



UNISUAM

Luis Felipe Macedo dos Santos

Desenvolvimento Móvel

Prof. Charles, Prof. Oswaldo, Prof. Artur.

29/05/2025

Objetivo: Ao desenvolver aplicativos móveis é comum precisar armazenar dados do usuário no aplicativo, essa prática se dá pelo fato de que informações como preferências de utilização, cache de dados para utilização offline da aplicação e credenciais de acesso do usuário precisam se encontrar próximas do cliente para garantir a consistência da aplicação, a entrega rápida de informações (visto que dependerá menos vezes de realizar conexões externas) e melhorar a experiência do usuário.

Existem diversas formas de armazenar dados no lado do cliente, vamos explorar cada algumas e explicar alguns de seus usos mais comuns.

Conteúdo: Quando falamos em armazenar dados, pensamos em banco de dados, e dos bancos de dados existem 3 tipos, são eles os bancos de dados relacionais, bancos de dados não relacionais e bancos de dados não somente relacionais.

Para os dispositivos móveis como o android, o próprio sistema operacional oferece um banco de dados do tipo relacional chamado “sqlite”, este é um banco relacional extremamente leve e é possível através de plugins nativos interagir com esse banco e persistir localmente na memória do aplicativo dados diversos. Entretanto bancos de dados relacionais possuem limitações pois sua estrutura de dados precisa ser muito bem definida e as consultas frequentes podem levar ao alto consumo de recursos de hardware do dispositivo.

Alternativamente os bancos de dados não relacionais, que são diversos, costumam oferecer flexibilidade na estrutura dos dados e baixo consumo de recursos, normalmente bancos de dados não relacionais persistem dados em documentos, para cada dado um documento, mas essa regra nem sempre se aplica a todos os bancos não relacionais.

Bancos de dados não relacionais também possuem tipos diferentes, como exemplo temos os bancos de dados de documentos, grafos, chave e valor e colunares, cada um possuindo seus prós e contras.

Nos dispositivos móveis a forma mais comum de armazenar dados é através do modelo chave e valor. Onde é persistido em memória um documento que é a chave, e seu conteúdo o valor.

Nos navegadores web possuímos implementações de bancos de chave e valor como o armazenamento local, os “Cookies” e mais atualmente o “IndexedDB”, todos baseados em chave e valor.

Além disso, para dispositivos móveis existem diversos frameworks e bibliotecas que implementam a funcionalidade de persistência de dados no lado do cliente. Uma delas, também baseado em chave/valor é a biblioteca “@ionic/storage” que faz parte do framework “ionic”.

Comumente dados de sessão e informações de preferências de usuário são armazenadas em bancos relacionais como o sqlite, dados para utilização offline em bancos de documentos e dados que são utilizados frequentemente em transações em bancos de chave e valor.

Entretanto, existem aplicações que precisam armazenar dados de forma eficiente e em que a regras de negócio da aplicação exigem que esses dados sejam armazenados em servidores na rede, nos casos em que por exemplo são necessárias trocas de mensagem rapidamente entre usuários. Para esses casos de uso é comum a utilização de bancos de dados de documentos como o “firestore”, além de bancos de dados de chave/valor como o “redis” , “memcacheDB” e “KeyVal” que são bancos de dados em memória se tornando ainda mais eficiente na comunicação entre usuários e servidores.

Conclusão: Dados são importantes, os persistir e mantê-los organizados ainda mais! Ferramentas como softwares e bibliotecas foram desenvolvidas ao longo dos anos para suportar diversos tipos de aplicações, se encaixando nas necessidades das regras de negócio de sistemas diferentes, desde aplicações do lado do servidor, aplicações web, aplicações móveis e “IoT”. Bancos de dados oferecem ao usuário experiência, informação e eficiência, enquanto que no lado dos que estão por de trás dos serviços agrega valor ao negócio, gera “insights” sobre como o usuário utiliza os aplicativos e suas preferências.

Referências: