TEXT CLASS REVIEW

TEMAS A TRATAR EN LA CUE

- Pruebas End To End: características, ventajas y limitaciones.
- Setup de herramientas con Vue-CLI para pruebas End-to-End.
- Herramientas para el testing End-to-End. Cypress, Nightwatch. Ventajas y limitaciones.

PRUEBAS END-TO-END: CARACTERÍSTICAS, VENTAJAS Y LIMITACIONES

Si bien las pruebas unitarias, incluidas las de componentes, son una gran herramienta a la hora de desarrollar, tienen utilidad limitada, pues no permiten comprobar el uso completo de la aplicación, antes del paso a producción. Aquí es donde cobran relevancia los test `end to end`, también conocidos como de punta a punta, o de extremo a extremo. Esta es una metodología que usa un navegador automatizado, simulando un escenario real, para comprobar que una aplicación funciona correctamente de principio a fin, viéndola desde la perspectiva del usuario.

Las pruebas de punta a punta, sirven para validar todas las capas de una aplicación. Esto no solo incluye el código, sino también todos los servicios externos, y la infraestructura asociada, que representan el entorno real en el que estarán inmersos los usuarios. Permiten ahorrar tiempo, y garantizar que la información se transmita correctamente entre los diversos componentes y sistemas involucrados. Una de sus fortalezas, es su capacidad para probar la aplicación en diferentes navegadores.

Sin embargo, también presentan algunas complicaciones. Uno de los principales problemas con las pruebas de extremo a extremo, es que ejecutar toda la suite lleva mucho tiempo; otro punto a tener en cuenta, es que probar la aplicación entre distintos navegadores requiere un tiempo adicional; otro inconveniente es su fragilidad, ya que una variación pequeña, como el cambio de una clase, puede hacer que el test falle; y una última complicación, surge cuando el test creado pasa, pero al volver a ejecutarlo falla, y viceversa, incluso cuando no se han hecho cambios en el código. Esto es conocido como: `“flaky”` test.

Cuando las pruebas `end to end` se ejecutan en líneas de implementación/integración continua, a menudo se ejecutan de tal manera que no se abre ningún navegador para que el desarrollador lo vea. Los `frameworks end to end` modernos, brindan un soporte que permite ver capturas instantáneas y/o videos de las aplicaciones, durante varias etapas de la prueba, para proporcionar una idea de por qué ocurren los errores.

A pesar de que los test `"end to end"` son bastante prácticos, éstos no son suficientes por sí solos. Debido a ello, su uso combinado con test unitarios, es una buena aproximación para verificar que nuestro código funcione correctamente.

SETUP DE HERRAMIENTAS CON VUE-CLI PARA PRUEBAS END-TO-END

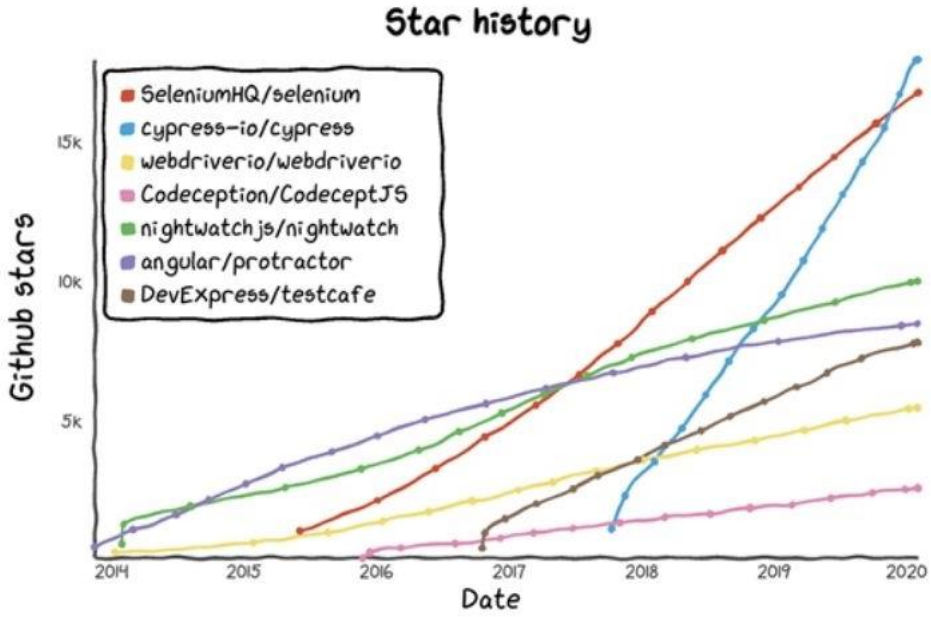
Antes de comenzar a trabajar en el desarrollo dirigido por pruebas, es necesario revisar las herramientas que usaremos para lograrlo.

- Hardware: Preprocesador Intel I3 o similar mínimo, RAM 8GB mínimo.
- Sistemas operativos: macOS 10.9 y superior (solo 64 bits); Linux: Ubuntu 12.04 y superior, Fedora 21, y Debian 8 (solo 64 bits); Windows 7 y superior.
- Arquitectura por defecto de la aplicación: `Vue CLI`.
- Ejecución de test mediante línea de comandos (CLI): `NodeJS`.
- `Framework` de pruebas `end to end`: `Cypress`.
- Biblioteca para afirmaciones: `Chai`, `Sinon` y `jQuery`.
- Biblioteca para `stubs` y `mocks`: `Chai` y `Sinon`.

Cuando comenzamos un proyecto con `Vue CLI`, usando la línea de comandos, existe la posibilidad de seleccionar manualmente las herramientas que vamos a utilizar. La opción para realizar pruebas `end to end`, viene con tres alternativas: `Cypress`, `Nightwatch` y `WebdriverIO`. A través de `Vue CLI` también se instala, de manera automática, `Vue Test Utils`, que es el paquete oficial para testeo de `Vue`.

HERRAMIENTAS PARA EL TESTING END-TO-END: CYPRES, NIGHTWATCH. VENTAJAS Y LIMITACIONES

El mercado de `frameworks` para realizar pruebas de extremo a extremo, es bastante amplio. Sin embargo, acá nos enfocaremos en las dos herramientas más populares según `Github`.



- **Cypress:** a pesar del poco tiempo transcurrido desde su lanzamiento, en el año 2017, se ha convertido en una de las herramientas más populares de testing. Si la aplicación web se basa en marcos `JavaScript` modernos, como: `React`, `Angular`, `Vue`, entre otros, es la mejor opción. Su mayor ventaja, es ser un `framework` todo en uno. Con una sola dependencia, que se agrega a través de `NPM`, podemos instalarla y tener configurada la librería de aserciones con `spies`, `mocks`, `stubs`, y sin depender de `Selenium`, ya que está hecho con `Electron`. Permite escribir pruebas de manera simple, y las ejecuta dentro del navegador, controlando el proceso a través del back end, en `NodeJS`. A diferencia de otras herramientas, `Cypress` dispone de una interfaz gráfica, que permite ver de forma interactiva cada uno de los pasos y las acciones ejecutadas durante la prueba, el estado de la aplicación, la duración

de la prueba, los test fallidos o pasados. La ejecución de las pruebas comienza tan rápido como se cargue la aplicación. Además, espera automáticamente a los elementos de funcionamiento asíncrono (DOM sin cargar por completo, `framework` sin terminar de arrancar, petición `AJAX` sin completar, entre otros), lo que minimiza los falsos positivos en las pruebas fallidas, y cuenta con buena documentación, que permite darse cuenta de que fue diseñado para ser fácil de usar. Y, por último, pero no menos importante, `Cypress` se puede extender con plugins, que vuelven esta herramienta aún más potente.

Pero, como toda herramienta, también presenta algunas limitaciones. Una de ellas, es que está restringida solo a navegadores basados en `Chromium` y `Firefox`, para los cuales se agregó soporte recientemente. Algunas funciones aún no están integradas, y requieren soluciones y bibliotecas externas, como por ejemplo, la carga de archivos..

- **NightwatchJS:** es un `framework` para implementar pruebas automáticas de aplicaciones y sitios web, escrito en `Node.js`, y que nos proporciona una `API` para el estándar de `W3C WebDriver API` (o `Selenium WebDriver`), por lo que tiene acceso a todos los navegadores; éste se ejecuta en un proceso aislado, y que se puede desactivar si `Selenium` se aplica desde otra máquina. Tiene una sintaxis sencilla, que nos da la posibilidad de escribir test con rapidez, permitiendo el uso de selectores de tipo `css` y `xpath`; además, tiene unas librerías de comandos y aserciones propias muy flexibles, las cuales pueden ser fácilmente extendidas por nosotros. Dispone de un runner que, entre otras funcionalidades, permite lanzar los test por grupos, por etiquetas, y en uno o varios navegadores en paralelo, pero que también se puede ejecutar desde `task runners`, como `Grunt` o `Gulp`. Entrega los informes en formato `JUnit XML`, por lo que pueden incluirse dentro de servidores de integración continua, como `Jenkins` y `Teamcity`. A parte de todo lo anterior, su curva de aprendizaje es muy rápida, y gracias a su `API` podemos cubrir la gran mayoría de casuísticas o necesidades que nos surjan.

  Sin embargo, la documentación no incluye toda la información necesaria, proporciona una funcionalidad limitada, y su rendimiento es lento.