

Tutorial de instalación y configuración de Cypress.

INSTALAR CYPRESS:

IMPORTANTE: De momento **solo soporta Chrome** como navegador.

Para ello vamos a seguir los pasos que se indican en este enlace:

<https://docs.cypress.io/guides/getting-started/installing-cypress.html#npm-install>

1. Lo primero que haremos es crear una carpeta en la que se ubicará el proyecto, los casos de prueba, etc.
P.ej.: C:\Users\yourname\Desktop\Cypress
2. Instalamos Node.js: <https://nodejs.org/es/download/>
3. Una vez instalado lo abrimos y vamos hasta la dirección de la carpeta que creamos en el paso 1.
4. Una vez dentro, lanzamos el instalador de Cypress con este comando: `npm install cypress --save-dev`
5. Después, ejecutamos este otro comando: `npm init`. Esto creará el archivo `package.json`.

Con estos pasos ya habríamos completado la instalación de Cypress. Ahora veremos cómo abrirlo y empezar a crear tests.

ABRIR CYPRESS:

Se puede abrir de varias formas.

1. Ir al directorio donde se encuentra el .exe: `\node_modules\.bin\` y ejecutar este comando: `cypress open`
2. Desde la consola de Node.js cualquier directorio: `npx cypress open`
3. Ir al directorio `node_modules` desde la consola de node.js y ejecutar este comando: `cypress open`

Pero, en cualquier caso, la mejor forma es crear un script en el `package.json` para poder ejecutarlo directamente desde el IDE donde vamos a crear los tests.

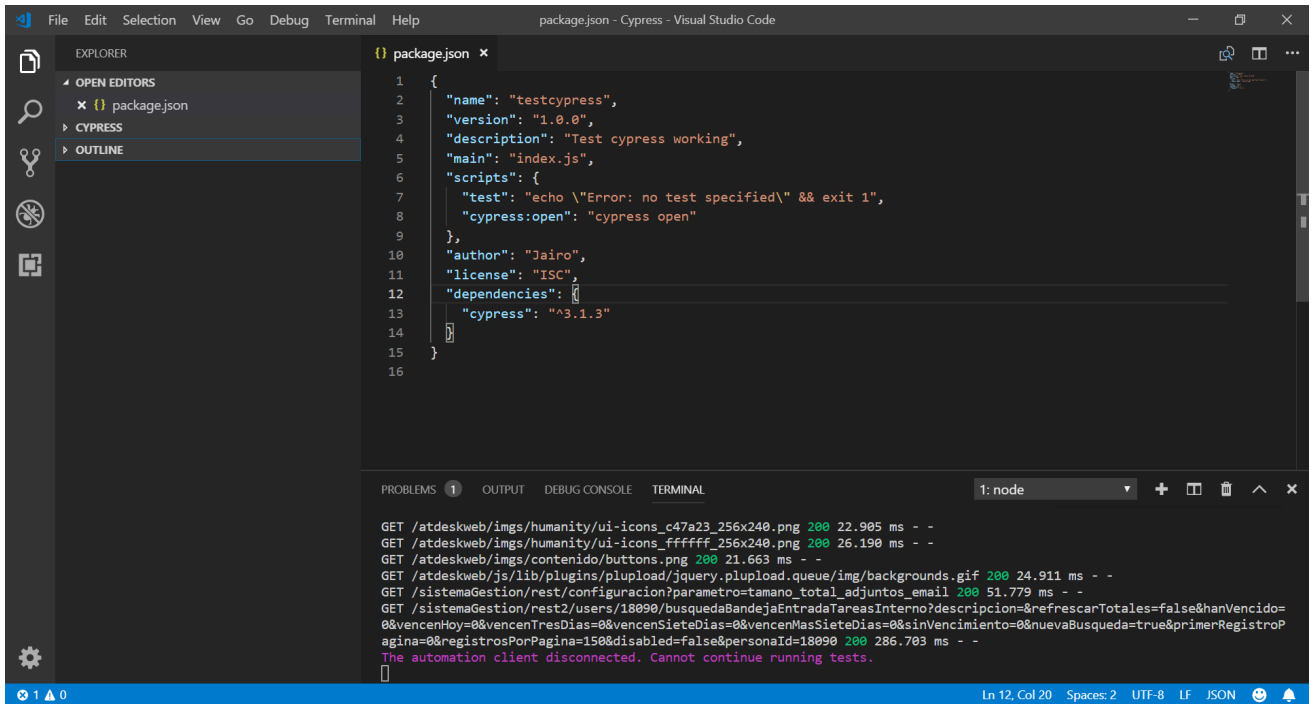
Para ello, debemos abrir el proyecto con un IDE. Yo estoy usando el que mencionan en el vídeo tutorial:

<https://code.visualstudio.com/download>

Lo instalamos, iniciamos el programa y, en el explorador vamos hasta la carpeta con el proyecto de Cypress:

1. File
2. Open folder
3. C:\Users\yourname\Desktop\Cypress

Luego abrimos el archivo `package.json` y añadimos la línea `"cypress:open": "cypress open"` dentro de `"scripts"`.

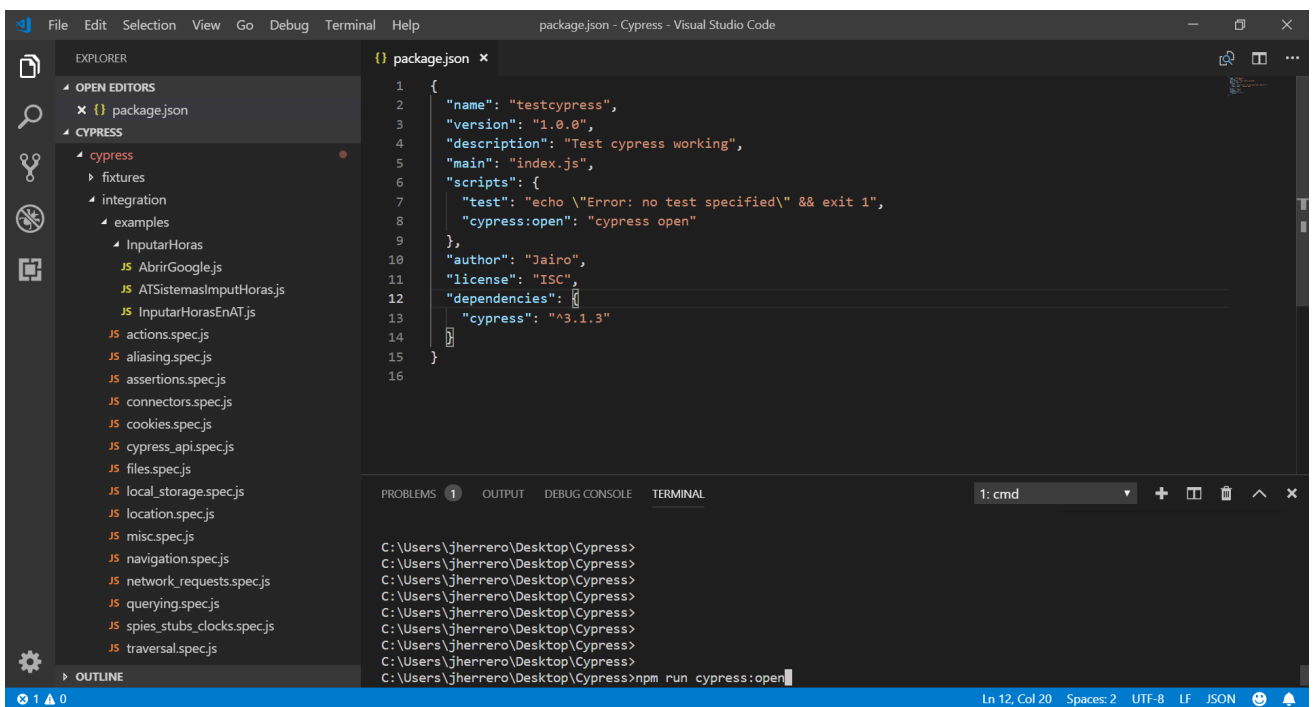


The screenshot shows the Visual Studio Code interface with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with 'package.json' selected. The main editor displays the contents of 'package.json', which is a JSON object with fields for name, version, description, main, scripts, author, license, and dependencies. The 'scripts' field includes 'test' and 'cypress:open'. The 'dependencies' field includes 'cypress'. The terminal window shows the output of a command, displaying a series of GET requests and their response times, followed by an error message: 'The automation client disconnected. Cannot continue running tests.'

```
1 {
2   "name": "testcypress",
3   "version": "1.0.0",
4   "description": "Test cypress working",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",
8     "cypress:open": "cypress open"
9   },
10  "author": "Jairo",
11  "license": "ISC",
12  "dependencies": {
13    "cypress": "^3.1.3"
14  }
15 }
16
```

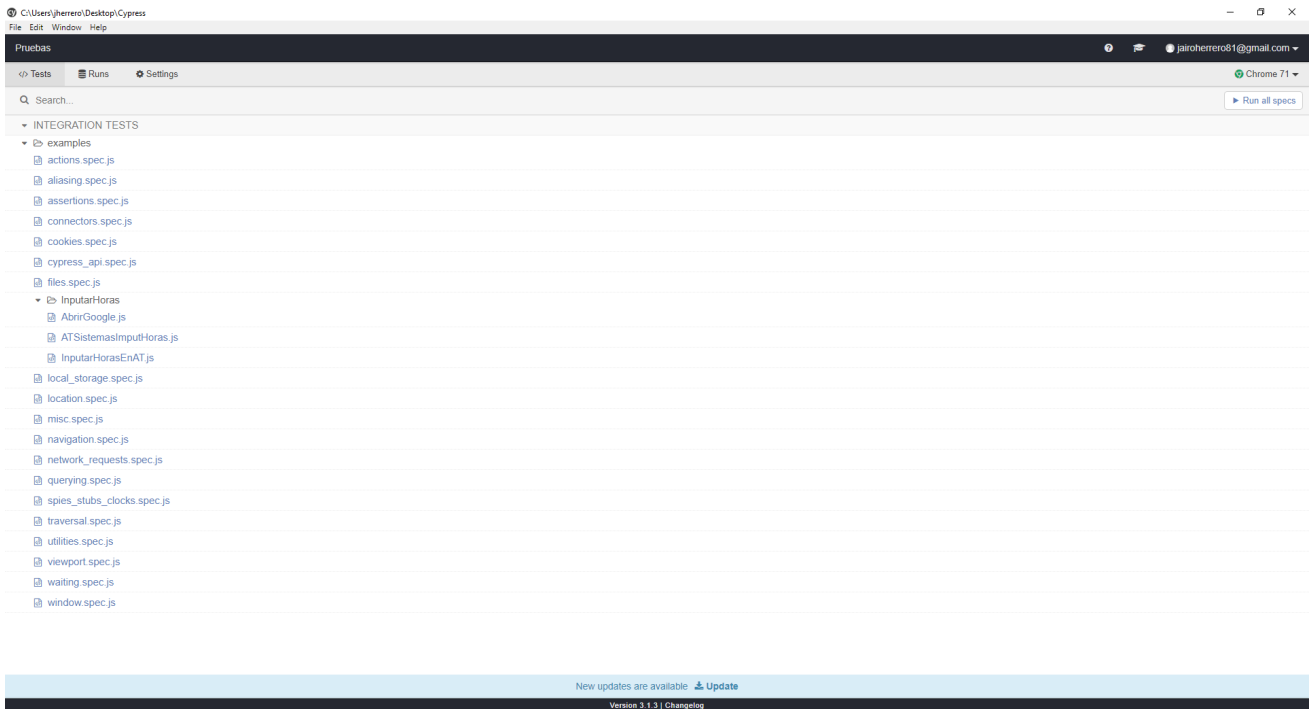
```
GET /atdeskweb/imgs/humanity/ui-icons_c47a23_256x240.png 200 22.905 ms - -
GET /atdeskweb/imgs/humanity/ui-icons_ffffff_256x240.png 200 26.190 ms - -
GET /atdeskweb/imgs/contenido/buttons.png 200 21.663 ms - -
GET /atdeskweb/js/lib/plugins/plupload/jquery.plupload.queue/img/backgrounds.gif 200 24.911 ms - -
GET /sistemaGestion/rest/configuracion?parametro=tamano_total_adjuntos_email 200 51.779 ms - -
GET /sistemaGestion/rest2/users/18090/busquedaBandejaEntradaTareasInterno?descripcion=&refrescarTotales=false&hanVencido=
0&vencenHoy=0&vencenTresDias=0&vencenSieteDias=0&vencenMasSieteDias=0&sinVencimiento=0&nuevaBusqueda=true&primerRegistroP
agina=0&registrosPorPagina=150&disabled=false&personaId=18090 200 286.703 ms - -
The automation client disconnected. Cannot continue running tests.
```

Luego, desde la consola, vamos al directorio de Cypress, donde se encuentran los archivos package.json y cypress.json y ejecutamos este comando: npm run cypress:open



Lo podemos lanzar desde la terminal del propio visual studio.

Cuando lo ejecutemos, se abrirá Cypress, algo parecido a esto:



Ya tenemos Cypress instalado y configurado con todo lo necesario para empezar a crear nuestros casos de prueba.

En el siguiente paso veremos cómo hacerlo.

CREANDO TESTS:

A la hora de crear casos de prueba es tan sencillo como empezar con un describe:

```
describe('nombre de las pruebas', () => {  
  
})
```

Dentro del describe vamos metiendo las pruebas, que se construyen con un it y deben tener esta estructura:

```
it('nombre del test', () => {  
    Aquí vamos poniendo los pasos necesarios.  
    P.ej. Que se abra una página, hacer click en algo, etc.  
})
```

Para los casos de prueba donde se necesita estar logado, es conveniente crear un método que mantenga la sesión abierta. Para ello se debe crear un método que se use siempre que se ejecuta un caso de prueba y así evitar repetir código ya usado.

Lo que haremos es lo siguiente:

Editamos el "commands.js" y añadimos esto:

```
Cypress.Commands.add('login', () => {  
    cy.visit('página de login')  
    cy.get('id o lo que sea de la caja de usuario').type('usuario')  
    cy.get('id o lo que sea de la caja de password').type('pass')  
    cy.get('id o lo que sea del boton entrar, login').click()  
  
})
```

Una vez creado este método, vamos de nuevo a los casos de prueba y en la parte del describe añadimos lo siguiente:

```
beforeEach(() => {  
  
    cy.login()  
  
})
```

Con esto ya no tenemos que preocuparnos más de que caduque la sesión de inicio, ni de cookies ni tokens, ni de volver a meter los datos de login.

EJECUTAR TESTS MEDIANTE CONSOLA:

Una de las cosas que nos permite hacer Cypress es ejecutar distintos comandos a través de la consola. En este enlace podemos encontrar una lista de ellos:

<https://docs.cypress.io/guides/command-line.html#cypress-run>

Así podremos, por ejemplo, lanzar los tests creados mediante un comando, sin visualizar el explorador, de modo que estos se ejecutarán mucho más rápido. Los casos así lanzados, generarán un informe posterior, en el que podremos ver los casos fallados, los casos pasados y se guardarán vídeos y capturas de pantalla de los errores ocurridos que pueden ser consultados en las carpetas "videos" y "screenshots" respectivamente.

Para lanzar todos los tests que tenemos creados en el proyecto, usaremos este comando:

`npx cypress run <folder>` por ejemplo si tenemos la carpeta del proyecto en `C:/cypress` desde este directorio ejecutaremos el comando de esta manera: `npx cypress run cypress/integration`

Y se ejecutarán todos los tests creados en la carpeta de integration.

EJECUTAR UN SOLO CASO DE PRUEBA O UNA SELECCIÓN:

Puede ser que solo queramos ejecutar unos casos específicos. En ese caso, podemos usar el siguiente comando:

`npx cypress run --spec` seguido de la ruta donde se encuentre el caso de prueba. Por ejemplo, imaginemos que, en el directorio anteriormente mencionado tenemos un caso de prueba llamado "AbrirGoogle.js". El comando quedaría así:

```
npx cypress run --spec cypress/integration/AbrirGoogle.js
```

Ahora imaginemos que queremos ejecutar un segundo caso de prueba, pero no el resto. En este caso, imaginemos que el caso que queremos añadir al paso anterior se llama "actions.spec.js". El comando sería el siguiente:

```
npx cypress run --spec cypress/integration/AbrirGoogle.js,cypress/integration/actions.spec.js
```

Una cosa importante: para poder ejecutar estas pruebas específicas debemos añadir en el `cypress.json` el directorio "integration". Siguiendo el ejemplo que hemos estado usando, el directorio integration se localizaría en "cypress/integration/". Si dentro de integration por ejemplo, tuviésemos otra carpeta, por ejemplo una que hemos llamado "pruebas" si añadimos a la línea anterior "/" con esto ya incluiría todos los tests de todas las carpetas que puedan existir dentro de la carpeta integration.

Entonces, abrimos el `cypress.json` y añadimos lo siguiente: "integrationFolder": "cypress/integration/".

EJECUTAR LOS TESTS POR COMANDO Y GRABARLO:

Cypress nos permite también guardar los resultados de las ejecuciones y mostrar en pantalla un informe detallado. Es decir, lo que nos muestra en consola, nos lo muestra en pantalla de modo que, desde la interfaz de Cypress, podemos ver un informe y acceder a los vídeos y screenshots guardados.

Podemos seguir los pasos de configuración en este enlace: <https://docs.cypress.io/guides/core-concepts/dashboard-service.html#Overview>

Por resumir un poco los pasos:

1. Lo primero que necesitamos es tener una cuenta de GitHub y logarnos en ella.
2. Abrimos Cypress.
3. Pulsamos en Runs.
4. Pulsamos en Setup Project to Record.
5. Le damos un nombre al proyecto.
6. Elegimos quién puede ver el proyecto.
7. Click Setup Project.
8. Entonces se mostrará una explicación de cómo se graban las pruebas.
9. Nos mostrará un key record que necesitaremos para ejecutar el comando.
10. Lo copiamos y vamos a la consola para lanzar el comando (también lo podemos encontrar en settings).
11. Introducimos en la consola lo que hemos copiado. Tiene que quedar algo así.

```
cypress run --record --key 70d5dd84-8ee5-4d67-93bf-7260e5382c4b
```

Y de esta manera, se ejecutarán todos los tests que tenemos en la carpeta de integration. Los resultados los podemos ver en el dashboard de Cypress, en la pestaña runs.

Descripción del HOW TO

Con este breve tutorial se pretende describir el proceso de intalación de Cypress y cómo crear los primeros casos de prueba.

Autor/es

Jairo Herrero Martín

Autor/es

Enlaces

Documentación de apoyo	https://docs.cypress.io/guides/getting-started/installing-cypress.html
	https://docs.cypress.io/guides/getting-started/writing-your-first-test.html
Repositorio de código	