des

**Configurar el ambiente para cypress:**

1. **Descargar NodeJS:** 
   1. Al descargarlo también vendrá NPM (Gestor de paquetes de Node) por default.
   2. Para comprobar si contamos ya con esto, o si queremos saber si se descargó e instaló correctamente, abrimos el CMD y ejecutamos:
      1. Node -v, este es para ver la versión de NodeJS.
      2. NPM -v, este para ver la versión del NPM.
2. **Crear una carpeta donde se alojará nuestro proyecto.**
3. **Npm init**: para más rápido **npm init -y:**
   1. Este es para levantar un proyecto de node. Hay que ejecutarlo en la carpeta creada con GitBash.
   2. Pedirá unos datos, los cuales se pueden quedar por default. Se puede configurar el test Command (test), git repository (Poner el repositorio de git), KeyWord (Poner las palabras clave).
   3. Al finalizar se creará un archivo .Json.
   4. Desde terminal:
      1. Darle a más gitbach.
4. **Npm install cypress --save-dev:**
   1. Ejecutar este comando desde la carpeta creada con GiTBash.
   2. Se ejecutar este comando para decirle a Cypress que se instale como dependencia en nuestro Package.json.
   3. Una vez instalado veremos la dependencia de Cypress en nuestro Package.json.
   4. También se verá reflejado el Node\_Modules que es donde se aloja cypress como tal.
5. **Abrir cypress, hay dos comandos**:
   1. ./node\_modules/.bin/cypress open
   2. npx cypress open
   3. Script del. json para crear comando para correr el cypress.

"test": "cypress run", abre por abajo las pruebas.

"cy:open": "cypress open", abre las pruebas en el navegador

**Npm test**, es lo que se le pasa a desarrollo para que corra las pruebas. Ahí se puede ver el estado de las pruebas.

Tambié: **Npm run cy:open.**

**Ojo con Install colors.**

**Códigos de NPM:**

-**Npm init**:para inicializar un proyecto.

-**Npm install cypress --save-dev**: para instalar cypress.

**Códigos de GitBash:**

-**Code .** : este es para abrir Visual studio code desde la carpeta.

-**./node\_modules/.bin/cypress open**: abrir Cypress.

- **npx cypress open**: abrir Cypress.

**Estructura recomendada:**

**Subir el Proyecto a git:**

-Crear archivo README.md

-Git init.

Crear archivo:

.gitignore : para ignorar las carpetas o archivos que no le daremos seguimientos.

Después pegamos:

**Ejemplo:** node\_modules.

-git add .

-git commit -m ‘Primer’

-git log: para traer el commit y nos da un id del commit.

-git config --global user.email “”

-git config --global user.name “”

-git reset --hard iddelcommit, regresa al commit del id.

**Clonar proyecto de git:**

-Code.

-Copy.

-ir a Carpeta y dar git batch.

-Git clone DARLE a pegar.

-ls

-cd rest-server-base/

-ls

-npm install

**Crear Branch:**

Git Branch PONERLEELNOMBRE.

Git checkout NOMBREDELBRANCH

**CAPTURAR CAMBIOS:**

Git clone

Moverse al main: Git checkout NOMBREDELBRANCH

Git pull

Git merge main

**Hace pull request:**

Git push origin NOMBREDELBRANCH

**TRABAJAR COLABORATIVAMENTE:**

Setting.

Collaboradores: Agregarlos.

Otra firma:

La persona entra a su git

Your organization.

NOTAS DE CYPRESS EN SÍ:

**Tipo de archivo**: .cy.js

**Codigo cypress:**

cy.visit('http://ec2-18-214-37-244.compute-1.amazonaws.com/login.php')

Para visitar un lugar

    cy.get('.mb-3 > .input-group > .form-control').type('PruebasAutomatizada')

Escribir en un elemento

cy.get('#signInBtn').click()

Clickear en un elemento

cy.get('.avatar > img').should('exist')

Verificar la existencia de un elemento

cy.get(':nth-child(1) > div > .nav-link > .nav-link-text').should('have.text', 'Dashboards')

Verificar un texto

**Para que se repita antes de cada prueba:**

beforeEach(() => {

    cy.visit('https://www.demoblaze.com/index.html')

  })

**COMMAND.jS en Support se utiliza para poder utilizar un caso de prueba en otro y no repetir cosas:**

Cypress.Commands.add('login', () => {

    cy.get('#login2').click()

    cy.get('#loginusername').type('jorge2022')

    cy.get('#loginpassword').type('12345678')

    cy.get('#logInModal > .modal-dialog > .modal-content > .modal-footer > .btn-primary').click()

})

Para pasarlo con parámetros:

Cypress.Commands.add('login', (usuario, contrasena) => {

    cy.get('#login2').click()

    cy.get('#loginusername').type('jorge2022')

    cy.get('#loginpassword').type('12345678')

    cy.get('#logInModal > .modal-dialog > .modal-content > .modal-footer > .btn-primary').click()

})

Se invoca de la siguiente manera:

cy.login('jorge2022', '12345678')

**Otra forma de agrupar los casos de pruebas:**

  context ('Titulos', () => {

    it('Titulo/ image', () => {

      cy.login('jorge2022', '12345678')

      cy.get('#nava > img').should('exist')

    })

**Forma para no pasar la cosa tan directa:**

**Fixtures (Promesas):**

describe('Home test', () => {

    beforeEach(() => {

        cy.visit('https://www.demoblaze.com/index.html')

        cy.fixture('example').then((examples) => {

            cy.login(examples.user, examples.Contrasena)

            })

    })

    it('verify image', () => {

        cy.get(':nth-child(1) > .thumbnail > .caption > .grrrr > b').should('have.text', 'About Us')

    })

})

Así se ve fixtures:

{

  "name": "Using fixtures to represent data",

  "email": "hello@cypress.io",

  "body": "Fixtures are a great way to mock data for responses to routes",

  "user": "Jorge2022",

  "Contrasena": "12345678"

}

Import y export de data.

**Clase 12/03/2022:**

COMPONENTES:

