Reporte de la experiencia

- **Autores:**
  - María Camila Martínez
  - Manuel Bejarano
  - Juan Sebastián Vargas
  - Héctor Franco

- **Curso:** MISW4103 - Pruebas Automatizadas de Software

1. Descripción de la prueba

Durante las pruebas de reconocimiento con las herramientas de **Monkey-Cypress** y **RIPuppet** sobre la aplicación **Ghost**, el equipo se encontró con distintos resultados a la hora de ejecutar las pruebas. A continuación, se describen el proceso de instalación del ambiente de pruebas y su ejecución en cada una de las herramientas descritas, posteriormente se hace un análisis grupal de los resultados obtenidos y los beneficios y contras encontrados de igual manera:

2. Ambiente de ejecución de la prueba

Lo primero que se realizó fue preparar las máquinas locales de cada uno de los miembros del equipo con base a la siguiente configuración de librerías y entornos de desarrollo de pruebas.

2.1 Especificaciones técnicas del ambiente de pruebas usado:

- SO: Windows y MacOS
- Node Version: v12.22.12
- NPM Versión: v6.14.16
- GIT: Versión más reciente o predefinida en sistemas UNIX
- Visual Studio Code

2.2 Instalar Node JS

Para poder ejecutar bien el set de herramientas disponible fue requerido instalar en las máquinas locales la versión de Node JS descrita en la sección 2.1.

Para comprobar la instalación, se procedió a ejecutar los siguientes comandos en la terminal o línea de comando de windows: Node:

```
> node --version
v12.22.12
```

npm:

```
> npm --version
v6.14.16
```

2.3 Instalar GIT

Para clonar los repositorios en donde se encuentran las herramientas, es necesario usar la herramienta GIT, la cual puede ser instalada siguiendo los pasos de su página oficial en la sección downloads.

Una vez instalada la herramienta, se puede comprobar su correcto funcionamiento con el siguiente comando. El resultado debe ser algo parecido a esto.

```
> git --version
git version 2.39.3 (Apple Git-145)
```

2.4 Instalar Visual Studio Code

Se necesita un IDE robusto que permita modificar el código fuente las herramientas para adaptarlo a las necesidades y poder ejecutar las pruebas sobre Ghost. Para ello se hizo uso de Visual Studio Code, el cual será más que suficiente para editar los archivos de las herramientas.

2.4 Descargar Monkey-Cypress

Esta herramienta de Random Testing puede ser obtenida mediante el siguiente comando en la terminal o CMD en el directorio de su preferencia.

```
> git clone https://github.com/TheSoftwareDesignLab/monkey-cypress.git
```

2.5 Descargar RIPuppet

Esta herramienta de Random Testing en la GUI puede ser obtenida mediante el siguiente comando en la terminal o CMD en el directorio de su preferencia.

```
> git clone https://github.com/TheSoftwareDesignLab/RIPuppetCoursera.git
```

2.6 Instalar las librerías de las herramientas

Una vez se han descargado ambas herramientas desde Github es indispensable instalar las dependencias propias de cada una de las herramientas. Para ello solo es necesario hacer dos pasos:

1. Abrir una terminal o consola de comandos (CMD) sobre la carpeta raíz del proyecto (Cypress-Monkey y RIPuppet)
2. Ejecutar el siguiente comando de Node:

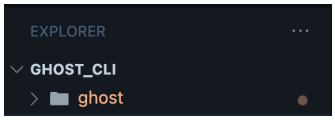
```
> npm install
```

Esto descargará e instalará las dependencias necesarias para poder ejecutar las herramientas de las pruebas sobre Ghost

3. Levantar servicio de Cliente de Ghost

Las pruebas de reconocimiento se ejecutarán sobre la versión de Ghost `v5.81.1`. Para levantar el servicio de Ghost se deben realiza dos pasos muy simples:

1. Dirigirise a la ruta donde está instalada la instancia de Ghost CLI y el Cliente de Ghost
   ejemplo:



2. Ejecutar el siguiente comando sobre la carpeta `ghost`

```
> ../node_modules/ghost-cli/bin/ghost start
```

```
> ../node_modules/ghost-cli/bin/sudo ghost start (usuarios UNIX)
```

Una vez se haya ejecutado este comando y tomando como referencia que se tenía instalado Ghost con antelación el servicio correrá sobre el puerto **2368**, accedido desde la siguiente URL: `http://localhost:2368`

4. Ejecutar Monkey-Cypress (Random-Testing)

4.1 Modificar código fuente

El proyecto descargado viene por defecto para ser ejecutado contra un sitio web distinto, en este paso se deben modificar una serie de archivos para que la prueba corra de forma exitosa.

Hay dos tipos de pruebas disponibles aquí **monkey** y **smart-monkey**, se comienza con m **monkey**

4.1.1 Monkey

1. Editar archivo `monkey-config.json` Este archivo hace la configuración inicial con datos importantes tales como la url base sobre la cual será ejecutada la prueba, cantidad de eventos, y el porcentaje de peso de los eventos a testear. Esta configuración quedó más o menos así:

```
{
    "projectId":"TSDL-Monkey-with-cypress",
    "baseUrl":"http://localhost:2368/ghost/#/sigin",
    "env":{
        "appName":"GHOST",
        "events":50,
        "delay":1000,
        "seed":1235,
        "pctClicks":20,
        "pctScroll":1,
        "pctSelectors":20,
        "pctKeys":20,
        "pctSpKeys":20,
        "pctPgNav":19
    },
    "integrationFolder": "./cypress/integration/monkey",
    "pluginsFile": "./cypress/plugins/index.js",
    "pageLoadTimeout":120000,
    "defaultCommandTimeout":120000,
    "testFiles": "monkey.js",
    "videosFolder":"./results"
}
```

Siendo **baseUrl** el valor de la url de ghost que se desea probar

2. El segundo archivo es `monkey/monkey.js` Aquí es donde va la lógica de las pruebas, realizando eventos aleatorios como clicks, inputs, selects y demás. Pero para poder pasar de la sección de sigin de Ghost es necesario realizar un pequeño ajuste para hacer que los robots hagan login sin problema y avanzar.

Sobre el it de visitar el sitio, más o menos sobre la línea **549** del archivo, se agrega el siguiente código fuente para hacer login y que loe eventos puedan ser ejecutados sobre otras interfaces de Ghost

```
    cy.wait(1000)
    cy.get('#identification').type('correo')
    cy.wait(1000)
    cy.get('#password').type('contraseña')
    cy.wait(1000)
    cy.get('#ember5').click()
    cy.wait(1000)
```

4.1.2 Smart Monkey

1. Editar archivo `smart-monkey-config.json` Este archivo hace la configuración inicial con datos importantes tales como la url base sobre la cual será ejecutada la prueba, cantidad de eventos, y el porcentaje de peso de los eventos a testear. Esta configuración quedó más o menos así:

```
{
    "projectId":"TSDL-Monkey-with-cypress",
    "baseUrl":"http://localhost:2368/ghost/#/sigin",
    "env":{
        "appName":"GHOST",
        "events":50,
        "delay":1000,
        "seed":1235,
        "pctClicks":20,
        "pctScroll":1,
        "pctSelectors":20,
        "pctKeys":20,
        "pctSpKeys":20,
        "pctPgNav":19
    },
    "integrationFolder": "./cypress/integration/monkey",
    "pluginsFile": "./cypress/plugins/index.js",
    "pageLoadTimeout":120000,
    "defaultCommandTimeout":120000,
    "testFiles": "monkey.js",
    "videosFolder":"./results"
}
```

Siendo **baseUrl** el valor de la url de ghost que se desea probar

2. El segundo archivo es `monkey/smart-monkey.js` Aquí es donde va la lógica de las pruebas, realizando eventos aleatorios como clicks, inputs, selects y demás. Pero para poder pasar de la sección de sigin de Ghost es necesario realizar un pequeño ajuste para hacer que los robots hagan login sin problema y avanzar.

Sobre el it de visitar el sitio, más o menos sobre la línea **679** del archivo, se agrega el siguiente código fuente para hacer login y que loe eventos puedan ser ejecutados sobre otras interfaces de Ghost

```
    cy.wait(1000)
    cy.get('#identification').type('correo')
    cy.wait(1000)
    cy.get('#password').type('contraseña')
    cy.wait(1000)
    cy.get('#ember5').click()
    cy.wait(1000)
```

4.2 Resultados hallados

1. Exploración de interacciones inusuales: Monkey-Cypress demostró ser eficiente en generar interacciones inusuales con la interfaz de usuario de la aplicación, logrando identificar escenarios no convencionales que no habían sido detectados mediante pruebas exploratorias tradicionales.
2. Identificación de errores de sintaxis CSS: La herramienta detectó errores de sintaxis en selectores CSS, los cuales pueden causar fallos en la ejecución de scripts y afectar la funcionalidad de la aplicación.
3. Cobertura de pruebas aleatorias: Monkey-Cypress abarcó una amplia gama de interacciones aleatorias, proporcionando una buena cobertura de pruebas para escenarios inesperados.

4.3 Pros y Contras

A continuación, se muestra una tabla con los pros y contras que el equipo encontró con esta herramienta de random testing.

Pros	Contras
Rápidas de ejecutar	Puede ser difícil replicar un caso de prueba específico que causó un fallo al ser aleatorio.
No necesitan una máquina muy potente.	Puede generar casos de prueba que no agregan valor y requieren de revisión manual para evaluar los resultados.
Son muy baratas de implementar	Menos efectivo para probar escenarios específicos que requieren pasos predefinidos.
Capacidad para grabar y reproducir pruebas.	Posibilidad de eventos inválidos o sin efecto.
No se requiere un profundo entendimiento de la aplicación a testear.	Interrupción de pruebas tras detectar errores

Buena cobertura de pruebas para escenarios inesperados	
El código fuente para su implementación es fácil de entender.	

5. Ejecutar RIPuppet

5.1 Modificar código fuente

El proyecto descargado viene con un punto de inicio el cual es **index.js** y con un archivo de configuración, **config.json**. Este último debe ser editado para que los rippers se ejecuten sobre el cliente de ghost

```
{
    "url": "http://localhost:2368/ghost/#/signin",
    "headless": true,
    "depthLevels": 1,
    "inputValues": true,
    "values": {
        "password": "contraseña",
        "identification": "correo"
    },
    "browsers": [
        "chromium", "webkit", "firefox"
    ]
}
```

El valor de **url** debe ser cambiado por el servidor local donde está corriendo Ghost, el arreglo de **values**, debe ser modificado con los id como claves y los valores el correo y contraseña del formulario de login de Ghost, por último, el campo inputValues debe ser modificado a true para que tome los valores del arreglo de inputs.

5.2 resultados hallados

1. RIPuppet realizó una inspección de los nodos por donde iba a pasar a realizar las pruebas.
2. Toma pantallazos sobre las vistas, antes y después de realizar una acción como el llenar un formulario o hacer click sobre un botón.
3. Exporta un reporte en HTML interactivo con los grafos y los puntos de los errores obtenidos en cada uno de ellos en caso de existir.

5.3 Pros y contras

Pros	Contras
Realiza un mapeo de la interacción de la prueba sobre la aplicación.	Requiere mayor conocimiento técnico y experiencia en pruebas automatizadas.
Puede proporcionar resultados más precisos en pruebas de interfaz.	Se ejecuta más lentamente que Cypress Monkey debido a su enfoque sistemático.
Proporciona informes detallados sobre los pasos de prueba, resultados, errores y métricas relevantes.	Posibilidad de generación de eventos inválidos o sin efecto.
Muestra el punto exacto en donde hubo un error con un pantallazo y su descripción.	Para poder ver el reporte en HTML de la prueba es necesario lanzar un servidor local o una herramienta que sirva como tal.

6. Incidente encontrado

Comportamiento esperado	Comportamiento actual	Enlace del Incidente
Se espera que el selector CSS utilizado en la función querySelectorAll identifique correctamente los elementos del DOM especificados sin lanzar errores, permitiendo la manipulación o análisis de estos elementos en el documento.	Al ejecutar el comando cypress run --config-file ./monkey-config.json, se reporta el error SyntaxError: Failed to execute 'querySelectorAll' on 'Document': '.gh-content > table:not(.gist table)' is not a valid selector. Este error indica que el selector utilizado no es válido, impidiendo la ejecución de pruebas que dependan de la selección de estos elementos en el documento.	Issue 28

7. Comparativa de las dos herramientas

Las dos herramientas son muy buenas para realizar pruebas de reconocimiento sobre las interfaces de la aplicación, siendo Monkey Cypress mucho más intuitiva de usar y de modificar su código fuente en especial para alguien que no es experto en pruebas y/o en esta herramienta.

De igual manera, la replicabilidad del set de pruebas es muy fácil de hacer con Monkey-Cypress, ya que en su código fuente se encuentra la semilla con la que se realizaron los eventos aleatorios y en caso de tener que replicarlo en otra máquina, el usuario podría ver el mismo paso a paso de eventos.

Por otro lado, RIPuppet nos muestra de forma más clara el grafo de interfaces visitadas y en caso de hallar un error, marca el nodo en color rojo y muestra la interfaz y el mensaje del error encontrado.