■ # INSTALL NECESSARY DEPENDENCIES

 # ¿Do you want to avoid installing everything manually?

 # YES:
-Execute inside the project by console:

npm run installDeps

 # NO:
0-DEPENDENCIES
npm i

 # 1-PROTRACTOR
npm install - g protractor

 # 2-SELENIUM DRIVER
npm install selenium-standalone@latest -g

 # 3-ChromeDriver(if you want Chrome) and GeckoDriver(if you want Mozilla)
node node_modules/protractor/node_modules/webdriver-manager/bin/webdriver-manager update

 # 4-JASMINE(TO UPDATE JASMINE)
npm install --save-dev @types/jasmine

 # ¿How I know if everything is right in the installation?
In the location <Project>\node_modules\protractor\node_modules\webdriver-manager\selenium , there is three main files:
-chromedriver<Version>.exe
-chrome-response.xml
-geckodriver<Version>.exe
-gecko-response.xml
-selenium-server-standalone-<Version>.jar
-standalone-response.xml
-update-config.json

 # Be carefull
-Be carefull that the ChromeDriver binaries and the GeckoDriver binaries must match the version of Chrome or Firefox

■ # ¿HOW TO CREATE CASES?

1-General file: From this file we are going to call the other cases.

 # Template-General-Case.json:

```
{
```

```
 "emailsReport":
 {
  "sendReports":true,
       "server":"smtp.gmail.com",
       "port":587,
  "from":"email passwordEmail",
       "to":
       [
            "email",
     "email",
     "email"
       ]
 },
 "casesToRun":
 [
  {
    "location": "https://raw.githubusercontent.com/IsaacDavidMoreraVargas/WEB-TESTER-SELENIUM/main/SAVETEMPLATESFOLDERS/case1.json",
       "timesToRepeat":2
       },
       {
        "location":
"C:/Users/Alcatraz/Desktop/WEB-TESTER-SELENIUM/e2e/FolderToSaveTemplates/
case.json",
       "timesToRepeat":1
       }
 ]
}
```

As you can see:

- You can send emails or not with the results of the test(replace in "from" the "email" and "password" with the email that you want to use to send the results and replace in "to" the "email" where you want to send the results. I recommend to use gmail to send the emails and in the gmail security configurations you must allow the access to unsecure apps since I use powershell script to send the emails)
- In location, you set the url of the Template-Specific-Case.json to reach (you can charge configuration files online or inside you computer), be aware that if you want to read online files the data must be raw and being able to download locally since I use a powershell script to download and the script is not that powerfull against errors. You can charge as many Template-Specific-Case.json as you want, please just follow the template of Template-General-Case.json
- In timesToRepeat: you set up the times that you want to execute each Template-Specific-Case.json

2-Specific file: From this file we are going to set up the component's webpage to interact with.

 # Template-Specific-Case.json:

```
{
 "nameCase":"test case 2",
```

```
 "url": "https://www.calculator.net",
 "waitAngular":true,
 "maximunTimeScripts":30000,
 "components":
 [
  {
        "name":"button2",
        "type":"GETBYXPATH",
        "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/div/div[3]/
span[2]",
        "action":"DOCLICK",
        "sendData":" "
      },
  {
        "name":"button4",
        "type":"GETBYXPATH",
        "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/div/div[2]/
span[1]",
        "action":"DOCLICK",
        "sendData":" "
      },
  {
        "name":"result",
        "type":"GETBYXPATH",
        "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/div/div[5]/
span[4]",
        "action":"DOCLICK",
        "sendData":" "
      },
  {
        "name":"get result",
        "type":"GETBYXPATH",
        "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[1]/td/div/div[2]",
        "action":"DOGETTEXT",
        "sendData":" "
      },
      {
       "name":"DO PROMISE-SET-GET",
        "type":"DO PROMISE",
        "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[1]/td/div/div[2]",
        "action":"DO PROMISE",
        "sendData":" ",
        "fatherAction":
        {
          "name":"result",
          "type":"GETBYXPATH",
          "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/div/div[5]/
span[4]",
          "action":"DOCLICK",
```

```
        "sendData":" "
       },
       "childAction":
       {
        "name":"get result",
        "type":"GETBYXPATH",
        "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[1]/td/div/div[2]",
        "action":"DOGETTEXT",
        "sendData":" "
       }
      }
 ]
}
```

As you can see:

- "NameCase" is to differentiate the name of each test, in that way you can recognize easier everything in the moment to read the results
- "url" is to charge the url where the app is running and is the app that you want to test(the app must be running at least in localhost so selenium can interact with the app)
- ¿Do you want to wait for angular to charge or not? then setup "waitAngular" in true or false
- "maximunTimeScripts" is the maximun time that the case has to finish the chores, if for any reason the script can´t finish in the maximun time the case will be force to fail
- "components" in this space we will set up the components of the web page that we want to interact with, every component must have 5 properties: name, type, route, action and sendData
- "name" is to difference every component so you will be able to understand in the results what is happening
- "type" is the flag that you use to find the html tag in the web page (i'll explain later, I do recommend "GETBYXPATH")
- "route" is the route to find the component according to the type (i'll explain later)
- "action" is the action that you wanto to apply in the component (i'll give to you a list of actions, later)
- "sendData" in case you choose the "action" "DOSENDTEXT" this is the data that you want to send to the component (even if you use another "action" the property "sendData" must be defined to ensure the quality of the template)

LIST OF "TYPE"(FLAGS) TO FIND A HTML TAG

- **GETBYID** ,¿Does your or another html tag has an "ID"?, Do you want to find it by the "ID"?. This is your flag
- **GETBYNAME** ,¿Does your or another html tag has a "NAME ID"?, Do you want to find it by the "NAME ID"?. This is your flag
- **GETBYCLASSNAME** ,¿Does your or another html tag has a "CLASS NAME"?, Do you want to find it by the " CLASS NAME"?. This is your flag
- **GETBYTAGNAME** ,¿Do you want to find it by a html tag?. This is your flag
- **GETBYCSS** ,¿Does your or another html tag has a "CSS NAME"?, Do you want to find it by the "CSS NAME"?. This is your flag
- **GETBYXPATH** ,¿Do you want to find A or another html tag by the "XPATH"?. This is your flag

- **GETBYLINKTEXT** ,¿Does your or another html tag possess a "href"?, Do you want to find it by a string match present in the "LINK" of the "href"?. This is your flag
- **DO PROMISE** ,¿Do you want to do promise?("I'll explain later what is a promise"). This is your flag
- **GO BACK** ,¿Do you want to emulate pressing the GO BACK BUTTON of the browser?. This is your flag
- **GO FOWARD** ,¿Do you want to emulate pressing the GO FOWARD BUTTON of the browser?. This is your flag
- **REFRESH** ,¿Do you want to emulate pressing the REFRESH BUTTON of the browser?. This is your flag
- **CLOSE** ,¿Do you want to emulate CLOSE the browser?. I don't know why you want to do this, because the framework automatically closes the browser after finishing, but it is always nice to have more options. This is your flag

LIST OF "ACTION" TO INTERACT WITH THE COMPONENT

- **DOCLICK** ,¿Do you want to CLICK in the component?. This is your action.
- **DOCLEAR** ,¿Do you want to CLEAR the data in the component?. This is your action.
- **DOSENDTEXT** ,¿Do you want to SEND TEXT DATA to the component?. This is your action.
- **DOGETTEXT** ,¿Do you want to GET TEXT DATA from the component?. This is your action.
- **DOGETACTUALURL** ,¿Do you want to GET THE URL from the component?. This is your action. (I don't remember what this options does xD, but is allways nice to have it)
- **ISDISPLAYED** ,¿Do you want to know if the component is DISPLAYED in the screen?. This is your action.
- **ISENABLED** ,¿Do you want to know if the component is ENABLED to interact with it?. This is your action.
- **ISSELECTED** ,¿Do you want to know if the component was SELECTED?. This is your action. (I don't know how this is usefull xD, but is allways nice to have it)

EXAMPLES

GETBYID:

"type":"GETBYID",
"route":"mainFrame"

GETBYNAME:

"type":"GETBYNAME",
"route":"mainFrame"

GETBYCLASSNAME:

"type":"GETBYCLASSNAME",
"route":"mainFrame"

GETBYTAGNAME:

"type":"GETBYTAGNAME",
"route":"div"

GETBYCSS:

"type":"GETBYCSS",
"route":".mainFrame"

GETBYXPATH:

"type":"GETBYXPATH",
"route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/div/div[5]/span[4]"

GETBYLINKTEXT:
Example:

"type":"GETBYLINKTEXT",
"route":"online-calculator"

■ # ¿DO YOU WANT TO KNOW HOW TO DO A PROMISE?

A promise is an action Father-Child, it means that you expect to execute immediately a second action(child) after the first action finished(father), it is a chain of actions.

-Template of a component of promise

```
{
        "name":"DO PROMISE-SET-GET",
        "type":"DO PROMISE",
        "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[1]/td/div/div[2]",
        "action":"DO PROMISE",
        "sendData":" ",
        "fatherAction":
        {
          "name":"result",
          "type":"GETBYXPATH",
          "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/div/div[5]/
span[4]",
          "action":"DOCLICK",
          "sendData":" "
        },
        "childAction":
        {
```

```
      "name":"get result",
      "type":"GETBYXPATH",
      "route":"/html/body/div[3]/div/table/tbody/tr/td[1]/table/tbody/tr[1]/td/div/div[2]",
      "action":"DOGETTEXT",
      "sendData":" "
    }
  }
```

 # THINGS TO HAVE IN COUNT

- If you are behind a security proxy, download the drivers and selenium sucks
- The configuration to use Firefox instead of Chrome

 # Template protractor.headless.js version Firefox:

```
const { SpecReporter, StacktraceOption } = require('jasmine-spec-reporter');
exports.config = {
  allScriptsTimeout: 11000,
  specs: [
    './**/*.codeTester.ts'
  ],
  capabilities: {
    'moz:firefoxOptions':
          {
        args: ['--safe-mode','--verbose","--headless"],
        binary: 'C:/Program Files/Mozilla Firefox/firefox.exe'
     }
  },
  params:
  {
    menu:"default"
  },
  directConnect: true,
  baseUrl: 'http://localhost:4200/',
  framework: 'jasmine',
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  },
  onPrepare() {
    require('ts-node').register({
      project: require('path').join(__dirname, './tsconfig.json')
    });
    jasmine.getEnv().addReporter(new SpecReporter({
      spec: {
        displayStacktrace: StacktraceOption.PRETTY
```

```
    }
   }));
  }
};
```

 # Template protractor.notHeadless.js version Firefox:

```
const { SpecReporter, StacktraceOption } = require('jasmine-spec-reporter');
exports.config = {
  allScriptsTimeout: 11000,
  specs: [
    './**/*.codeTester.ts'
  ],
  capabilities: {
    'moz:firefoxOptions': {
        args: ['--safe-mode','--verbose'],
        binary: 'C:/Program Files/Mozilla Firefox/firefox.exe'
     }
  },
  params:
  {
    menu:"default"
  },
  directConnect: true,
  baseUrl: 'http://localhost:4200/',
  framework: 'jasmine',
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  },
  onPrepare() {
    require('ts-node').register({
      project: require('path').join(__dirname, './tsconfig.json')
    });
    jasmine.getEnv().addReporter(new SpecReporter({
      spec: {
        displayStacktrace: StacktraceOption.PRETTY
      }
    }));
  }
};
```

 # Template protractor.headless.js version Chrome:

```
const { SpecReporter, StacktraceOption } = require('jasmine-spec-reporter');
```

```
exports.config = {
  allScriptsTimeout: 11000,
  specs: [
    './**/*.codeTester.ts'
  ],
  capabilities: {
    "browserName": 'chrome',
    "chromeOptions":
    {
      args: ["--headless","--disable-gpu","--window-size=2048,1080" ],
      binary: "C:/Program Files/Google/Chrome/Application/chrome.exe"
    },
  },
  params:
  {
    menu:"default"
  },
  directConnect: true,
  baseUrl: 'http://localhost:4200/',
  framework: 'jasmine',
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  },
  onPrepare() {
    require('ts-node').register({
      project: require('path').join(__dirname, './tsconfig.json')
    });
    jasmine.getEnv().addReporter(new SpecReporter({
      spec: {
        displayStacktrace: StacktraceOption.PRETTY
      }
    }));
  }
};
```

 # Template protractor.notHeadless.js version Chrome:

```
const { SpecReporter, StacktraceOption } = require('jasmine-spec-reporter');
exports.config = {
  allScriptsTimeout: 11000,
  specs: [
    './**/*.codeTester.ts'
  ],
  capabilities: {
    "browserName": 'chrome',
    "chromeOptions":
```

```
    {
      args: ["--disable-gpu","--window-size=2048,1080" ],
      binary: "C:/Program Files/Google/Chrome/Application/chrome.exe"
    },
  },
  params:
  {
    menu:"default"
  },
  directConnect: true,
  baseUrl: 'http://localhost:4200/',
  framework: 'jasmine',
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  },
  onPrepare() {
    require('ts-node').register({
      project: require('path').join(__dirname, './tsconfig.json')
    });
    jasmine.getEnv().addReporter(new SpecReporter({
      spec: {
        displayStacktrace: StacktraceOption.PRETTY
      }
    }));
  }
};
```

- Be carefull with "binary" in the "capabilities" because there you set up the location of the browser to avoid potencials errors

■ # FINALLY, ¿HOW TO RUN THE PROGRAM?

- You can reach files online or inside the computer
- Open the terminal inside the project and execute:
- Dont forget to follow the sintax:

npm run testH -- --params.menu="ChargeCase <url file online or location of file inside the pc>"

or

npm run testNH -- --params.menu="ChargeCase <url file online or location of file inside the pc>"

testH: Means that is going to work showing the browser
testNH : Means that is going to work not showing the browser

Example:
npm run testH -- --params.menu="ChargeCase C:\Users\Alcatraz\Desktop\PROYECTOS-PROGRAMACION\WEB-TESTER-SELENIUM\e2e\FolderToSaveTemplates\tests.json"

 # npm run testNH or npm run testH To open the menu
npm run testNH
npm run testH