Eureka Xpert By Eureka Edutech

talenthub

"Architecture Components"

Ikrima Amanda

Architectural Pattern

Deskripsi

- pendekatan yang memberikan pedoman dan struktur untuk mengorganisir sistem perangkat lunak secara keseluruhan.
- Pola arsitektur => membantu dalam merancang sistem yang terstruktur, modular, dan mudah dimengerti.
- Membantu dalam mengatur hubungan antara komponen sistem, menentukan tanggung jawab masing-masing komponen, dan mengatur alur data dan logika bisnis.
- Membantu dalam mencapai tujuan-tujuan seperti keterbacaan kode, pengujian yang baik, dan pemeliharaan yang mudah.

Pola Arsitektur yang umum digunakan

- MVC
  - pola arsitektur yang memisahkan aplikasi menjadi tiga komponen utama: Model, View, dan Controller.
  - Model => data dan logika bisnis
  - View => antarmuka pengguna
  - Controller => mengatur interaksi antara Model dan View
- MVP
  - pola arsitektur yang memisahkan tampilan (View), logika bisnis (Presenter), dan data (Model).
  - Presenter => penghubung antara View dan Model. Presenter menerima input dari pengguna melalui View, memprosesnya menggunakan Model, dan mengubah tampilan di View sesuai dengan hasilnya.
  - MVP membantu memisahkan kode tampilan dan meningkatkan pengujian dengan memungkinkan pengujian logika bisnis secara terpisah dari antarmuka pengguna.
- MVVM
  - pola arsitektur yang paling populer dalam pengembangan Android
  - MVVM memisahkan logika bisnis (Model), antarmuka pengguna (View), dan ViewModel yang bertindak sebagai penghubung antara keduanya.
  - ViewModel => menyediakan data yang diperlukan oleh View dan memproses aksi pengguna.

View Model

- kelas yang dirancang untuk menyimpan dan mengelola data terkait UI serta logika bisnis yang terkait dengan tampilan (View). ViewModel bertahan selama siklus hidup aktivitas atau fragment, yang berarti data yang disimpan di dalam ViewModel tetap ada bahkan saat terjadi perubahan konfigurasi seperti rotasi layar.
- Keuntungan menggunakan ViewModel adalah sebagai berikut:
    - Memisahkan logika bisnis dan data dari komponen tampilan (aktivitas/fragment).
    - Memudahkan pengujian karena logika bisnis dapat diuji secara terpisah dari komponen tampilan.

Live Data

- komponen yang digunakan untuk mengamati (observe) perubahan data dan memberikan notifikasi ke penerima (observer) saat data berubah. LiveData memastikan bahwa penerima (observer) selalu mendapatkan data terbaru secara otomatis dan hanya saat data berubah.
- LiveData biasanya digunakan bersama dengan ViewModel.
- ViewModel menyimpan data dan logika bisnis, sedangkan LiveData digunakan untuk mengamati perubahan data dan memberikan notifikasi ke komponen UI yang terkait.
- LiveData dapat diakses dari komponen UI seperti aktivitas atau fragment dengan menggunakan metode observe().
- Ketika ada perubahan pada LiveData, komponen UI akan menerima notifikasi dan dapat memperbarui tampilan sesuai dengan data yang terkini.

Eureka Xpert talenthub

Coroutines

Deskripsi

- fitur yang diperkenalkan dalam bahasa pemrograman Kotlin untuk melakukan pemrograman konkuren (concurrency) secara lebih mudah dan responsif. Coroutines membantu untuk menulis kode yang bersifat asinkron tanpa menggunakan callback atau pemrograman berbasis thread secara langsung.
- Untuk menggunakan coroutines dalam proyek Android Kotlin, perlu menambahkan dependensi kotlinx-coroutines-android pada file build.gradle

Konsep Penting

- Suspended Functions => fungsi yang dapat memberikan sinyal untuk menangguhkan (suspend) eksekusi tanpa memblokir thread yang sedang berjalan.
- Coroutine Scope => lingkungan tempat coroutines berjalan dan dikendalikan. Anda dapat membuat coroutine scope dengan menggunakan fungsi CoroutineScope(). Coroutine scope memungkinkan untuk mengelola coroutines, membatalkan coroutines yang sedang berjalan, atau mengendalikan perilaku coroutines.
- Coroutine Builders => fungsi yang memulai pembuatan coroutines. Dalam pengembangan Android Kotlin, dua coroutine builders yang sering digunakan adalah launch dan async. Launch digunakan untuk menjalankan coroutines tanpa mengembalikan nilai, sedangkan async digunakan untuk menjalankan coroutines yang mengembalikan nilai.
- Coroutine Context dan Dispatcher => kumpulan properti yang mendefinisikan lingkungan dan aturan untuk menjalankan coroutines. Coroutine dispatcher adalah komponen dalam coroutine context yang menentukan di mana coroutines akan dijalankan, misalnya pada thread utama (Dispatchers.Main) atau pada thread latar belakang (Dispatchers.IO).

Crashlytics

Deskripsi

- layanan pelaporan kecelakaan (crash reporting) yang disediakan oleh Firebase untuk pengembangan aplikasi Android Kotlin.
- Firebase Crashlytics membantu dalam mengumpulkan dan melacak laporan kecelakaan yang terjadi pada aplikasi di berbagai perangkat, memungkinkan developer untuk dengan cepat mengidentifikasi dan memperbaiki masalah dalam aplikasi Anda.
- Membantu developer untuk dengan cepat menemukan dan memperbaiki masalah dalam aplikasi, meningkatkan stabilitas, dan memberikan pengalaman yang lebih baik kepada pengguna Anda.

Poin Penting Penggunaan Crashlytics

- Mengintegrasikan Firebase crashlytics dengan proyek android
- Menginisialisasikan crashlytics ke dalam aplikasi (umumnya di dalam method onCreate())
- Dapat melaporkan exceptions atau error secara manual dengan method recordException() atau logException() => umumnya diletakkan di blok catch()
- Dapat mengumpulkan informasi kustom => log error, id pengguna yang mengalami error, atau informasi perangkat pengguna untuk membantu dalam menganalisis bug.

Eureka Xpert talenthub

ProGuard

Deskripsi

- alat pengoptimasi kode yang disediakan oleh Android SDK.
- Digunakan untuk melindungi dan mengamankan aplikasi Android Kotlin dengan melakukan beberapa tindakan seperti penghilangan kode yang tidak terpakai, mengurangi ukuran file APK, serta menyembunyikan struktur dan logika program dari dekompilasi.
- Dengan mengaktifkan ProGuard, developer dapat meningkatkan keamanan aplikasi dan mengurangi ukuran file APK, serta mengurangi risiko terhadap dekompilasi dan penggunaan kode Anda oleh pihak yang tidak berwenang.

Poin Penting Penggunaan ProGuard

- ProGuard biasanya sudah terintegrasi secara default dalam alat build seperti Gradle pada proyek Android.
- Developer hanya perlu mengaktifkan atau menyesuaikan konfigurasi ProGuard dalam file build.gradle pada tingkat aplikasi pada blok konfigurasi seperti buildTypes atau release yang dapat digunakan untuk mengaktifkan atau menyesuaikan ProGuard.
- Konfigurasi ProGuard terdiri dari aturan dan opsi yang mengendalikan perilaku ProGuard saat memproses aplikasi Anda.
- Aturan ini bisa berupa aturan default yang telah ditentukan oleh SDK Android, serta aturan yang ditentukan oleh Anda untuk melindungi kode atau mengabaikan kelas tertentu. Konfigurasi ProGuard umumnya disimpan dalam file proguard-rules.pro.

Eureka Xpert talenthub

Dependency Injection

Deskripsi

- Sebuah konsep dalam pengembangan perangkat lunak yang digunakan untuk mengelola dan menyediakan dependensi (objek yang digunakan oleh kelas lain). Pada dasarnya, DI membantu dalam mengurangi ketergantungan yang kuat antara kelas-kelas dalam aplikasi, sehingga memudahkan pengujian, memperbaiki, dan memodifikasi kode.

Manfaat DI

- Memudahkan pengujian => mengganti implementasi ketergantungan dengan versi yang dioptimalkan untuk pengujian, yang memudahkan pengujian unit dan pengujian fungsional.
- Memperbaiki fleksibilitas => DI memungkinkan penggantian ketergantungan dengan mudah, sehingga memudahkan perubahan dan memperbaiki kode tanpa mengubah banyak bagian lain dari aplikasi.
- Meningkatkan modularitas => DI membantu dalam memisahkan tanggung jawab dan memastikan kelas hanya bertanggung jawab untuk tugas tertentu, yang meningkatkan modularitas dan memudahkan pengembangan dan pemeliharaan.
- Mempermudah penggunaan ulang kelas => Dengan DI, kelas-kelas dapat digunakan kembali dengan lebih mudah karena mereka tidak terikat pada implementasi ketergantungan yang spesifik.

Referensi

- https://github.com/android/architecture-components-samples
- https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern
- https://developer.android.com/kotlin/coroutines?gclid=CjwKCAjw-7OlBhB8EiwAnoOEkydvN3tmmo5KhnzQIdnax5EHgo9nkMP6qW3pPEES8svpGzCVn7OUWhoCqCcQAvD_BwE&gclsrc=aw.ds
- https://firebase.google.com/docs/crashlytics
- https://developer.android.com/build/shrink-code
- https://developer.android.com/training/dependency-injection

Tugas

- https://forms.gle/wwSUqMRuPSxjS1An9

Stuck?

talk with ur best friend...
chatgpt, google, stack over flow, medium

The Most Important Thing…

https://www.instagram.com/reel/CtoupCfADAv/?igshid=Y2IzZGU1MTFhOQ%3D%3D

“Practice doesn’t make perfect. Perfect practice makes perfect”

- Vince Lombardi

Terima kasih!

telegram : @ikrimaamandaa

LinkedIn : https://www.linkedin.com/in/ikrimaamanda/