ARCHITECTURE EVOLUTION: INTRODUCCIÓN

En esta charla Randy Shoup, vicepresidente de ingenieros y arquitecto jefe en eBay, nos viene a hablar sobre cómo fue evolucionando la infraestructura de eBay desde sus inicios. Trata temas sobre como cambió la arquitectura de eBay pasando por monolitos a aplicaciones independientes o como fueron evolucionando las bases de datos a lo largo del tiempo. También nos habla de cómo se trabaja en la actualidad en eBay.

Ahora pasaremos a comentar cada una de las 3 iteraciones (versiones) que comenta Randy en el podcast.

PRIMERA ITERACIÓN

En la primera iteración el fundador de eBay, Pierre Omidyar, hizo la página con un back-end de Pearl y cada producto era un fichero que almacenaba en su ordenador con un Intel 486. Esto no era muy escalable y no estaba hecho para que lo fuese.

SEGUNDA ITERACIÓN

En la segunda iteración se pasó a tener un monolito en C++, un ISAPI DLL. Pero esto hizo que solo hubiese un archivo de 3.4 millones de líneas de código que hacía que los cambios fuesen mucho más complejos. Tenían dos bases de datos enormes: la principal y la de backup. Ambas máquinas de Oracle. Con el paso del tiempo esta base de datos principal se dividió en bases de datos específicas para un dominio o entidades (user IDs, item ID, etc).

TERCERA ITERACIÓN

La transición a esta tercera iteración tardo en ser completada un tiempo aproximado de 5 años.

En esta iteración se pasó a usar Java y se movió el código a muchos ficheros EAR, que son archivos que contienen las librerías, beans y archivos JAR que se requieren para el deploy de la aplicación. Se tenía alrededor de 220 ficheros de esos en aquella época. Muchas de estas aplicaciones se relacionaban con otras en relaciones muchos a muchos.

Cada dos semanas la aplicación se mejoraba y los 220 archivos se tenían que actualizar por sus interdependencias. Para poder hacer las mejoras había que hacerlas en un cierto orden por estas dependencias.

BASES DE DATOS

Las bases de datos se usaban tanto para la V2 y V3, después de la partición usada en la iteración 1 (monolito). Cuando se probaba la aplicación en su versión 3 lo que se hacía era usar la base de datos que se usaba para la V2.

DIFERENCIAS ENTRE MONOLITOS Y LAS SIGUIENTES VERSIONES

Una de las ventajas de la segunda y tercera iteración era la independencia de los diferentes equipos de desarrollo, permitiendo que se pudiese avanzar de forma eficiente. Una de las desventajas por ejemplo es que si había un fallo en la aplicación no se sabía de donde provenía el error, ya que había una gran cantidad de aplicaciones y no se sabía de donde provenía la ejecución.

Para manejar las diferentes llamadas a la base de datos lo que desarrollaron fue un log de la base de datos para almacenar la información de las diferentes peticiones, creando así un su propio sistema de trazas.

CLOUD SERVICE

eBay es tan eficiente en cuanto a costes y beneficios que se pueden permitir tener data centers propios y controlar todo de forma interna sin tener que depender de ningún proveedor de servicios en la nube. Esto como dice el propio Randy para muchas empresas que recién empiezan no es rentable, ya que es mucho coste y trabajo montar tu propia nube y servidores.

CAMBIOS ARQUITECTURA

La evolución de la arquitectura del software ha sido grande y esto se ha visto reflejado en distintos avances. Podemos encontrar entre estos avances la evolución desde el monolito, que consiste en tener un solo bloque de código que contiene todas las funcionalidades de una aplicación y que puede resultar problemático a medida que la aplicación crece y se vuelve más compleja lo que hace que los cambios en una parte del código pueden tener un impacto en todo el sistema, hasta los microservicios, que separan la aplicación en servicios más pequeños, cada uno con su propia responsabilidad y funcionalidad específica permitiendo por ejemplo que los equipos de desarrollo puedan trabajar de manera más independiente mejorando la velocidad de desarrollo y que cada servicio sea el propietario de sus datos comunicándose con otros servicios a través de interfaces bien definidas.

CAMBIOS DERIVADOS (BBDD)

La evolución en la arquitectura del software afectó a otros aspectos como es el de la fragmentación de las BBDD. Con el monolito las BBDD eran grandes bloques de información y centralizadas. Con la evolución hacia los microservicios se promovió la fragmentación de las bases de datos, en la que cada microservicio puede tener su propia base de datos. Es decir, la fragmentación de las bases de datos, aunque en algunas empresas ya existía, es una consecuencia de la evolución hacia los microservicios, y proporciona distintos beneficios en comparación con las grandes BBDD de los monolitos.

LIBRO ACCELERATE

Este libro posee una gran información sobre cómo las empresas pueden mejorar su rendimiento y la calidad software a través de la implementación de prácticas DevOps y la colaboración entre equipos. Además, en él se proporcionan una gran cantidad de datos y estadísticas para respaldar sus afirmaciones y ofrece consejos prácticos para implementar estos cambios. Una de estas estadísticas por ejemplo son las 4 técnicas para medir la eficacia, que son frecuencia de despliegue, tiempo de espera para el cambio, tasa de fracaso del cambio y tiempo medio de restauración. Otra puede ser la clasificación por rendimiento que depende de en que unidades estes midiendo la eficacia, es decir si mides en meses es un nivel bajo, en semanas medio, en días alto y en horas está la élite.

TIEMPOS DE DESARROLLO

Muchas veces los tiempos de desarrollo eran muy altos, de manera que necesitaban bajarlos. Para ello hicieron varias cosas, la primera, hablar con los equipos para ver como podían acortarlos, en el caso de un equipo era el “build time” lo que los retrasaba. Otro equipo era la frecuencia de “code reviews y pull requests”. Para descubrir y mejorar estos aspectos también invirtieron en Traffic Mirroring (técnica que consiste en comparar el comportamiento de la aplicación con el código sin modificar y modificado cuando se

hacen cambios que no modifican el comportamiento de la misma). Para mejorar los tiempos de desarrollo actualmente se deben hacer pull requests y deploys con mayor frecuencia.

SCRUM MEETINGS

Una de las cosas importantes que se comentan es hacer reuniones scrum una vez a la semana con los equipos involucrados para recibir ayuda, ver que se ha hecho y que se hará en la semana. Básicamente mejora la comunicación y facilita la detección de problemas, también permite encontrar herramientas y servicios de otros equipos que se pueden usar por otros.

LEGACY SERVICES Y CLUSTERS

Otro tema importante del que se habla en el podcast es el uso de servicios y clusters hechos hace 15/20 años, que ya estaban en la V2 o V3 y que todavía siguen en el código de la aplicación actual. Estos se deberían migrar para adaptarse a las necesidades actuales. Sin embargo, no es una tarea fácil porque si todavía están es porque son muy importantes, perteneces al core de la aplicación, son difíciles de migrar e incluso pueden darles miedo a los desarrolladores cambiarlo.

REFERENCIAS

Shoup, R. (2022, agosto 22). Randy Shoup on Evolving Architecture and Organization at eBay. https://www.se-radio.net/2022/08/episode-525-randy-shoup-on-evolving-architecture-and-organization-at-ebay/

Martin Fowler. Microservices. https://martinfowler.com/articles/microservices.html

Accelerate by Nicole Forsgren. https://itrevolution.com/product/accelerate/

GRUPO ES1-09

Jorge Casatejada Santamarta - UO282294

Jonathan Arias Busto- UO283586

Alejandro Campa Martínez- UO282874