Reporte de Experiencia

Fecha:	3 de noviembre de 2024
Autores:	Calletana López Juan Tapia Raúl Ramos Sergio Andrés Gelves
Empresa:	The Software Design Company (TSDC)

Tabla de Contenido

1 Introducción

Este informe busca documentar los resultados de la pruebas iniciales exploratorias automatizadas, con el propósito de entender las diferencias clave entre los enfoques "monkey" y "ripper", identificando puntos fuertes y limitaciones de cada herramienta en términos de cobertura de pruebas, manejo de errores, y capacidad para descubrir problemas en la interfaz y en la experiencia del usuario.:

Herramientas:

- **Monkey-cypress**: Utilizada para generar eventos de usuario aleatorios en la interfaz gráfica de Ghost, simulando acciones como clics, entradas de texto, y navegación.
- **RIPuppet**: Herramienta de exploración de interfaces que permite la navegación automatizada en el sistema a diferentes niveles de profundidad, rellenando campos y buscando interacciones significativas.

2 Configuración y Preparación de las Pruebas

2.1 Configuración de Cypress:

Cypress es un proyecto de código abierto alojado en un repositorio público de GitHub accesible desde el siguiente enlace: https://github.com/cypress-io/cypress. Se recomienda instalar cypress de manera global debido a su tamaño. Para crear las pruebas se debe seguir los siguientes pasos

Instalar Node.js. Cypress se ejecuta en Node.js, se puede descargar desde https://nodejs.org/

Crea una carpeta para el proyecto. Abrir una terminal y crea una nueva carpeta donde se instalara Cypress.

- mkdir mi-proyecto-cypress
- cd mi-proyecto-cypress

Inicializa el proyecto con npm init: Esto crea un archivo package.json para gestionar dependencias.

- npm init -y

Instalar Cypress usando npm: Ejecutar el siguiente comando en la terminal para instalar Cypress como dependencia de desarrollo en caso de requerir versiones especificas para cada proyecto, de lo contrario se instala de manera global.

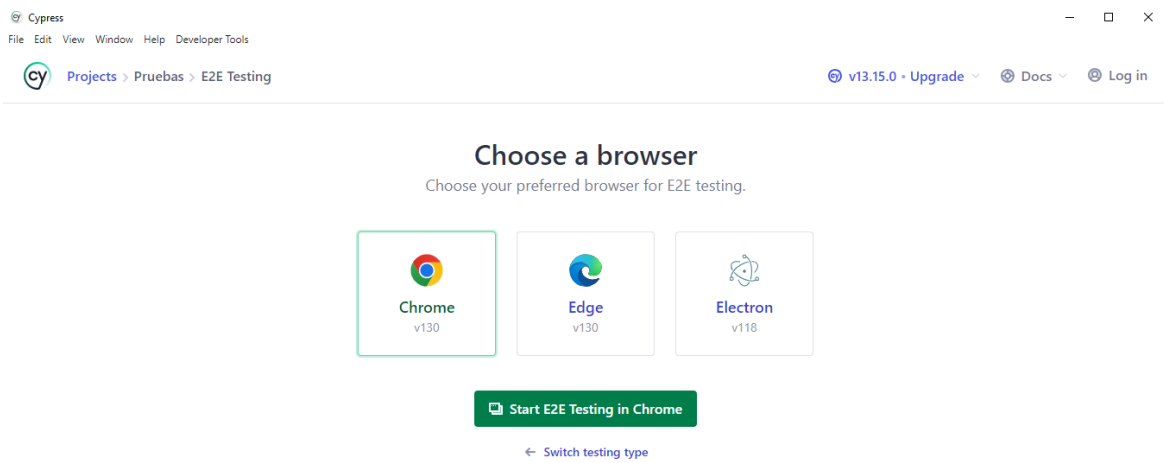
- npm install cypress --save-dev
- npm install -g cypress // instalación de manera global

Abrir Cypress: Una vez instalado, abrir Cypress:

- npx cypress open

Esto abrirá la interfaz de Cypress ahí se puede agregar un nuevo proyecto, agregue la carpeta donde se inicio el proyecto y luego de clic en el botón E2E Testing, esto creará una estructura de carpetas (cypress/) donde se podrá almacenar las pruebas,

De clic en el botón del navegador y luego clic en el botón Start E2E.



Luego de clic en el botón New Spec, esto permite crear la primera prueba y adicionara una carpeta llamada e2e donde se guardara la prueba, al final habrá un árbol de directorios similar al siguiente:

```
pruebas
├── cypress
│   ├── fixtures          # Datos de prueba estáticos en formato JSON
│   │   └── example.json
│   ├── e2e               # Carpeta para almacenar los archivos de pruebas
│   │   └── example.cy.js    # Archivo de ejemplo de prueba E2E
│   ├── support           # Funciones de soporte y configuraciones globales
│   │   ├── commands.js      # Comandos personalizados
│   │   └── e2e.js           # Configuración que se carga antes de cada prueba
│   └── downloads             # Carpeta opcional donde se guardan archivos descargados en
pruebas
├── node_modules          # Módulos de npm instalados (si Cypress no es global)
├── .gitignore            # Ignora archivos y carpetas de Git, como node_modules
```

```
├── cypress.config.js       # Archivo de configuración principal de Cypress
├── package.json            # Archivo de configuración de npm para dependencias y scripts
└── package-lock.json       # Archivo de bloqueo de versiones de dependencias
```

Por ultimo desde un editor de codigo se puede crear o modificar pruebas, el nombre de los archivos debe seguir la siguiente nomenclatura

- nombreArchivo.cy.js

El siguiente es un ejemplo del codigo de las pruebas.

```
describe('Pruebas a la aplicacion Ghost', () => {

   beforeEach(() => {
      // Visitar la página de inicio de sesión de Ghost
      cy.visit('http://localhost:2368/ghost/#/signin');

      // Iniciar sesión
      cy.get('input[name="identification"]').type('admin@redfox.com.co');
      cy.get('input[name="password"]').type('Admin123+');
      cy.get('button[type="submit"]').click();

      // Esperar a que la página principal de Ghost se cargue
      cy.url().should('include', '/ghost/#/dashboard');
   });


   it('01-Debería crear un nuevo post', () => {

      // Navegar al editor de publicaciones
      cy.get('a[href="#/editor/post/"]').click();

      // Esperar a que el editor de publicaciones se cargue
      cy.url().should('include', '/ghost/#/editor/post');

      // Completar el formulario del nuevo post
      cy.get('textarea[placeholder="Post title"]').type('Post Generado con Cypress');
      cy.get('a[data-test-link="posts"]').click();

   });

});
```

Configuración de monkey-cypress:

Para configurar monky-cypress se realiza clonando el proyecto monkey-cypress desde el repositorio github: https://github.com/TheSoftwareDesignLab/monkey-cypress.git

Esta versión de monkey-cypress funciona con la versión 4.2.0 de cypress, actualmente la versión de cypress esta en la vesion 13.15.1. por lo cual si tiene instalado la ultima versión se recomienda des instalar y volver a instalar cypress en la versión 4.2.0 con el siguiente comando desde la raíz del proyecto en la consola de comandos.

- npm install -g cypress@4.2.0

E monkey se configura desde el archivo monkey-config.json aquí se configura diferentes parámetros para la ejecución de las pruebas con el monkey.

```
{
   "projectId":"TSDL-Monkey-with-cypress",
   "baseUrl":"http://localhost:2368/ghost/",
   "env":{
      "appName":"App prueba",
      "events":50,
      "delay":300,
      "seed":1234,
      "pctClicks":19,
      "pctScroll":17,
      "pctSelectors":16,
      "pctKeys":16,
      "pctSpKeys":16,
      "pctPgNav":16
   },
   "integrationFolder": "./cypress/integration/monkey",
   "pluginsFile": "./cypress/plugins/index.js",
   "pageLoadTimeout":120000,
   "defaultCommandTimeout":120000,
   "testFiles": "monkey.js",
   "videosFolder":"./results"
}
```

Para ejecutar el monkey desde la raíz del proyecto en la consulta de comandos ejecute el siguiente comando:

- cypress run -C monkey-config.json

Esto ejecutara la prueba con el monkey de manera automática.

2.2 Configuración de RIPuppet:

RIPuppet es una herramienta de automatización de pruebas para aplicaciones web que permite simular acciones de usuario, como clics y la entrada de datos en formularios. Es fácil de usar y permite a los desarrolladores verificar la funcionalidad de sus aplicaciones de manera rápida y eficiente. Además, RIPuppet puede interactuar con diferentes navegadores, lo que ayuda a garantizar que una aplicación funcione correctamente en múltiples entornos.

La herramienta de prueba se puede descargar desde github clonando el siguiente repositorio en el folder que desee.

https://github.com/TheSoftwareDesignLab/RIPuppetCoursera.git

Despues de clonar el respositorio busque el siguiente archivo dentro del folder creado con la clonación. “config.json”

El archvio tiene la siguiente estructura.

```
{
        "url": "http://localhost:2368",
        "headless": true,
        "depthLevels": 1,
        "inputValues": true,
        "values": {
                "identification": "admin@redfox.com.co",
                "password": "Admin123+"
        },
        "browsers": ["chromium","webkit","firefox"],
        "waitBetweenActions": 3000
}
```

Aquí puede indicar aspectos clave como la URL de prueba (p. ej., http://localhost:2368/ghost), niveles de profundidad, valores de entrada automáticos, y los navegadores probados (chromium, firefox, etc.)

Para ejecutar la prueba ejecute el siguiente comando desde la raíz del proyecto en la consola de comandos:

- node index.js

Esto ejecutara la prueba de acuerdo al archivo de configuración.

3 Resultados de la Ejecución

3.1 Monkey-cypress

Después de ejecutar monkey-cypress sobre la interfaz administrativa de ghost estos son los resultados.

- Monkey-cypress no fue capaz de pasar de la ventana de inicio de sesión.
- Se genero un video de la prueba
- Se genero un archivo html de los resultados de la prueba

Después de ejecutar monkey-cypress sobre la interfaz publica del usuario estos son los resultados.

Cypress versión 4.2.0 se ejecuta en modo headless usando el navegador Electron 80. La prueba se basa en el archivo monkey.js, configurado para realizar una serie de interacciones aleatorias en la aplicación web.

Durante la ejecución, la herramienta monkey intenta una variedad de eventos en la aplicación, como navegación de páginas, clics aleatorios, eventos de desplazamiento, y presiones de teclas. Estos se generan aleatoriamente basándose en una "seed" o semilla de valor 1234.

Se produce un error específico de: cy.scrollTo() failed because this element is not scrollable: <window>.

Esto puede indicar que el comando cy.scrollTo() intentó desplazarse en un elemento que no admite desplazamiento (<window>). Es probable que el script monkey.js esté configurado para realizar un desplazamiento en una sección de la página que no tiene contenido desplazable o podría estar intentando desplazarse en la ventana principal (window) sin comprobar primero si es posible realizar el desplazamiento.

La prueba registró 0 exitosos y 1 fallido en un tiempo total de 2 minutos y 17 segundos. Se generó una captura de pantalla y un video que podrían ayudar a identificar visualmente en qué punto se produce el error.

3.2 Resultados de la Ejecución de RIPuppet

Este resultado de prueba con RiPuppet muestra los siguientes puntos clave de lo que está ocurriendo:

RiPuppet comienza explorando desde el nivel de profundidad 0, indicando que empieza en la URL raíz (http://localhost:2368).

Identifica varias páginas hijas desde este nivel de profundidad, enumerando las URL que se encuentran dentro del sitio, como páginas de inicio de sesión, registro y otras.

Se presenta un error al intentar hacer clic en un elemento que RiPuppet detecta pero que ya no está adjunto al DOM. Este problema puede ocurrir cuando el DOM cambia (por ejemplo, debido a una recarga de la página o animación) justo antes de que el script intente interactuar con el elemento. RiPuppet intenta interactuar con un elemento en la estructura de dentro de una sección con la clase gh-container is-list gh-outer, pero el elemento queda fuera de la vista o pierde su conexión con el DOM en el momento de intentar el clic.

La ejecución se detiene por un error debido a una promesa no controlada en el código de RiPuppet (triggerUncaughtException). Esto indica que el flujo de la prueba no maneja adecuadamente el fallo de la acción de clic, provocando una excepción en el script.

.

4 Comparativa entre monkey-cypress y RIPuppet

Aspecto	monkey-cypress	ripuppet
Propósito Principal	Pruebas aleatorias de interacciones con la aplicación en el navegador	Exploración aleatoria y automatización de pruebas visuales de la interfaz
Curva de Aprendizaje	Moderada a alta; requiere conocimiento en Cypress y configuración de eventos aleatorios.	Moderada; es necesario entender los archivos de configuración de la librería.
Configuración Inicial	Usa un archivo de configuración en formato JSON para definir eventos y porcentajes de interacción.	Usa un archivo de configuración en formato JSON para definir eventos y porcentajes de interacción.
Soporte de Navegadores	Electron y otros navegadores soportados por Cypress, principalmente en modo headless.	Soporte de Chromium y navegadores basados en puppeteer, incluyendo opciones de pantalla completa y personalización del viewport.
Errores Comunes	Problemas con elementos fuera del viewport o no interactuables por eventos aleatorios.	Problemas al encontrar elementos que no están adjuntos al DOM, reintentos automáticos.
Logging y Reportes	Incluye logs de eventos como navegación, clics, y errores comunes, con opciones de captura de video.	Permite capturar logs detallados de interacciones en DOM y screenshots de estados específicos.
Rendimiento	Adecuado para ejecuciones rápidas en pruebas simples de interfaz.	Más lento en ejecuciones complejas debido al nivel de detalle y reintentos.

5 Conclusión

A lo largo de las pruebas realizadas en la aplicación Ghost, se ha identificado la necesidad de un conocimiento más profundo del código interno de la plataforma. Este entendimiento es crucial para diseñar casos de prueba más precisos y efectivos, ya que cada interacción y flujo en Ghost puede presentar desafíos específicos que requieren familiaridad con su estructura y lógica de navegación. Por lo tanto, es recomendable que los desarrolladores y testers que trabajen en esta plataforma se comprometan con una curva de aprendizaje dedicada al código fuente de Ghost, lo que permitirá anticiparse a posibles problemas y optimizar la cobertura de prueba.

Adicionalmente, la experiencia con las herramientas monkey-cypress y ripuppet ha demostrado que, si bien ambas son potentes y flexibles, su uso efectivo depende de la comprensión de sus configuraciones y opciones avanzadas. Invertir tiempo en dominar estas herramientas ofrece beneficios significativos, ya que permite maximizar su robustez y adaptabilidad en escenarios de prueba variados. Con un mayor conocimiento sobre las capacidades y limitaciones de monkey-cypress y ripuppet, los testers pueden realizar ajustes más precisos y aprovechar características como el control de eventos aleatorios, navegación de múltiples niveles y manejo de errores específicos, aumentando así la calidad y efectividad de las pruebas.

Fomentar una curva de aprendizaje en el código de Ghost y en las herramientas de prueba no solo incrementa la efectividad de las pruebas, sino que también abre la puerta a una estrategia de prueba más robusta, flexible y adaptada a los desafíos particulares de la aplicación.

En cuanto a Cypress, las versiones más recientes han mejorado en gran medida el control sobre los elementos de la página, permitiendo crear pruebas más precisas y detalladas. Estas mejoras ayudan a que las pruebas interactúen de

manera más efectiva con los distintos elementos de la interfaz. Por otro lado, la herramienta monkey-cypress está algo desactualizada respecto a estas versiones más modernas de Cypress, lo cual limita su funcionalidad. Para aprovechar al máximo las capacidades de Cypress, es importante conocer las versiones actuales, que ofrecen mejores opciones para diseñar pruebas específicas y efectivas.