

SEMESTRÁLNA PRÁCA VAMZ

KeepInMind Tomáš Kotrík 559142



11. júna 2022

Žilinská univerzita

Fakulta Riadenia a Informatiky

# KeepInMind

**Špecifikácia zadania:**

KeepInMind je aplikácia na tvorbu a spracovanie vlastných poznámok.

Názov aplikácie som zvolil Keep In Mind ako metaforu aby si človek nemusel pamätať všetko, tak si to môže zapísať do aplikácie ktorá to bude držať v jej „mysli“. Aplikácia funguje len na *portrait* mód keďže som nevidel prípad kedy užívateľ potreboval si písať poznámky v landscape režime z tohto dôvodu je orientácia aplikácia konštantne daná na portrait mód.

**Aplikácie pre porovnanie zadania:**

Color Note:

*Odlišnosti KeepInMind:*

1. Ako prvú odlišnosť by som upozornil na vzhľad aplikácie ktoré podľa môjho názoru je mierne nevkusný... áno spĺňa prehľadnosť a podobne ale nieje ničím výnimočný.
2. Ako druhú odlišnosť nemá žiadne potvrdzovacie tlačítko na pridanie poznámky, poznámka sa pridá tým že sa posuniete späť čo mi príde ako zastrašujúci spôsob kedže užívateľ môže byť v domienke že sa mu poznámka neuloží správne. V mojej aplikácii pre pridanie poznámky je jasné tlačítko na uloženie poznámky a taktiež na vymazanie.
3. Ako tretie do aplikácie Color Note si nieje možné pridať fotku ani sa odfotiť ako je možné v aplikácii KeepInMind

Poznámkový blok – poznámky:

*Odlišnosti KeepInMind:*

1. V aplikácii nieje možné si zadať názov poznámky
2. Taktiež nezobrazuje dátum vytvorenia danej poznámky
3. Nedá sa vložiť fotka ani sa odfotiť
4. Nieje možné filtrovať jednotlivé poznámky
5. Nedá sa zmeniť layout zobrazenia poznámok

# Návrh riešenia problému

**Analýza spracovania problému:**

Vytvorenie úvodnej obrazovky

Vytvorenie layoutu zobrazovania poznámok

Pridávanie poznámok

Notifikácia pridania poznámok

Ukladanie poznámok

Otváranie už vytvorených poznámok

Update-ovanie poznámok

Mazanie poznámok

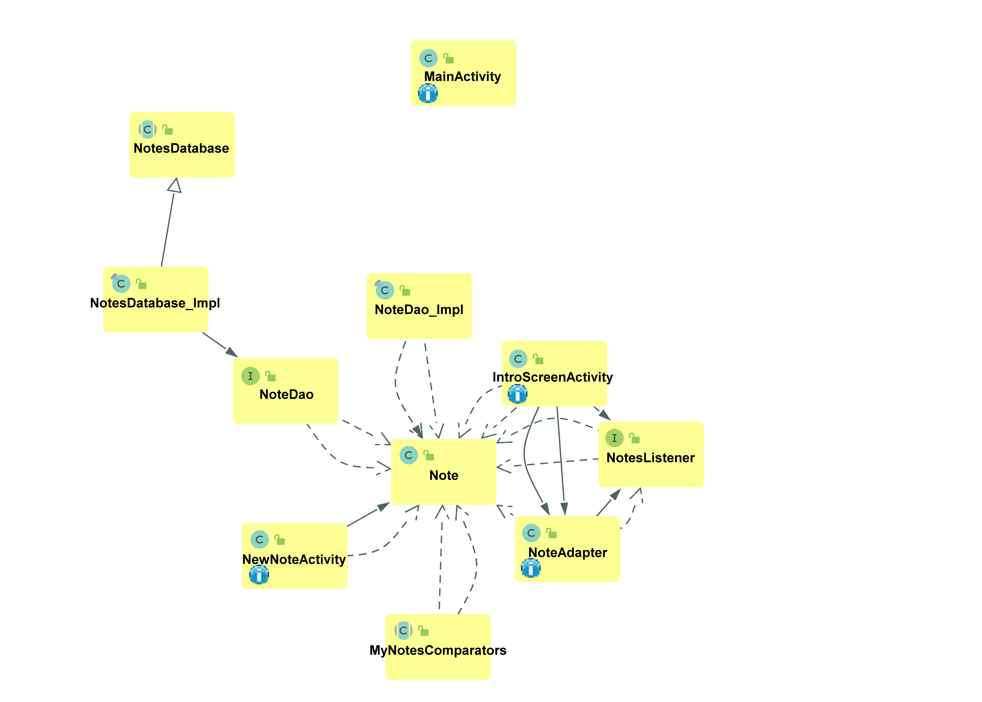
Pridávanie fotiek do poznámok

Customizácia poznámok

Vyhľadávanie poznámok

Triedenie poznámok podľa parametrov

Menenie layoutu zobrazenia poznámok



# Riešenie problému, aplikovanie analýzy

1. Ako oživenie a odlíšenie mojej aplikácie od iných som pridal na štart aplikácie do MainActivity náhodný obrázok z 20 tich obrázkov ktoré su vo forme citátu aby si človek spríjemnil deň alebo prišiel k nejakej zaujímavej myšlienke.

**XML:** Linear layout s horizontálnou orientáciou v ktorom je len jeden ImageView.

**Activity:** do array listu sa postupne pridajú obrázky z R.drawable adresára.

Následne pomocou Random generátora čísla z intervalu veľkosti array listu sa vygeneruje platný index a na Image View sa nastaví resource daného obrázka. Tento Image View má onClickListener ktorý užívateľa presunie na ďalšiu Aktivitu s názvom IntroScreenActivity.

1. *IntroScreenActivity* táto Aktivita rieši vytvorené poznámky ich triedenie, vyhľadávanie, otváranie,menenie layoutu zobrazenia poznámok.

Poďme sa bližšie pozrieť na XMLtejto Aktivity.

**XML:** Constraint Layout , ako prvé si môžeme povšimnút TextView ktorý zobrazuje nadpis aplikácie teda jej názov *KeepInMind*, názov je vycentrovaný a constraintnuté Top na parenta.

Ďalší komponent je ImageView tento komponent je viditelný hore vpravo, takže jeho constrainty sú na vrch parenta a jeho vrch na spodok vyhľadávacieho baru.

Na pomoc constraint layoutu prichádza Linear Layout ktorý bude pod sebou v horizonálnej orientácii podporovať rozmiestnenie search baru ktorý pozostáva s ImageView ktorý má obrázok „lupy“ a EditText ktorý ma v sebe hint SEARCH nech užívateľ vie že v tomto poli je možné vyhľadávať.

Následne na listovanie poznámok je použitý RecyclerView ktorý má jemný padding k úprave detailu a vzhľadu.

Pod RecyclerView sa nachádza tlačítko + ktorým sa pridáva poznámka.

Napokon spodná triediacia/filtrovacia „lišta“ vo forme TextView s názvami A-Z čo známená triedenia od A po Z.

Z-A triedenie od Z po A, tieto dve triedenia majú parameter Title poznámky čiže ich názov. TIME je triedenie poznámok podľa najstaršej poznámky. Tu sa ešte aj nachádza ImageView ktorý slúži ako refresh do pôvodného stavu.

**Activity:** obsahuje metódy na nastavenie RecyclerViewera na zobrazenie poznámok, triedenie poznámok, a vyhľadávanie a prekrytú metódu onNoteClicked, poďme sa bližšie pozrieť na tieto metódy.

private void initializeRecyclerView()

*Metoda initializeRecyclerView sluzi na uvodne nastavenie recycler view  
\* vytvori sa arraylist poznamok  
\* nastavi sa adapter pre recycler view*

private void getNotes(final int reqCode,final boolean isDeleted)

*metoda getNotes dostava do parametra request kod a ci nahodou nieje poznamka prave v stave deletovania  
\* vlozi sa asyncTask kde sa z databazy vytiahnu data.  
\* V onPostExecute sa podla request kodu uskutocnia dane algoritmy a potrebne funkcie*

private void filtering()

*/\*\*  
\* metoda filtering sluzi na sortovanie danych poznamok  
\* \*/*

private void initializeSearcher()

*\*\*  
\* uvedenie vyhladavania poznamok  
\* onTextChanged zastavi sa casovac pre vyhladavanie  
\* afterTextChanged ak je v liste aspon jedna poznamka tak zacne prehladavat  
\* \*/*

public void onNoteClicked(Note note, int position)

*/\*\*  
 \* Vlastna metoda onNoteClicked ktora je zdedena z interface NotesListener  
 \* kde sa zoberie momentalna pozicia poznamky a chysta sa na otvorenie poznamky  
 \* Request kod je v zmysle otvorenia cize pravdepodobne sa bude updatovat poznamka alebo minimalne prezerat  
 \*/*

3.*NewNoteActivity* je Aktivita ktorá sprostredkúváva pridávanie/mazanie a updatovanie poznámok.

**XML:**  Relative Layout v ktorom vo vnútri je ScrollView následne v Linear Layoute horizontálnej orientácii sú navkladané vrchné navigačné ImageView

Prvý image X je vrátenie späť, v strede image odpadkového koša, slúži na vymazanie poznámky, v pravo image + na pridanie poznámky.

Pod ním môžeme vidieť EditText ktorý je na zadanie názvu poznámky, pod ním je EditText ktorý slúži na obsah poznámky

**Activity:** obsahuje metódy na nastavenie počiatočného stavu, nastavenie či sa poznámka otvorila a updatovanie, vybratie obrázku z galérie, odfotenie z kamery, vytvorenie súboru na uloženie z kamery, nájdenie cesty daného súboru, vytvorenie notifikácie, vymazanie poznámky, uloženie poznámky inicializovanie customizácie.

private void setStuff()

*/\*\*  
 \* Zakladne inicializovanie vnutra poznamky  
 \* findViewById ...  
 \* \*/*

private void setViewOrUpdateNote()

*/\*\*  
 \* setViewOrUpdateNote je metoda na nastavenie danych fieldov ktore boli zapisane do poznamky a nasledne zobrazenie  
 \* ak bol pridany obrazok tak sa umozni tento obrazok vymazat  
 \* \*/*

private void handleOpenedNote()

*/\*\*  
 \* Ak sa dostalo z vnutorneho intentu aj boolean parameter true tak to znamena ze sa poznamka neotvori lebo bola vymazana  
 \* inak sa otvori a prebehne metoda setViewOrUpdateNote  
 \* \*/*

private void saveNote()

*/\*\*  
 \* saveNote metoda sluzi na ulozenie poznamky a poslanie do Room Databazy  
 \* nastavia sa potrebne parametre ktora poznamka dostala od uzivatela  
 \* ak uzivatel nezada Titulok poznamky tak mu aplikacia neumozni ist dalej  
 \* \*/*

private void initializeCustomDrawer()

*/\*\*  
 \* initializeCustomDrawer je metoda kde sa nastavia vsetky potrebne parametre pre customizovanie poznamky  
 \* cize farba font a pridany obrazok.  
 \* Ked si uzivatel voli farby alebo fonty tak sa zobrazuje aj ukazatel aku farbu alebo font si prave vybral  
 \* \*/*

private void selectImage()

*/\*\*  
 \* Vybratie z galerie obrazkov  
 \* \*/*

private File createImageFile()

*/\*\*  
 \* Vytvorenie obrazku z pouzitej kamery  
 \* kod zo stranky developer.android.com  
 \* \*/*

makeNotificationADDNote(Note note,String message)

*/\*\*  
 \* Vytvorenie notifikacie po pridani poznamky  
 \* do parametru poznamka a sprava ktora sa ma zobrazit v notifikacii  
 \* \*/*

*/\*\*  
 \* Uskutocnenie povolenia uzivatela na fotenie alebo vyberanie z galerie  
 \* \*/*   
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)

*/\*\*  
 \* openCameraToCaptureImage sluzi na zaznamenanie fotky z kamery pouzivatela  
 \* kod taktiez pomocka z developer.android.com  
 \* \*/*private void openCameraToCaptureImage()

*/\*\*  
 \* Kontrolovanie povolenia pre zapisovanie obrazka do suboru  
 \*  
 \* \*/* private void checkStoragePermission()

*/\*\*  
 \* Nasledne pridanie obrazka do suboru  
 \* \*/*private void galleryAddPic()

*\*\*  
 \* Zmensenie obrazka aby nezaberal tolko miesta riesenie tvorene v zmysle zmensenia bitmapy  
 \* \*/* private void compressImage(Uri imageUri)

*/\*\*  
 \* onActivityResult ak je request kod na snimanie kamery tak sa bude snimat kamera  
 \* ak je request kod na vybratie obrazka z galerie tak sa vybera z galerie  
 \* \*/* @Override  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)

*/\*\*  
 \* getPathFromUri vrati cestu k obrazku aby som ho vedel nasledne zobrazit  
 \* \*/* private String getPathFromUri(Uri content)

*/\*\*  
 \* deleteNote sluzi na vymazanie poznamky z databazy a celkovo z listu poznamok  
 \* do intentu delete sa vlozi boolean parameter true aby sa vedelo ze sa ide mazat poznamka  
 \* \*/*public void deleteNote()

# Triedy

**Note :**

*/\*\*  
 \** ***@author*** *TOMAS KOTRIK 2022  
 \* Trieda Note je entita pre Room databazu  
 \* sklada sa z idcka, nazvu poznamky, obsahu poznamky,datumu+casu, obrazku, a farby poznamky  
 \* \*/*@Entity(tableName = "notes")  
public class Note implements Serializable {  
  
 @PrimaryKey(autoGenerate = true)  
 private int id;  
  
 @ColumnInfo(name = "title")  
 private String title;  
  
 @ColumnInfo(name = "time")  
 private String time;  
  
  
 @ColumnInfo(name = "inputNote")  
 private String inputNote;  
  
 @ColumnInfo(name = "imagePath")  
 private String imagePath;  
  
 @ColumnInfo(name = "color")  
 private String color;

Trieda Note nadalej obsahuje jednoduche gettere a settere pre dané fieldy.

**NoteDao:**

*/\*\*  
 \* NoteDao je DAO od room databazy kde je jednoduchy query select pre list poznamok kde sa vybere vsetko  
 \* a insert pre vlozenie poznamky , delete pre odstranenie poznamky  
 \* \*/*@Dao  
public interface NoteDao {  
  
 @Query("SELECT \* FROM notes ORDER BY id DESC")  
 List<Note> notesList();  
  
 @Insert(onConflict = OnConflictStrategy.*REPLACE*)  
 void insertNote(Note note);  
  
 @Delete  
 void deleteNote(Note note);  
}

**NotesDatabase:**

*/\*\*  
 \** ***@author*** *TOMAS KOTRIK 2022  
 \* Room Databaza robena z cviceni  
 \* \*/*@Database(entities = Note.class,version = 1,exportSchema = false)  
public abstract class NotesDatabase extends RoomDatabase {  
  
 private static NotesDatabase *notesDatabase*;  
  
 public static synchronized NotesDatabase getDatabase(Context context) {  
 if (*notesDatabase* == null) {  
 //vytvorenie databazy  
 *notesDatabase*= Room.*databaseBuilder*(context,NotesDatabase.class,"notes\_database").build();  
 }  
 return *notesDatabase*;  
 }  
 //pripustenie DAO  
 public abstract NoteDao noteDao();  
}

**NotesListener:**

*/\*\*  
 \** ***@author*** *TOMAS KOTRIK 2022  
 \* NotesListener je moj interface kde vlastne sa riesi len metoda  
 \* ked sa klikne na danu poznamku nech sa vie aplikacia spravat  
 \* \*/*public interface NotesListener {  
 void onNoteClicked(Note note , int position);  
}

**MyNotesComparator:**

*/\*\*  
 \** ***@author*** *TOMAS KOTRIK 2022  
 \* Vlastna trieda na komparovanie poznamok pomocou Comparable porovnavam jednotlive fieldy poznamok  
 \* a vraciam ich v poradi akom potrebujem cize nazov zostupne/vzostupne a datum zostupne  
 \* \*/*public abstract class MyNotesComparators implements Comparable<Note> {  
  
  
 public static Comparator<Note> *compareAZ* = new Comparator<Note>() {  
 @Override  
 public int compare(Note o1, Note o2) {  
 return o1.getTitle().compareTo(o2.getTitle());  
 }  
 };  
  
 public static Comparator<Note> *compareZA* = new Comparator<Note>() {  
 @Override  
 public int compare(Note o1, Note o2) {  
 return o2.getTitle().compareTo(o1.getTitle());  
 }  
 };  
  
 public static Comparator<Note> *compareTime* = new Comparator<Note>() {  
 @Override  
 public int compare(Note o1, Note o2) {  
 return o1.getTime().compareTo(o2.getTime());  
 }  
 };  
  
}

**NoteAdapter:**

*/\*\*  
 \** ***@author*** *TOMAS KOTRIK 2022  
 \* Trieda NoteAdapter dedi od recyclerViewu jeho adapter a dotvara ho este poznamkovy view holder  
 \* Tu sa vpodstate nastavuje Vizualizacia poznamok pre recycler view  
 \* \*/*public class NoteAdapter extends RecyclerView.Adapter<NoteAdapter.NoteViewHolder> {  
 //notes su vseobecne poznamky  
 private List<Note> notes;  
 private NotesListener notesListener;  
 //casovac pre priebeh vyhladavania poznamky  
 private Timer timer;  
 // source notes su poznamky pre vyhladavanie  
 private List<Note> sourceNotes;  
  
 public NoteAdapter(List<Note> notes, NotesListener notesListener) {  
 this.notes = notes;  
 this.notesListener = notesListener;  
 this.sourceNotes = notes;  
 }  
  
 */\*\*  
 \* Vytvorenie View holdera ktory vytvori vzhlad danej poznamky  
 \* \*/* @Override  
 public NoteViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
 return new NoteViewHolder(LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*item\_note\_view*,parent,false));  
 }  
  
 */\*\*  
 \* Bindovanie view holdera v zmysle nastavenie onclick Listenera kde prekriva metodu onNoteClicked  
 \* \*/* @Override  
 public void onBindViewHolder( NoteViewHolder holder, int position) {  
 holder.initializeNote(notes.get(position));  
 holder.layoutNoteItem.setOnClickListener(view -> notesListener.onNoteClicked(notes.get(position),position));  
 }  
  
 */\*\*  
 \* Vrati velkost vsetkych poznamok cize kolko poznamok je momentalne v liste  
 \* \*/* @Override  
 public int getItemCount() {  
 return notes.size();  
 }  
  
 */\*\*  
 \* Vrati poziciu danej poznamky  
 \* \*/* @Override  
 public int getItemViewType(int position) {  
 return position;  
 }

*/\*\*  
 \* Vyhladavanie poznamok  
 \* nastavi sa casovy Task kde je delay 500ms  
 \* postupne sa vyhlada dana poznamka v zmysle containst takze nemusi byt presne zadana  
 \* staci ak si pouzivatel pamata priblizny nazov  
 \*  
 \* \*/* public void searchNotes(final String key) {  
 timer = new Timer();  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 if (key.trim().isEmpty()) {  
 //ak parameter zadaneho textboxu je prazdny tak sa nic nevyhlada teda zobrazia sa vsetky poznamky  
 notes = sourceNotes;  
 } else {  
 //ak parameter nieje "" tak sa ide hladat temp list ako docasny kde sa budu vkladat zhody najdenia  
 ArrayList<Note> temp = new ArrayList<>();  
 for (Note note : sourceNotes) {  
 //for each cez poznamky ak sa zadany nazov nachadza tak sa prida do temp listu  
 if (note.getTitle().toLowerCase().contains(key.toLowerCase()) ||  
 note.getInputNote().toLowerCase().contains(key.toLowerCase())) {  
 temp.add(note);  
 }  
 }  
 //po pridany sa povodny list nahradi zhodami vo vyhladavani  
 notes = temp;  
  
 }  
 new Handler(Looper.*getMainLooper*()).post(new Runnable() {  
 @Override  
 public void run() {  
 //kym uzivatel hlada tak sa updatuje view  
 notifyDataSetChanged();  
 }  
 });  
 }  
  
 },500);  
 }  
 //zrusenie timeru  
 public void cancelTimer() {  
 if (timer != null) {  
 timer.cancel();  
 }  
 }  
}

**NoteViewHolder:**

*/\*\*  
 \* Vnutorna trieda ktora nastavuje cely obsah danej poznamky na zovnajsku cize v IntroScreenActivity  
 \* \*/*public class NoteViewHolder extends RecyclerView.ViewHolder {  
  
 TextView titleText, timeText,imageNoteText;  
 LinearLayout layoutNoteItem;  
  
 public NoteViewHolder( View itemView) {  
 super(itemView);  
 //nastavenie fieldov  
 titleText = itemView.findViewById(R.id.*titleText*);  
 timeText = itemView.findViewById(R.id.*timeText*);  
 layoutNoteItem = itemView.findViewById(R.id.*item\_note\_view\_layout*);  
 imageNoteText = itemView.findViewById(R.id.*imageNoteText*);  
  
 }  
  
 */\*\*  
 \* initializeNote metoda sluzi na Nastavenie potrebnych fieldov a zobrazenie farby a  
 \* ak je aj obrazok vo vnutri poznamky tak sa ukaze text ze poznamka obsahuje obrazok  
 \* \*/* void initializeNote(Note note) {  
 titleText.setText(note.getTitle());  
 timeText.setText(note.getTime());  
 //ak pouzivatel zadal customizovanu farbu poznamky tak sa nastavi inak je defaultne tmava  
 GradientDrawable gradientDrawable = (GradientDrawable) layoutNoteItem.getBackground();  
 if (note.getColor() != null) {  
 gradientDrawable.setColor(Color.*parseColor*(note.getColor()));  
  
 } else {  
 gradientDrawable.setColor(Color.*parseColor*("#292929"));  
 }  
 //ak sa nachadza obrazok tak nastavi text ze poznamka obsahuje obrazok inak zakryje dany text  
 if (note.getImagePath() != null && !note.getImagePath().trim().isEmpty()) {  
 imageNoteText.setVisibility(View.*VISIBLE*);  
 } else {  
 imageNoteText.setVisibility(View.*GONE*);  
 }  
 }  
}

# Použité prvky

Aktivity fungujú s Room databázou to znamená že priebežný stav a celkové uloženie riešia asyncTasky do databázy. Životný cyklus postačuje na on Create a on Destroy keďže sa z aplikácie nikde netreba presúvať a neukrýva žiadne skryté údaje kde by sa užívateľ nemal dostať pomocou stlačenia tlačítka vrátiť späť. Použitý Recycler View a sním aj adaptér a View Holder. V možnosti dva layouty rozloženia Recycler View a to grid staggered layout so spanom 2 a klasický grid layout. Notifikácia pri vytvorení poznámky. Použitie managovania storagu pre výber obrázka z galérie a tak isto aj senzor v podobe kamery a odfotenie. Vlastné témy a štýly použité pri orámovaní niektorých komponentov.

# Zdroje:

<https://developer.android.com/guide>

[https://stackoverflow.com](https://stackoverflow.com/)

<https://www.youtube.com/watch?v=4BuRMScaaI4>

<https://www.youtube.com/watch?v=lwAvI3WDXBY>

<https://www.youtube.com/watch?v=UtfFgJxvM4Q&t=458s>

<https://developer.android.com/reference/android/os/AsyncTask>

<https://developer.android.com/training/camera/photobasics>

<https://www.youtube.com/watch?v=FRGkKMWpn08>