Fasi di modellazione di una App.

Questo schema fa riferimento all'App NoteTakingAppAdvanced

1. Creare un modello di UI e di navigazione con carta e penna e successivamente su modellazione digitale con programmi come Figma

} Non si programma!

2. Dopo aver chiarito gli obbiettivi della app, e la sua UI, bisogna scegliere un tema, e quindi una Color palette.

   2.1 Creare tutti i colori in Android Studio con relativi Background

} Non si programma ma si implementano gli oggetti grafici necessari

3. Implementare le view più importanti, come la Home e le viste dei singoli oggetti.
   ! Ricorda di assegnare un ID ad ogni componente grafico che dovrà essere reperito successivamente.

} Programmazione della UI con XML → Packaging activities

4. **Modellazione degli oggetti:** In questa fase si modellano gli oggetti che dovranno essere visualizzati; Se si usa un Database (Room - locale) si deve annotare ogni componente dello schema, oltre alla classe che va annotata come `@Entity (tableName = "test")`

} Programmazione Java degli oggetti di tipo Modellazione → Packaging Entities

5. Implementare i Dao degli oggetti da modellare. Usare l'annotazione: `@Dao` → classe e `@Query` `@Insert` ... Comportamenti

} I Dao vanno nel package dao

6 Creazione della classe che gestisce il Database: "NotesDatabase" che estende RoomDatabase, classe astratta. Va specificato la Versione del DB e il tipo di entità

la classe DB va nel package database

7 Per salvare le note (o altro) nel database, creiamo un metodo saveNote nella Activity che implementa il comportamento della View usata per aggiungere l'oggetto

Il salvataggio nel DB va fatto su un secondo thread.

8 Per leggere le note nella View principale ci affidiamo ad un metodo simile a quello usato per salvarle. Anche in questo caso il task è Asincrono ed eseguito in background.

Metodo getNotes con classe GetNotesTask che estende AsyncTask<Void, Void, List<Note>>

8.1 A questo punto possiamo leggere le note salvate nel DB, ma dobbiamo creare una view per la visualizzazione delle singole note, (o risorse in generale). Per testare il funzionamento usiamo Logcat

Per loggare ci basta Log.d("TAG_OBJ", String)

9 Ora che il DB funziona, possiamo creare la visualizzazione del singolo elemento che verrà poi visualizzato (con gli altri) nella RecyclerView.

La view, fatta in XML conterrà la rappresentaz. di tutti gli attributi dell'oggetto. Non dimentichiamo l'ID per ogni campo che verrà modificato!

10 Dopo aver creato una visualizzazione XML dell'oggetto, dobbiamo creare un adapter che si interfacci tra DB e RV. L'Adapter ha una inner class NoteViewHolder che estende RecyclerView.ViewHolder.
La classe adapter estende RecyclerView.Adapter<NotesAdapter.NoteViewHolder>

L'adapter è di fondamentale importanza perché consente ad una particolare View di mostrare il contenuto che ci serve.

11 Dopo aver creato l'adapter, lo aggiungiamo alla view interessata (R.V.) creando prima l'adapter: notesAdapter = new NotesAdapter(noteList) e poi lo aggiungiamo alla view con: recyclerView.setAdapter(notesAdapter); Il tutto nella MainActivity.
Implementiamo il metodo onActivityResult() per aggiornare la lista dopo essere tornati dalla view di aggiunta. Questo metodo usa getNotes() che legge da DB e aggiorna.