

UTFSM

Departamento de Informática



Proyecto Entrega 2

Asignatura: INF331

Pruebas de Software

Profesora: Oscar Reyes H

Pablo Contreras, 201973572-1

Vania Gallardo, 201973619-1

Noviembre-2024

Repositorio: www.github.com/Equipo-7-PDS-Proyecto/pruebas-software-frontend

1 Alcances de la solución implementada

1.1 Uso de Selerium

Selerium se utilizó para realizar las pruebas de del Front-end mediante el uso de la IDE Selerium. Se realizaron 5 test que son los siguientes:

1. Anadir inventario: Agrega un producto en el inventario en la sección de "Administración"
2. Editar Perfil: Edita el perfil de usuario en la sección "Perfil"
3. Ingresar como administrador: Ingresa como administrador en la sección "Login"
4. Ingresar como usuario: Ingresa como usuario en la sección "Login"
5. Registro usuario con datos incorrectos (diferentes contraseñas): La ejecución del test termina con el mensaje de alerta "Las contraseñas no coinciden".

Los test son ejecutados por el pipeline del Jenkins corriendo el comando "selenium-side-runner test/TestSelenium.side".

2 Descripción del trabajo realizado

En esta entrega, se llevó a cabo la integración completa del frontend del proyecto a una *pipeline* de **Jenkins**, asegurando un flujo de integración y despliegue continuo (*CI/CD*) confiable. Este trabajo involucró varios pasos clave, detallados a continuación:

2.1 Integración del Frontend a Jenkins

El frontend fue configurado como un proyecto dentro de Jenkins para automatizar su proceso de despliegue. Esto incluyó:

- Configuración de un *pipeline declarativo*, dividiendo el proceso en varias etapas:
 - **Instalación de dependencias:** Se utiliza *npm* para instalar las dependencias necesarias del proyecto.
 - **Compilación de la aplicación:** Se ejecuta el comando `npm run build` para compilar los archivos del frontend, generando el contenido en la carpeta `.next`.
 - **Pruebas automatizadas:** Se integraron pruebas funcionales usando *Selenium*, ejecutadas a través de `selenium-side-runner` en un entorno configurado con *Xvfb* para manejar pruebas en un entorno gráfico sin interfaz.
 - **Despliegue:** Los archivos compilados se copian al servidor de producción utilizando `systemctl` para detener el servicio previo, copiar los archivos y reiniciar el servicio con el nuevo código.
- Configuración de un *webhook* en GitHub para que cada *push* a la rama principal dispare automáticamente el *pipeline*.
- Configuración de notificaciones en Slack para informar el estado de los despliegues (éxito o fallo).

2.2 Automatización de Pruebas con Selenium

Se integraron pruebas funcionales para el frontend, implementadas con *Selenium*. Estas pruebas permiten validar el comportamiento de la aplicación antes de proceder con el despliegue en producción. Los pasos principales fueron:

- Configuración de pruebas usando un archivo `.side` para especificar los casos de prueba.
- Instalación de dependencias necesarias, como `chromedriver` y `selenium-side-runner`.

- Uso de `Xvfb` para emular un entorno gráfico en Amazon Linux 2, necesario para ejecutar las pruebas en Jenkins.
- Ejecución de las pruebas contra un servidor temporal levantado durante la etapa de pruebas para garantizar que se pruebe el código nuevo y no el código actualmente desplegado.

2.3 Despliegue Automatizado

El despliegue automatizado se realizó utilizando `systemd` para gestionar el servicio del frontend. Se configuraron los siguientes pasos dentro del *pipeline*:

- Detención del servicio actual (`frontend.service`).
- Copia de los archivos compilados desde la carpeta de `Jenkins workspace` al directorio de producción.
- Reinicio del servicio para que la aplicación use el nuevo código.

2.4 Resolución de Problemas

Durante el proceso de integración, se enfrentaron y resolvieron los siguientes problemas:

- **Pruebas en entornos gráficos:** La falta de un entorno gráfico en Amazon Linux 2 fue resuelta configurando y utilizando `Xvfb`.
- **Acceso a chromedriver:** Se configuró correctamente el `PATH` para incluir `chromedriver` y permitir que Jenkins lo reconozca.
- **Conflictos de versiones:** La integración de las dependencias de *Selenium* y *Chromedriver* se resolvió asegurando que estuvieran correctamente instaladas en el entorno del *pipeline*.
- **Ejecución de pruebas contra código antiguo:** Se levantó un servidor temporal para ejecutar las pruebas contra el nuevo código antes de desplegarlo en el servicio principal.