Ingeniería de Software 2

Iris Ayala – 23965

Gabriel Bran, 23590

David Domínguez, 23712

Luis Padilla, 23663

Anggie Quezada, 23643

Tarea 4: Pruebas automatizadas – Plan de pruebas

1. Herramientas o frameworks para prueba automatizada

Las pruebas automatizadas son procesos ejecutados por software que validan automáticamente el comportamiento correcto de una aplicación sin intervención humana. Existen tres tipos clave: las pruebas funcionales, que aseguran que las funcionalidades del sistema se comportan como se espera desde la perspectiva del usuario final; las pruebas de integración, que validan que los distintos módulos o servicios del sistema funcionen correctamente cuando se combinan, como por ejemplo la comunicación entre el frontend y la API; y las pruebas de regresión, que detectan si cambios recientes en el código han roto funcionalidades previamente estables. Automatizar estas pruebas es importante porque reduce errores humanos, mejora la velocidad de entrega del software, garantiza mayor estabilidad en cada versión y permite escalar los proyectos sin comprometer la calidad (Global Business IT, s. f.).

Fronted - React

1. **Playwright**

   Tipo: Pruebas E2E, funcionales, regresión visual.

   Ventajas:

   - Soporte nativo para React y aplicaciones Vite.
   - Permite testing en múltiples navegadores (Chromium, Firefox, WebKit).
   - Muy rápido y confiable.
   - Buen soporte para grabar pruebas.

   Ideal para: Pruebas E2E y regresión visual con capturas.

2. **Testing Library (React Testing Library)**

Tipo: Pruebas unitarias y de integración de componentes.

Ventajas:

- Testing centrado en la experiencia del usuario (interacción real).
- Integración con Vitest (equivalente moderno a Jest para Vite).
- Fácil de mantener y escalar.

Ideal para: Componentes y lógica de UI, integración con hooks y estados.

3. **Cypress**

Tipo: E2E, integración y pruebas funcionales.

Ventajas:

- Interfaz visual para depurar tests.
- Muy popular y con comunidad activa.
- Compatible con Vite con configuración mínima.

Ideal para: Pruebas funcionales y flujos completos de usuario.

Tecnología seleccionada: Playwright

Porque es una de las herramientas más modernas, completas y robustas para pruebas automatizadas de interfaces web. Ha ganado popularidad en la comunidad de desarrollo debido a su enfoque confiable, su rendimiento superior y su amplio soporte multiplataforma. A diferencia de otras herramientas, Playwright permite ejecutar pruebas en todos los navegadores principales, lo que garantiza una mayor cobertura de compatibilidad real con distintos entornos de usuario (Playwright, s. f).

Una gran ventaja de Playwright es su documentación clara y actualizada. Además, soporta funciones avanzadas como capturas de pantalla, grabación de videos, testing de componentes, pruebas móviles emuladas y pruebas paralelas. Playwright cubre un amplio espectro de pruebas: desde pruebas funcionales (interacción de usuarios con formularios, navegación, flujos de autenticación), pruebas de integración (validación de la comunicación entre frontend y backend) hasta pruebas de regresión visual (comparación de capturas para detectar cambios inesperados en la UI) (Playwright, s. f).

Evidencia ejecución de pruebas: https://youtu.be/vKTm0KbyoG8

Backend – NodeJS

1. **Jest + Supertest**

Tipo: Pruebas unitarias y de integración para APIs REST.

Ventajas:

- Supertest simula peticiones HTTP sin levantar el servidor.
- Jest tiene buenas utilidades para mocks y pruebas asincrónicas.

Ideal para: Testing de endpoints REST y lógica del backend.

2. **Mocha + Chai + Sinon**

Tipo: Pruebas unitarias, funcionales y de integración.

Ventajas:

- Muy flexible y modular.
- Sinon permite espiar, stubear y mockear funciones fácilmente.

- **Ideal para**: Proyectos con estructura personalizada, pruebas controladas.

3. **Pact (Consumer Driven Contracts)**

Tipo: Pruebas de integración mediante contratos entre frontend y backend.

Ventajas:

- Verifica que las APIs cumplan con lo esperado por el frontend.
- Reduce regresiones por cambios no controlados.

Ideal para: Proyectos con comunicación API entre frontend y backend.

Tecnología seleccionada: Jest + Supertest

Porque es uno de los frameworks de pruebas más populares en el ecosistema JavaScript gracias a su velocidad, simplicidad y robustez. Ofrece características integradas como mocks, aserciones, cobertura de código y pruebas en paralelo. Por su parte, Supertest es una librería especializada en pruebas de APIs HTTP, que permite simular peticiones directamente al servidor sin necesidad de levantarlo en un puerto real. Esto mejora el rendimiento de las pruebas y permite realizar pruebas de integración completas (Asmare, 2023).

Con Jest + Supertest se pueden cubrir pruebas unitarias (validación de funciones internas y controladores), pruebas de integración (comprobación de interacciones entre rutas, base de datos y lógica del servidor) y pruebas funcionales (verificar que la API responde como se espera desde la perspectiva del cliente). Esta combinación es eficiente, confiable y adecuada tanto para proyectos pequeños como escalables (Asmare, 2023).

Evidencia ejecución de pruebas: https://youtu.be/fS2HwfbHDFg

2. Plan de pruebas máster

Fronted

1. Introducción
Este plan de pruebas maestro describe el enfoque, alcance, recursos y cronograma de las actividades de prueba destinadas a validar el correcto funcionamiento del sistema de login del frontend. La automatización se realizará usando Playwright.

2. Alcance de las pruebas
El alcance de estas pruebas incluye la validación de la funcionalidad de autenticación del usuario en la interfaz web. Las pruebas se centrarán en los flujos más críticos del login.

3. Herramientas de Prueba

- Playwright: Todas las pruebas (funcionales, de integración y regresión)

4. Casos de Prueba

- Casos de prueba funcionales:
  - Validar login exitoso
  - Mostrar error con campos vacíos
  - Mostrar error con credenciales incorrectas
- Casos de prueba de regresión visual:
  - Captura visual tras error de login
  - Verificar interfaz inicial del login
  - Asegurar consistencia visual post-login
- Casos de prueba de integración:
  - Redirección a /home tras login exitoso
  - Manejo correcto de token de sesión
  - Recuperación del nombre del usuario desde la API

Casos seleccionados para automatización:

1. Mostrar mensajes de error si los campos están vacíos.

2. Mostrar error con credenciales incorrectas.

3. Redirección a /home tras login exitoso.

Backend

1. Introducción

Nos centraremos en validar que el sistema cumple con los requisitos funcionales desde la perspectiva del usuario final, sin analizar la implementación interna del código.

Ya que nuestro enfoque fue de pruebas de validación es de tipo Caja Negra, validamos que el sistema hace lo que debe hacer según lo especificado, además solo interactuamos con inputs y outputs (no nos importa cómo se procesan internamente) y verificamos los casos de uso principales y alternativos.

2. Alcance de las pruebas

El alcance incluye:

- Interacción con base de datos PostgreSQL
- Validación del sistema y flujo completo de autenticación
- Control de acceso basado en roles
- Gestión básica de proyectos

3. Herramientas de Prueba

- Jest: Framework de pruebas con soporte para cobertura de código
- Supertest: Para pruebas de endpoints HTTP
- pg-mock: Para mockear conexiones a PostgreSQL

4. Casos de Prueba

Listado de casos de prueba:

- **Flujo de Cuenta de usuario**:
    - Registro exitoso de nuevo usuario (para realizar las pruebas en sí)
    - Login con credenciales válidas
    - Verificación de token JWT y mantener sesión
    - Logout correcto
- **Proyectos**:
    - Listado de proyectos según rol
    - Validación de estructura de datos
- **Roles**:

- Asignación de roles a usuarios
- Validación de permisos

Casos seleccionados para automatización:

1. Flujo completo de autenticación (AUTH1)
2. Acceso a proyectos según rol (PROJ1)
3. Asignación de roles (ROLE1)

3. Conclusiones

- Las pruebas de caja negra demostraron ser altamente efectivas para verificar que el backend cumple con los flujos críticos definidos.
- Este enfoque permitió identificar problemas de integración entre componentes con el manejo de tokens JWT que podrían pasar desapercibidos en pruebas unitarias. Sin embargo, se evidenció la necesidad de complementar con pruebas de carga y seguridad para cubrir escenarios más complejos.
- Playwright ofrecio un rendimiento es alto: las pruebas corren rápido y de forma estable, incluso en entornos sin interfaz gráfica (headless), ideal para automatización continua.
- Playwright es fácil de entender y configurar

4. Referencias

Fast and reliable end-to-end testing for modern web apps | Playwright. (s. f.). https://playwright.dev/

Global Business IT. (s. f.). ¿Qué son las pruebas de automatización? - Global Business IT. https://gbitcorp.com/blog/posts/qu-son-las-pruebas-de-automatizacin/

Musings, E. (2023, 15 agosto). Overview of integration testing frameworks for React apps. Medium. https://medium.com/@aksharsanjose/overview-of-integration-testing-frameworks-for-react-apps-177fd540d8ae

Asmare, E. (2023, 5 octubre). Node.js Express testing with Jest and SuperTest - Ermias Asmare - Medium. Medium. https://medium.com/@it.ermias.asmare/node-js-express-with-jest-and-supertest-e58aaf4c4514