P14 - Faites passer une librairie jQuery vers React

•••

par Thomas RANQUE

Le projet

WealthHealth est une grande société financière qui utilise une application web interne pour gérer les dossiers des employés.

L'application est ancienne et utilise jQuery côté front end, ce qui entraîne des bugs considérables et une augmentation des plaintes en interne.

Phase 1 : Bascule sur une application React

Phase 2 : Conversion des plugins jQuery

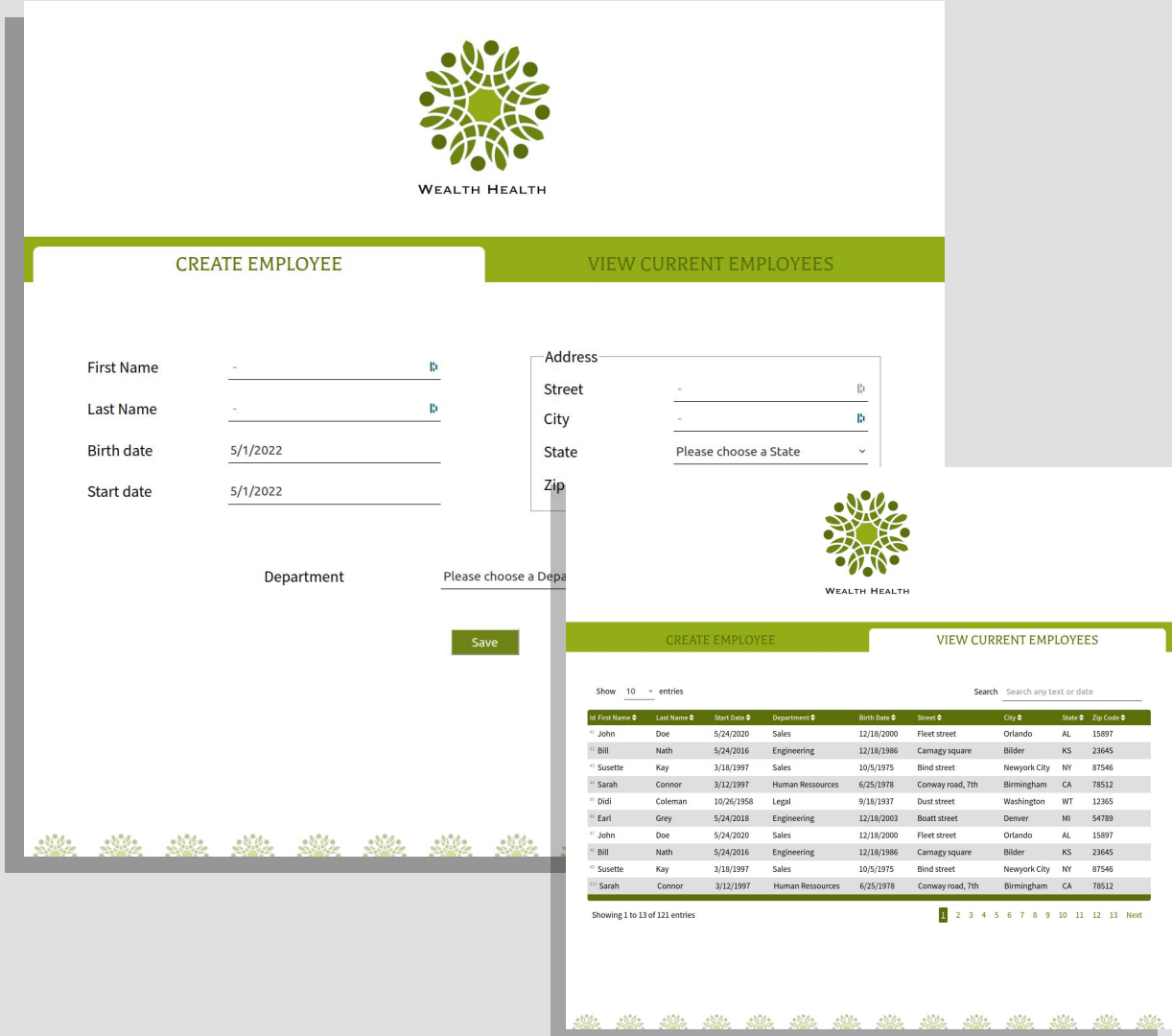
Phase 3: Création de librairie React – NPM

La structure du projet

Le projet est développé en tant qu'application **React**.

React-router v6 est employé pour le routage.

React Context est employé pour la gestion d'un store global.



PHASE 1 - Utilisateur

Router & Store Provider

Index.js

La librairie FontAwesome fournit les icônes requises.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { library } from '@fortawesome/fontawesome-svg-core'
import {faChevronUp, faChevronDown, faSort, faSortUp, faSortDown} from '@fo
import './style/index.css';
import App from './components/App';
import reportWebVitals from './reportWebVitals';

library.add(faChevronUp, faChevronDown, faSort, faSortUp, faSortDown)

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <div className='container'>
      <App />
    </div>
  </React.StrictMode>
);
```

```
import { BrowserRouter, Routes, Route } from 'react-router-dom'
import '../style/app.css';
import Header from './Header'
import Footer from './Footer'
import Create from '../pages/Create'
import View from '../pages/View'
import Error from '../pages/Error'
import { EmployeeProvider } from '../context/employeesCtx'

function App() {
  return (
    <EmployeeProvider>
      <BrowserRouter basename="/P14-wealth_health">
        <div className="App">
          <Header />
          <Routes>
            <Route path='/' element={<Create />} />
            <Route path='/view' element={<View />} />
            <Route path='*' element={<Error />} />
          </Routes>
          <Footer />
        </div>
      </BrowserRouter>
    </EmployeeProvider>
  );
}

export default App;
```

App.js

- EmployeeProvider -> Store avec useContext Hook
- Etablissement des routes de l'application
- Gestion des routes erronées (Error)
- Basename pour la production (démo)

Store

useContext()

L'emploi d'un state manager comme Redux semble disproportionné pour cette mini-app!

Le choix du manager du Store s'est porté sur le React Hook useContext().

Un context est créé, qui est placé dans un Custom Hook Context.

En parallèle, un Provider est défini, qui dispose d'un initialState, de variables (getter/setter) et de méthodes qui viennent alimenter le Store qui wrap l'application.

On exporte le Store /Provider et le custom hook useEmployeeContext() qui mettent les données globales du store à disposition des composants requis.

Context

InitialState

Provider

Provided data & functions

Custom Hook

```
1  import { createContext, useContext, useEffect, useState } from "react";
2  import { employees } from '../utils/defaultEmployees'
3
4  // Context
5  export const EmployeesCtx = createContext()
6
7  // Initial State
8  const initialState = JSON.parse(localStorage.getItem('WH_employees')) || employees
9
10 // Provider
11 function EmployeeProvider(props) {
12     const [employees, setEmployees] = useState(initialState)
13     const [initForm, setInitForm] = useState(false)
14
15     // SORT table management
16     const [sortBy, setSortBy] = useState(null)
17     const [sortWay, setSortWay] = useState(null)
18     const setSorting = { setSortBy, setSortWay }
19     const sortInfo = { sortBy, sortWay }
20
21     // INIT form components
22     const [init, setInit] = useState(false)
23     const initComponent = { init, setInit }
24
25     useEffect(() => {
26         localStorage.setItem('WH_employees', JSON.stringify(employees))
27     }, [employees])
28
29     function add(employee) {
30         setEmployees([...employees, employee])
31     }
32
33 >   function removeByIndex(index) {…
37     }
38
39 >   function removeByName(name) {…
51     }
52
53     const employeesData = {
54         employees,
55         initForm,
56         setInitForm,
57         setSorting,
58         sortInfo,
59         initComponent,
60         add,
61         removeByIndex,
62         removeByName
63     }
64     return (<EmployeesCtx.Provider value={employeesData} {...props} />)
65 }
66
67 // Custom Hook Context
68 function useEmployeesContext() {
69     return useContext(EmployeesCtx)
70 }
71
72 // Export provider & custom hook ctx
73 export { EmployeeProvider, useEmployeesContext }
```

Conversion plugins jQuery

#1.Dropdown menu

C'est le plugin qui sera converti en librairie React. Le plugin ***create-component-lib*** est utilisé.

Six arguments peuvent être employés:

- options – Required
- setvalue – Required
- initComponent – Required
- label – Optional
- placeholder – Optional
- log – optional

```
import React, { useEffect, useRef, useState } from 'react'
import { Fragment } from "react"
import PropTypes from 'prop-types'
import './simpleSelectMenu.css'

/**
 Simple select menu Library
 @param {string} label - Label for the select menu. Optional
 @param {array} options - Array of all the options. Required
    @param {string} elt - Option 1 : array of strings. Value returned is the value.toLowerCase() with white spaces converted to underscore
    @param {object} elt - Option 2 : array of objects. Value returned is the value property of object
        @param {string} name - Text to display in select menu.
        @param {string} value - Value returned when selected.
@param {string} placeholder - Text to display at start. Optional.
@param {boolean} log - Displays nodeElement & value returned in console. Optional. Default to true.
@param {function} setvalue - Setter to return the selected value to parent Component. Required
@param {object} initComponent - Init getter/setter. Required
    @param {boolean} init - Getter to init action state.
    @param {function} setInit - Setter to set init action to false.
@returns SimpleSelectMenu component.
*/
const SimpleSelectMenu = ({ label = 'Label', options = ['Option 1', 'Option 2'], placeholder, log = true, setvalue, initComponent }) => {

    const selectMenu = useRef()
    const {init, setInit} = initComponent
    useEffect(() => {
        if (init === true) {
            if (placeholder !== undefined && placeholder !== false) {
                setVal('')
            } else {
                if (typeof options[0] === 'string') {
                    setVal(options[0])
                } else {
                    setVal(options[0].value)
                }
            }
        }
        setInit(false)
    }, [init])

    const [val, setVal] = useState('')

    function _returnValue(e) {
        log === true && console.log(e.target, `Value : ${e.target.value}`)
        setVal(e.target.value)
        setvalue(e.target.value)
    }

```

```
    return (
        <Fragment>
            <label name={label.toLowerCase()} className="simple-select-menu-label">{label}<
            <select
                ref={selectMenu}
                className="simple-select-menu-select"
                value={val}
                onChange={_returnValue}
            >
                {placeholder !== undefined && placeholder !== false && (
                    <option className="simple-select-menu-option" value="">{placeholder}</option>

                )}
                {options && typeof options[0] === 'string' && options.map((option, i) =>
                    (<option className="simple-select-menu-option" value={option.toLowerCase().replace(' ', '_')} key={`ssm-${i}`}>{option}</option>)
                )
                }
                {options && typeof options[0] === 'object' && options.map((option, i) =>
                    (<option className="simple-select-menu-option" value={option.value} key={`ssm-${i}`}>{option.name}</option>)
                )
                }
            </select>
        </Fragment>
    )
```

Conversion plugins jQuery

#1.Dropdown menu

PropTypes permet de vérifier les arguments employés avec la lib pour prévenir les erreurs de paramétrage.

Un fichier de mise en situation de différents cas d'usage est créé pour tester la lib en direct. Ce fichier sera repris dans le README.md pour aider l'utilisateur futur dans son paramétrage.

```
77  SimpleSelectMenu.propTypes = {
78      label: PropTypes.string,
79      options: PropTypes.arrayOf(
80          PropTypes.oneOfType([
81              PropTypes.string,
82              PropTypes.objectOf(PropTypes.string)
83          ]).isRequired
84      ),
85      placeholder: PropTypes.oneOfType([
86          PropTypes.string,
87          PropTypes.bool
88      ]),
89      log: PropTypes.bool,
90      setvalue: PropTypes.func.isRequired,
91      initComponent: PropTypes.objectOf(
92          PropTypes.oneOfType([
93              PropTypes.bool,
94              PropTypes.func,
95          ]).isRequired
96      )
97  }
```

```
1   import { render } from "react-dom";
2   import { SimpleSelectMenu } from "./lib";
3   import { useEffect, useState } from 'react'
4   import './index.css'
5
6   const App = () => {
7     const [menu1Value, setMenu1Value] = useState()
8     const [menu2Value, setMenu2Value] = useState()
9     const [init, setInit] = useState(false)
10    const initComponent = {init, setInit}
11
12    useEffect(() => console.log('INIT MENUS -', init), [init])
13
14    function resetMenu(e) {
15      e.preventDefault()
16      setInit(true)
17    }
18
19    return (
20      <form className="container" onSubmit={resetMenu}>
21        <h1>Simple select menu</h1>
22
23        <SimpleSelectMenu
24          label="Select menu with strings (log = false)"
25          options={["Option 1", "Option 2"]}
26          placeholder="Please choose an option"
27          log={false}
28          setvalue={setMenu1Value}
29          initComponent={initComponent}
30        />
31        {menu1Value !== '' && menu1Value !== undefined && <span><em>returned value: {menu1Value}</em></span>}
32        <br />
33
34        <SimpleSelectMenu
35          label="Select menu with objects (log = true)"
36          options={[{ name: 'Option 1', value: 'opt1' }, { name: 'Option 2', value: 'opt2' }]}
```

Conversion plugins jQuery

#1.Dropdown menu

Le *build* de la lib est effectué, puis le *publish* afin de la mettre à disposition des utilisateurs.
Publish met en ligne la lib et le README.md.

La lib est ensuite importée dans le projet et employée afin de gérer les menus select de la page *Create.*

Nascent Prototype Metaverse

Products Pricing Documentation

npm

Search packages

Search

simple-select-menu

0.1.10 • Public • Published a day ago

Readme | Explore BETA | 3 Dependencies | 0 Dependents | 11 Versions | Settings

simple-select-menu - REACT

Create a select menu providing options to select. It can take a placeholder as desired, returns value and can be set back to default state once form has been sent.

Installation

```
npm install simple-select-menu
```

Usage

Six parameters shall be implemented

- **label** {string} - The text to display in front of the select menu. *Optional*
- **options** {array} - Options MUST BE an array. *Required*
  - **case 1** : {string} - Strings used as-is for displayed text, value returned is a lower case version of the string with white spaces converted to underscore ('_').
  - **case 2** : {object} - {name: 'optionText', value: 'returnedValue'}
    - **name** {string} - Displayed text.
    - **value** {string} - Returned value on select.
- **placeholder** {string} - Text displayed at start when no selection has been done. *Optional. Can be either unset or set to false.*
- **log** {boolean} - Displays nodeElement and returned value in console. *Optional. Default to true.*
- **setvalue** {function} - Setter to return the selected value to parent Component. *Required*
- **initComponent** {object} - Init getter/setter to set menu back to default. *Required*
  - **init** {boolean} - Getter to init action state.
  - **setInit** {function} - Setter to set init action to false once menu has gone back too default state.

Get selected value

Install

```
> npm i simple-select-menu
```

Weekly Downloads

426

Version 0.1.10

License none

Unpacked Size 8.68 kB

Total Files 5

Last publish

a day ago

Collaborators

>_Try on RunKit

Report malware

Conversion plugins jQuery

#2.Modal

C'est le plugin le plus simple. Un simple écran opaque qui est géré via le CSS, qui affiche un message de confirmation d'enregistrement.

Une fonction de fermeture de la modale permet de l'effacer au click du bouton.

```
1   import React from 'react'
2   import '../style/modal.css'
3
4   const ModalConfirm = ({show}) => {
5       function closeModal() {
6           show(false)
7       }
8
9       return (
10          <div className='modal-confirm'>
11              <div className='modal-confirm-message'>
12                  Employee's data have been successfully stored!
13                  <button className='modal-confirm-btn' onClick={closeModal}>Close</button>
14              </div>
15          </div>
16      )
17  }
18
19  export default ModalConfirm        You, 2 days ago • Modal done & Reset inputs to default when form is…
```

Conversion plugins jQuery

#3. Table

La table de présentation des employés prends 2 arguments :

- Un ARRAY des employés.
- Un NUMBER de l'index du 1er résultat affiché.

Elle comporte une fonction toggleSort() qui permet d'indiquer au composant parent (View) la colonne de tri choisie et le sens du tri.

Les fonctions de tri et de recherche sont implémentés dans View.

```
1  import React from 'react'
2  import '../style/dataTable.css'
3  import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
4  import PropTypes from 'prop-types'
5  import { useEmployeesContext } from '../context/employeesCtx'
6
7
8  /**
9   * It renders a table of employees, with a header row that allows the user to sort the table by
10  * clicking on the header
11  * @param {Array} data - List of employees.
12  *   @param {Object} employee - Employee props.
13  * @param {Number} start - Beginning number of the displayed employees.
14  * @returns A table of employees
15  */
16 const DataTable = ({ data, start }) => {
17     // LEGENDS
18     const columns = {
19         'firstName': 'First Name',
20         'lastName': 'Last Name',
21         'startDate': 'Start Date',
22         'department': 'Department',
23         'birthDate': 'Birth Date',
24         'street': 'Street',
25         'city': 'City',
26         'stateName': 'State',
27         'zipCode': 'Zip Code'
28     }
29
30     // SORT global variable management
31     const employeesCtx = useEmployeesContext()
32     const { setSortBy, setSortWay } = employeesCtx.setSorting
33     const { sortBy, sortWay } = employeesCtx.sortInfo
34
35     // SORT function - toggle WAY up / down / null
36     function toggleSort(e) {
37         let target
38         if (e.target.classList.contains('employee-legend-col')) {
39             target = e.target.querySelector('svg')
40         } else {
41             if (e.target.classList.contains('svg-inline--fa')) {
42                 target = e.target
43             } else {
44                 target = e.target.parentNode
45             }
46         }
47         const sortColumn = target.id.split('sort-')[1]
48         setSortBy(sortColumn)
49         if (sortWay === null) {
50             setSortWay('up')
```

Conversion plugins jQuery

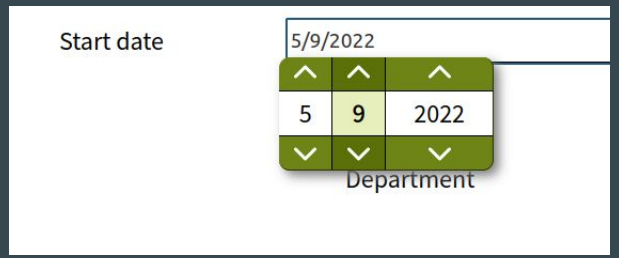
#4. Date picker

Le Datepicker prend 2 arguments :

- Un LABEL
- Un SETVALUE pour retourner la date choisie par l'utilisateur au composant parent.

Le choix a été fait d'une ergonomie simplifiée avec déroulé des dates à la molette ou au click sur des chevrons.

Le *mouseLeave* déclenche un "display: none" sur le composant, et la sélection de la date.




```python
import '../style/datepicker.css'
import React, { Fragment, useEffect, useRef, useState } from 'react'
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import PropTypes from 'prop-types'
import { useEmployeesContext } from '../context/employeesCtx'

/**
 * It's a date picker component that allows the user to select a date using a mouse wheel or chevrons
 * up/down
 * @param {string} label - Text to display in front of input.
 * @param {function} setvalue - Setter to update date selected to parent component.
 * @returns A date picker component.
 */
const Datepicker = ({ label, setvalue }) => {
    // GET actual date
    const now = new Date()
    const today = {}
    today.month = now.getMonth()
    today.day = now.getDate()
    today.year = now.getFullYear()
    // { month: 12, day: 29/30/31, year: 1900-2050 }

    // SET component back to default once form is sent
    const employeesCtx = useEmployeesContext()
    const {init, setInit} = employeesCtx.initComponent
    const picker = useRef()
    const [inputDate, setInputDate] = useState('')

    // DATE variables
    const monthNumbers = [...Array(12).keys()].map(i => i + 1)
    const [monthNum, setMonthNum] = useState(today.month)
    const [dayNumbers, setDayNumbers] = useState([...Array(31).keys()].map(i => i + 1))
    const [dayNum, setDayNum] = useState(today.day - 1)
    const yearNumbers = [...Array(2050).keys()].map(i => i + 1).filter(i => i >= 1900)
    const [yearNum, setYearNum] = useState(today.year - 1900)

    // RESET DatePicker once form is sent
    useEffect(() => {
        if (init === true) {
            console.log('INIT DATE PICKER', today);
            setMonthNum(today.month)
            setDayNum(today.day - 1)
            setYearNum(today.year - 1900)
            setInit(false)
        }
    // eslint-disable-next-line react-hooks/exhaustive-deps
    }, [init])

    // UPDATE number of days when changing month
    useEffect(() => {
        if ([3, 5, 8, 10].includes(monthNum)) {
```

Page Create

Ce composant dispose des fonctionnalités suivantes :

- Formulaire controlé React.
- Fonction de check de la validité des entrées utilisateur.
- Gestion d'alerte des erreurs.
- Reset de valeur des inputs.
- Ajout de l'employé à la base.
- Display de la modale.

```
104     // DISPLAY ERRORS OR SAVE DATA //
105     if (errorCounter > 0) {
106       return
107     } else {
108       // SAVE datas to Context Store
109       employeesCtx.add(res)
110       employeesCtx.setInitForm(true)
111
112       // RESET all form
113       setFirstName('')
114       setLastName('')
115       setBirthDate('')
116       setStartDate('')
117       setStreet('')
118       setCity('')
119       setStateName('')
120       setZipCode('')
121       setDepartment('')
122
123       // SET menu components back to default
124       setInit(true)
125
126       // DISPLAY confirmation modal
127       setDataStored(true)
128     }
```

```
1  import '../style/create.css'
2  import React, { useRef, useState } from 'react'
3  import Datepicker from '../components/Datepicker'
4  import ModalConfirm from '../components/ModalConfirm'
5  import { SimpleSelectMenu } from 'simple-select-menu'
6  import { states } from '../utils/states'
7  import { useEmployeesContext } from '../context/employe
8
9  /**
10  * It's a form that allows the user to create a new emp
11  * @returns A form with a few inputs and a submit butto
12  */
13 const Create = () => {
14   const departmentOptions = ["Sales", "Marketing", "Eng
15
16   // DISPLAY MODAL when new data is stored
17   const [dataStored, setDataStored] = useState(false)
18
19   // SET component back to default once form is sent
20   const employeesCtx = useEmployeesContext()
21   const initComponent = employeesCtx.initComponent
22   const {setInit} = initComponent
23
24   // FORM values
25   const [firstName, setFirstName] = useState('')
26   const [lastName, setLastName] = useState('')
27   const [startDate, setStartDate] = useState(new Date()
28   const [birthDate, setBirthDate] = useState(new Date()
29   const [street, setStreet] = useState('')
30   const [city, setCity] = useState('')
31   const [stateName, setStateName] = useState('')
32   const [zipCode, setZipCode] = useState('')
33   const [department, setDepartment] = useState('')
34
35   // ERRORS refs
36   const errFirstName = useRef(null)
37   const errLastName = useRef(null)
38   const errStartDate = useRef(null)
39   const errBirthDate = useRef(null)
40   const errStreet = useRef(null)
41   const errCity = useRef(null)
42   const errStateName = useRef(null)
43   const errZipCode = useRef(null)
44   const errDepartment = useRef(null)
45
46   // CHECK FORM values & RECORD new employee
47   function handleSubmit(e) {
48     e.preventDefault()
49     const address = { street, city, stateName, zipCode }
50     const res = { firstName, lastName, birthDate, startDate, address, department }
51     let errorCounter = 0
```

```
46   // CHECK FORM values & RECORD new employee
47   function handleSubmit(e) {
48     e.preventDefault()
49     const address = { street, city, stateName, zipCode }
50     const res = { firstName, lastName, birthDate, startDate, address, de
51     let errorCounter = 0
52
53     // VALIDITY check
54     if (firstName.length < 2) {
55       errFirstName.current.innerText = `First name should be at least 2
56       errorCounter++
57     } else {
58       errFirstName.current.innerText = ` `
59     }
60
61     if (lastName.length < 2) {
62       errLastName.current.innerText = `Last name should be at least 2 ch
63       errorCounter++
64     } else {
65       errLastName.current.innerText = ` `
66     }
67
68     const age = new Date().getFullYear() - new Date(birthDate).getFullYe
69     if (age < 17) {
70       errBirthDate.current.innerText = `Employee's age is ${age}. It sho
71       errorCounter++
72     } else {
73       errBirthDate.current.innerText = ` `
74     }
75
76     if (city.length < 2) {
77       errCity.current.innerText = `City is required.`
78       errorCounter++
79     } else {
80       errCity.current.innerText = ` `
81     }
```

```
94    // SEARCH table for value & FILTER table
95    function searchData(e) {
96      setSortBy(null)
97      setSortWay(null)
98      const target = e.target.value.toString().toLowerCase()
99
100     if (target.length >= 3) {
101       // USER errases letter
102       if (target.length < searchLength) {
103         const searchResult = employeesCtx.employees.filter(employee =>
104           Object.values(employee).some(field => field.toString().toLowerCase().includes(target) ||
105             Object.values(employee.address).some(field => field.toString().toLowerCase().includes(target))))
106         setAllEmployees(searchResult)
107         setSearchLength(target.length)
108         return
109       }
110       // USER adds letter
111       const searchResult = allEmployees.filter(employee =>
112         Object.values(employee).some(field => field.toString().toLowerCase().includes(target) ||
113           Object.values(employee.address).some(field => field.toString().toLowerCase().includes(target))))
114       setAllEmployees(searchResult)
115       setSearchLength(target.length)
116
117     } else {
118       // RESET table to general table
119       setAllEmployees([...employeesCtx.employees])
120       setSearchLength(0)
121     }
```

```
28    // SORT table management
29    const { setSortBy, setSortWay } = employeesCtx.setSorting
30    const { sortBy, sortWay } = employeesCtx.sortInfo
31
32    // SORT table by column & way
33    useEffect(() => {
34      // RESET sorting
35      if (sortBy === null) {
36        setAllEmployees([...employeesCtx.employees])
37        return
38      }
39      // LAUNCH sorting
40      setAllEmployees(allEmployees.sort((a, b) => {
41        let valueA = a[sortBy]
42        let valueB = b[sortBy]
43        if (['street', 'city', 'stateName', 'zipCode'].includes(sortBy)) {
44          valueA = a.address[sortBy]
45          valueB = b.address[sortBy]
46        }
47
```

Page View

Ce composant dispose des fonctionnalités suivantes :

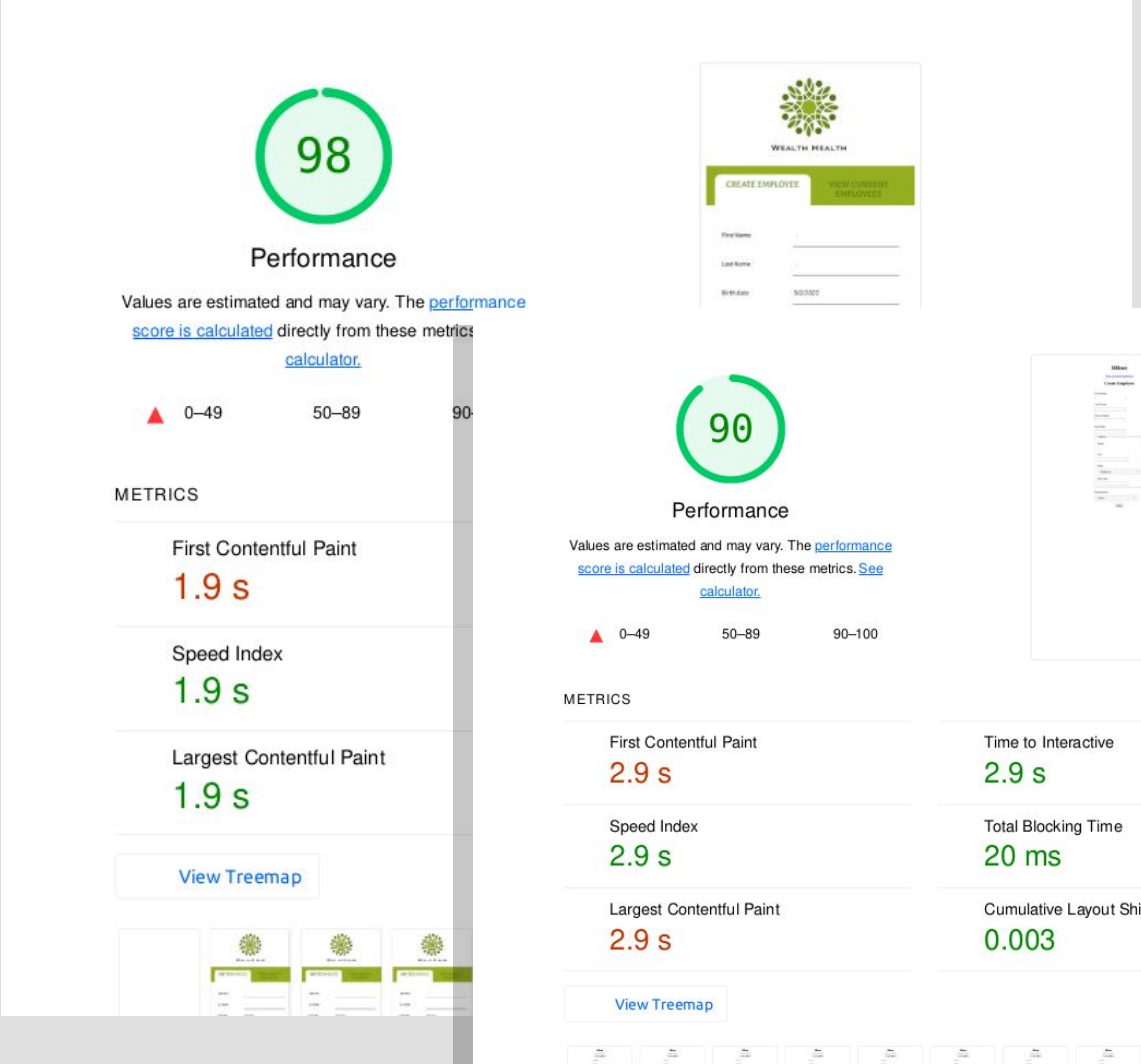
- Nombre de résultats par page.
- Nombre de pages.
- Array de résultats à afficher.
- Page active
- Fonction de tri.
- Fonction de recherche

L'array des employés et les variables de nom de colonne de tri et de sens d'affichage (up/down) sont importés du Store global.

LightHouse

Le test réalisé sur la version de production du projet révèle d'excellentes performances, meilleures que la version précédente disposant de plugins jQuery.

L'adresse de prod est : https://peanuts-83.github.io/P14-wealth_health/



GitHub

Le projet est hébergé sur Github avec un fichier README.md explicite :

- Contexte / technos
- Installation
- Utilisation
- Paramétrage

Les commits sont réguliers et explicites.