# **¿Qué es Selenium?**

Es una **SUIT de herramientas** para la automatización de navegadores Web.

El objetivo de Selenium NO fue para el Testing ni para el Web Scraping (aunque se puede usar para eso), por lo tanto, no es el más óptimo para estas actividades.

**Protocolo**: WebDriver, herramienta que se conecta a un API.

Selenium WebDriver es la herramienta que utilizaremos en el curso.

* Selenium **NO** es un Software, **ES** una **SUIT de Softwares**.
* DDT: **D**ata **D**rive **T**esting: Ingresar datos para que realice varias pruebas (sin intervención humana).



## **Historia**

Todo comienza en 2004, **Jason Huggins** buscaba automatizar pruebas manuales, creando así una aplicación llamada “JavaScriptTestRunner” que posteriormente fue llamada **Selenium Core**.

* **Paul Hammant** vio el demo y busco una solución a la “Same Origin Policy” y se creó **Selenium RC**.

**Más fácil y mejor**

* **Shinya Katasani** en Japón envolvió el código de Selenium convirtiéndolo en un plugins para Firefox.
* **Selenium IDE** es capaz de grabar, repetir, importar y exportar automatización.

**Un cliente para todos**

* **Simon Stewart** trabajo en varias herramientas llamadas **WebDriver**. Reemplazando JS por un cliente para cada navegador y una API de alto nivel.
* **Selenium RC** se fusiono a este proyecto, dando lugar a **Selenium WebDriver**.

## **Origen del nombre**

En su momento, la compañía de desarrollo que creo Selenium originalmente, tenían competidores, uno de ellos era Mercurial. Un dato curioso es que cuando una persona padece de mercurio debe consumir suplementos a base de selenio para poderse curar, entonces el equipo de desarrollo considero que era una buena idea utilizar el nombre de Selenium porque ellos ofrecían una “cura” para lo que hacían u ofrecían sus competidores. Esto era muy cierto ya que Selenium era un producto con capacidades superiores a lo que hacia la competencia.

## **Selenium IDE**

* No es un lenguaje, sólo funciona con clics.
* Ofrece reportes de las pruebas.

## **Pros**

* Excelente para iniciar en el mundo del testing.
* No requiere saber programar.
* Exporta scripts para Selenium RC y Selenium WebDriver.
* Genera reportes.

## **Contras**

* Disponible para Google Chrome y Firefox.
* No soporta DDT. No permite colocar datos para múltiples pruebas.

## **Selenium RC**

### **Pros**

* Soporte para.
  + Varias plataformas, navegadores y lenguajes.
  + Operaciones lógicas y condicionales.
  + DDT.
* Posee una API madura.

### **Contras**

* Complejo de instalar.
* Necesita de un servidor corriendo.
* Comandos redundantes en una API.
* Navegación no tan realista.

## **Selenium Web Driven**

### **Pros**

* Soporte para múltiples lenguajes.
* Fácil de instalar.
* Comunicación directa con el navegador.
* Interacción más realista.

### **Contra**

* No soporta nuevos navegadores tan rápido.
* No genera reportes o resultados de pruebas.
* Requiere de saber programar.

## **Selenium Grid**

### **Características**

* Se utiliza junto a Selenium RC.
* Permite correr pruebas en paralelo.
* Conveniente para ahorrar tiempo.

# **Otras herramientas de testing y automatización**

## [**Puppeteer**](https://pptr.dev/)

### **PROS**

* Soporte por parte de Google, te brinda datos del Performance Analysis de Chrome y un mayor control de este navegador. No requiere archivos externos como lo hace Selenium con WebDriver.

### **CONTRAS**

* Solo funciona para Google Chrome con JavaScript, tiene una comunidad pequeña así que el apoyo será poco.

## [**Cypress.io**](https://www.cypress.io/)

### **PROS**

* Tiene una comunidad emergente y va creciendo a pasos acelerados, tiene muy buena documentación para implementar Cypress en los proyectos. Es muy ágil en pruebas E2E, está orientado a desarrolladores y tiene un excelente manejo del asincronismo, logrando que las esperas sean dinámicas y también se puedan manejar fácilmente.

### **CONTRAS**

* Solo funciona en Google Chrome con JavaScript, se pueden realizar pruebas en paralelo únicamente en la versión de pago.

# **Configurar entorno de trabajo**

1. Descargamos Python desde el [sitio oficial](https://www.python.org/downloads/).
2. Se necesita tener python 3.6 en adelante.
3. Instalamos Selenium con el comando **pip3 install selenium** en el command prompt (admin) de windows.
4. Instalamos la librería pyunitreport que nos permitirá obtener reportes en HTML, el comando es **pip3 install pyunitreport**

# **Unittest (PyTest)**

* **Text Fixture**: Es todo lo que va ocurrir **antes** y **después** de la automatización.
* **Text Case**: Unidad de código que le indicaremos a Selenium que es lo que queremos que haga.
* **Test Suite**: Colección de **Tests Cases** en un sólo archivo para que podamos ejecutar todos de forma secuencial.
* **Test Runner**: Orquestador de la ejecución. Es decir, es la parte encargada de que se va a ejecutar, en que orden, como y los resultados que nos arrojará.
* **Test Report**: Resumen de resultados.

# **Encontrar elementos con find\_element**

En ocasiones algunos sitios pueden tener bloqueos regionales o no estar disponibles por la alta cantidad de solicitudes que llegan a tener.

Si el sitio de práctica no abre, puedes intentar ingresando a [OneStepCheckout Responsive Demo](http://demo.onestepcheckout.com/).