Alquifiestas: Sensacional Eventos
Tarea 3
Herramientas para pruebas automatizadas

Hugo Eduardo Rivas Fajardo - 22500
José Santiago Pereira Alvarado - 22318
Nancy Gabriela Mazariegos Molina - 22513
Giovanni Alejandro Santos Hernández - 22523
Mauricio Julio Rodrigo Lemus Guzmán- 22461
Alexis Mesías Flores - 22562

Herramientas o frameworks existentes para test automáticos de pruebas unitarias en el o los lenguajes que está trabajando el grupo, tanto para frontend como para backend.

- Frontend:
  - Vue Test Utils
    - Descripción: Biblioteca oficial de Vue.js para pruebas unitarias de componentes.
    - Características: Permite montar y probar componentes de Vue, simulación de eventos, y verificación de la salida renderizada.
  - Jest
    - Descripción: Framework de pruebas de JavaScript que funciona bien con Vue.
    - Características: Ofrece un entorno de pruebas con soporte para mocks, pruebas asíncronas, y snapshots.
  - Cypress
    - Descripción: Conocido principalmente por pruebas end-to-end, pero también se puede usar para pruebas unitarias de componentes.
    - Características: Ofrece pruebas rápidas, automáticas y con un detallado reporte de errores.
  - Selenium
    - Descripción: Herramienta para pruebas automatizadas de aplicaciones web, ideal para pruebas end-to-end.
    - Características: Soporte para múltiples navegadores, integración con frameworks de pruebas como pytest, y posibilidad de automatizar interacciones complejas del usuario.
- Backend:
  - unittest
    - Descripción: Framework de pruebas integrado en la biblioteca estándar de Python.
    - Características: Proporciona herramientas para crear pruebas, definir conjuntos de pruebas y comprobar resultados.
  - pytest
    - Descripción: Alternativa popular a unittest, conocida por su simplicidad y capacidad de extensión.
    - Características: Sintaxis sencilla, soporta fixtures, y extensible mediante plugins.
  - Flask-Testing
    - Descripción: Extensión para Flask que facilita la creación de pruebas unitarias y funcionales.

        - Características: Soporte para pruebas de aplicaciones Flask, incluyendo pruebas de rutas y contexto de la aplicación.

    - Postman
        - Descripción: Herramienta para pruebas de API que permite realizar pruebas manuales y automatizadas.
        - Características: Creación de colecciones de peticiones, ejecución de scripts de prueba, y posibilidad de usar Newman para ejecutar colecciones desde la línea de comandos.
-

Describir las herramientas que utilizará el equipo de desarrollo y por qué se ajustan mejor a su proyecto. Explicar las razones de su elección.

Identifique 4 pruebas de TDD para su proyecto.

- Para las pruebas TDD se utilizó Postman Collections. Se realizaron pruebas para distintas funcionalidades de clientes y productos. Una prueba que se realizó para clientes fue la de loginVisual la cual verifica que el código de estado de la respuesta es 200, que el tipo de contenido de la respuesta es text/html, que el tiempo de respuesta sea menor a 200 ms, que el HTML contiene los elementos de formulario esperados y que los recursos de Bootstrap se cargan desde las URLs esperadas. Similarmente, la prueba de CrearCliente Verifica que el código de estado de la respuesta es 200, que el tiempo de respuesta es menor a 200 ms, que la respuesta contenga las etiquetas html, head, body y script y que el tipo de contenido de la respuesta es text/html.

  Para la parte de
-
-

Identifique 4 pruebas de BDD para su proyecto.
Más de 3 ejemplos de automatización de pruebas unitarias tanto en frontend como en backend, usando las herramientas seleccionadas.

Conclusiones: El uso de Selenium para pruebas automatizadas en un sistema de alquiler de fiestas construido con Flask y Vue.js ha demostrado ser muy eficiente y beneficioso. La configuración inicial y la creación de scripts requieren una inversión considerable de tiempo y aprendizaje, pero la capacidad de automatizar tareas repetitivas, realizar pruebas en múltiples navegadores y detectar errores tempranamente compensa ampliamente este esfuerzo. La rapidez en la ejecución de pruebas automatizadas y la cobertura exhaustiva que ofrecen mejoran significativamente la calidad y la fiabilidad de la aplicación.

A pesar de los desafíos asociados con el mantenimiento de scripts y la posibilidad de generar falsos positivos o negativos, Selenium se integra bien con herramientas de

desarrollo continuo, como Jenkins o GitHub Actions. Esto facilita el flujo de trabajo y asegura el buen funcionamiento del sistema a lo largo de su ciclo de desarrollo. En resumen, Selenium proporciona una solución robusta y efectiva para mejorar la calidad del software, a pesar de los desafíos que presenta.