Problemas encontrados y soluciones

Durante el desarrollo de este proyecto, se han encontrado varios problemas. Este capítulo proporciona una visión general de estos problemas y cómo se resolvieron.

Problemas de compilación relacionados con OneDrive

Para facilitar el trabajo entre varios dispositivos y para mantener una copia de seguridad adicional, al principio la carpeta padre del proyecto se encontraba en OneDrive. Sin embargo, el nombre de algunos ficheros generados durante la compilación que contenían caracteres extraños como '$' o nombres demasiado largos provocaban que los cambios no se escribieran en el disco porque OneDrive está integrado como parte de Windows, y errores de sincronización de OneDrive pueden provocar problemas a nivel del manejo por parte del sistema operativo de esos ficheros.

No era algo esperado y los problemas eran extraños por lo que a pesar de ser algo aparentemente menor encontrar la causa del problema fue bastante complicado.

Solución: Mover la carpeta del proyecto fuera de OneDrive permitió resolver el problema y la compilación de cambios pudo continuar como se esperaba.

Cambios en la API de Google

Al comienzo del proyecto se intentó realizar el flujo de autenticación con Google de una forma distinta, consistía en utilizar un flujo propio de Google para el que había muchas librerías ya adaptadas y una gran cantidad de documentación, sin embargo esa forma de autenticarse contra los servicios de Google quedaba obsoleta a partir de marzo de 2023, por lo que cuando fui consciente de esto, a mitad del desarrollo de la autenticación usando el sistema antiguo tuve que cambiar al flujo nuevo basado en OAuth.

Esto además supuso un problema debido a que la propia documentación de Google seguía haciendo referencia a la forma antigua en algunos documentos y a la nueva en otros. Y la documentación de la comunidad y librerías era escasa, debido a que el cambio se había anunciado y la nueva metodología ya estaba disponible pero no era obligatorio el cambio hasta marzo.

Solución: Se realizó una investigación de como funciona OAuth2 y de la nueva forma de trabajar de Google, en algunos casos adaptando ejemplos de la documentación oficial pensados para el flujo antiguo al flujo nuevo.

Imposibilidad de usar el flujo de Micronaut para oauth

Relacionado con el problema anterior a la hora de implementar el nuevo flujo había que pedir dos tipos de token, uno genérico que contiene información básica del perfil del usuario y otro que permite obtener a su vez otra serie de tokens que permiten hacer llamadas a la API de Google Calendar y en el que a la hora de pedirlo especificas los permisos que vas a requerir.

Este segundo tipo de token requiere que sepas el código OAuth de los permisos que requieras y ha de ser pedido por el frontend, para obtener un idToken que luego el backend pueda intercambiar por los tokens de acceso y refresco. Este proceso en Micronaut en teoría puede ser automatizado de forma relativamente sencilla, pero la necesidad de usar la librería de Google para Java hacía imposible utilizar el flujo estandarizado de Micronaut ya que es la librería de Google la que gestiona los tokens en base de datos.

Solución: Se decidió hacer una implementación manual del flujo OAuth en el backend usando la interfaz AuthenticationProvider de Micronaut para interceptar las peticiones a endpoints securizados y usar la librería de Google de Java dentro de ese "interceptor" para validar y refrescar los tokens necesarios.

Falta de documentación oficial para la obtención de tokens en el backend

Si bien ya se ha mencionado que existe una librería de Google para Java que facilita la gestión de estos tokens en el backend lo cierto es que esta librería no dispone de una documentación adecuada. En el momento del desarrollo no existían ejemplos oficiales de ningún proceso. Por ejemplo, para poder obtener tokens de acceso y refresco en OAuth hay que usar un proceso denominado explícito, distinto del implícito que es un procedimiento simplificado para aplicaciones que no persisten datos, este procedimiento explicito requiere que se le pase en la llamada del backend para intercambiar el idToken por los tokens de acceso y refresco una uri a la que la API de Google responderá.

En la documentación explicativa de Google no se menciona la posibilidad de pasar el valor "offline" como accessType que permite obtener los tokens sin dar una uri de redireccionamiento como contestación a la propia petición de la librería. Sí está documentado en el código.

Solución: Estudiar el funcionamiento genérico de OAuth 2.0 e inferir el funcionamiento del flujo dentro de la librería de Google de Java y de la de Javascript con la ayuda de los comentarios disponibles en el código para poder avanzar con el desarrollo.

CORS

Tuve problemas relacionados con la configuración de Cross-Origin Resource Sharing (CORS) entre el backend y el frontend.

Solución: Estos problemas se resolvieron configurando correctamente el CORS en el backend del servidor y utilizando la librería Axios en el frontend para realizar las llamadas al backend en lugar de la librería nativa de JavaScript (fetch).

Reactive Streams y la biblioteca de Google

En un primer momento estaba planificado el uso de programación reactiva en el backend, pero la biblioteca de Google no está diseñada de forma reactiva ni lo soporta de forma nativa, por lo que la finalidad de usar programación reactiva que era hacer un código más compacto pero legible no iba a poder conseguirse porque el código se complicaba demasiado al tener que hacer envoltorios reactivos para la librería de Google.

Solución: Se desecho la idea porque consumía demasiado tiempo y empeoraba la legibilidad del código.

Problemas con WSL

Existen problemas con Windows Subsystem for Linux (WSL) y su comportamiento después de la hibernació. Estos problemas impiden utilizarlo con normalidad cuando el equipo sale de estado de hibernación, impidiendo el uso de Docker basado en WSL.

Solución: A través de la investigación en línea, se descubrió que este es un problema conocido con WSL y se mitigó evitando poner el sistema en estado de hibernación.

Problemas entre Azure, Gradle y Micronaut

La combinación de Azure App Services con Gradle y Micronaut no es nada común. Existe gran cantidad de documentación para el caso de uso de aplicaciones desarrolladas con Spring Boot y Maven que son desplegadas en Azure pero no así para Gradle y Micronaut.

Micronaut utiliza el plugin shadowJar de Gradle para crear el fat jar de la aplicación aunque en un primer momento cuando se observa la configuración parece que se utiliza el plugin de Java para eso. Por defecto este plugin nombra el jar con una combinación de nombre de aplicación y versión, pero Azure solo reconoce los .jar que se nombran como app.jar por lo que el despliegue fallaba porque Azure no era capaz de encontrar el .jar de la aplicación.

Solución: Tras investigar las causas encontré un post de un desarrollador que había tenido este mismo problema en 2019 y después de leerlo configuré adecuadamente el plugin shadowJar para que genere un fichero app.jar allí donde Azure espera encontrarlo.

LEVANTAMIENTO INEFICIENTE DE CONTENEDORES DURANTE LAS PRUEBAS

Una característica importante que me motivo a escoger Micronaut como framework para el backend fue que la rapidez con la que inicia permite hacer tests de integración con la aplicación al completo levantada sin que eso consuma mucho tiempo. Como parte de la batería de pruebas hay tests de integración que acceden a una base de datos que se levanta de manera automática gracias a TestContainers.

El problema que enfrenté es que se levantaba una base de datos por cada clase de prueba, y aunque no era un problema grave ya que se levantan bastante rápido en mi equipo pueden no hacerlo en otro y además no es algo que me gustara porque era poco eficiente y consumía tiempo de CI/CD.

Solución: Decidí usar la posibilidad que ofrece Junit de establecer métodos que se ejecuten antes y después de que comience la batería de tests mediante la interfaz *TestExecutionListener*, además mediante la interfaz *PropertySourceLoader* de Micronaut puedo inyectar directamente las propiedades de la base de datos levantada en la configuración de Micronaut.

Para poder ejecutar todo esto antes de la carga de Micronaut y Junit uso el Service Locator de Java que me permite instanciar la clase necesaria para levantar Docker, marcada además con la palabra reservada '*volatile'* para asegurar que las dos instancias que se crean (una para micronaut y otra para Junit) puedan acceder de forma sincronizada a el estado de la clase.

CONCLUSIONES

A lo largo del proyecto se ha realizado un ejercicio no solo de desarrollo si no de planificación, investigación de tecnologías nuevas o poco exploradas por mí como los framework Micronaut y React, la autenticación con servicios de terceros o el montaje de un proceso de integración y despliegue continuo en la nube. Además, se han mantenido reuniones periódicas y se ha escuchado feedback de manera regular por parte de un stakeholder, el representante de HP que ha propuesto este TFG.

Todo este ejercicio de trabajo e investigación ha derivado en un sistema robusto, modular, preparado para la ampliación con nuevos proveedores de calendario y que ha satisfecho las expectativas de los stakeholders. Un sistema compuesto por un backend escrito en Java con Micronaut, un frontend escrito en JavaScript y React y un sistema de base de datos en la nube con un sistema de integración, testeo y despliegue automatizado.

A modo de reflexión me gustaría destacar que por el camino me ha permitido enfrentarme a un proyecto de manera autónoma y aprender sobre nuevas tecnologías y formas de trabajo que me generaban

curiosidad como es el uso de recursos en la nube, el CI/CD o el uso de Docker para levantar contendores de prueba de manera automática.

Trabajo futuro

A lo largo del desarrollo de NoMoreMeetings se han identificado varias áreas de mejora y características adicionales que podrían implementarse en el futuro para mejorar aún más la eficacia y la utilidad de la aplicación. Algunas de ellas como la conectividad con Exchange ya se han previsto en la fase de diseño y han influido en la toma de decisiones como ya se ha explicado. A continuación, se detallan algunas de las posibles características que podrían desarrollarse en futuras versiones de la aplicación:

Reuniones Repetitivas

Una funcionalidad importante sería que la aplicación pudiera identificar y tener en cuenta las reuniones que se producen con asiduidad. Esta característica aumentaría la precisión del análisis del tiempo de reunión y proporcionaría una representación más realista de la forma en que los usuarios gastan su tiempo ya que permitiría distinguir estas reuniones en la página de resultados de análisis.

Conexión con Exchange

Actualmente, NoMoreMeetings es compatible solo con Google Calendar. Sin embargo, Microsoft Exchange también es ampliamente utilizado en el entorno empresarial. Por lo tanto, sería beneficioso para muchos usuarios si la aplicación pudiera conectarse con Exchange.

Exportación de Informes

Por último, aunque la aplicación proporciona un análisis detallado del tiempo de reunión, sería útil poder exportar estos informes en un formato normalizado, como CSV o JSON. Esto permitiría a los usuarios compartir fácilmente sus informes con otros o utilizar los datos en otras aplicaciones o herramientas.

Ver Informes Anteriores

Otra posible ampliación es la capacidad de acceder y visualizar informes de análisis de tiempo de reunión generados previamente. En la versión actual de NoMoreMeetings, los usuarios solo pueden ver el informe más reciente generado. Sin embargo, el acceso a informes anteriores permitiría a los usuarios hacer un seguimiento de su tiempo de reunión a lo largo del tiempo y detectar patrones o tendencias.

Esta característica implicaría almacenar los informes generados de forma segura y proporcionar una interfaz de usuario para que los usuarios puedan acceder a ellos. La implementación de esta funcionalidad permitiría a los usuarios no solo tener una instantánea de su tiempo de reunión actual, sino también una visión más amplia y una perspectiva a lo largo del tiempo.