Arquitectura de software monolítica vs. microservicios

Elegir el diseño adecuado para el desarrollo de tu app

Willem L. Middelkoop
Mar. 3, 2020

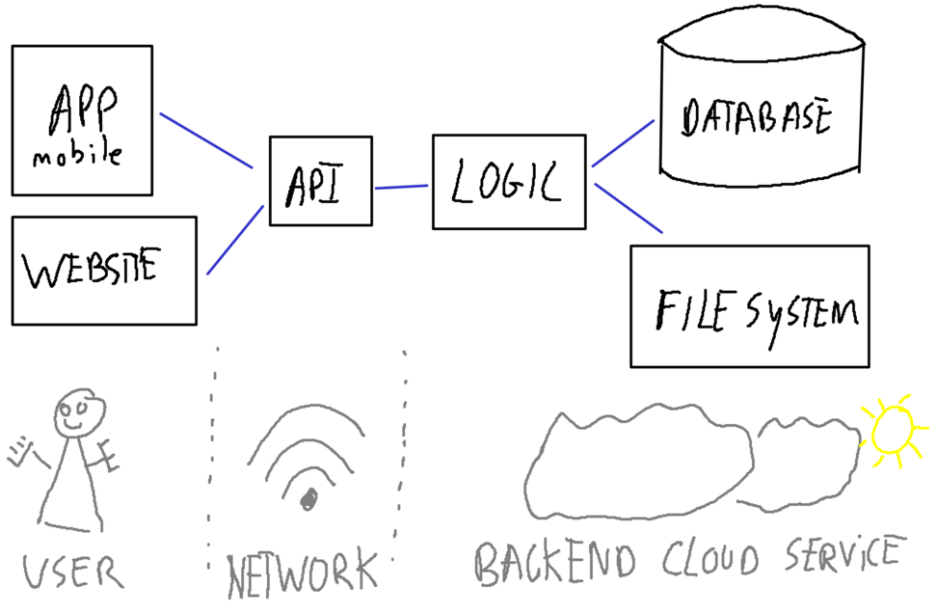
Esta semana volé a Gotemburgo para reunirme con gente de una gran compañía naviera internacional, para hablar sobre el desarrollo de software a nivel empresarial. Durante la reunión había varios expertos en la sala, uno de ellos me preguntó sobre la elección de la arquitectura de software correcta (para aplicaciones grandes, complejas, a nivel empresarial). Una muy buena pregunta, que bien merece una entrada de blog.

Visita a una gran empresa naviera internacional para hablar de software empresarial

Arquitectura de software

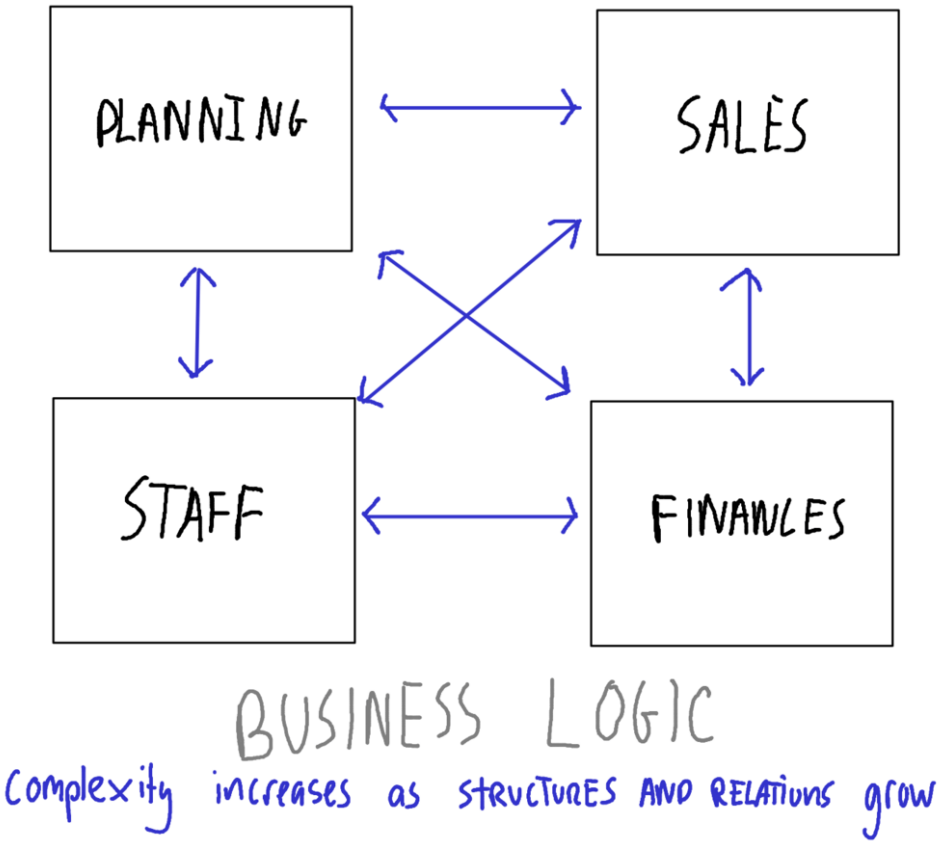
Al diseñar aplicaciones, no solo debe pensar en la mera apariencia, un buen diseño también abarca cómo funciona todo en conjunto. La arquitectura de software se refiere a las estructuras fundamentales de un sistema de software, cada estructura comprende elementos, relaciones entre ellos y propiedades tanto de los elementos como de sus relaciones.

Es análogo a la arquitectura de un edificio. Una vez que se define e implementa una arquitectura, resulta costoso cambiarla después. Por lo tanto, es una buena idea elegir la arquitectura de software correcta *antes* de comenzar a construir.



Ejemplo simple de arquitectura de software: diferentes estructuras que componen un sistema con distintas relaciones entre ellas

Para aplicaciones simples, la arquitectura del software podría ser obvia, pero a medida que su aplicación crece en tamaño y funcionalidad, la cantidad de estructuras en un sistema puede crecer rápidamente. Esto puede suceder con el tiempo, a medida que crecen las necesidades de su cliente. Cuantas más estructuras, más relaciones entre ellas, más complejo se vuelve.



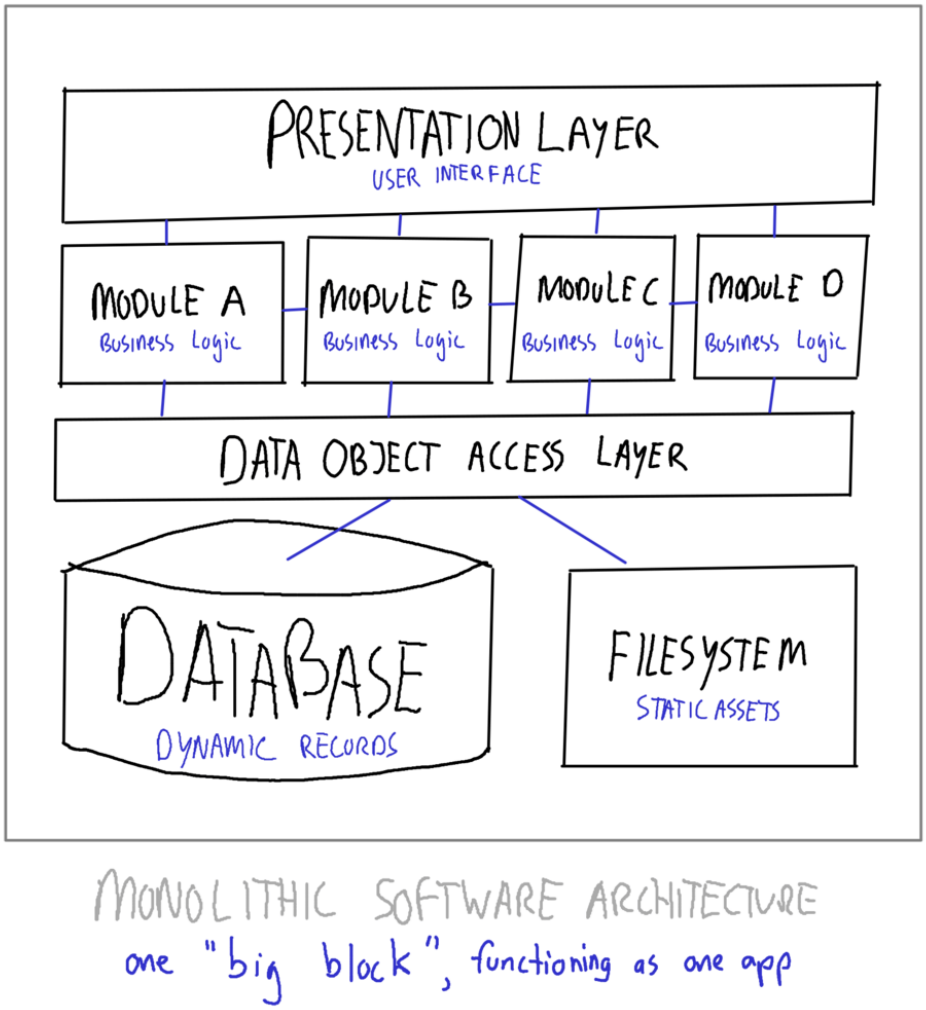
Mayor complejidad a medida que crece el número de estructuras (y sus relaciones)

Lidiando con el diseño de software complejo

Si espera que su software maneje muchas cosas diferentes, es probable que tenga que lidiar con la complejidad. Desde un punto de vista arquitectónico, existen diferentes enfoques para lidiar con esta complejidad del software.

Arquitectura monolítica

La arquitectura monolítica se describe mejor como ”un gran bloque”. Aquí, todo lo que hace su aplicación es parte de la misma base de código.



Arquitectura de software monolítica

En una ”arquitectura monolítica”, el software a menudo está en capas, donde cada capa consta de diferentes tipos de componentes:

- **capa de presentación**: responsable de la interfaz de usuario y/o la interfaz de programación de aplicaciones (UI / API).
- **lógica de negocios**: aquí es donde se definen las reglas de negocio reales. Es el núcleo de la aplicación que se adapta muy bien a los procesos comerciales reales.
- **capa de acceso a objetos de datos**: proporciona una interfaz común entre la aplicación en ejecución y los almacenes backend persistentes.
- **base de datos y sistema de archivos**: aquí es donde se almacenan todos los datos y activos estáticos (archivos, imágenes, etc.).

Cuanto más grande se vuelve la aplicación, más grande se vuelve cada capa. Esto dificulta la escalabilidad de una arquitectura monolítica, en cierto punto las cosas se vuelven simplemente demasiado grandes y complejas.

Cada vez que realiza un cambio en la aplicación, toda la base de código se ve afectada. Para aplicaciones (muy) grandes, esto dificulta el mantenimiento del software, ya que requiere una comprensión completa de toda la aplicación.

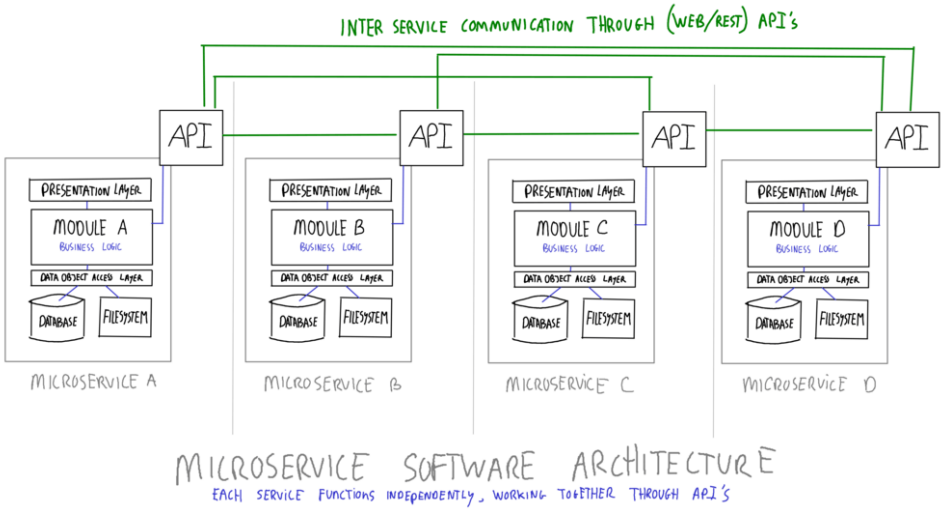
Como toda la aplicación es interdependiente, los problemas en una parte de la aplicación a menudo afectan significativamente a todo el sistema. La comunicación entre las

diferentes partes del sistema se realiza internamente, a nivel de código. Esto significa que las diferentes partes deben programarse en el mismo lenguaje de programación (o compatible). Esto es una barrera para adoptar nuevas tecnologías (en el futuro), ya que no puede cambiar fácilmente el lenguaje de programación de una parte del sistema.

Arquitectura de microservicios

En lugar de construir una gran aplicación monolítica, puede dividir su aplicación en servicios interconectados más pequeños. Cada microservicio es una pequeña aplicación en sí misma, completa con su propia arquitectura (pero más pequeña + simple), lógica de negocios y capas de datos.

Los diferentes servicios se comunican entre sí utilizando API que se definen independientemente de la tecnología utilizada, a menudo en una "estructura de API" común como REST, JSON, Redis y XML. Esto permite que la combinación de microservicios funcione como "un sistema".



Arquitectura de software de microservicios

Al descomponer la aplicación de "bloque grande" en servicios más pequeños, se vuelve más fácil comprender las diferentes partes del sistema. Esto facilita el mantenimiento del sistema, ya que permite que cada microservicio se desarrolle de forma independiente. También reduce la barrera para adoptar nuevas tecnologías, ya que es libre de elegir la tecnología que mejor se adapte a un servicio determinado. Una parte del sistema se puede programar en un idioma diferente al de otras partes. Esto hace posible que diferentes equipos (con diferentes conocimientos) trabajen juntos.

El arte y la habilidad de obtener la arquitectura de microservicios correcta provienen de la capacidad de definir correctamente un microservicio. Demasiado amplio o demasiado granular puede destruir la arquitectura. Puede utilizar procesos de negocio, separación jerárquica o de dominio para definir cada microservicio.

El desafío con la arquitectura de microservicios es que agrega complejidad a todo el sistema por el hecho de que requiere un mecanismo de comunicación distribuido. Las diferentes API que se conectan entre sí pueden ser difíciles de probar y depurar. Se vuelve más difícil implementar cambios que abarcan múltiples servicios.

Definir, diseñar e implementar las diferentes API entre servicios requiere un acuerdo

sobre los datos intercambiados, esto puede ser difícil de lograr cuando se pretende compartir estructuras de datos extendidas y no universales.

Supervisar e implementar una arquitectura de microservicios puede ser complejo cuando los servicios crecen en número. Simplemente hay más que administrar, lo que requiere más planificación y coordinación para cada uno de los servicios.

Elegir la arquitectura correcta

Mi difunto padre me dijo una vez que se puede "ganar más con las cosas que no se hacen". Construir aplicaciones complejas es inherentemente difícil, lo mejor que puede hacer es intentar simplificar las cosas simplemente no haciendo todo. Concéntrese en las cosas que realmente importan, haga esas cosas bien.

Si su aplicación es simple, liviana, una arquitectura monolítica es el camino a seguir. El desarrollo, la implementación, el mantenimiento, todo es más fácil cuando las cosas son pequeñas.

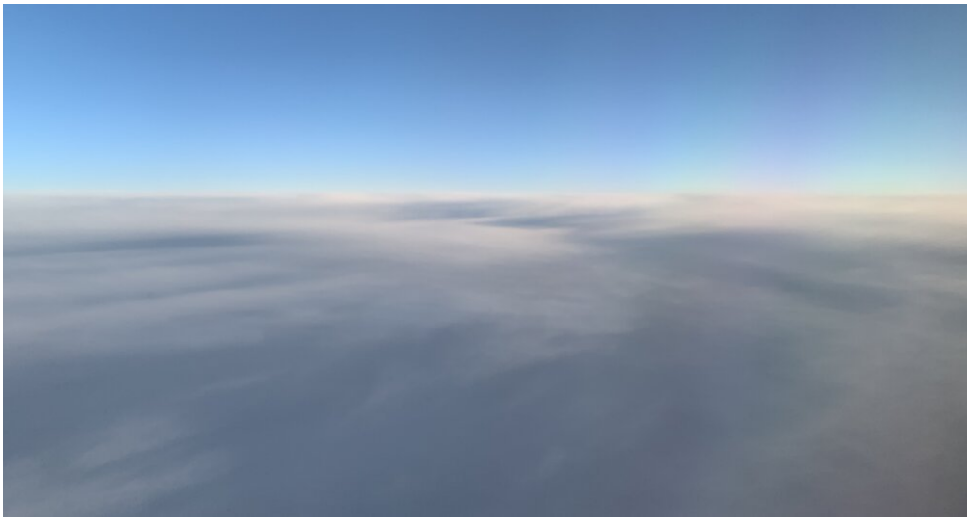
Si no puede escapar de la complejidad, definitivamente debería considerar la arquitectura de microservicios. Hágase las siguientes preguntas:

- ¿Puede definir un "idioma común" entre sus microservicios?
- ¿Existen procesos de negocio claramente distintos?
- ¿Qué tan significativamente son interdependientes estos procesos de negocio?
- ¿Cómo espera que evolucione su aplicación con el tiempo, espera que crezca o se reduzca?
- ¿Cuántas tecnologías de programación diferentes planea utilizar? ¿Espera que esto cambie en el futuro?
- ¿Quién va a construir y mantener el software, tienen experiencia en la gestión de dicha arquitectura?

Conclusión

No hay una respuesta "mágica" a la pregunta de qué arquitectura de software es la correcta. Ambos enfoques tienen sus pros y sus contras.

¡Lo mejor que puede hacer es tomar una decisión consciente, bien pensada, y redoblar la apuesta!



"El cielo es el límite" si dominas la arquitectura que has elegido (¡simplemente no la cambies durante el vuelo!)