SEGUNDA EDICIÓN

DISEÑO DE SISTEMAS

UNA GUÍA DE INFORMACIÓN PRIVILEGIADA

Alex Xu

ANAYA MULTIMEDIA

Índice de contenidos

la fecha de caducidad no sea demasiado corta, ya que eso provocará que el sistema vuelva a cargar los datos de la base con demasiada frecuencia. Asimismo, tampoco es aconsejable que la fecha de caducidad sea demasiado larga, puesto que los datos pueden resultar obsoletos.

- **Consistencia:** Esto implica mantener sincronizado el almacén de datos y la caché. Puede producirse inconsistencia porque las operaciones que modifican datos en el almacén y la caché no estén en una única transacción. Si se amplía a varias regiones, mantener la consistencia entre el almacén de datos y la caché se convierte en un desafío. Para más detalles se puede consultar el artículo "*Scaling Memcache at Facebook*" publicado por Facebook[7].
- **Mitigar los fallos:** Un solo servidor caché representa un posible SPOF (*Single Point Of Failure,* punto único de fallo), definido en la Wikipedia de la siguiente manera: "Un punto único de fallo o punto de fallo único (en inglés, *single point of failure*, abreviado SPOF) es un componente de un sistema que tras un fallo en su funcionamiento ocasiona un fallo global en el sistema completo, dejándolo inoperante"[8]. Por lo tanto, se recomienda que los distintos servidores caché de distintos centros de datos eviten los SPOF. Otro planteamiento recomendado es excederse en la cantidad prevista de memoria requerida en determinados porcentajes, disponiéndose así de un búfer de almacenamiento a medida que el uso de la memoria aumenta.
- **Política de desalojo:** Cuando la caché está llena, las siguientes peticiones de añadir elementos a la caché pueden dar como resultado la eliminación de los elementos existentes. Esto se denomina desalojo de caché. LRU (*Least-recently-used*), o menos utilizado recientemente, es la política de desalojo de caché más conocida. Se pueden igualmente adoptar otras políticas de desalojo, como LFU (*Least Frequently Used,* utilizado con menos frecuencia) o FIFO (*First in First Out,* primero en entrar, primero en salir), para ajustarse a distintos escenarios de uso.

7. "Scaling Memcache at Facebook" de R. Nishtala, artículo publicado dentro del 10º simposio USENIX sobre diseño e implementación de sistemas en red (NSDI '13). https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/nishtala.
8. Punto único de fallo o SPOF (*Single point of failure*): https://es.wikipedia.org/wiki/Punto_%C3%BAnico_de_fallo.

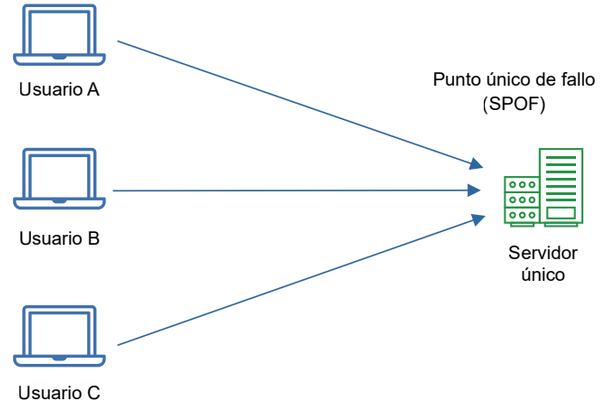


Figura 1.8.

Red de entrega de contenidos o CDN (*Content Delivery Network*)

Una red de entrega de contenidos o CDN (*Content Delivery Network*) es una red de servidores geográficamente dispersados empleada para entregar contenido estático. Los servidores CDN almacenan en la caché contenido estático como imágenes, vídeos, archivos CSS y JavaScript, etc. El almacenamiento en caché de contenido dinámico es un concepto relativamente nuevo, que está más allá del objetivo de este libro. Permite el almacenamiento en caché de páginas HTML que se basan en *request path* (ruta de petición), *query strings* (cadenas de consulta), *cookies* y cabeceras de petición. Si desea más información al respecto, puede consultar el artículo de Amazon[9] mencionado al pie de esta página. Este libro se centra en cómo utilizar CDN para almacenar en caché contenido estático.

Así es como CDN funciona en el nivel alto: cuando un usuario visita una página web, el servidor CDN situado más cerca del usuario entregará contenido estático. Resulta intuitiva la idea de que, cuanto más lejos estén los usuarios de los servidores CDN, más despacio se cargará la web. Por ejemplo, si los servidores CDN están en San Francisco, los usuarios ubicados en Los Ángeles obtendrán contenido más rápido que los usuarios de Europa. La figura 1.9 es un estupendo ejemplo que muestra cómo CDN mejora el tiempo de carga.

9. Entrega de contenido dinámico de Amazon: https://aws.amazon.com/es/cloudfront/dynamic-content/.

3 Estructura para una entrevista de diseño de sistemas

Acaba usted de conseguir una de esas codiciadas entrevistas presenciales en la empresa de sus sueños. El coordinador de contrataciones le envía el programa del día. Revisando la lista todo le parece de maravilla, hasta que sus ojos se encuentran con esta parte: la entrevista de diseño de sistemas. A menudo las entrevistas de diseño de sistemas intimidan bastante. Podría ser algo tan indefinido como "¿Puede diseñar un producto X muy conocido?". Las preguntas son ambiguas y parecen desproporcionadamente generales, así que es fácil desanimarse. Después de todo, ¿cómo podría alguien diseñar en una hora un producto conocido cuya creación ha necesitado de la participación de cientos, si no miles, de ingenieros?

La buena noticia es que nadie espera que haga esto. El diseño de sistemas en la vida real es complicado en extremo. Por ejemplo, una búsqueda en Google puede parecer engañosamente sencilla; sin embargo, la cantidad de tecnología que subyace bajo esa simplicidad es sorprendente. Si nadie espera que diseñe un sistema de verdad en una hora, ¿cuál es el beneficio de una entrevista de diseño de sistemas?

Una entrevista de este tipo simula la resolución de problemas de la vida real, como por ejemplo cuando dos compañeros de trabajo colaboran en un problema ambiguo y acaban con una solución que cumple sus objetivos. El problema es abierto y no existe la respuesta perfecta. El diseño final es menos importante comparado con el trabajo invertido en el proceso de diseño, con lo que puede demostrar sus habilidades de diseño, defender sus decisiones y responder a los comentarios de una forma constructiva. Demos la vuelta a la tortilla y pensemos en lo que pasa por la cabeza del entrevistador cuando entra en la sala de reuniones para encontrarse con usted. Su principal objetivo es evaluar con la máxima precisión posible las

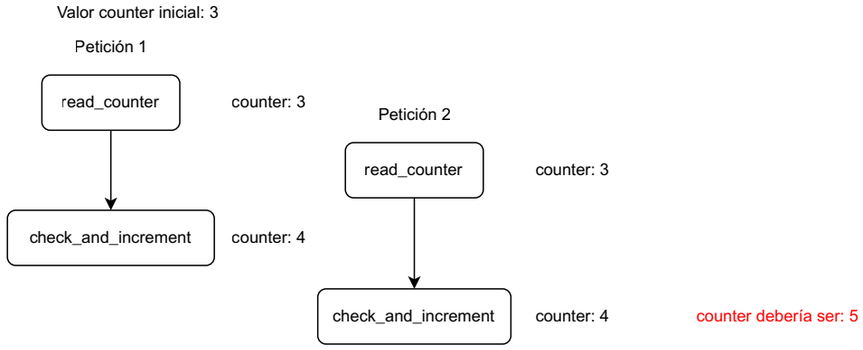


Figura 4.14.

Los bloqueos son la solución más obvia para resolver la condición de carrera. Sin embargo, ralentizan notablemente el sistema. Para resolver el problema se suelen utilizar dos estrategias: código Lua[13] y estructura de datos *sorted sets* (conjuntos ordenados) en Redis[14].

El problema de la sincronización

La sincronización es otro factor importante que tener en cuenta en un entorno distribuido. Para soportar millones de usuarios, un solo servidor limitador de frecuencia podría no ser suficiente para gestionar el tráfico. Cuando se utilizan varios servidores, se necesita sincronización. Por ejemplo, en la parte izquierda de la figura 4.15, el cliente 1 envía peticiones al limitador de frecuencia 1 y el cliente 2 las envía al limitador 2. Como la capa web no tiene estado, los clientes pueden enviar peticiones a un limitador de frecuencia distinto, como se muestra en la parte derecha de la figura 4.15. Si no se produce la sincronización, el limitador 1 no contiene datos sobre el cliente 2, de ahí que el limitador no pueda funcionar correctamente.

Una posible solución es utilizar sesiones *sticky* (persistentes), que permiten a un cliente enviar tráfico al mismo limitador de frecuencia. Pero esta solución no es aconsejable, porque no es ni escalable ni flexible. Es mejor método utilizar almacenes de datos centralizados, como Redis. El diseño se muestra en la figura 4.16.

13. "Scaling your API with rate limiters": https://gist.github.com/ptarjan/e38f45f2dfe601419ca3af937fff574d.
14. "Better Rate Limiting with Redis Sorted Sets": https://engineering.classdojo.com/blog/2015/02/06/rolling-rate-limiter/.

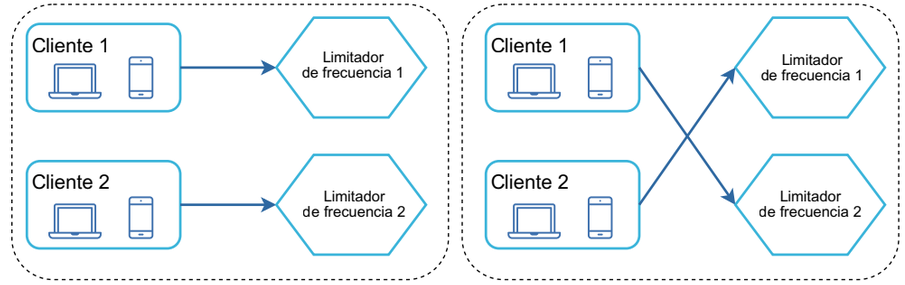


Figura 4.15.

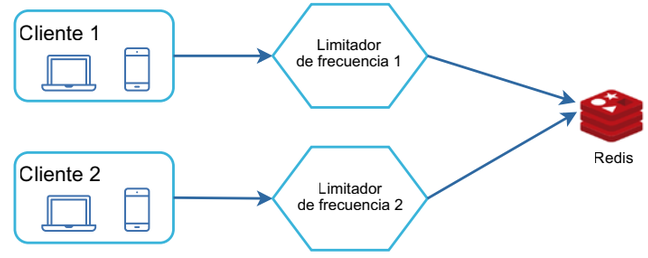


Figura 4.16.

Optimización del rendimiento

La optimización del rendimiento es un tema habitual en las entrevistas de diseño de sistemas. Hablaremos de dos ámbitos que conviene mejorar.

En primer lugar, la configuración de centros multidatos es crucial para un limitador de frecuencia porque la latencia es elevada para usuarios ubicados lejos del centro de datos. La mayoría de los proveedores de servicios de la nube crean muchas ubicaciones de servidor perimetrales por todo el mundo. Por ejemplo, desde el 20 de mayo de 2020, Cloudflare tiene 194 servidores perimetrales geográficamente distribuidos[15]. El tráfico se encamina de forma automática al servidor perimetral más próximo para reducir la latencia.

En segundo lugar, sincronice los datos con un modelo de consistencia provisional. Si no tiene claro esto del modelo de consistencia provisional, consulte la sección "Consistencia" del capítulo 6.

15. Qué es *Edge Computing* o informática de perímetro: https://www.cloudflare.com/es-es/learning/serverless/glossary/what-is-edge-computing/.

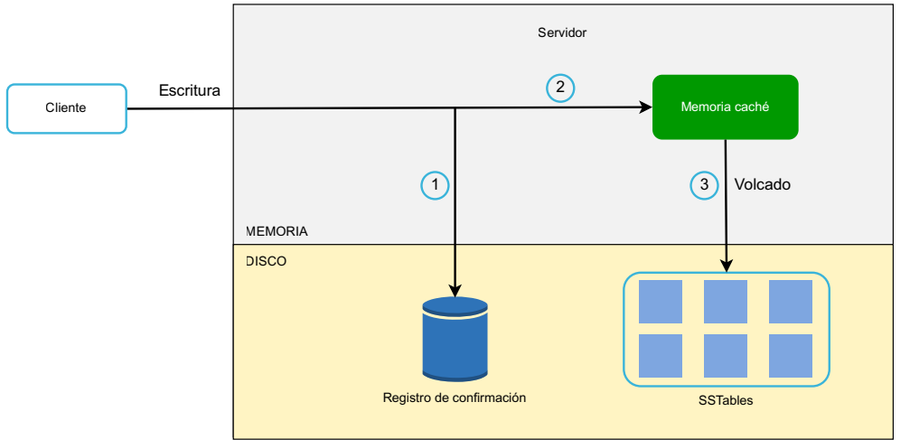


Figura 6.19.

Ruta de lectura

Una vez que una petición de lectura se dirige a un determinado nodo, este comprueba primero que los datos estén en la memoria caché. Si es así, los datos se devuelven al cliente, como muestra la figura 6.20.

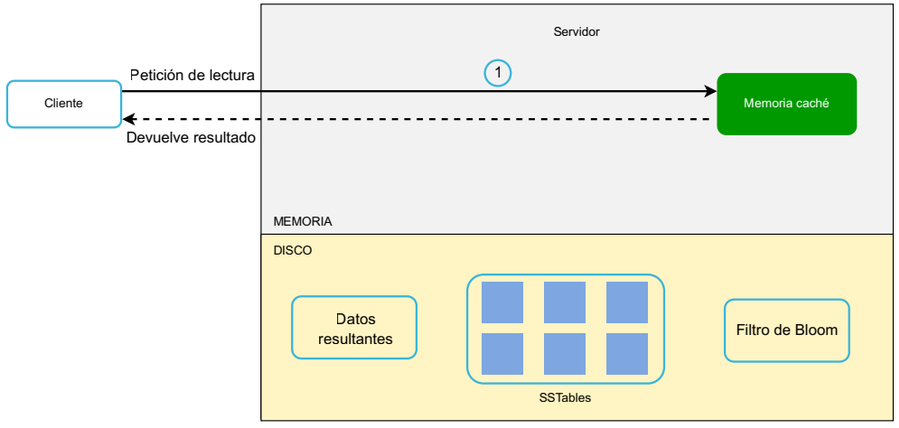


Figura 6.20.

Si los datos no están en la memoria, lo que ocurrirá es que se recuperarán del disco. Necesitamos una manera eficaz de averiguar qué *SSTable* contiene la clave. Para resolver este problema se utiliza el filtro de Bloom[12].

La ruta de lectura se muestra en la figura 6.21 cuando los datos no están en la memoria.

1. El sistema comprueba antes que nada si los datos están en la memoria. Si no lo están, continúa al paso 2.
2. Si los datos no están en la memoria, el sistema comprueba el filtro de Bloom.
3. El filtro de Bloom se utiliza para intentar averiguar qué *SSTables* pueden contener la clave.
4. Las *SSTables* devuelven el resultado del conjunto de datos.
5. El resultado del conjunto de datos se devuelve al cliente.

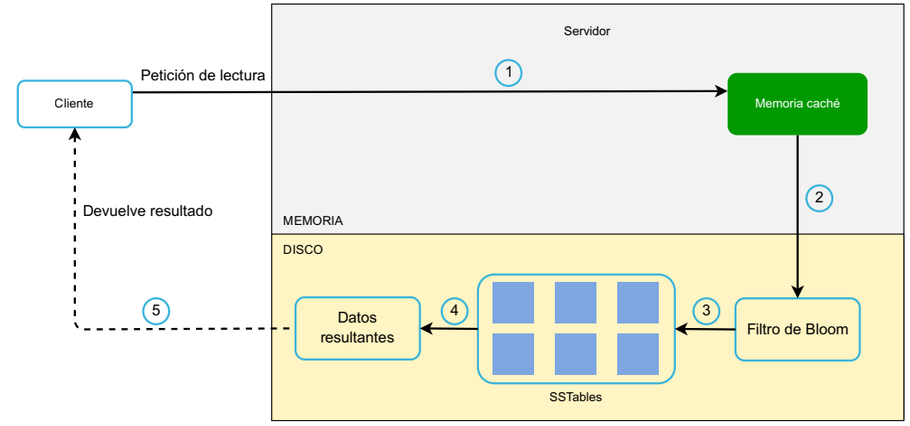
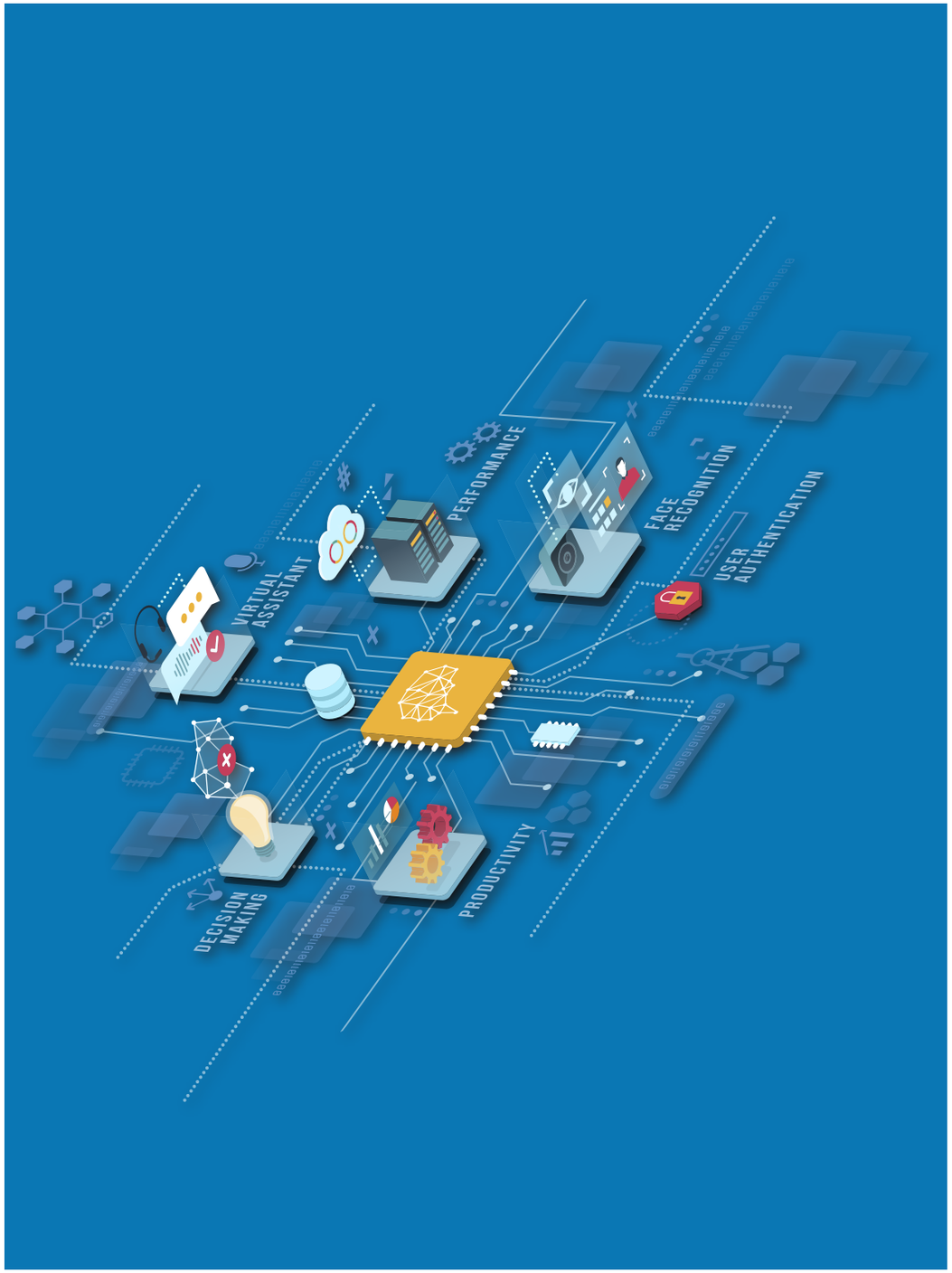


Figura 6.21.

Resumen

Este capítulo abarca muchos conceptos y técnicas. Para refrescarle la memoria, la siguiente tabla resume los aspectos tratados y las correspondientes técnicas empleadas para diseñar un almacén de clave-valor distribuido.

12. Filtro de Bloom: https://es.wikipedia.org/wiki/Filtro_de_Bloom.



9 Diseñar un rastreador web

En este capítulo vamos a centrarnos en el diseño de un rastreador web, una pregunta interesante y bastante habitual en las entrevistas de diseño de sistemas.

Los rastreadores web, también llamados arañas web, son *bots* o robots. Los utilizan los motores de búsqueda para descubrir en la web contenido nuevo o actualizado, por ejemplo una página web, una imagen, un vídeo, un archivo PDF, etc. El rastreador web empieza recopilando páginas web y sigue después los enlaces a dichas páginas para recuperar el contenido nuevo. La figura 9.1 muestra un ejemplo visual del proceso de rastreo.

Una araña web tiene muchos usos diferentes:

- **Indexado de motores de búsqueda:** Es el uso más frecuente. El rastreador recoge páginas web con las que crea un índice local para los motores de búsqueda. Por ejemplo, Googlebot es el rastreador web que está detrás del motor de búsqueda de Google.
- **Archivado web:** Es el proceso de recopilar información de la web con el objetivo de conservar datos para el futuro. Por ejemplo, muchas bibliotecas nacionales manejan rastreadores para archivar sitios web, como la Librería del Congreso de los Estados Unidos[1] y el archivo web de la Unión Europea[2].
- **Minería web:** El explosivo crecimiento de la web presenta una oportunidad sin precedentes para la minería de datos. Esta metodología ayuda a descubrir conocimiento útil dentro de Internet. Por ejemplo, las empresas

1. Librería del Congreso de los Estados Unidos: https://www.loc.gov/web-archives/.
2. Archivo web de la Unión Europea: https://op.europa.eu/es/web/web-tools/euwebarchive.

Flujo de recogida de información de contacto

Para enviar notificaciones es necesario recopilar *tokens* de dispositivo móvil, números de teléfono o direcciones de correo electrónico. Como se muestra en la figura 10.7, cuando un usuario instala nuestra aplicación o se registra por primera vez, los servidores API recogen información de contacto del usuario y la almacenan en la base de datos.

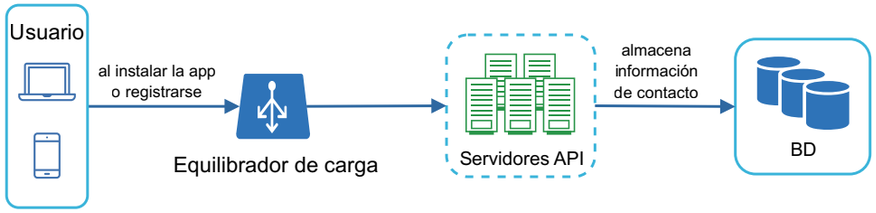


Figura 10.7.

La figura 10.8 muestra tablas de base de datos simplificadas para almacenar información de contacto. Las direcciones de email y los números de teléfono se almacenan en la tabla *user* (usuario), mientras que los *tokens* de dispositivo se guardan en la tabla *device* (dispositivo). Un usuario puede tener varios dispositivos, con lo cual se puede enviar una notificación *push* a todos sus dispositivos.

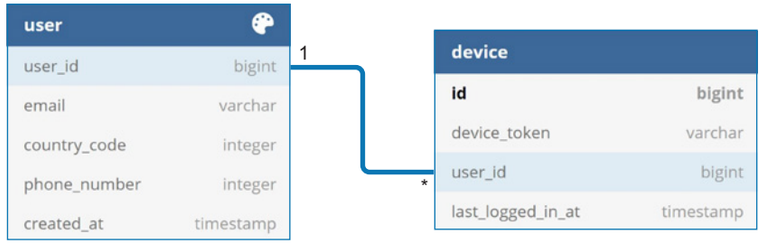


Figura 10.8.

Flujo de envío y recepción de notificaciones

Presentamos a continuación el diseño inicial y después propondremos algunas optimizaciones.

Diseño en el alto nivel

La figura 10.9 muestra el diseño, explicándose cada componente del sistema a continuación.

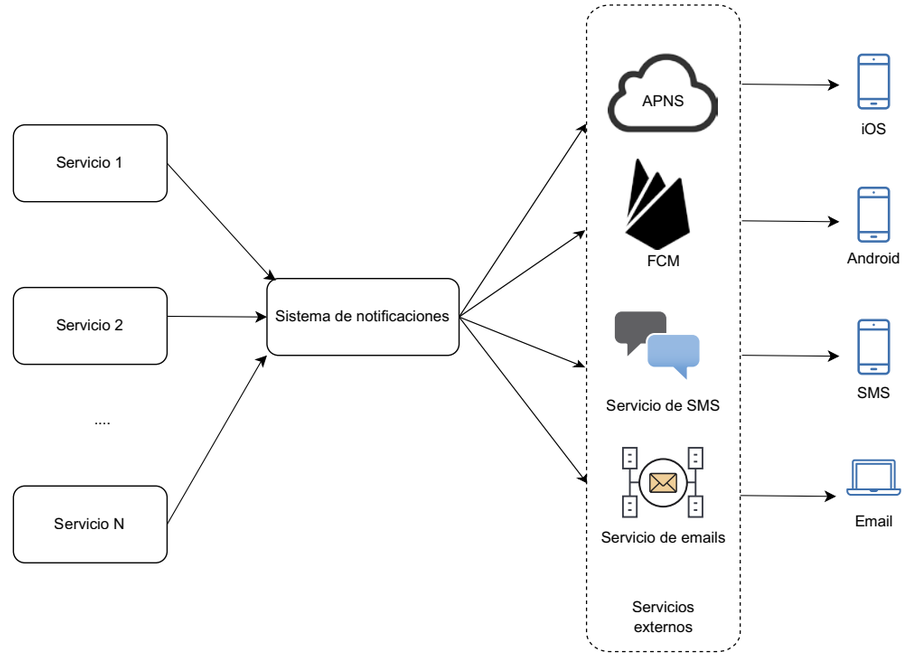


Figura 10.9.

- **Servicio 1 a N:** Puede ser un microservicio, un *cron job* o un sistema distribuido que activa eventos de envío de notificaciones. Por ejemplo, un servicio de facturación envía emails para recordar a los clientes el vencimiento de sus pagos, o un sitio web de compras les dice a sus clientes mediante mensajes SMS que sus paquetes serán entregados mañana.
- **Sistema de notificaciones:** Es la pieza central del envío y recepción de notificaciones. Por empezar por algo sencillo, solo se utiliza un servidor de notificaciones, que proporciona API para los servicios 1 a N y crea cargas útiles de notificación para servicios externos.

13 Diseñar un sistema para completar palabras de forma automática en un motor de búsqueda

Cuando se busca o consulta algo en Google o se va de compras en Amazon, al escribir en el cuadro de búsqueda suelen aparecer una o varias alternativas que pueden coincidir con el término buscado. A esta función se le denomina autocompletar, búsqueda anticipada o búsqueda incremental. La figura 13.1 presenta un ejemplo de una búsqueda en Google que muestra una lista de posibles resultados coincidentes cuando se introduce la palabra "dinero" en el cuadro de búsqueda. Estamos frente a una función importante de muchos productos, lo cual nos lleva a la pregunta de la entrevista: diseñe un sistema para completar palabras de forma automática, también formulada como "diseñe un *top k*" o "diseñe las *k* consultas más buscadas".

Paso 1 – Comprender el problema y establecer el alcance del diseño

El primer paso para enfrentarse a cualquier pregunta en una entrevista de diseño de sistemas es plantearle al entrevistador las cuestiones suficientes que le permitan aclarar los requisitos. Este es un ejemplo de interacción entre candidato y entrevistador:

Candidato: ¿Las coincidencias se soportan solamente al principio de la consulta o también a la mitad?

Entrevistador: Solo al principio de la búsqueda.

Paso 4 – Conclusión

En este capítulo hemos propuesto un diseño para crear Google Drive. La combinación de una elevada consistencia, un bajo ancho de banda de red y una rápida sincronización lo hacen interesante. Nuestro diseño contiene dos flujos: gestión de metadatos de archivo y sincronización de archivos. El servicio de notificaciones es otro componente del sistema de gran importancia. Utiliza sondeo largo para mantener a los clientes actualizados con los cambios de archivos.

Como ocurre con cualquier otra pregunta de una entrevista de diseño de sistemas, no existe la respuesta perfecta. Cada empresa tiene sus limitaciones particulares, por lo tanto el sistema que se diseñe se debe adaptar a ellas. Conocer las concesiones y las opciones tecnológicas ofrecidas por el diseño ideado es también primordial. Si queda tiempo al final de la entrevista, puede hablar de las distintas opciones de diseño.

Por ejemplo, se pueden subir archivos directamente a la nube desde el cliente en lugar de pasar por los servidores de bloques. La ventaja de este método es que acelera la subida de archivos, ya que basta con transferir los archivos una sola vez al almacenamiento en la nube. En nuestro diseño el archivo se transfiere primero a los servidores de bloques y después a la nube. No obstante, esta nueva visión tiene varios inconvenientes:

- Primero, hay que implementar la misma lógica de fragmentación, compresión y codificación en distintas plataformas (iOS, Android, Web). Es propensa a los errores y requiere un gran esfuerzo de ingeniería. En nuestro diseño, todas estas lógicas se llevan a cabo en un único lugar: los servidores de bloques.
- Segundo, como es muy fácil manipular un cliente o acceder a él ilegalmente, implementar lógica de codificación en el lado del cliente no es lo ideal.

Otra evolución del sistema que se puede considerar es llevar la lógica de conexión y desconexión a un servicio distinto. Llamémoslo servicio de presencia. Sacando el servicio de presencia de los servidores de notificaciones, otros servicios pueden integrar fácilmente la funcionalidad de conexión y desconexión.

¡Felicidades por haber llegado tan lejos! Dese una palmadita en la espalda. ¡Buen trabajo!

Esta obra es una fuente de información privilegiada, basada en las preguntas que se realizan en las más difíciles entrevistas técnicas. Los entrevistadores piden a los entrevistados que diseñen una arquitectura para un sistema de software, que puede ser un *news feed* (noticias), una búsqueda en Google, un sistema de chat, etc.

Las empresas adoptan las entrevistas de diseño de sistemas porque las habilidades de comunicación y resolución de problemas que se ponen a prueba en estas entrevistas son similares a las requeridas en el trabajo cotidiano de un ingeniero de software. Son evaluados basándose en el modo en el que cada uno analiza un problema impreciso y en cómo resuelve el problema paso a paso. Las habilidades sometidas a prueba implican además el modo en el que explica la idea, discute con otros y evalúa y optimiza el sistema.

El objetivo de este libro es ofrecer una estrategia fiable y una base sólida de conocimiento para abordar no solo las preguntas de diseño de sistemas en una entrevista laboral, sino para que el ingeniero de software pueda contar con la estrategia correcta para crear un sistema escalable o solucionar problemas en la arquitectura de los sistemas.

Esta obra cuenta con:

- Una estructura en cuatro pasos para resolver cualquier pregunta en una entrevista de diseño de sistemas.
- 16 preguntas realizadas en la vida real en entrevistas de diseño de sistemas con soluciones detalladas.
- 188 diagramas que explican de manera visual el funcionamiento de los distintos sistemas.
- Ejemplos que ilustran un acercamiento sistemático y práctico con pasos detallados.

ANAYA MULTIMEDIA
www.anayamultimedia.es

ISBN 978-84-415-4555-7
2315174
9 788441 545557