Educação Profissional Paulista

Técnico em
Desenvolvimento de Sistemas

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Armazenamento de dados e repositórios

Introdução ao armazenamento de dados em aplicações móveis

Conceitos básicos de armazenamento de dados

Código da aula: [SIS]ANO2C1B2S8A1

Mapa da Unidade 2 Componente 1

Armazenamento de dados e repositórios

semana 8

Você está aqui!

Introdução ao armazenamento de dados em aplicações móveis

semana 9

Implementação de banco de dados

semana 12

Avançando com repositórios de dados

semana 14

Integração de repositórios com APIs externas

semana 17

Banco de dados avançados

Armazenamento de dados e repositórios

Mapa da Unidade 2 Componente 1

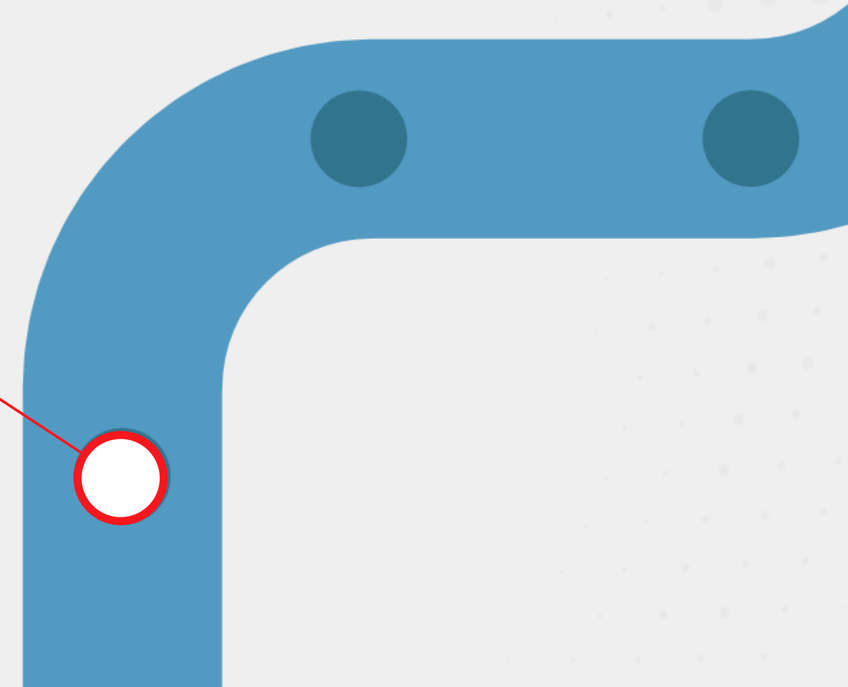
Você está aqui!

Introdução ao armazenamento de dados em aplicações móveis

8

Aula 1 - Conceitos básicos de armazenamento de dados

Código da aula: [SIS]ANO2C1B2S8A1



Objetivos da aula

- Conhecer os conceitos básicos de armazenamento de dados em aplicações móveis.

Recursos didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Lápis e caderno para anotações.

Duração da aula

50 minutos.

Habilidades técnicas

- Criar um banco de dados básico para uma aplicação móvel.

Habilidades socioemocionais

- Promover a curiosidade com o interesse em novas maneiras de armazenar dados.

Ponto de **partida**

Imagine que você está desenvolvendo um aplicativo de lista de tarefas para dispositivos móveis. Cada usuário pode adicionar suas tarefas diárias, e você precisa garantir que essas informações sejam salvas de forma que o usuário possa acessá-las mesmo depois de fechar o aplicativo. Além disso, o aplicativo precisa funcionar off-line, de modo que, mesmo sem conexão, os dados das tarefas sejam preservados e sincronizados com o servidor quando a conexão for restabelecida.

A partir dessa situação, surge a pergunta: Como armazenar os dados das tarefas de forma eficiente em um dispositivo móvel?

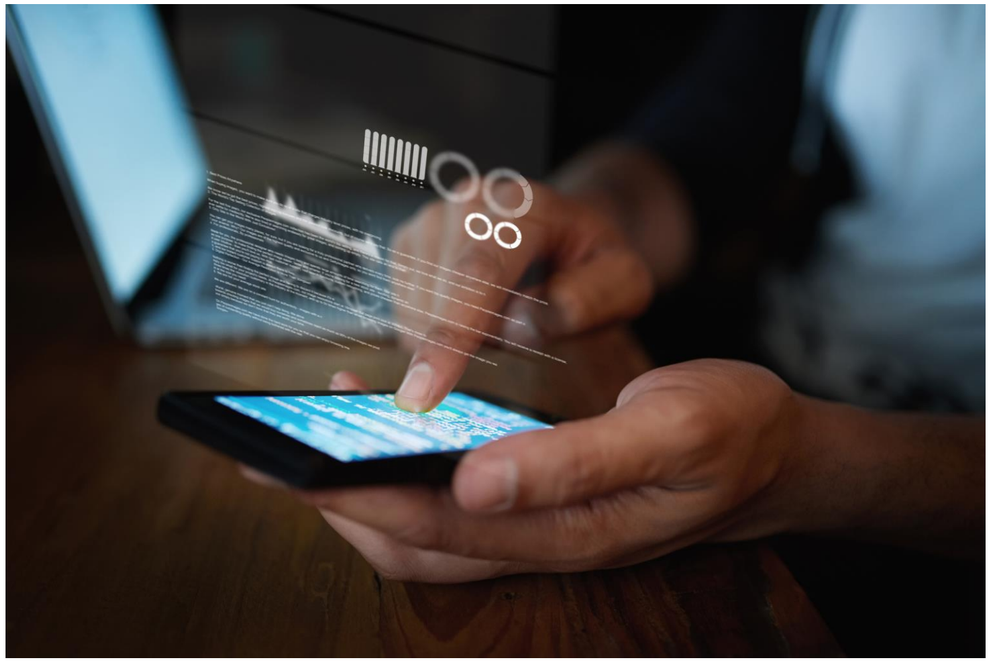
Situação fictícia produzida pela SEDUC-SP.

- Quais são os principais tipos de armazenamento de dados em dispositivos móveis?
- Quando devemos utilizar um banco de dados local em vez de um armazenamento temporário (memória)?
- Como o armazenamento de dados off-line pode ser sincronizado com o servidor em uma aplicação móvel?

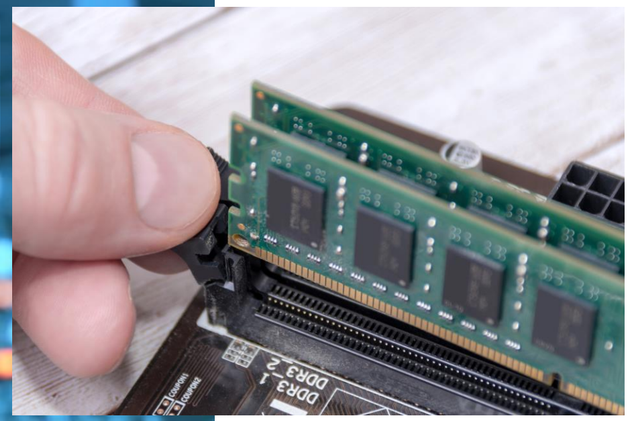
Construindo o **conceito**

Armazenamento em dispositivos móveis

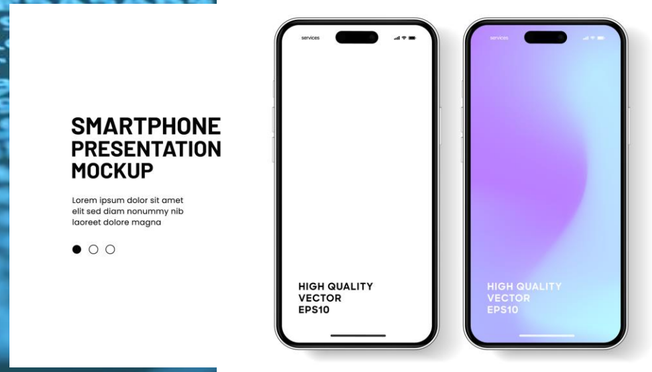
Armazenamento de dados em dispositivos móveis é a base para aplicações que precisam persistir informações além do ciclo de vida da aplicação. Existem vários tipos de armazenamento, cada um adequado a um propósito específico. Vamos explorar os mais comuns:



Colocando em **prática**



© Getty Images



© Getty Images

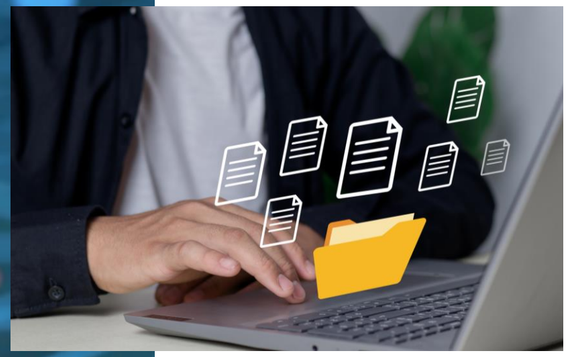
Memória Volátil (RAM)

Utilizada para armazenar dados temporários durante a execução do aplicativo. Esses dados são perdidos quando o aplicativo é fechado. Ideal para armazenar informações de curto prazo ou que não precisam ser persistidas, como dados de sessões temporárias.

SharedPreferences (Android) e UserDefaults (iOS)

Armazenam pares chave-valor e são ideais para pequenas quantidades de dados que precisam ser persistidos entre sessões, como configurações de usuário ou preferências.

Colocando em **prática**



© Getty Images

Reprodução - INFORTREND, [s.d.]. Disponível em: https://www.infortrend.com.br/post/servidor-de-banco-de-dados. Acesso em: 7 nov. 2024.

Arquivos Locais

Arquivos podem ser utilizados para armazenar dados não estruturados (por exemplo, documentos, imagens, vídeos) ou dados estruturados em formatos como JSON ou XML. Esse tipo de armazenamento é útil para dados que precisam ser acessados diretamente pelo usuário ou compartilhados com outros aplicativos.

Banco de Dados Local (SQLite, Room)

Um banco de dados é mais adequado quando a aplicação precisa lidar com grandes volumes de dados estruturados e realizar operações complexas, como consultas ou manipulação de registros. O SQLite é o banco de dados embutido mais comum em dispositivos móveis, e Room é uma biblioteca que facilita o uso do SQLite no Android, oferecendo uma camada de abstração que simplifica a criação e manipulação de bancos de dados.

Construindo o **conceito**

Exemplo prático (Android - Room):

```
@Entity
data class Tarefa(
    @PrimaryKey val id: Int,
    val descricao: String,
    val status: String
)

@Dao
interface TarefaDao {
    @Query("SELECT * FROM Tarefa WHERE status = :status")
    fun obterTarefasPorStatus(status: String): List<Tarefa>

    @Insert
    fun inserirTarefa(tarefa: Tarefa)
}

@Database(entities = [Tarefa::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun tarefaDao(): TarefaDao
}
```

Produzido pela SEDUC-SP com a ferramenta Visual Studio Code.

Construindo
o **conceito**

Exemplo prático (iOS - Core Data):

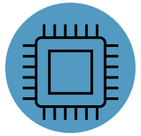
```swift
let tarefa = NSEntityDescription.insertNewObject(forEntityName: "Tarefa", into: context)
tarefa.setValue("Estudar Swift", forKey: "descricao")
tarefa.setValue("pendente", forKey: "status")

do {
    try context.save()
} catch {
    print("Falha ao salvar tarefa: \(error)")
}
```

Produzido pela SEDUC-SP com a ferramenta Visual Studio Code.

Construindo o **conceito**

Quando usar cada tipo de armazenamento



Memória

Use para dados temporários, como informações que só precisam estar disponíveis enquanto o aplicativo está em execução. Por exemplo, resultados de cálculos ou variáveis que controlam o estado atual da interface do usuário.

SharedPreferences/UserDefaults

Ideal para pequenas configurações ou estados simples. Por exemplo, o tema escolhido pelo usuário (escuro ou claro) ou preferências de notificações.

Arquivos Locais

Útil para dados grandes que precisam ser armazenados no formato de arquivos, como fotos ou relatórios em PDF. Por exemplo, uma aplicação de notas pode salvar arquivos de áudio ou documentos como anexos.

Banco de Dados Local

Ideal para aplicações que precisam armazenar dados relacionais e fazer consultas frequentes, como uma aplicação de gerenciamento de tarefas ou uma agenda de contatos.

Construindo o **conceito**

Sincronização de dados off-line com o servidor

Para garantir que os dados armazenados localmente sejam sincronizados com o servidor quando a conexão for restabelecida, as aplicações móveis geralmente utilizam estratégias como:

- **Tarefas em segundo plano:** ferramentas como o WorkManager (Android) e BackgroundTasks (iOS) permitem que tarefas sejam executadas em segundo plano para sincronizar os dados com o servidor, mesmo quando o aplicativo está fechado.

Continua...

Construindo o **conceito**

Sincronização de dados off-line com o servidor

- **Fila de sincronização**: uma técnica comum é manter uma fila de operações que precisam ser sincronizadas com o servidor. Quando a conexão é restabelecida, essas operações são processadas de forma sequencial. Isso garante que nenhuma informação seja perdida ou duplicada.

Exemplo prático (Android - WorkManager):

```
val uploadWorkRequest = OneTimeWorkRequestBuilder<UploadWorker>().build()
WorkManager.getInstance(context).enqueue(uploadWorkRequest)
```

Produzido pela SEDUC-SP com a ferramenta Visual Studio Code.

Situação

Desenvolvimento mobile

Você está trabalhando em um projeto de um aplicativo de lista de tarefas, que armazena os dados localmente no dispositivo e sincroniza com o servidor quando há conexão à internet. Durante os testes finais, foi identificado um problema: as tarefas criadas quando o aplicativo está off-line não estão sendo sincronizadas corretamente com o servidor quando a conexão é restabelecida.

A equipe de desenvolvimento está dividida: alguns membros acreditam que o problema é secundário e que não vale a pena atrasar o lançamento do aplicativo para resolver a questão. Já outros, incluindo você, percebem que o problema de sincronização pode gerar uma experiência ruim para o usuário, especialmente para aqueles que usam o aplicativo em áreas com cobertura de internet limitada.

Sua tarefa é argumentar com a equipe sobre a importância de corrigir o problema, explicando como o armazenamento off-line e a sincronização de dados são essenciais para a experiência do usuário, e propor uma solução técnica que resolva o problema de maneira eficiente. Além disso, você precisa abordar como trabalhar colaborativamente para resolver o problema sem criar tensões desnecessárias na equipe.

Situação fictícia produzida pela SEDUC-SP.

Ser
sempre +

Ação

- Argumente de forma construtiva com a equipe, explicando por que o problema de sincronização precisa ser resolvido antes do lançamento. Utilize exemplos técnicos e pense na perspectiva do usuário.
- Proponha uma solução técnica, sugerindo o uso de um mecanismo de sincronização confiável, como o WorkManager no Android ou BackgroundTasks no iOS, e explique como essa solução resolveria o problema identificado.
- Administre o conflito na equipe, buscando uma solução que permita o lançamento do aplicativo sem comprometer a qualidade e, ao mesmo tempo, atenda à preocupação com prazos.

O que nós **aprendemos hoje?**

Então ficamos assim...

1. Compreendemos quais são os principais tipos de armazenamento de dados em dispositivos móveis;

2. Conhecemos quando devemos utilizar um banco de dados local em vez de um armazenamento temporário (memória);

3. Entendemos como o armazenamento de dados off-line pode ser sincronizado com o servidor em uma aplicação móvel.

Saiba mais

Curioso para saber mais sobre o desenvolvimento de apps móveis e como isso pode ser aplicado no mercado?

Explore ideias práticas e relevantes com este vídeo que conecta teoria e prática de forma dinâmica!

ATTEKITA DEV. **Como se tornar um desenvolvedor mobile em 2024? (Roadmap de tecnologias atualizado)**. Disponível em: https://www.youtube.com/watch?v=COY5vTa778g. Acesso em: 6 nov. 2024.

Referências da aula

ANDROIDPRO. **4 maneiras de armazenar dados de aplicativos localmente**, [s.d.]. Disponível em: https://www.androidpro.com.br/blog/armazenamento-de dados/armazenar-dados-de-aplicativos-localmente/. Acesso em: 6 nov. 2024.

ATTEKITA DEV. **Como se tornar um desenvolvedor mobile em 2024? (Roadmap de tecnologias atualizado).** Disponível em: https://www.youtube.com/watch?v=COY5vTa778g. Acesso em: 6 nov. 2024.

CIÊNCIA DA COMPUTAÇÃO LINKEDIN. Quais são os padrões de design de armazenamento de dados mais comuns para aplicativos móveis? **Linkedin**, [s.d.]. Disponível em: https://www.linkedin.com/advice/0/what-most-common-data storage-design-patterns-ns9hc?lang=pt&originalSubdomain=pt. Acesso em: 6 nov. 2024.

IBM. **Armazenamento de dados de aplicações nativas da nuvem**, [s.d.]. Disponível em: https://www.ibm.com/br-pt/products/cloud-object-storage/cloud-native-app data-storage. Acesso em: 6 nov. 2024.

Orientações ao professor

Slide 6

Orientações: Professor, a seção **Ponto de partida** aparece no início de cada aula e tem como objetivo ativar o conhecimento prévio dos estudantes sobre o tema da aula e estimular seu pensamento crítico e suas habilidades comunicativas. Por meio de uma situação-problema ou exemplo próximo da realidade do estudante, pretende-se sair da abstração conceitual e promover um diálogo dinâmico para explorar hipóteses, soluções e compartilhar eventuais experiências que os estudantes já possam ter com os tópicos a serem abordados na aula. Também é um momento de engajá-los em relação ao tema da aula.

Tempo: 8 minutos

Expectativas de respostas:

1. Principais tipos de armazenamento em dispositivos móveis incluem: Armazenamento em memória (volátil), SharedPreferences (para dados simples como configurações), Arquivos locais (para dados estruturados ou não estruturados) e Bancos de Dados locais (SQLite, Room).

2. Utilizamos um banco de dados local quando precisamos armazenar grandes quantidades de dados estruturados, que precisam ser persistidos por longos períodos de tempo e acessados mesmo após fechar o aplicativo. O armazenamento em memória, por outro lado, é útil para dados temporários que não precisam ser mantidos após o ciclo de vida da aplicação.

3. O armazenamento off-line pode ser sincronizado com o servidor usando técnicas como sistema de fila para envio dos dados quando a conexão é reestabelecida. Isso pode ser feito utilizando APIs como WorkManager em Android ou BackgroundTasks em iOS, que garantem que as tarefas de sincronização ocorram mesmo quando o aplicativo não está em primeiro plano.

Slide 15

Orientações: A seção **Ser sempre +** tem como objetivo desenvolver e aprimorar as competências socioemocionais dos estudantes, focando especificamente nas situações desafiadoras que podem surgir no ambiente profissional.

Tempo: 15 minutos

Gestão de sala de aula:
Mantenha um ambiente de diálogo aberto e respeitoso.
Assegure a participação equitativa, promovendo uma discussão inclusiva.
Reconheça a complexidade do tema e a diversidade de perspectivas que os estudantes podem trazer.
Forneça feedback construtivo e direcionamento à medida que os estudantes exploram possíveis soluções para o cenário proposto.
Ajude os estudantes a refinarem suas ideias e a considerarem todas as implicações de suas sugestões.

Condução da dinâmica:

Divida a turma em pequenos grupos de estudantes.

Expectativas de Respostas:

Justificativa Socioemocional
Ao comunicar-se com a equipe, você pode começar mencionando a importância do feedback dos usuários para o sucesso contínuo do aplicativo. A retenção de clientes está diretamente relacionada à experiência do usuário, e travamentos podem resultar em avaliações negativas que afetam diretamente o sucesso do aplicativo.

Argumento chave: "As críticas dos usuários podem nos ajudar a identificar problemas que talvez não tenhamos percebido no desenvolvimento. Ignorar esses sinais pode custar clientes. O travamento relatado durante o pagamento ocorre porque o processamento é feito na UI thread, o que interrompe a fluidez do aplicativo. Ao mover essa tarefa para uma thread em segundo plano, garantimos que o usuário tenha uma experiência mais suave, aumentando a satisfação e a retenção."

Slide 17

Orientações: Professor, a seção **O que nós aprendemos hoje?** tem o objetivo de reforçar e esclarecer os conceitos principais discutidos na aula. Essa revisão pode ser uma ferramenta de avaliação informal do aprendizado dos estudantes, identificando áreas que possam precisar de mais atenção em aulas futuras.

Tempo: 2 minutos

Gestão de sala de aula:

- Mantenha um tom positivo e construtivo, reforçando o aprendizado em vez de focar em correções;
- Seja direto e objetivo nas explicações para manter a atividade dentro do tempo estipulado;
- Engaje os estudantes rapidamente, pedindo confirmações ou reações breves às definições apresentadas.

Condução da dinâmica:

- Explique que esta parte da seção, “Então ficamos assim...”, é um momento de reflexão e esclarecimento sobre os conceitos abordados na aula;
- Informe que será uma rápida revisão para assegurar que os entendimentos dos estudantes estão alinhados com as definições corretas dos conceitos;
- Apresente o slide com a definição sintética de cada conceito principal discutido na aula, ampliando em forma de frases completas;
- Destaque se as contribuições dos estudantes estavam alinhadas com o conceito e ofereça esclarecimentos rápidos caso haja discrepâncias ou mal-entendidos.;
- Finalize resumindo os pontos principais e reiterando a importância de cada conceito e como ele se encaixa no contexto maior da aula;
- Reforce a ideia de que essa revisão ajuda a solidificar o entendimento dos estudantes e prepará-los para aplicar esses conceitos em situações práticas.

Expectativas de respostas:

Os estudantes devem sair da aula com um entendimento claro e preciso dos conceitos principais. A atividade serve como uma verificação rápida do entendimento dos estudantes e uma oportunidade para corrigir quaisquer mal-entendidos.

Educação Profissional Paulista

Técnico em
Desenvolvimento de Sistemas

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO