Arrow Player

курсов проект по Мобилни приложения

на Ивелина Иванова, фак. номер 61572

Съдържание

Описание на задачата

Създаване на мобилно приложение. Условно ще разделим проектите на 3 нива – базово, междинно, високо.

Разделянето е от гледна точка на усилията, които ще костват разработката на проект и Android APIs, които ще използвате за това.

Създаване на мобилно приложение. Условно ще разделим проектите на 3 нива – базово, междинно, високо.

Разделянето е от гледна точка на усилията, които ще костват разработката на проект и Android APIs, които ще използвате за това.

1) Базово – в тази групата са проектите представляващи управление на UI контроли.

Какво представлява базовото ниво?

Android app, който визуализира поне едно Activity, което съдържа базови UI компоненти; Като единствено изискване към приложението -да осигурява коректен вход от UI компонентите, тоест те да проверяват потребителските данни и евентуално предупреждават потребителят за некоректен вход.

Като бонус за тази група, ще смятаме поддръжката на layouts за различни устройства, според Google Guidelines (ref: 4.what-are-resources.pdf).

2) Междинно – изискванията на ниво 1), като към тях добавяме: Persistence (по най-лекият за Вас начин) и повече от едно Activity. Пример за достатъчно лесен начин да сътворим второ Activity с достатъчен смисъл за приложението – Shared Preferences за това да помним настройките на приложението и Activity което ги управлява.

3) Високо – изискванията на 2), като към тях добавяме „background processing“ или „мрежа“.

Под мрежа разбираме, най-много прости http get/post заявки и съответната „инфраструктура“, която ги обезпечава. Всеки проекти, би следвало да представлява работещо приложение, което човек може да изпробва поне в среда на емулатор.

Реализираният проект е мобилно приложение музикален плейър. Приложението е имплементирано за операционна система Android, минимално ниво на SDK 16 (Android 4.1, 4.1.1- Jelly Bean) и целево ниво на SDK 23 (Android 6.0 - M).

Функционални изисквания

- Приложението трябва да достъпва музикалните файлове на устройството и да ги показва по папки.
- Създаване, редактиране и изтриване на плейлисти от песни
- Възпроизвеждане на уеб радио
- Контрол върху възпроизвеждането през нотификационния панел на устройството
- Повтаряване на песен или на всички песни от списък, разбъркване на реда за възпроизвеждане
- Търсене на песен по заглавие или артист
- Възпроизвеждане на подкаст
- Възможност за команди с разтърсване на устройството

Необходими разрешения

- READ_PHONE_STATE (read phone status and identity): необходимо за засичане на входящо повикване, за да бъде оставено на пауза приложението;
- WAKE_LOCK (prevent phone from sleeping): необходимо, за да продължи музиката, когато телефонът заспи;
- ACCESS_NETWORK_STATE (view network connections): необходимо за засичане на мрежи;
- INTERNET (full network access): необходимо за слушане на радио и сваляне на подкасти;
- READ_EXTERNAL_STORAGE (test access to protected storage): необходимо за добавяне на списък от уеб радиа;
- WRITE_EXTERNAL_STORAGE (modify or delete contents of your SD card): необходимо за експорт на списък с уеб радиа.

Реализация

Използвани Android APIs

- *android.content*

Съдържа класове за достъп и публикуване на данни. Включва три основни типа API-та:

  - Content sharing (android.content)

За споделяне на съдържание между компонентите на приложението. Най-важните класове са:

    - ContentProvider и ContentResolver за управление и публикуване на persistent данни, асоциирани с приложението.
    - Intent и IntentFilter, за предаване на структурирани съобщения между компонентите на приложението —позволявайки на компоненти да инициират други компоненти и да върнат резултат.
  - Package management (android.content.pm)

За достъпване на информация за пакет на Android (APK), включвайки информация за неговите дейности, разрешения, сървиси, сигнатури. Най-важният клас за такава информация е PackageManager.

  - Resource management (android.content.res)

За връщане на данни, асоциирани с приложението като стрингове, drawables, медиа и конфигурационни детайл. Най-важният клас е Resources.

- *android.os*

Предлага базови сървиси на операционната система, предаване на съобщения и комуникация между процесите на устройството.

- *android.support*

Android Support Library предлага различни свойства, които не са включени във фреймуърка. Тези библиотеки предлагат съвместимост със стари версии на операционната система, предлагат полезни елементи на потребителския интерфейс, които не са част от фреймуърка и предлагат широк набор от средства, на които приложението може да разчита.

- *android.text*

Предлага класове, изолзвани за рендиране или проследяване на текст и спанове с текст по екрана.

Тези класове могат да се използват за дизайн на собствени widget-и за управление на текст, за управление на спорни случаи с промени на спанове с текст.

Интерфейсите и класовете, започващи със Span… се използват за създаване и управление на спанове от текст в View елемент. Може да се използват за стилизиране на текст или фон или за следене на

промени. Ако бъде създаден собствен widget, трябва да се наследи DynamicLayout, за да се управлява компонента, обвиващ тескта.

- *android.widget*

Пакетът съдържа най-вече визуални елементи от потребителския интерфейс. Също могат да бъдат създавани и собствени.

За създаване на собствен widget, трябва да се наследи View или негов подклас. За да се използва собствен widget в layout XML, има два допълнителни файла, които трябва да бъдат създадени:

    - Java implementation file – Това е файла, носител на поведението на widget-а. Ако може да се инстанциира обекта от layout XML, ще трябва да се напише и конструктор, който намира всички стойности на атрибути от layout XML.
    - XML definition file - XML файл в res/values/, който дефинира елемента, използва за инстанциране на widget-а и атрибутите, които поддържа.
    - Layout XML [по желание]- Незадължителен XML файл в res/layout/, който описва дизайна на widget-а.
- *android.app*

Съдържа класове от високо ниво, които енкапсулират целия модел на андроидското приложение.

Приложение за Андроид се дефинира с един или повече от четирите главни андроидски компонента. Два от тези компонента са дефинирани в този пакет – Activity и Service. Другите да се намират в android.content пакета : BroadcastReceiver and ContentProvider.

Activity е компонент, който осигурява екран, с който потребителя може да взаимодейства, за да извърши дейност, като набиране на номер, заснемане на кадър и други. Едно activity може да даде начало на други activity-та, включително такива, които живеят в рамките на други приложения.

Service е компонент, който извършва операции с дълъг живот на заден фон без потребителски интерфейс. Например, service може да се занимава с мрежови транзакции, да възпроизвежда музика и др.

Класът Fragment е също важен за дизайна на приложението- особено когато приложението е предназначено за по-големи резолюции като таблети. Един fragment дефинира определена част от поведението на activity, включително и свързано с потребителския ѝ интерфейс. Има собствен цикъл на живот, подобен на този на activity, и може да съществува заедно с друг фрагменти, внедрени в това activity. Докато едно activity върви, може да се добавят и премахват фрагменти.

Пакетът също дефинира средства като диалози, нотификации и action bar.

- *android.preference*

Осигурява класове, които управляват настройките на приложението и имплементира потребителския интерфейс на настройките. Грижи се настройките във всяко приложение да са поддържани по един и същ начин и потребителското изживяване да е консистентно в системата и приложенията.

Настройките в приложението трябва да бъдат изпълнявани като отделно activity, което наследява класа PreferenceActivity. В PreferenceActivity обектът PreferenceScreen трябва да бъде кореновия елемент на изгледа. PreferenceScreen съдържа елементи като CheckBoxPreference, EditTextPreference, ListPreference, PreferenceCategory или RingtonePreference.

Всички настройки направени за определен Preference ще бъдат автоматично запазени в инстанцията на SharedPreferences за приложението. Достъпът до SharedPreferences е лесен чрез getSharedPreferences().

Важно е, че запазените настройки са достъпни само през приложението, което ги е създало.

- *android.view*

Осигурява класове, които дават достъп до основни класове на потребителския интерфейс, които управляват екранния изглед и взаимодействието с потребителя.

- *android.graphics*

Осигурява графични средства на ниско ниво като canvas, цветови филтри, правоъгълници, които позволяват рисуване върху екрана директно.

- *android.util*

Осигурява основни полезни методи като манипулация на дата/час, base64 енкодери и декодери, преобразуване на стринг и число, XML средства и др.

- *android.media*

Осигурява класове, които управляват различни медиа интерфейси за аудио и видео.

Апитата на Media се използват за възпроизвеждане, а в някои случаи и запис на медийни файлове. Това включва аудио (например MP3, рингтонове, аудио ефекти за игри и други) и видео (видео стрийминг от интернет или локално запаметено).

Други специални класове от пакета предлагат възможността за откриване на човешки лица в Bitmap изображения (FaceDetector), управление на аудио рутинга (до устройството или слушалките) и контрол върху рингтоновете или вибрациите (AudioManager).

- *android.telephony*

Осигурява апита за следене на основна информация за телефона като тип на мрежата и състояние на връзката.

- *android.hardware*

Осигурява съпорт за хардуерни свойства като камера и сензори.

- *android.net*

Класове, които помагат при мрежова връзка освен основните апита на java.net.*

Архитектура на приложението

Адаптери

Адаптерът служи за мост между AdapterView и данните към това view. Адаптерът осигурява достъп до данните. Също така адаптерът е отговорен да направи View за всеки елемент от дейтасет. В случая на това приложение има два адаптера: MusicPlayerAdapter и SearchResultsAdapter. Първият се занимава с данните на browser-а, а втория с тези на търсачката.

```
public class MusicPlayerAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder>
```

```
public class SearchResultsAdapter extends
RecyclerView.Adapter<SearchResultsAdapter.SearchResultsViewHolder>
```

База данни

Използва се SQLite. В случая в нея се пазят плейлисти, подкасти и радиа.

```
public class PlaylistsDatabase extends SQLiteOpenHelper
```

```
public class PodcastsDatabase extends SQLiteOpenHelper
```

```
public class RadiosDatabase extends SQLiteOpenHelper
```

Фрагменти

Фрагмент е елемент на потребителския интерфейс, който може да бъде поставен в activity. Взаимодействието с фрагментите се осъществява през FragmentManager, който се достъпва през Activity.getFragmentManager() и Fragment.getFragmentManager(). Основният е абстрактният клас MusicPlayerFragment.

```
public abstract class MusicPlayerFragment extends Fragment
```

BrowserFragment го наследява, като добавя специфични за браузъра методи, свързани с навигирането към началната папка, към друга папка и сменянето на директория. PlaylistFragment също го наследява, като добавя методи за създаване, редакция и изтриване на плейлист, добавяне и изтриване на песен от плейлист, ъпдейт на плейлист и сортиране на плейлист. PodcastsFragment отново го наследява, като добавя методи за добавяне и управление на подкасти. RadioFragment е последният фрагмент, наследяващ MusicPlayerFragment. Той добавя методи за създаване и управление на уеб радиа.

Модели

Моделите са класове, описващи типовете елементи/данни, с които ще се работи. В приложението те са:

```
public class BrowserDirectory
```

```
public class BrowserSong implements PlayableItem, Serializable
public class Information
public interface PlayableItem
public class Playlist
public class PlaylistSong implements PlayableItem
public class Playlists
public class Podcast
public class PodcastEpisode implements PlayableItem
public class Radio implements PlayableItem
```

Viewholders

Един Viewholder описва един view и метаданните за мястото му в RecyclerVew. В случая има по един за директория, хедър, плейлист, подкаст, подкаст епизод, радио и песен.

Activities

Освен основното activity MainActivity, бяха отделени такова за информация за приложението, за настройките и за търсенето.

Services

Бяха отделени сървиси като MusicService, който отговаря за целия процес по възпроизвеждане на музиката, PodcastEpisodeDownloaderService, който се грижи за свалянето на епизод на подкаст. Също така бяха отделени различни класове тулове, в които за целите на документацията няма да задълбаваме.

Екрани из приложението