Luggage Organizer

Мобилно приложение за по-лесно приготвяне на багаж за пътуване

Изготвил: Радослав Здравков

Фак. № 121221004

ФКСТ, КСИ, 47гр



Съдържание:

1.Увод

2.Проектиране и идея на приложението

3.Реализация на приложението

4.Потребителско ръководство

5.Заключение за приложението

1. Увод

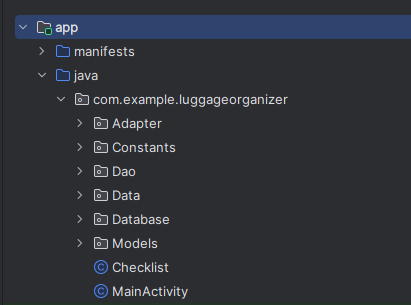
Luggage Organizer е мобилно приложение, целящо да помогне на хора, които предстои да пътуват и трябва да приготвят своя багаж без да пропуснат нещо важно. Приложението предоставя чеклисти за различни категории, в които са включени най-нужните неща при пътуване. Luggage Organizer има функционалност за добавяне на потребителски предмети към съществуващите категории, както и за изготвяне на персонален чеклист. Често пътуващите се случва да забравя вещи, които е нужно да включат в багажа си. Чеклистът би им помогнал за организацията на нещата, които трябва да се вземат при пътуване и би гарантирал, че нищо важно няма да бъде пропуснато. Приложението може да предложи списък с неща, които са разпределени в подходящи за типа на пътуването категории(), което би спестило време на пътуващите при подготовката на багажа. Друг голям поблем, който приложението разрешава е стресът при приготвяне на багаж – всички знаем колко е стресиращо в последния момент преди важно пътуване да започнем да мислим „Ами ако съм забравил нещо?“, „Дали взех всичко нужно?“. Подобни мисли неминуемо водят до голям стрес и в крайна сметка и да сме забравили нещо, шансът да се сетим навреме е малък. Голям плюс на Luggage Organizer е възможността за персонализация. Така всеки потребител има възможност да направи чеклистите с вещи за пътуване според своя вкус и нужди. Освен всичко в My Selections могат да се видят всички текущо маркирани вещи за предстоящото пътъване, което прави по-лесно подсещането, ако все пак сме пропуснали нещо.

2. Проектиране и идея на приложението

Приложението трябва да съдържа основни категории с най-често носените от човек принадлежности при пътуване. Важна част от Luggage Organizer ще е възможността всеки да може да добавя към съществуващите основни категории още предмети, както и да има къстъм списък, в който да може също да се добавят потребителски вещи. Приложението ще използва база данни, за да може селектираните предмети и новите предмети добавени от потребителя да се запазват след затварянето му. Потребителският интерфейс ще е интуитивен с анимирани картинки, по които потребителят лесно ще може да се ориентира. Крайният продукт ще е подходящ за всякакви потребителски групи – без значение от пол и възраст. Продуктът ще е удобен за използване от хора, които просто искат да подсигурят пътуването си и да му се насладят без излишни притеснения. Приложението няма да изисква регистрация и няма да съхранява лични данни, затова не е нужно и предпазване на данните от изтичане.

3. Реализация на приложението

В основната папка на приложението са разделени следните папки: Adapter, Constant, Dao, Data, Database и Models, в които кодът е логически раделен.



В папката Adapter се намират класовете Adapter.java и CheckListAdapter.java. Адаптерът ни служи, за да може да „закачим“ всички view-та и налседява RecyclerView.ViewHolder. Използва се за дефиниране на компоненти на елементите в RecyclerView. Този клас държи препратки към всички изгледи, които са част от един елемент на списъка и се използват за показване на данни. В конструктора на класа се инициализират препратки към изгледите, които ще бъдат попълнени с данни за всеки елемент на списъка.

Конструктор на класа Adapter:

public Adapter(Context context, List<String> titles, List<Integer> images, Activity activity) {  
 this.titles = titles;  
 this.images = images;  
 this.activity = activity;  
 this.inflater = LayoutInflater.*from*(context);  
}

Методът onCreateViewHolder се извиква при създаването на нов ViewHolder обект. Той създава нов ViewHolder обект, който ще бъде свързан с XML шаблона за изгледа на един елемент на списъка.

Методът onBindViewHolder се извиква, когато елементът в списъка трябва да бъде попълнен с данни. Той се използва за връзката между данните и изгледите на елементите.

Методът getItemsCount връща броят на добавените категории.

Класът MyViewHolder е вътрешен в класа Adapter.

public MyViewHolder(@NonNull View itemView) {

super(itemView);

title = itemView.findViewById(R.id.title);

img = itemView.findViewById(R.id.img);

linearLayout = itemView.findViewById(R.id.linearLayout);

}

Този конструктор служи за инициализиране на ViewHolder обекта със съответните изгледи, които са част от един елемент на списъка. След като ViewHolder е инициализиран, той може да се използва за показване на данни за съответния елемент в списъка.

Клас CheckListAdapter

В папката Constants се намира класът AppConstants.java, който съдържа стрингове, които се използват на други места в кода.

В папка Dao имаме интерфейса ItemsDao, тъй като приложението изпозлва RoomDB за базата данни. Room е библиотека за управление на бази от данни в Android, която осигурява абстракция над SQLite базите данни и предоставя по-лесен начин за работа с тях. Грижи се за много от досадните детайли като отваряне и затваряне на връзката с базата данни, генериране на SQL заявки и други. DAO (Data Access Object) интерфейсът е част от архитектурата на бази от данни при използване на Room в Android Studio. Той представлява интерфейс, който дефинира методите за извършване на операции върху базата данни. Интерфейсът е обозначен с анотация @Dao. Използвани са и анотациите @Delete – за изтриване на предмет, @Insert – за добавяне на предмет и @Query за къстъм заявки към базата.

@Insert(onConflict = *REPLACE*)  
void saveItem(Items items);

@Query("select \* from items where category=:category order by id asc")  
List<Items> getAll(String category);

@Delete  
void delete(Items items);

В папката AppData е класът AppData, който наследява Application и служи за управление на приложението. Когато приложението се стартира, методът onCreate се изпълнява преди всички други останали компоненти на приложението. Именно затова в класа се инициализират системните предмети, разпределени в различните категории. За всяка категория има get метод, който създава съдържанието на всяка категория чрез метода prepareItemsList.

public List<Items> getBasicData() {  
 String[] items = {"Bank Card", "Passport", "Driver License", "Wallet", "Tickets", "Keys", "Umbrella"};  
 return prepareItemsList(*BASIC\_NEEDS\_CAMEL\_CASE*, items);  
}

В класа са и методите getAllData и persistAllData. Първият връща списък от списъците с предметите по категории, а вторият – служи за запазване на всичко в базата данни.

public void persistAllData(){  
 List<List<Items>> listOfAllItems = getAllData();  
 for(List<Items> list: listOfAllItems){  
 for(Items items: list){  
 database.mainDao().saveItem(items);  
 }  
 }  
 System.*out*.println("Data added.");  
}

В папката Database, логично, се намира класът RoomDB, който наследява RoomDatabase и в него се създава самата база данни, ако тя не съществува.

*database* = Room.*databaseBuilder*(context.getApplicationContext(), RoomDB.class, *DATABASE\_NAME*)  
 .allowMainThreadQueries()  
 .fallbackToDestructiveMigration()  
 .build();

Класът Items от папката Models пресъздава всеки един предмет, като съдържа информация за името му, категорията, към която спада, кой го е добавил и дали е отбеляза/чекнат. Чрез анотацията @Entity се посочва, че класът ще представлява таблица в базата данни. ID-то е с анотация @PrimaryKey, защото играе ролята на първичен ключ в таблицата и започва от 0 и е autoincremented. Останалите член-полета на класа са с анотация @ColumnInfo, което показва, че представляват колкони в таблицата.

@PrimaryKey(autoGenerate = true)  
int ID = 0;  
  
@ColumnInfo(name = "item\_name")  
String itemName;

За всяка член-променлива на класа има създадени аксесори и мутатори.

Използваните ресурси и xml файлове за FrontEnd частта се намират в папка res – папките drawable, layout, values.

A screenshot of a computer

Description automatically generated

В xml файловете чрез тагове се позиционират различните елементи върху екрана.

4. Потребителско ръководство

Използването на приложението е много лесно и интуитивно. След стартирането му се зарежда екран, съдържащ всички категории за организиране на багаж – Basic Needs, Clothing, Personal Care, Baby Needs, Health, Technology, Food, Beach Supplies, Car Supplies, Needs и My List, в който потребителят може да добавя персонални неща. Всяка категория освен подходящо име има и подходящо изображение. След натискане на някоя от категориите се визуализира списък с предмети, които могат да бъдат добавени към багажа или изтрити чрез кошчето отдясно на предмета. В долната част на екрана има теккстово поле за добавяне на предмет към списъка в текущата категория. При добавяне се написва името на предмета и се натиска тикчето вдясно на текстовото поле. My List служи за добавяне на потребителски принадлежности, а My Selections показва всички избрани до момента предмети. За излизане от дадената категория има back бутон в горния ляв ъгъл на екрана.

5. Заключение

Продуктът е готов да бъде използван от хора, които искат да улеснят приготвянето на багажа си. Той може да бъде верен другар на всеки, който често пътува или има постоянни притеснения при подготвяне на багажа си за път да не пропусне нещо важно.