Path Intellisense

Git pour la gestion de version: en intégration continue directement dans VS Code.

Utilisation de TypeScript: phase de développement, par le typage statique => éviter de nombreuses erreurs (pour le plugin Table utilisé aussi pour builder l'application)

TypeDoc: génération de documentation pour TypeScript => une documentation claire

Débuggage: Chrome DevTools ainsi que React Developer Tools

Jest => tests unitaires, rapports dans le répertoire "coverage".

Lighthouse, outil de Google intégré dans DevTools => tests de performance et d'accessibilité

Création du plugin Table

Préserver toutes les fonctionnalités d'origine :

=> listage des fonctionnalités existantes

=> réécriture en composants react distincts avec quelques ajouts pertinents

- boutons de tri rendu optionnel sur chaque colonne

```
- example for a column :  
{  
  label: 'First Name',  
  property: 'firstName',  
  disableSort: true, // for disable sorting  
}
```

- gestion du tri par type de date en string

```
{  
  label: 'Date of Birth',  
  property: 'dateOfBirth',  
  dateFormat: 'DD/MM/YYYY' // Use this value for formats 'DD/MM/YYYY',  
  'DD-MM-YYYY' or 'DD.MM.YYYY'  
}
```

```
- For a date string 'DD/MM/YYYY', 'DD-MM-YYYY', or 'DD.MM.YYYY', use:  
`dateFormat: 'DD/MM/YYYY`  
- For a date string 'MM/DD/YYYY', 'MM-DD-YYYY', or 'MM.DD.YYYY', use:  
`dateFormat: 'MM/DD/YYYY`  
- For a date string 'YYYY/MM/DD', 'YYYY-MM-DD', or 'YYYY.MM.DD', use:  
`dateFormat: 'YYYY/MM/DD`  
The accepted and automatically managed separators are '/', '-', and '.'.
```

Compléter l'offre du plugin lors de futures utilisations dans de nouvelles applications :

=> Améliorer l'expérience développeur et utilisateur

- Ajouter de nouvelles fonctionnalités (en option)
- Rendre le style customisable

Ajout de fonctionnalités :

- **Recherche par colonne** (optionnelle sur chaque colonne)

```
- example for a column :  
{  
  label: 'First Name',  
  property: 'firstName',  
  disableFilter: true // for disable filtering  
}
```

- **Management de la visibilité des colonnes prévu :**
 - ne rendre que les colonnes dont la propriété isVisible est en true

- **Exportation des data** en option par l'ajout d'un composant publié à part sur npm

```
import {ExportDataComponent} from 'typescript-exportdata'
```

```
<Table  
  data={datasExample}  
  columns={columnsExample}  
  renderExportDataComponent={({filteredData, columnsManaged}) => (  
    <ExportDataComponent  
      filteredData={filteredData} //don't change this  
      columnsManaged={columnsManaged} //don't change this  
      headerProperty='label' //don't change this  
      csvExport={true} // to have an export bouton for csv format  
      excelExport={true} // to have an export bouton for excel format  
      pdfExport={true} // to have an export bouton for pdf format  
    />  
  )}  
/>
```

- **Boutons d'action** sur chaque ligne de données qui déclenche une fonction qui utilise à la fois l'ID de l'objet et en option l'événement de clic pour effectuer une action spécifique sur cet objet:
 - boutons optionnels: modifier, archiver et supprimer

```
function App() {
  document.title = 'typescript-table';

  const handleEditRow = (id: number | string, e?: Event) => {
    console.log(id);
    if (e) {
      console.log('Click event:', e);
    }
  };

  const handleArchiveRow = (id: number | string, e?: Event) => {
    console.log('archive: ' + id);
    if (e) {
      console.log('Click event:', e);
    }
  };

  const handleDeleteRow = (id: number | string, e?: Event) => {
    console.log('delete: ' + id);
    if (e) {
      console.log('Click event:', e);
    }
  };
}
```

```
return (
  <div className="App">
    <Table
      data={datasExample}
      columns={columnsExample}
      editRowColumnVisible
      handleEditRow={handleEditRow}
      archiveRowColumnVisible
      handleArchiveRow={handleArchiveRow}
      deleteRowColumnVisible
      handleDeleteRow={handleDeleteRow}
    />
  </div>
);


export default App;
```

Option de style possible :

Option de style de couleurs pour les boutons, et select row

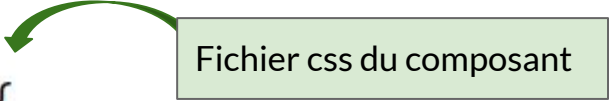
```
<Table
  data={datasExample}
  columns={columnsExample}
  background='red' //change background button and dropdown
  color='#fff' //change color button and dropdown
  hoverBackground='salmon' //change background :hover button and
  dropdown
  selectedRowsBackground='#ffdcd8' //change background selected rows
/>
```

```
export function Table<T>({
  data,
  columns,
  background = '#677e11',
  color = 'white',
  hoverBackground = '#7e9b16',
  selectedRowsBackground = 'rgba(175 228 145 / 20%)',
```

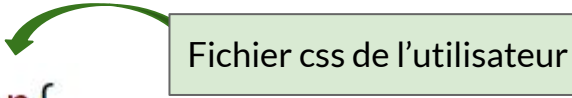


Couleur par défaut

Customiser par l'augmentation de la spécificité des règles CSS



```
.thColor{  
  border-bottom: 1px solid #1b1818;  
  background-color: #b1c46c;  
}
```



```
.box_table .thColor{  
  border-bottom: 1px solid #1b1818;  
  background-color: blue;  
}
```

Documentation du plugin

composée de 3 parties :

- **Le code annoté**

explique le "comment" de mon composant

=> destinés à être consultés directement dans le code source

- **Le fichier README**

explique le "quoi", le "pourquoi" et le "comment" (avec des exemples d'utilisation)

=> première chose que les utilisateurs verront lorsqu'ils visiteront la page du package npm,

=> doit donner une vue d'ensemble claire

- **La documentation TypeDoc**

documentation plus formelle et structurée, peut être consulté séparément, actuellement mis en ligne sur vercel (lien mis dans le readme)

=> produit une documentation HTML interactive

Création de l'application HRnet

Création des 3 autres plugins

Philosophie de React : minimalisme et modularité.

Plugin datePicker :

=> utilisation de la bibliothèque MUI pour créer un composant réutilisable

Plugins liste déroulante et modal :

=> pas de recours à une bibliothèque de composants d'interface utilisateur préexistante

=> évite les dépendances inutiles

- création de composants réutilisables avec gestion de l'accessibilité (aria, tabulation, focus)

Phase de tests

Tests unitaires : Utilisation de Jest

All files src/lib/components

75% Statements 258/344 70.68% Branches 176/249 71.92% Functions 82/114 75.14% Lines 254/338

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

Table tests coverage

File		Statements		Branches		Functions		Lines	
ActionButton.tsx	<div><div></div></div>	80%		4/5		50%		1/2	
Dropdown.tsx	<div><div></div></div>	67.12%		49/73		64.91%		37/57	
EntriesInfo.tsx	<div><div></div></div>	100%		7/7		100%		4/4	
ManageColumns.tsx	<div><div></div></div>	70%		7/10		70%		7/10	
ManageTable.tsx	<div><div></div></div>	81.81%		9/11		77.77%		14/18	
Modal.tsx	<div><div></div></div>	93.1%		27/29		90%		18/20	
Pagination.tsx	<div><div></div></div>	100%		6/6		100%		4/4	
RowsPerPage.tsx	<div><div></div></div>	100%		3/3		100%		0/0	
SearchByProperty.tsx	<div><div></div></div>	80%		8/10		0%		0/2	
SearchDropdown.tsx	<div><div></div></div>	100%		6/6		100%		6/6	
SortButton.tsx	<div><div></div></div>	71.42%		5/7		81.25%		13/16	
Table.tsx	<div><div></div></div>	70.76%		121/171		62.74%		64/102	
TableHeader.tsx	<div><div></div></div>	100%		4/4		100%		8/8	
searchAndResetGlobal.tsx	<div><div></div></div>	100%		2/2		100%		0/0	

All files src/lib/utills

93.75% Statements 60/64 84.81% Branches 67/79 100% Functions 10/10 93.75% Lines 60/64

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
filterData.ts	<div><div></div></div>	100%		10/10		87.5%		7/8	
sortDatas.ts	<div><div></div></div>	92.59%		50/54		84.5%		60/71	

All files

73.73% Statements **306/415** **53.73%** Branches **115/214** **72.8%** Functions **83/114** **73.69%** Lines **297/403**

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

Application tests coverage

File	Statements	Branches	Functions	Lines
src	<div><div></div></div> 66.66%6/9	<div><div></div></div> 100%0/0	<div><div></div></div> 75%3/4	<div><div></div></div> 57.14%4/7
src/components	<div><div></div></div> 68.66%195/284	<div><div></div></div> 52.09%87/167	<div><div></div></div> 67.53%52/77	<div><div></div></div> 68.92%193/280
src/mocks	<div><div></div></div> 100%2/2	<div><div></div></div> 100%0/0	<div><div></div></div> 100%0/0	<div><div></div></div> 100%2/2
src/pages	<div><div></div></div> 100%6/6	<div><div></div></div> 50%1/2	<div><div></div></div> 100%5/5	<div><div></div></div> 100%5/5
src/services	<div><div></div></div> 90.9%20/22	<div><div></div></div> 75%3/4	<div><div></div></div> 100%3/3	<div><div></div></div> 90.9%20/22
src/store	<div><div></div></div> 77.08%37/48	<div><div></div></div> 66.66%6/9	<div><div></div></div> 77.77%14/18	<div><div></div></div> 77.27%34/44
src/utlis	<div><div></div></div> 90.9%40/44	<div><div></div></div> 56.25%18/32	<div><div></div></div> 85.71%6/7	<div><div></div></div> 90.69%39/43

All files src/pages

100% Statements 6/6 50% Branches 1/2 100% Functions 5/5 100% Lines 5/5

Press n or j to go to the next uncovered block, b , p or k for the previous block.

Filter:

File		Statements	Branches	Functions	Lines				
Home.tsx	<div></div>	100%	1/1	100%	0/0	100%	1/1	100%	1/1
ListEmployees.tsx	<div></div>	100%	3/3	50%	1/2	100%	2/2	100%	2/2
NewEmployee.tsx	<div></div>	100%	1/1	100%	0/0	100%	1/1	100%	1/1
Spinner.jsx	<div></div>	100%	1/1	100%	0/0	100%	1/1	100%	1/1

All files src/components

68.66% Statements 295/284 52.09% Branches 87/167 67.53% Functions 52/77 68.92% Lines 193/280

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements	Branches	Functions	Lines
AddressAndDepartmentForm.tsx	100%	3/3	100%	2/2
ArchiveEmployeeContent.tsx	100%	2/2	100%	2/2
DatePickerComponent.tsx	100%	18/18	75% 15/20	100%
DeleteEmployeeContent.tsx	100%	3/3	100%	2/2
Dropdown.tsx	64.93%	50/77	59.67% 37/62	72.22% 13/18
EditEmployeeContent.tsx	100%	2/2	100%	1/1
FormNewEmployee.tsx	65.95%	31/47	21.05% 4/19	75% 6/8
Header.tsx	100%	2/2	100%	1/1
InputField.tsx	100%	14/14	80% 12/15	100%
Modal.tsx	66.66%	20/30	46.87% 15/32	85.71% 6/7
ModalEmployeesContent.tsx	88.88%	16/18	50% 2/4	100%
NavLinkComponent.tsx	100%	2/2	100%	1/1
TableEmployees.tsx	48.48%	32/66	13.33% 2/15	26.08% 6/23

All files src/utlis

90.9% Statements 49/44 56.25% Branches 15/32 85.71% Functions 6/7 90.69% Lines 39/43

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements	Branches	Functions	Lines
controlDate.tsx	100%	11/11	66.66% 4/6	100%
controlName.tsx	77.77%	14/18	56.25% 9/16	77.77% 14/18
department.tsx	100%	1/1	100%	0/0
linkData.tsx	100%	2/2	100%	0/0
modalConstants.tsx	100%	11/11	50% 5/10	66.66% 2/3
states.tsx	100%	1/1	100%	0/0

Test de performance Lighthouse de devTools

Test de performance : page Create Employee

Jquery

React

Desktop

99

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice.](#)



▲ 0-49 ■ 50-89 ● 90-100

100

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice.](#)



▲ 0-49 ■ 50-89 ● 90-100

Mobile

91

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice.](#)



▲ 0-49 ■ 50-89 ● 90-100

95

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice.](#)



▲ 0-49 ■ 50-89 ● 90-100

Test de performance : page View Current Employees

Jquery

React

Desktop

100

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice](#).

▲ 0-49 ■ 50-89 ● 90-100



100

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice](#).

▲ 0-49 ■ 50-89 ● 90-100



96

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice](#).

▲ 0-49 ■ 50-89 ● 90-100



95

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice](#).

▲ 0-49 ■ 50-89 ● 90-100



Analyse des différences

- + **React: utilisation d'un Dom virtuel plus rapide que le DOM réel utilisé par jQuery**
=> minimisation du nombre de manipulations de DOM réelles nécessaires.
- **React: plus lourd que jQuery en termes de taille de bundle,**
=> affecte les performances de chargement initial de l'application
=> possibilité de minimiser cet impact grâce à des techniques de fractionnement du code (code-splitting)
ex: utilisation de `React.lazy()`



La comparaison ne vaut que lorsque l'on compare des applications équivalentes,

Application React : Ajout de style et de nombreuses fonctionnalités

La comparaison est-elle alors dénuée de sens ?

=> Bien au contraire, react apparaît d'autant plus performant au vu des résultats obtenus

Test d'accessibilité, de bonne pratique et SEO

Lighthouse de devTools

jQuery

Create Employee desktop

React

 http://127.0.0.1:5502/index.html




 http://localhost:63954/newemployee



View Current Employees desktop

 http://127.0.0.1:5502/employee-list.html



 http://localhost:63954/listemployees



Conclusion :

Le résultat des tests de comparaison montre l'intérêt d'avoir fait passer l'application de jQuery en React.

Evolution de l'application :

A terme il serait judicieux de publier également les composants modal et liste déroulante pour ensuite pouvoir les importer car ces composants sont utilisés dans le table ainsi que dans l'application HRnet.