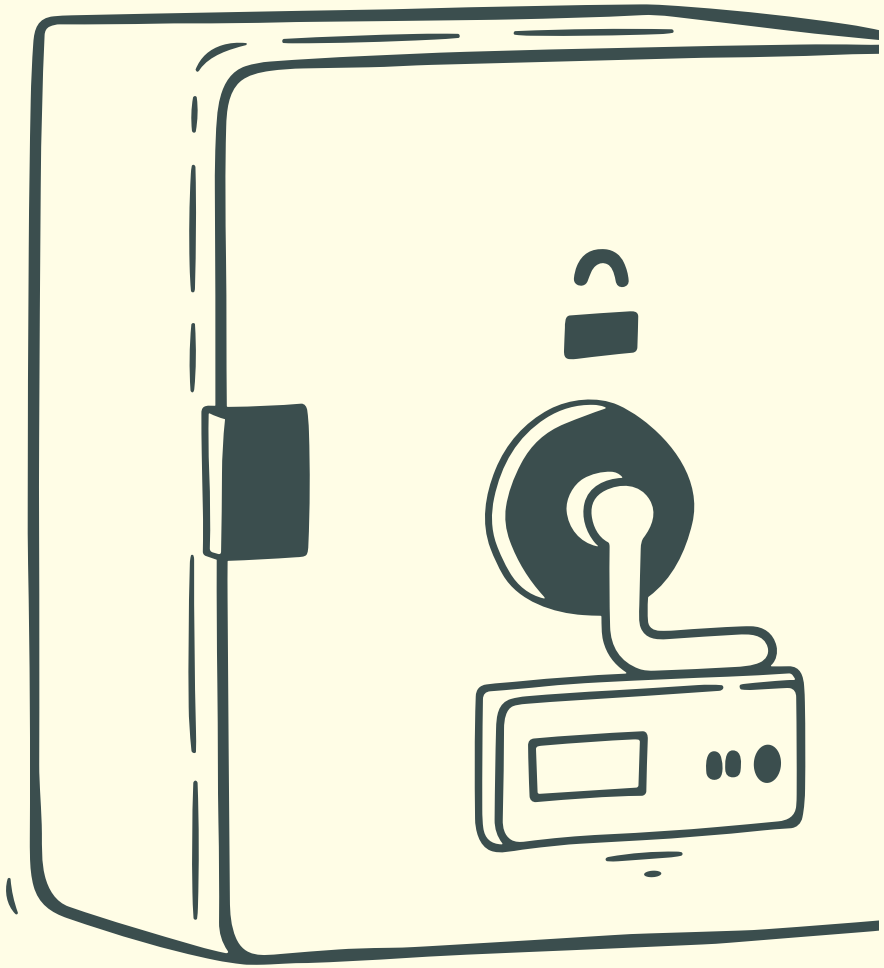
Handling Data and Logic in Android Apps

Master the art of persistent data storage and efficient logic handling in Android applications.

This course covers essential techniques from simple preferences to complex database operations and API interactions.

by Australian Teachers



Storing Data Using SharedPreferences

Basic Concept

A lightweight key-value storage system for simple app data. Perfect for user settings and preferences.

Implementation

Easy to implement with minimal code. Values are stored as primitive types or strings.

Limitations

Not suitable for complex data structures. Limited to your app's private storage.

SQLite Databases in Android

Create Database

Define schema and tables with SQL statements

Implement Helper

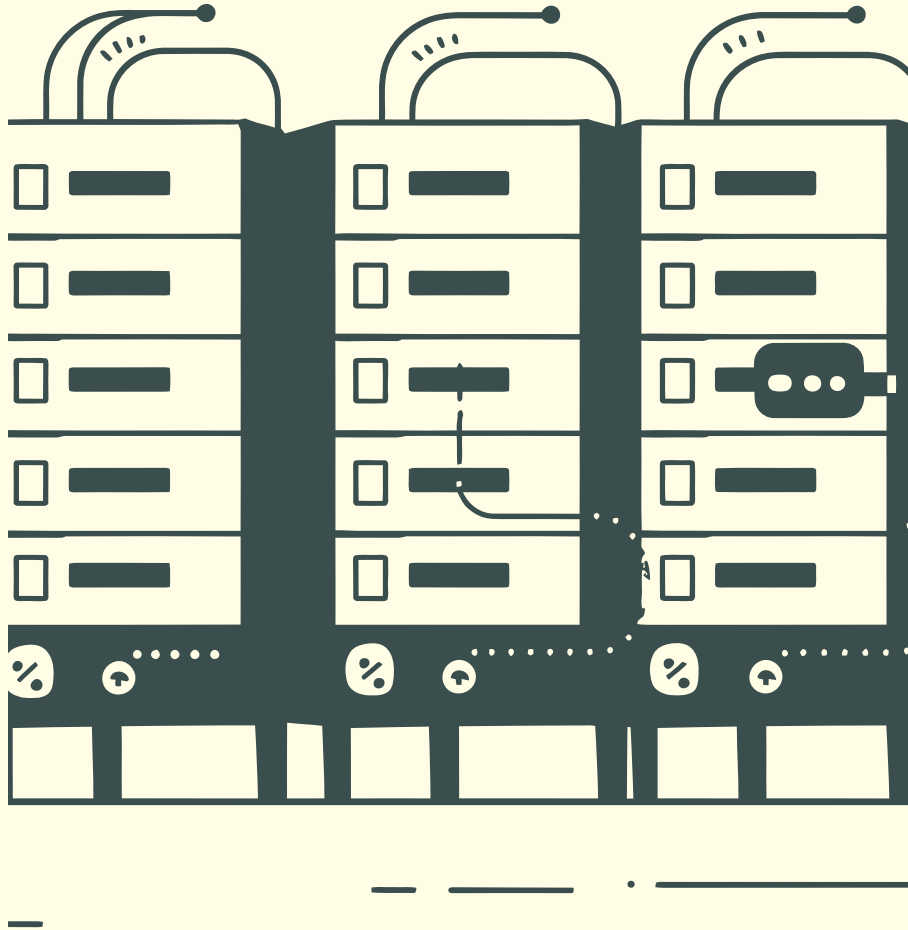
Extend SQLiteOpenHelper for version management

Query Data

Use SQL queries to retrieve structured information

Maintain

Handle upgrades and migrations gracefully



Simplifying Persistence with Room

Define Entities

Use @Entity annotation to create database tables from Kotlin classes

Create DAOs

Data Access Objects define SQL operations as interface methods

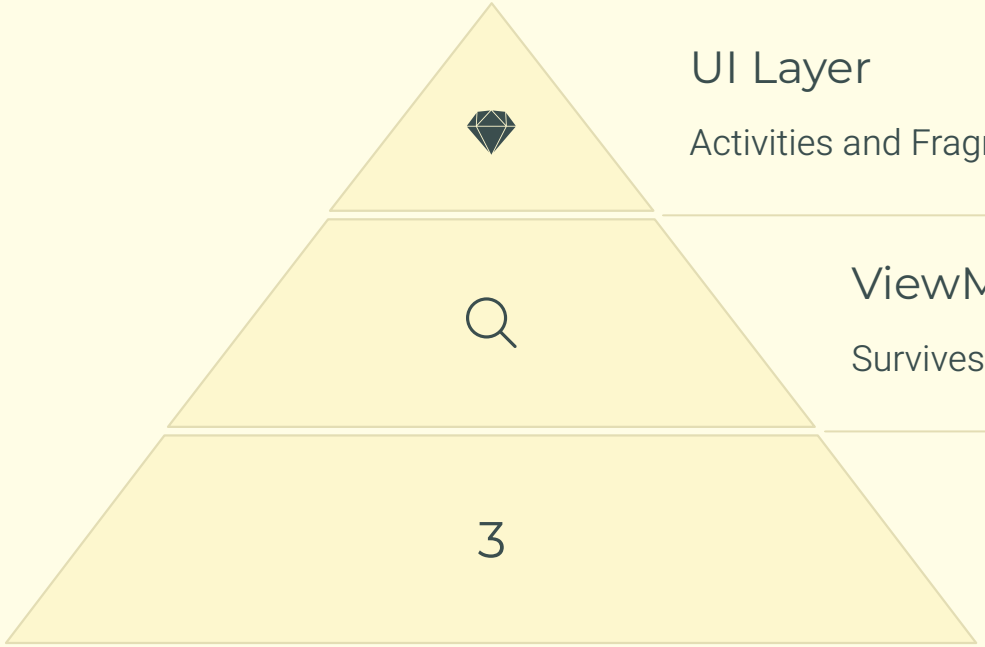
Setup Database

Abstract class extends RoomDatabase with migration strategies

Connect Components

Link entities, DAOs, and database class for type-safe operations

ViewModel and LiveData for App Logic



UI Layer

Activities and Fragments observe LiveData for updates

ViewModel Layer

Survives configuration changes and exposes LiveData

Repository Layer

Manages data operations from multiple sources

Validating User Input

Client-Side Validation

Check data format and integrity before processing. Use regex patterns for emails and passwords.

Prevent SQL Injection

Sanitize inputs before database operations. Use parameterized queries for security.

Provide Real-Time Feedback

Show error messages immediately. Highlight fields needing correction with clear instructions.

Server-Side Validation

Always verify data on the server too. Never trust client-only validation for security.

Connecting to REST APIs & Handling JSON

Configure Client

Setup Retrofit or Volley with appropriate converters

Define Endpoints

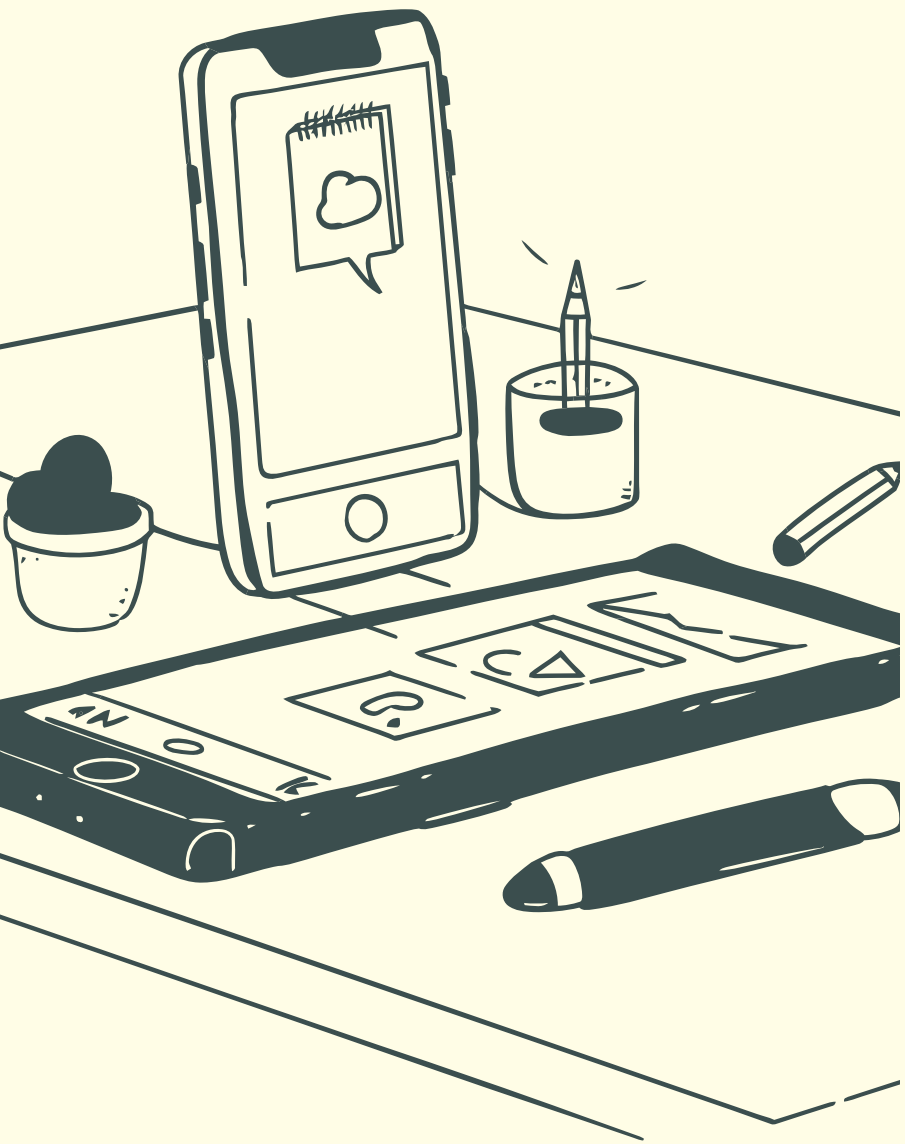
Create interface methods for each API endpoint

Handle Errors

Implement robust error handling for network failures

Parse Responses

Convert JSON to Kotlin data classes automatically



Summary & Best Practices

Choose The Right Storage

SharedPreferences for simple data. Room for complex relational data. Cloud storage for shared data.

Implement Clean Architecture

Separate concerns with ViewModel and Repository patterns. Keep UI logic-free and responsive.

Validate Everything

Never trust user input. Validate on client and server sides before processing.

Test Edge Cases

Ensure your app handles network failures gracefully. Test with different data scenarios.