Departamento de Ciencias de la Computación(DCCO)

Carrera de Ingeniería de Software

Curso de Aplicaciones Distribuidas

Pruebas y Validación

Presentado por: Marley Almeida, Sebastian Bolaños, Nicole Lara, Axel Pullaguari

Tutor: Morales, Dario.

Ciudad: Sangolquí, Ecuador

Fecha: 01/02/2025

Contenido

5. Pruebas y Validación

5.1 Pruebas de Integración y Sistema.

El propósito de este informe es detallar las pruebas de integración entre los microservicios y la capa de presentación en el proyecto, así como las pruebas de sistema ejecutadas para validar el comportamiento general de la aplicación. Estas pruebas son cruciales para garantizar que todos los componentes del sistema interactúan correctamente y que la aplicación funcione como se espera en un entorno de producción.

El objetivo principal de las pruebas es asegurar que:

1. **Integración entre microservicios**: Los microservicios estén correctamente conectados y puedan intercambiar datos de manera efectiva entre sí y con la capa de presentación.
2. **Comportamiento de la aplicación**: La capa de presentación debe comunicarse de manera correcta con los microservicios y responder adecuadamente a las acciones de los usuarios.
3. **Validación del sistema**: Se comprueba que el sistema en su conjunto cumple con los requisitos funcionales y no funcionales del proyecto.

Descripción de la Arquitectura

El sistema está compuesto por varios microservicios que se encargan de tareas específicas, como la gestión de empleados, asistencia, evaluaciones y reportes. La capa de presentación está implementada utilizando React, que interactúa con estos microservicios a través de APIs RESTful.

Pruebas de Integración entre Microservicios y la Capa de Presentación

Las pruebas de integración se llevaron a cabo para asegurar que los microservicios interactúan correctamente con la capa de presentación. Estas pruebas involucran los siguientes pasos:

1. **Prueba de comunicación entre front-end y back-end**: Se realizó un conjunto de pruebas para verificar que las solicitudes HTTP realizadas desde la capa de presentación (React) son correctamente manejadas por los microservicios. Esto incluye la verificación de solicitudes GET, POST, PUT y DELETE.
    - **Ejemplo**: Se probó la creación, edición y eliminación de empleados mediante el microservicio correspondiente. La capa de presentación debe mostrar los datos de manera dinámica y actualizar la UI sin errores.
2. **Prueba de validación de datos**: Se verificó que los datos proporcionados desde la interfaz de usuario se enviaran de forma correcta a los microservicios y que estos datos fueran validados antes de ser procesados o almacenados.
    - **Ejemplo**: Al registrar una asistencia o evaluación, se verificó que los datos de la interfaz de usuario fueran validados adecuadamente antes de ser enviados al microservicio correspondiente.
3. **Prueba de autenticación y autorización**: Se verificó que la capa de presentación maneja correctamente la autenticación y autorización. Se probó que solo los usuarios con roles adecuados pudieran acceder a ciertas rutas y realizar operaciones específicas.

4. **Prueba de manejo de errores**: Se validó que los microservicios gestionan adecuadamente los errores y responden con los códigos de estado HTTP apropiados. La capa de presentación debe manejar estos errores y mostrar mensajes al usuario cuando sea necesario.

Pruebas de Sistema

Las pruebas de sistema se llevaron a cabo para garantizar que la aplicación, como un todo, cumple con los requisitos y comporta correctamente en un entorno real.

1. **Prueba de rendimiento**: Se probó la capacidad de la aplicación para manejar múltiples solicitudes simultáneas, especialmente cuando se gestionan grandes volúmenes de datos (por ejemplo, en la consulta de asistencia y reportes). El objetivo fue verificar que el sistema mantuviera un tiempo de respuesta adecuado bajo condiciones de carga.

2. **Prueba de flujo completo**: Se realizó una prueba que abarca todo el flujo de trabajo del usuario, desde la autenticación hasta la generación de reportes. El objetivo fue asegurar que la aplicación funcione correctamente desde el momento en que un usuario inicia sesión hasta que genera un reporte en PDF de los empleados.

    - **Ejemplo de flujo**: Un usuario inicia sesión, accede a la lista de empleados, registra asistencia para un empleado, luego accede a las evaluaciones y genera un reporte PDF que incluye los datos de asistencia y evaluaciones.

3. **Prueba de UI/UX**: Se verificó la funcionalidad de la interfaz de usuario, asegurándose de que los componentes se muestren correctamente y que los botones y enlaces funcionen como se espera. Se probó la usabilidad en diferentes dispositivos y resoluciones.

4. **Prueba de integración de reportes**: En la funcionalidad de reportes, se verificó que la generación de un reporte en PDF incluya correctamente los datos de los empleados, sus asistencias, evaluaciones y comentarios.

5. **Prueba de compatibilidad**: Se realizaron pruebas en diferentes navegadores y dispositivos para asegurar que la aplicación sea compatible y funcione correctamente en todos los entornos posibles.

Resultados

- **Pruebas de integración**: Todas las pruebas de integración entre los microservicios y la capa de presentación fueron exitosas. Los microservicios respondieron adecuadamente a las solicitudes de la interfaz de usuario, y la capa de presentación manejó correctamente los datos enviados y recibidos.

- **Pruebas de sistema**: La aplicación pasó todas las pruebas de sistema. El flujo completo de trabajo del usuario funcionó sin problemas, y la aplicación fue capaz de manejar múltiples usuarios y generar reportes correctamente. La interfaz de usuario fue intuitiva y fácil de usar.

- **Mejoras y optimizaciones**: Aunque la aplicación funcionó correctamente en todas las pruebas, se identificaron algunas áreas para optimización, como la mejora de la velocidad de generación de reportes y la gestión de errores en ciertos casos extremos.

5.2 Implementación de OAuth 2.0 en el Proyecto

En este sistema basado en microservicios, hemos implementado OAuth 2.0 para gestionar la autenticación de usuarios y proteger las rutas sensibles, como la gestión de empleados, asistencia, evaluaciones, y reportes.

1. **Autenticación:** El proceso de autenticación comienza cuando un usuario intenta acceder al sistema. Si el usuario no está autenticado, es redirigido a la página de login. En esta página, el sistema valida las credenciales del usuario contra un servidor de autorización, que emite un **token de acceso** (access token) tras una autenticación exitosa.

2. **Autorización:** Una vez que el usuario está autenticado, el sistema utiliza el token de acceso para acceder a los recursos del backend. El token de acceso es enviado en las cabeceras HTTP de cada solicitud. Las rutas protegidas (como /home, /employees, /attendance, etc.) solo pueden ser accesadas si el usuario tiene un token de acceso válido.

3. **Flujo de OAuth 2.0:** El flujo típico de OAuth 2.0 en este proyecto sigue los siguientes pasos:
    - El usuario ingresa sus credenciales en la página de login.
    - El servidor de autorización verifica las credenciales y devuelve un token de acceso.
    - El cliente (la aplicación frontend) utiliza este token para realizar solicitudes a las APIs protegidas.
    - Si el token es inválido o ha expirado, el cliente redirige al usuario a la página de login para reautenticarse.

4. **Seguridad:**
    - **Tokens de acceso:** Los tokens son guardados de forma segura en el navegador del cliente (en el almacenamiento local o en las cookies), y son enviados en las cabeceras de cada solicitud.
    - **Expiración de tokens:** Los tokens tienen una fecha de expiración que obliga al usuario a volver a autenticarse después de cierto tiempo. Esto mejora la seguridad al limitar el tiempo de validez de los tokens.
    - **Refrescar tokens:** Si un token expira, el sistema ofrece un mecanismo de refresco de tokens (usualmente mediante un **refresh token**), lo que permite que el usuario continúe utilizando la aplicación sin necesidad de autenticarse nuevamente.

Beneficios de OAuth 2.0

- **Seguridad mejorada:** Al evitar el almacenamiento de credenciales del usuario en el cliente, OAuth 2.0 ofrece una capa adicional de seguridad.
- **Experiencia de usuario fluida:** Los usuarios pueden iniciar sesión una vez y mantener su sesión activa sin tener que introducir sus credenciales constantemente.
- **Escalabilidad:** OAuth 2.0 permite integrar fácilmente servicios de terceros (por ejemplo, Google, Facebook) si en el futuro se desea ofrecer opciones de login externas.