Ejercicios del Tema 11

Ejercicio 1

Para este ejercicio partiremos de un proyecto nuevo creado con:

create-react-app Ej1Tema11

Vamos a crear una aplicación to-do en la cual vamos a modificar los cambios mediante Redux. Después iremos creando las clases que se indican para tener la aplicación funcionando.
Empezamos modificando nuestro index.js para usar Provider y cargar un App que crearemos más adelante.

index.js

```
import React from 'react'
import { render } from 'react-dom'
import { Provider } from 'react-redux'
import { createStore } from 'redux'
import todoApp from './reducers'
import App from './components/App'

let store = createStore(todoApp)

render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
)
```

Seguidamente crearemos una carpeta llamada actions donde añadiremos las acciones de nuestra aplicación, estas acciones serán las encargadas de modificar el estado de la aplicación.

 © Imagina Formación

actions/index.js

```
let nextTodoId = 0
export const addTodo = text => {
  return {
    type: 'ADD_TODO',
    id: nextTodoId++,
    text
  }
}

export const setVisibilityFilter = filter => {
  return {
    type: 'SET_VISIBILITY_FILTER',
    filter
  }
}

export const toggleTodo = id => {
  return {
    type: 'TOGGLE_TODO',
    id
  }
}
```

 © Imagina Formación

Ahora creamos la carpeta reducers para incluir todas las acciones que hemos creado arriba.

reducers/todos.js

```
const todos = (state = [], action) => {
  switch (action.type) {
    case 'ADD_TODO':
      return [
        ...state,
        {
          id: action.id,
          text: action.text,
          completed: false
        }
      ]
    case 'TOGGLE_TODO':
      return state.map(todo =>
        (todo.id === action.id)
          ? {...todo, completed: !todo.completed}
          : todo
      )
    default:
      return state
  }
}
```

 © Imagina Formación

reducers/visibilityFilter.js

```
const visibilityFilter = (state = 'SHOW_ALL', action) => {
  switch (action.type) {
    case 'SET_VISIBILITY_FILTER':
      return action.filter
    default:
      return state
  }
}

export default visibilityFilter
```

reducers/index.js

```
import { combineReducers } from 'redux'
import todos from './todos'
import visibilityFilter from './visibilityFilter'

const todoApp = combineReducers({
  todos,
  visibilityFilter
})

export default todoApp
```

 © Imagina Formación

Creamos la clase referente a único item To-do

components/Todo.js

```
import React from 'react'
import PropTypes from 'prop-types'

const Todo = ({ onClick, completed, text }) => (
  <li
    onClick={onClick}
    style={ {
      textDecoration: completed ? 'line-through' : 'none'
    }}
  >
    {text}
  </li>
)

Todo.propTypes = {
  onClick: PropTypes.func.isRequired,
  completed: PropTypes.bool.isRequired,
  text: PropTypes.string.isRequired
}

export default Todo
```

 © Imagina Formación

Ahora creamos una lista donde mostraremos todos los elementos To-do disponibles.

components/TodoList.js

```
import React from 'react'
import PropTypes from 'prop-types'
import Todo from './Todo'

const TodoList = ({ todos, onTodoClick }) => (
  <ul>
    {todos.map(todo => (
      <Todo key={todo.id} {...todo} onClick={() =>
onTodoClick(todo.id)} />
    ))}
  </ul>
)

TodoList.propTypes = {
  todos: PropTypes.arrayOf(
    PropTypes.shape({
      id: PropTypes.number.isRequired,
      completed: PropTypes.bool.isRequired,
      text: PropTypes.string.isRequired
    }).isRequired
  ).isRequired,
```

 © Imagina Formación

Aquí tenemos el código del enlace con una llamada onClick

components/Link.js

```
import React from 'react'
import PropTypes from 'prop-types'
const Link = ({ active, children, onClick }) => {
  if (active) {
    return <span>{children}</span>
  }
  return (
    <a
      href=""
      onClick={e => {
        e.preventDefault()
        onClick()
      }}
    >
      {children}
    </a>
  )
}
Link.propTypes = {
  active: PropTypes.bool.isRequired,
  children: PropTypes.node.isRequired,
  onClick: PropTypes.func.isRequired
}

export default Link
```

 © Imagina Formación

Mediante la siguiente clase podemos que To-dos se muestran.

components/Footer.js

```
import React from 'react'
import FilterLink from '../containers/FilterLink'

const Footer = () => (
  <p>
    Show:
    {' '}
    <FilterLink filter="SHOW_ALL">
      All
    </FilterLink>
    {', '}
    <FilterLink filter="SHOW_ACTIVE">
      Active
    </FilterLink>
    {', '}
    <FilterLink filter="SHOW_COMPLETED">
      Completed
    </FilterLink>
  </p>
)

export default Footer
```

 © Imagina Formación

Este es el componente principal que muestra todo lo que hemos creado anteriormente.

components/App.js

```
import React from 'react'
import Footer from './Footer'
import AddTodo from '../containers/AddTodo'
import VisibleTodoList from '../containers/VisibleTodoList'

const App = () => (
  <div>
    <AddTodo />
    <VisibleTodoList />
    <Footer />
  </div>
)

export default App
```

 © Imagina Formación

Usaremos la siguiente clase para filtrar entre los to-dos disponibles y mostrar unos u otros.

containers/VisibleTodoList.js

```
import { connect } from 'react-redux'
import { toggleTodo } from '../actions'
import TodoList from '../components/TodoList'
const getVisibleTodos = (todos, filter) => {
  switch (filter) {
    case 'SHOW_COMPLETED':
      return todos.filter(t => t.completed)
    case 'SHOW_ACTIVE':
      return todos.filter(t => !t.completed)
    case 'SHOW_ALL':
    default:
      return todos
  }
}
const mapStateToProps = state => {
  return {
    todos: getVisibleTodos(state.todos, state.visibilityFilter)
  }
}
const mapDispatchToProps = dispatch => {
  return {
    onTodoClick: id => {
      dispatch(toggleTodo(id))
    }
  }
}

const VisibleTodoList = connect(
  mapStateToProps,
  mapDispatchToProps
)(TodoList)

export default VisibleTodoList
```

 © Imagina Formación

Esta clase ayudará a la anterior obteniendo el actual filtro de visibilidad que hay seleccionado.

containers/FilterLink.js

```
import { connect } from 'react-redux'
import { setVisibilityFilter } from '../actions'
import Link from '../components/Link'

const mapStateToProps = (state, ownProps) => {
  return {
    active: ownProps.filter === state.visibilityFilter
  }
}

const mapDispatchToProps = (dispatch, ownProps) => {
  return {
    onClick: () => {
      dispatch(setVisibilityFilter(ownProps.filter))
    }
  }
}

const FilterLink = connect(
  mapStateToProps,
  mapDispatchToProps
)(Link)

export default FilterLink
```

Vamos a crear ahora la clase encargada de poder añadir nuevos to-dos a nuestra lista, tendrá un campo para escribir y un botón para añadir.

 © Imagina Formación

containers/AddTodo.js

```
import React from 'react'
import { connect } from 'react-redux'
import { addTodo } from '../actions'
let AddTodo = ({ dispatch }) => {
  let input
  return (
    <div>
      <form
        onSubmit={e => {
          e.preventDefault()
          if (!input.value.trim()) {
            return
          }
          dispatch(addTodo(input.value))
          input.value = ''
        }}
      >
        <input
          ref={node => {
            input = node
          }}
        />
        <button type="submit">
          Add Todo
        </button>
      </form>
    </div>
  )
}
AddTodo = connect()(AddTodo)
```

 © Imagina Formación

Ejecuta la aplicación y comprueba que redux funciona correctamente. ¿Cuál es la función de redux en esta aplicación?

 © Imagina Formación