Universidad Don Bosco

Ingeniería en Ciencias de la Computación

Desarrollo de Software para Móviles (DSM941)

Foro I

Presentado por

Ardon Martinez, Marlin José	AM120254
David Ernesto Mejía Villalobos	MV191973
Velasco Crespín, Alejandro Ernesto	VC161941
Ramírez Guardado, Ronald Gerardo	RG110604

Fecha de Entrega

29 de octubre de 2023

Índice

Índice 2
Ventajas y desventajas de SQLite 5
Ventajas y desventajas de Firebase 6
Mejor opción para implementar en Android 7
Comparación y conclusiones 8

Historia y concepto de SQLite

SQLite es un sistema de gestión de bases de datos relacionales contenido en una biblioteca C. Fue creado en 2000 por D. Richard Hipp y se ha convertido en uno de los motores de bases de datos más utilizados en el mundo.

La idea original de SQLite era crear un motor de base de datos simple, liviano, autónomo, sin configuración y que no requiriera un proceso/demonio separado, diferente a los grandes sistemas de gestión de bases de datos clientes/servidor. Todo el motor de SQLite está contenido en una relativamente pequeña (~300 KB) biblioteca escrita en C que se vincula directamente en la aplicación.

SQLite utiliza un diseño de procesamiento integrado, lo que significa que el motor de base de datos no es un proceso separado con el que la aplicación se comunique. En su lugar, la biblioteca SQLite se vincula directamente en cada programa y el código SQL se utiliza para comunicarse con la biblioteca logica. Esto simplifica el diseño, reduce la latencia en el acceso a los datos y tiene beneficios de rendimiento.

El diseño simple de SQLite lo hace ideal para su uso en diversos dispositivos y plataformas como teléfonos, navegadores web, sistemas embebidos, etc. Al mismo tiempo, implementa la mayoría de los estándares SQL y tiene características avanzadas como transacciones ACID, claves foráneas, disparadores, vistas, índices, etc. Por estas razones se ha convertido en una de las bases de datos más utilizadas en aplicaciones móviles y web modernas.

SQLite se centra en los siguientes principios:

Arquitectura sin servidor: SQLite almacena la base de datos en un solo archivo, lo que facilita su uso y transporte. No necesita un servidor dedicado, lo que reduce la complejidad y los recursos necesarios.

Transacciones ACID: SQLite garantiza la integridad de los datos a través de transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que lo hace adecuado para aplicaciones que requieren operaciones confiables.

Tipos de datos y consultas SQL completas: SQLite admite tipos de datos comunes y ofrece una amplia gama de características de SQL, lo que le permite manejar consultas complejas y relaciones entre tablas.

Licencia de dominio público: SQLite se distribuye bajo una licencia de dominio público, lo que significa que se puede utilizar en aplicaciones comerciales sin costos de licencia.

Ampliamente compatible: SQLite es compatible con una variedad de lenguajes de programación, lo que lo hace una opción versátil para desarrolladores.

Evolución: A lo largo de los años, SQLite ha experimentado múltiples versiones, cada una con mejoras y optimizaciones. Se ha mantenido como un proyecto de código abierto activo con una comunidad de desarrollo sólida. A medida que las demandas de aplicaciones móviles y embebidas han crecido, SQLite ha evolucionado para ofrecer un mejor rendimiento y mayor capacidad.

Historia y concepto de Firebase

Firebase fue inicialmente creado en 2011 por James Tamplin y Andrew Lee con el nombre de Envolve. En 2012 fue renombrado a Firebase y en 2014 fue adquirido por Google.

Firebase nació como una plataforma en tiempo real que facilitaba el desarrollo de aplicaciones web y móviles. Originalmente proveía una base de datos en tiempo real hospedada en la nube a la que las aplicaciones podían conectarse para guardar y sincronizar datos entre los usuarios en tiempo real.

Posteriormente Firebase fue expandiendo sus capacidades y servicios ofrecidos. Hoy en día provee una plataforma de desarrollo de aplicaciones completa, con servicios como base de datos en tiempo real, autenticación de usuarios, alojamiento de archivos, notificaciones push, seguimiento de analytics, entre otros.

Algunos de los fundamentos conceptuales clave incluyen:

Desarrollo en tiempo real: Firebase ofrece una base de datos en tiempo real que permite a las aplicaciones mantener datos actualizados en tiempo real. Esto es fundamental para aplicaciones colaborativas, como chats en grupo y aplicaciones de seguimiento en tiempo real.

Almacenamiento en la nube: Firebase proporciona un servicio de almacenamiento en la nube que facilita la carga y descarga de archivos, como imágenes y videos, desde las aplicaciones. Esto es esencial para el contenido multimedia en aplicaciones.

Autenticación de usuarios: Firebase ofrece un sistema de autenticación de usuarios que permite a los desarrolladores administrar la identidad de los usuarios de sus aplicaciones. Esto facilita la implementación de registros de usuarios y sistemas de inicio de sesión seguros.

Servicios en la nube: Firebase incluye servicios en la nube que permiten realizar operaciones en la nube sin necesidad de configurar servidores propios. Esto simplifica tareas como el envío de notificaciones push y el procesamiento de datos en la nube.

Analíticas y prueba de aplicaciones: Firebase ofrece herramientas de analíticas que permiten a los desarrolladores comprender cómo los usuarios interactúan con sus aplicaciones y realizar pruebas para mejorar la experiencia del usuario.

Escalabilidad y confiabilidad: Firebase está diseñado para escalar con las necesidades de las aplicaciones, y Google proporciona una infraestructura de nube confiable para respaldar la plataforma.

Evolución: Desde su adquisición por Google, Firebase ha experimentado un crecimiento constante en términos de servicios y características. Se ha convertido en una plataforma completa para el desarrollo de aplicaciones móviles y web, y ha incorporado tecnologías como Cloud Functions, Hosting, Machine Learning, y más, para satisfacer las necesidades cambiantes de los desarrolladores. Firebase se ha convertido en una opción popular para empresas y desarrolladores individuales por igual, lo que demuestra su capacidad para proporcionar soluciones integrales en el mundo del desarrollo de aplicaciones.

Ventajas y desventajas de SQLite

Ventajas:

Rendimiento: SQLite es muy rápido y eficiente en consultas por su diseño liviano y por mantener los datos, índices y tablas en un solo archivo en disco. El acceso es muy rápido.

Escalabilidad: SQLite escala muy bien a grandes conjuntos de datos, soportando bases de datos de terabytes. Pero no escala bien porque es un sistema de un solo host.

Seguridad: SQLite proporciona seguridad a nivel de proceso para separar el acceso a la base de datos. También soporta cifrado para mayor seguridad de los datos.

Facilidad de uso: SQLite es simple de configurar y usar, no requiere un servidor dedicado, administración de usuarios, etc. La API es simple y limpia.

Portabilidad: SQLite se integra fácilmente en procesos y aplicaciones, y funciona en la mayoría de los sistemas operativos.

Desventajas:

Escalabilidad: No escala horizontalmente, solo soporta un host y requiere replicación para múltiples hosts.

Concurrencia: El bloqueo de SQLite limita la concurrencia en casos de mucho acceso concurrente por múltiples procesos o hilos.

Recuperación: No posee herramientas avanzadas de recuperación ante fallas o replicación automática. Requiere replicación manual.

Seguridad: Carece de algunas características avanzadas de seguridad como control de acceso, cifrado de conexiones, etc.

Consistencia: El modelo de consistencia eventual en lugar de fuerte puede ser problemático en algunos casos.

Ventajas y desventajas de Firebase

Ventajas:

Escalabilidad: Firebase está construido sobre la infraestructura de Google Cloud Platform, por lo que escala automáticamente para manejar grandes volúmenes de datos y usuarios.

Rendimiento: Los datos se almacenan en la memoria para una baja latencia y tiempos de respuesta muy rápidos. La sincronización de datos es también casi en tiempo real.

Facilidad de uso: Firebase provee una API intuitiva y fácil de usar. La consola simplifica el monitoreo y administración sin necesidad de un backend dedicado.

Flexibilidad: Firebase permite integrar solo los servicios que se necesiten en una aplicación. Se integra bien con otros servicios de Google Cloud.

Seguridad: Cumple con altos estándares de seguridad en cifrado de datos y autenticación. Permite reglas avanzadas de seguridad y permisos.

Desventajas:

Costo: Algunos servicios de Firebase tienen un costo adicional o requieren pasarse a un plan pago para mayor uso y capacidad.

Vendor lock-in: Por ser un servicio propietario, encadena la aplicación a la plataforma de Firebase/Google Cloud.

Curva de aprendizaje: Aunque simple al inicio, dominar todas sus funcionalidades avanzadas puede requerir una curva de aprendizaje.

Redundancia limitada: La replicación geográfica y redundancia de datos es limitada en el plan básico gratuito.

Seguridad: Algunos consideran que concentrar toda la lógica del backend en Firebase puede disminuir la seguridad general.

Mejor opción para implementar en Android

Ya que ambas son opciones sólidas para el almacenamiento de datos en aplicaciones Android, la mejor opción de implementación dependerá del enfoque y el alcance que se espera que tenga la aplicación, ya que varía según los requerimientos del proyecto.

Firebase ofrece una infraestructura sólida y escalable administrada por Google, lo que hace que sea una elección lógica para aplicaciones que requieren la gestión de grandes volúmenes de datos y la colaboración en tiempo real. Su capacidad para gestionar la sincronización de datos entre múltiples dispositivos y usuarios en tiempo real es inigualable, lo que lo convierte en una excelente opción para aplicaciones de redes sociales, aplicaciones de mensajería, aplicaciones colaborativas y otras aplicaciones que dependen de una interacción constante y en tiempo real.

Por otro lado, para desarrollos más pequeños y aplicaciones locales, donde la necesidad de sincronización en tiempo real es limitada o inexistente, SQLite es una elección más adecuada. SQLite ofrece un enfoque ligero y sin costo adicional para almacenar datos en el dispositivo local del usuario. Es especialmente útil en aplicaciones de utilidad, aplicaciones de lectura de datos sin conexión o aplicaciones que no requieren la colaboración en tiempo real. Además, SQLite es compatible con una amplia variedad de soluciones de almacenamiento local, como SharedPreferences y archivos, lo que brinda flexibilidad para adaptarse a las necesidades específicas de la aplicación.

Comparación y conclusiones

Algunas comparaciones a tomar en cuenta son las siguientes:

- **Tipo de Base de Datos:**

Firebase: Firebase es una plataforma de desarrollo de aplicaciones móviles de Google que ofrece una base de datos en tiempo real (Firebase Realtime Database) y una base de datos en la nube (Firestore). Ambos almacenan datos en la nube y ofrecen sincronización en tiempo real.

SQLite: SQLite es una base de datos SQL incorporada en el sistema operativo Android que almacena datos en el dispositivo local del usuario.

- Conectividad y Sincronización:

Firebase: Ofrece sincronización en tiempo real y permite a múltiples usuarios colaborar y ver los cambios en tiempo real. Es ideal para aplicaciones en las que la conectividad en tiempo real es crucial.

SQLite: Almacena datos localmente en el dispositivo, lo que significa que los datos no se sincronizan automáticamente con otros dispositivos sin implementar lógica adicional.

- Escalabilidad:

Firebase: Firebase es escalable de forma automática y gestionada por Google, lo que facilita el manejo de grandes volúmenes de datos y usuarios.

SQLite: La escalabilidad en SQLite depende de la capacidad del dispositivo local y no es tan escalable ni fácil de administrar en comparación con Firebase.

- Seguridad:

Firebase: Firebase ofrece autenticación incorporada y reglas de seguridad personalizables para proteger los datos almacenados en la nube.

SQLite: La seguridad en SQLite depende de cómo se implemente en la aplicación, y los datos locales pueden ser más vulnerables si no se aplican las medidas adecuadas.

- Costos:

Firebase: Firebase ofrece un modelo de precios basado en el uso, lo que significa que podría ser gratuito hasta cierto punto, pero los costos pueden aumentar a medida que crece el uso de la plataforma.

SQLite: No implica costos adicionales, ya que almacena datos localmente en el dispositivo del usuario.

Para proyectos pequeños desarrollados en Kotlin, SQLite es generalmente la mejor opción para implementar el almacenamiento de datos local. SQLite es un sistema de gestión de bases de datos relacional liviano, de alto rendimiento y ampliamente utilizado en aplicaciones móviles. Al estar completamente incorporado en Android, SQLite permite crear una base de datos en el dispositivo sin necesidad de instalar software adicional. Esto resulta muy beneficioso para proyectos de menor escala.

Una de las principales ventajas de SQLite para proyectos Kotlin pequeños es que, al ser un motor de base de datos contenido en una biblioteca, no requiere de un servidor dedicado ni configuraciones complejas. Simplemente se puede llamar a la API de SQLite desde Kotlin y realizar consultas SQL para leer y escribir en la base de datos, manteniendo un enfoque simple pero potente. Esto reduce significativamente la complejidad en comparación con soluciones clientes-servidor.

Además, dado que los datos se almacenan localmente, SQLite funciona sin conexión a internet y no depende de llamadas a servicios remotos. Esto es ideal para aplicaciones que necesitan estar disponibles sin conexión. SQLite también tiene un excelente rendimiento en lecturas y escrituras dado que los datos están en el dispositivo local, evitando latencias de red.

Otra ventaja clave para proyectos pequeños es que SQLite no tiene costos de licencia. Se puede distribuir en aplicaciones sin restricciones al estar bajo dominio público. Esto reduce costos para proyectos con presupuestos limitados. SQLite también tiene un pequeño uso de memoria y recursos computacionales, por lo que funciona muy bien en dispositivos móviles.

En cuanto a seguridad, SQLite permite aislar el acceso a la base de datos a nivel de proceso. Aunque no tiene capacidades avanzadas de autenticación y control de acceso, en proyectos simples se puede lograr un nivel adecuado de seguridad si se implementa correctamente. La encriptación también está disponible para proteger los datos confidenciales.

Un proyecto Kotlin pequeño normalmente no requerirá de alta concurrencia, replicación ni alta disponibilidad. Por lo tanto, las limitaciones de SQLite en estas áreas no son un problema crítico. Lo más importante es contar con una base de datos rápida, confiable y sin costo adicional.

Como grupo hemos llegado a la conclusión que para un proyecto Kotlin de menor escala donde la base de datos se utilizará localmente en el dispositivo móvil, SQLite es la mejor elección. Ofrece rendimiento, simplicidad, cero costos, bajos requerimientos y una API sencilla de utilizar desde Kotlin. Firebase u otras bases de datos en la nube resultarían excesivas y no necesarias para proyectos pequeños. SQLite cumple con creces las necesidades de almacenamiento local en estos casos. Por estas razones, es altamente recomendable utilizar SQLite para proyectos Kotlin pequeños.