📌Cypress. Plan de Pruebas automatizadas – Proyecto Web Colegio DVS

Este documento contiene los casos de prueba automatizadas diseñados para el sistema web del Colegio Secundario DVS (Dalmacio Vélez Sársfield). Y también el paso a paso para instalar y conectar Cypress.

—------------------------------------------------------

# **🚀 Guía paso a paso – Cypress + React (Proyecto DVS)**

## **0) Requisitos**

* Node.js 18+ y npm.
* Proyecto React funcionando (Vite, CRA o Next).
* URL local de desarrollo (ej.: http://localhost:5173 en Vite o http://localhost:3000 en CRA).

## **1) Instalar dependencias**

Bash

# dentro de tu proyecto React

npm install -D cypress start-server-and-test

start-server-and-test permite **levantar el front** y **correr Cypress** en una sola línea.

## **2) Inicializar Cypress**

npx cypress open

* Cypress creará la carpeta cypress/ y archivos base.
* Elegí **E2E Testing** cuando te lo pregunte.

## **3) Estructura recomendada de carpetas**

cypress/

e2e/ # specs de pruebas (nuevo estándar)

fixtures/ # datos mock (JSON)

support/ # comandos y bootstrap

Si usás el ZIP que te pasé, copiá el contenido del archivo dvs\_tests.spec.js adentro de cypress/e2e/ y renombralo a dvs\_tests.cy.js (formato moderno).

## **4) Configurar cypress.config.js**

Cypress 10+ usa este archivo en la raíz. Ajustá el **baseUrl** según tu dev server (Vite/CRA/Next).

Js

// cypress.config.js

import { defineConfig } from 'cypress'

export default defineConfig({

e2e: {

baseUrl: 'http://localhost:5173', // Vite. Usa 3000 si CRA.

specPattern: 'cypress/e2e/\*\*/\*.cy.{js,jsx,ts,tsx}',

supportFile: 'cypress/support/e2e.js',

video: false,

viewportWidth: 1280,

viewportHeight: 800,

env: {

apiUrl: 'http://localhost:3000/api' // si tenés backend local

},

setupNodeEvents(on, config) {

// hooks de node si los necesitás

},

},

})

## **5) Scripts en package.json**

Agregá estos scripts (ajustá dev y la URL según tu proyecto):

Json

{

"scripts": {

"dev": "vite", // o "react-scripts start" si CRA

"cy:open": "cypress open",

"cy:run": "cypress run",

"test:e2e": "start-server-and-test dev http://localhost:5173 cy:run",

"test:e2e:ui": "start-server-and-test dev http://localhost:5173 cy:open"

}

}

## **6) Selectores estables (data-testid)**

Asegurate de poner **atributos estables** en el HTML/JSX para que Cypress no dependa de clases o textos que cambian.

Ejemplos en tu React:

Jsx

<div data-testid="widget-clima">...</div>

<section data-testid="proyectos-proceso">...</section>

<div data-testid="mapa-zonal">...</div>

Y en Contacto:

Jsx

<input name="nombre" />

<input name="email" />

<textarea name="mensaje" />

<button type="submit">Enviar</button>

## **7) Crear los tests E2E**

Copiá tus casos (del ZIP) a cypress/e2e/dvs\_tests.cy.js.  
 Ejemplo (resumen):

Js

describe('Home', () => {

it('muestra el título del colegio', () => {

cy.visit('/')

cy.contains('Colegio Secundario DVS').should('be.visible')

})

it('muestra el widget de clima', () => {

cy.get('[data-testid="widget-clima"]').should('be.visible')

})

})

describe('Contacto', () => {

it('envía el formulario con datos válidos', () => {

cy.visit('/contacto')

cy.get('input[name="nombre"]').type('Juan Pérez')

cy.get('input[name="email"]').type('juan@test.com')

cy.get('textarea[name="mensaje"]').type('Consulta de prueba')

cy.get('button[type="submit"]').click()

cy.contains('Gracias por tu mensaje').should('be.visible')

})

it('valida errores al enviar vacío', () => {

cy.visit('/contacto')

cy.get('button[type="submit"]').click()

cy.contains('El campo nombre es obligatorio').should('be.visible')

})

})

## **8) Mock de APIs (clima, mapas, laboratorio)**

Para evitar fallas por red/datos en tus tests, **mockeá** respuestas.

1. Creá fixtures:

Bash

cypress/fixtures/weather.json

cypress/fixtures/muestreos.json

cypress/fixtures/weather.json

{

"timestamp": "2025-08-31T12:00:00Z",

"temperatura": 22.4,

"viento": { "velocidad": 12, "direccion": "NE" },

"lluvia": { "mm": 0.0, "ultimaHora": 0.0 }

}

1. Interceptá en el test o en cypress/support/e2e.js:

// cypress/support/e2e.js

beforeEach(() => {

cy.intercept('GET', '\*\*/api/meteo/actual', { fixture: 'weather.json' }).as('meteo')

cy.intercept('GET', '\*\*/api/muestreos\*', { fixture: 'muestreos.json' }).as('muestreos')

})

Ajustá las URLs a tus endpoints reales. Luego en tus tests podés esperar:

cy.visit('/meteorologia')

cy.wait('@meteo')

cy.get('[data-testid="widget-clima"]').should('be.visible')

## **9) Datos variables (años de Comisión)**

Si el filtro de años viene del backend, añadí otro fixture:

// cypress/fixtures/comision-gestiones.json

{

"2023": { "integrantes": ["..."], "logros": ["..."] },

"2024": { "integrantes": ["..."], "logros": ["..."] }

}

Y el intercept:

cy.intercept('GET', '\*\*/api/comision/gestiones', { fixture: 'comision-gestiones.json' }).as('gestiones')

## **10) Correr los tests**

* **Modo interactivo (útil para debug):**

Bash

npm run test:e2e:ui

* **Headless (para CI o rápida verificación):**

npm run test:e2e

## **11) Consejos anti-flaky**

* Usá data-testid y cy.findByTestId (si añadís Testing Library para Cypress).
* Evitá cy.wait(1000). Preferí cy.wait('@alias') o asserts que confirmen estado listo.
* Mockeá todas las integraciones externas (clima, mapas, laboratorio).
* Configurá **timeouts** si algunas cargas tardan (ej.: mapas).

## **12) Cobertura sugerida (qué automatizar primero)**

* **Críticos**: Contacto (envío/errores), Comisión (filtro por año), Meteorología (render + datos), Home (widgets clave).
* **Secundarios**: Cartelera (paginación/orden), Institucional (renders básicos).

## **13) (Opcional) Unit/Component Tests**

Para componentes React aislados:

npm i -D vitest @testing-library/react @testing-library/jest-dom

Usá Vitest + RTL para lógica/validaciones y **Cypress E2E** para flujos completos. (Podés también usar **Cypress Component Testing** si preferís unificar herramientas).

## **14) (Opcional) Accesibilidad rápida**

Bash

npm i -D cypress-axe axe-core

En cypress/support/[e2e.js](http://e2e.js):

Js

import 'cypress-axe'

Cypress.Commands.add('checkA11yPage', () => {

cy.injectAxe()

cy.checkA11y()

})

Y en un test:

Js

cy.visit('/')

cy.checkA11yPage()

## **15) CI con GitHub Actions (mínimo viable)**

.github/workflows/e2e.yml

name: E2E

on: [push, pull\_request]

jobs:

cypress-run:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- uses: actions/setup-node@v4

with:

node-version: 20

- run: npm ci

- run: npm run dev &

- run: npx wait-on http://localhost:5173

- run: npx cypress run

Cambiá el puerto si usás CRA (3000).

## **16) Trazabilidad: Manual ↔ Automatizado**

Mantené un mapeo simple (en tu Excel y en comentarios de specs):

* CT-011 (Contacto / envío válido) → Contacto › envía el formulario con datos válidos.
* CT-012 (Contacto / errores) → Contacto › valida errores al enviar vacío.
* CT-021 (Comisión / filtro año) → Comisión › filtra por año.
* CT-030/031/032 (Meteo/mapa/calendario) → Meteorología specs.

### **✅ Listo**