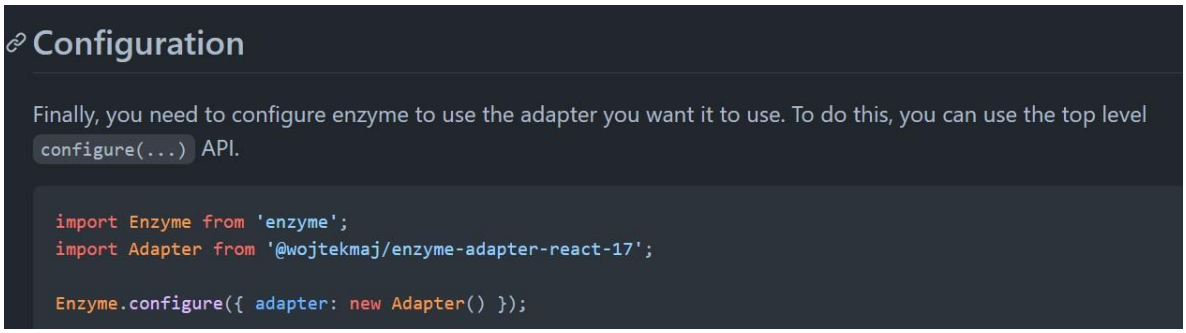
Pruebas unitarias o de integración

Repositorio: https://github.com/Garcia091/F4-firebase.git

1. Instalación: https://enzymejs.github.io/enzyme/

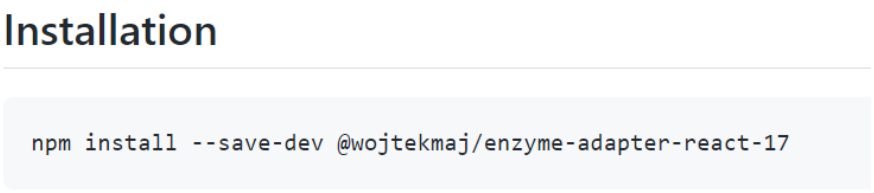
Si la versión de su React es superior a la 17 debe realizar la respectiva instalación y configuración con el siguiente enlace:



https://github.com/wojtekmaj/enzyme-adapter-react-17

configuración con npm:

https://www.npmjs.com/package/@wojtekmaj/enzyme-adapter-react-17



Ademas se debe reaalizar la instalacion de enzyme npm i --save-dev enzyme

De lo contrario seguir instalación oficial de Enzyme.

Pruebas unitarias o de integración

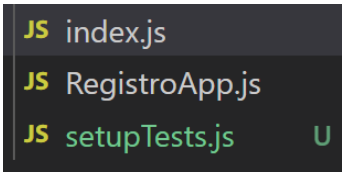
```
npm i --save-dev enzyme enzyme-adapter-react-16
```

```
import Enzyme from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';

Enzyme.configure({ adapter: new Adapter() });
```

2. Verificar la versión de react en el archivo

```
{} package.json > ...
  9     "firebase": "^9.0.2",
 10     "react": "^17.0.2",
```

1. Crear un archivo setupTests.js en la raíz de la carpeta src y agregar la configuración del adaptador
2. Instalar **npm install --save-dev enzyme-to-json** https://www.npmjs.com/package/enzyme-to-json
3. Realizar las configuraciones pertinentes de la dependencia de enzyme-tojson
4. Crear un archivo setupTests.js al interior de src



Versión 16

```
src > JS setupTests.js > ...
1   import Enzyme from 'enzyme';
2   import Adapter from 'enzyme-adapter-react-16';
3   import {createSerializer} from 'enzyme-to-json';
4
5   expect.addSnapshotSerializer(createSerializer({mode: 'deep'}));
6   Enzyme.configure({ adapter: new Adapter() });
```

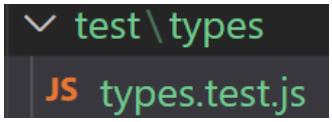
Pruebas unitarias o de integración

Version 17

```
src > JS setupTests.js > ...
1   import Enzyme from 'enzyme';
2   import Adapter from '@wojtekmaj/enzyme-adapter-react-17';
3   import {createSerializer} from 'enzyme-to-json';
4
5   Enzyme.configure({ adapter: new Adapter() });
6   expect.addSnapshotSerializer(createSerializer({mode: 'deep'}));
7
```


```
import Enzyme from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';
import {createSerializer} from 'enzyme-to-json';

Enzyme.configure({ adapter: new Adapter() });
expect.addSnapshotSerializer(createSerializer({mode: 'deep'}));
```


5. Ejecutar el comando **npm run test**
6. Crear la carpeta test y la carpeta types
7. Crear el archivo types.test.js



8. Verificar que no falle la instalación de `setupTests.js`
9. Para que la ejecución del archivo types debemos crear la estructura de una prueba

```
src > test > types > JS types.test.js > ...
1   import '@testing-library/jest-dom'
2
3   describe('Verificar types ', () => {
4       test('should ', () => {
5
6       })
7
8   })
```

10. Como la estructura es validad la prueba pasa

Pruebas unitarias o de integración

```
PASS src/test/types/types.test.js (9.854 s)
  Verificar types
    √ should  (4 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        12.892 s
Ran all test suites related to changed files.
```

Realizar la primera prueba:

Verificar que el objeto de types si tiene todo los elementos necesarios para la ejecución de la aplicación

Comparar el objeto de los types con el archivo types

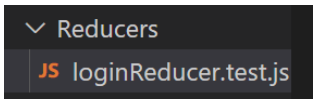
```
RUNS src/test/types/types.test.js

Test Suites: 0 of 1 total
Tests:       0 total
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        12.57 s
Ran all test suites related to changed files.
```

```jsx
import '@testing-library/jest-dom'
import { types } from '../../types/types'

describe('Verificar types ', () => {
    test('comparar objetos  ', () => {
        expect(types).toEqual({
            login: 'login',
            register: 'register',
            logout: 'Logout',

            taksAddNew: '[Taks] New taks',
            taksActive: '[Taks] Active taks',
            taksLoad: '[Taks] Load taks',
            taksUpdate: '[Taks] Update taks',
            taksDelete: '[Taks] Delete taks',
            taksClear: '[Taks] Clear taks',
            taksLogoutClean: '[Taks] Logout taks'
        }
        )
    })
})

```

Pruebas unitarias o de integración

Pruebas de un Reducer loginReducer.js

Crear una carpeta Reducer, un archivo con el nombre loginReducer.test.js



Verificar los datos que le llegan al reducer de Login

```
export const loginReducer = (state = {}, action) => {
    switch (action.type) {
        case types.login:

        return{
            id: action.payload.id,
            name: action.payload.displayname
        }
        case types.logout:
            return []
        default:
          return state;
    }
}
```

En este ejercicio podemos hacer 3 pruebas básicas.

1. Verificar los datos del usuario que inicia sesión

Pruebas unitarias o de integración

src > test > Reducers > JS loginReducer.test.js > ...

```
1   import '@testing-library/jest-dom'
2   import { loginReducer } from '../../reducers/loginReducer'
3   import { types } from '../../types/types'
4
5   describe('Pruebas en LoginReducer', () => {
6       test('debe de realizar el login', () => {
7
8           const initState = {};
9           const action = {
10              type: types.login,
11              payload: {
12                  id: 'abc',
13                  displayname: 'Fernando'
14              }
15          };
16          const state = loginReducer( initState, action );
17          expect( state ).toEqual({
18              id: 'abc',
19              name: 'Fernando'
20          })
21      })
22
23  })
24
```

Cerrar sesión :

```
test('Cerrar sesión - logout ', () => {
    const initState = {
        id: 'abc',
        name: 'Fernando'
    };

    const action = {
        type: types.logout,
    };

    const state = loginReducer( initState, action );
    expect( state ).toEqual([])
})
```

Pruebas unitarias o de integración

Enviar datos diferentes a las contempladas en las acciones para que se active el estado por default

```
test('State por default ', () => {
    const initState = {
        id: 'abc',
        name: 'Fernando'
    };

    const action = {
        type: types.Hola,
    };

    const state = loginReducer( initState, action );
    expect( state ).toEqual(initState)
})
```

Pruebas en acciones síncronas en actionLogin.js

```
export const loginSincrono = (id, displayname) => {

    return{
        type: types.login,
        payload: {
            id,
            displayname
        }
    }
}
```

Para hacer las pruebas correspondiente realizamos el siguiente procedimiento

1. Importar los types
2. Importar la función loginSíncrono desde el archivo de testing

Pruebas unitarias o de integración

src > test > actions > JS actionLogin.test.js > ...

```
1   import '@testing-library/jest-dom'
2   import { loginSincrono } from '../../actions/actionLogin';
3   import { types } from '../../types/types'
4
5   describe('Verificar acciones de Login', () => {
6
7       test('Validar login sincronico ', () => {
8           const id = 'ABC123';
9           const displayname = 'Fernando';
10
11          const loginAction = loginSincrono( id, displayname );
12
13          expect( loginAction ).toEqual({
14              type: types.login,
15              payload: {
16                  id,
17                  displayname
18              }
19          });
20
21      })
22  })
```

Crear el test para cerrar sesión

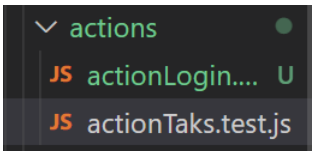
```
test('Cerrar sesión ', () => {
    const id = 'ABC123';
    const displayname = 'Fernando';

    const logoutAction = logout();

    expect( logoutAction ).toEqual({
        type: types.logout,
    });
})
```

Pruebas unitarias o de integración

Realizar una función asíncrona

Para realizar una dispatch debemos realizar unas buenas instalaciones y debemos configurar la información en firebase para no tener errores de permisos.

1. Crear la carpeta y el archivo del elemento a testear



2. Instalar redux-mock-store: https://www.npmjs.com/package/redux-mock-store

```
npm i redux-mock-store
```

3. `Configurar el store`

```
import configureStore from 'redux-mock-store';
import thunk from 'redux-thunk';

const middlewares = [thunk];
const mockStore = configureStore(middlewares);

const initState = {};

let store = mockStore(initState);
```

4. Crear la prueba y terminar la configuración

Pruebas unitarias o de integración

src > test > actions > JS actionTaks.test.js > ...

```
1   import configureStore from 'redux-mock-store';
2   import thunk from 'redux-thunk';
3
4   const middlewares = [thunk];
5   const mockStore = configureStore(middlewares);
6
7   const initState = {};
8
9   let store = mockStore(initState);
10
11  describe('Pruebas con las acciones Taks', () => {
12
13      beforeEach(()=> {
14          store = mockStore(initState);
15      })
16  })
17
```

1. Conocer en qué base de datos estamos enviando los datos

src > firebase > JS firebaseConfig.js > ...

```
1   import { initializeApp } from 'firebase/app';
2   import { GoogleAuthProvider } from  'firebase/auth';
3   import {getFirestore} from 'firebase/firestore'
4
5   const firebaseConfig = {
6     apiKey: "AIzaSyCjEJEW1Ml0UmhpW0HZP5TSs96AsgGkreo",
7     authDomain: "app-frontend-6.firebaseapp.com",
8     projectId: "app-frontend-6",
9     storageBucket: "app-frontend-6.appspot.com",
10    messagingSenderId: "66843513700",
11    appId: "1:66843513700:web:ac800485c5370d3e30a071"
12  };
13  console.log(process.env)
14
15  const app = initializeApp(firebaseConfig);
16  const google =  new GoogleAuthProvider();
17  const db = getFirestore(app)
18
19  export{
20      app,
21      google,
22      db
23  }
```

Pruebas unitarias o de integración

```
▼ Object i
    FAST_REFRESH: true
    NODE_ENV: "development"
    PUBLIC_URL: ""
    WDS_SOCKET_HOST: undefined
    WDS_SOCKET_PATH: undefined
    WDS_SOCKET_PORT: undefined
  ▶ [[Prototype]]: Object
```

Los datos de prueba deben quedar en una base de datos de prueba, por tal motivo debemos configurar el archivo

src > test > actions > JS actionTaks.test.js > ...

```
1  import configureStore from 'redux-mock-store';
2  import thunk from 'redux-thunk';
3  import { TaksNew } from '../../actions/actionTaks';
4
5  const middlewares = [thunk];
6  const mockStore = configureStore(middlewares);
7
8  const initState = {
9      login: {
10         id:'TESTING'
11     }
12 };
13
14 let store = mockStore(initState);;
15
16 describe('Pruebas con las acciones Taks', () => {
```

Pruebas unitarias o de integración

```
 beforeEach(()=> {
     store = mockStore(initState);
 })

test('Crear tareas ', async() => {
 await store.dispatch(TaksNew ({
         url:'123',
         nombre: '123',
         description: '123'
 }) );

 const actions = store.getActions();
})
```