Emiliano Cabrera - A01025453
Andrew Dunkerley - A01025076
Do Hyun Nam - A01025276

Pruebas de Software

Pruebas locales

Son las pruebas más básicas en el desarrollo de software y se centran en la verificación del funcionamiento individual de cada componente de un sistema. Se les conoce como pruebas unitarias y son rápidas y baratas de hacerse por su alcance limitado y ambiente de ejecución cerrado (Pittet, s.f.).

Pruebas de integración

La idea de esta prueba es verificar el funcionamiento correcto en conjunto de todos los componentes individuales de un proyecto. Estos componentes suelen funcionar apropiadamente al estar separados, pero al integrar puede ser que existan errores o fallas en envíos o accesos a datos entre ellos. Suele hacerse para comprobar que las partes visuales, de funcionamiento, de guardado de información y de respaldo sean capaces de actuar como un solo sistema. (Vargas, s.f.)

Pruebas alfa

Las pruebas alfa son aquellas que toman lugar antes del lanzamiento del sistema al público general o al cliente interesado (Ebooks Online, s.f.). Fungen para prototipar el proyecto de inicio, antes de avanzar con el desarrollo demasiado y entrar a la etapa de pruebas de calidad.

Pruebas de regresión

Al cambiar o mejorar partes de un sistema ya implementado, dichas modificaciones pueden causar que otras partes del sistema encuentren errores. Las pruebas de regresión sirven para que cualquier actualización o modificación al funcionamiento de un sistema no cree accidentalmente algún error que sea perjudicial para su uso. (Vargas, s.f.)

Pruebas dinámicas de validación.

Más comúnmente llamadas pruebas de aceptación de usuario, las pruebas de validación sirven para que, en un ambiente real de uso, el sistema cumpla con todos los requerimientos que necesita el cliente. Suelen hacerse durante la fase final de testing, ya que involucran dar acceso a individuos no

relacionados al desarrollo del sistema, pero que son los involucrados en su operación futura. (Vargas, s.f.)

Pruebas bajo condiciones frontera

Conocidas como pruebas no funcionales, el objetivo de estas pruebas es conocer los límites del funcionamiento del sistema y se dividen en carga, rendimiento y estrés. Cada una de estas mide un aspecto distinto de la ejecución del sistema, ya sea capacidad de respuesta de red, uso de recursos físicos en el sistema huésped, o número de usuarios simultáneos que puede soportar (Soto, 2021).

Herramientas de pruebas automatizadas

De acuerdo a Geekflare (2022), algunas de las herramientas más prominentes son:

- Selenium
  - tipo de prueba: funcionales
  - licencia: open source
  - plataforma/lenguajes: múltiples
  - particularidades: usa un add-on de navegador
- Gatling
  - tipo de prueba: de carga
  - licencia: open source
  - plataforma/lenguajes: Scala, Kotlin, Java
  - particularidades: puede usar la nube
- Testim
  - tipo de prueba: múltiples
  - licencia: comercial
  - plataforma/lenguajes: JS
  - particularidades: usa IA
- HeadSpin
  - tipo de prueba: múltiples
  - licencia: open source
  - plataforma/lenguajes: múltiples
  - particularidades: usa ML y utiliza infraestructura de máquinas virtuales

Documentación

Para nuestro programa se escribió una prueba unitaria que en conjunto cubre todos los funcionamientos requeridos por el sistema lo cual lo hace un sistema de pruebas de integración. La prueba consiste en corroborar que se puedan subir archivos al sistema y que este cree un render. En esencia, se realiza

primariamente una comparación del snapshot y del render. Esto significa que comparamos los renders del código escrito y de un render, para que nos podamos asegurar de que todo el UI no cambie de manera inesperada al interactuar con ella. Esto lo convierte en una prueba de integración porque estamos garantizando que todos los componentes que se incluyen en el árbol principal de <App /> funcionan de manera esperada durante su interacción en la aplicación.

En la imagen superior observamos que tenemos un error al momento de probar el componente de <FileUpload />, esto se debe a que se empezó a utilizar una librería deprecada llamada enzyme, la cual ya no tiene soporte para React > 16. Para poder generar pruebas de componentes de varios otros

componentes, se tendría que buscar librerías o frameworks alternativos para lograr esto de mejor manera y adaptándonos a las necesidades tecnológicas que van surgiendo.

Referencias

Vargas, C. (s.f.). *Tipos de pruebas funcionales para el aseguramiento de la calidad.* https://trycore.co/transformacion-digital/tipos-de-pruebas-funcionales/

Ebooks Online. (s.f.). *¿Qué es la prueba alfa? Proceso, ejemplo.* https://ebooksonline.es/que-es-la-prueba-alfa-proceso-ejemplo/

Soto, V.M. (20 de mayo de 2021). *Conoce qué son las pruebas no funcionales de software.* https://www.pragma.com.co/blog/conoce-que-son-las-pruebas-no-funcionales-de-software

Pittet, S. (s.f.). *Los distintos tipos de pruebas de software.* *https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing*

Khatri, V. (18 de julio de 2021). Las 19 mejores herramientas de prueba de software que debe conocer como evaluador. *https://geekflare.com/es/software-testing-tools/*