B202-BI-AND - Programování pro operační systém Android

Semestrální práce

Téma semestrální práce bude společné pro všechny studenty. Je jím vytvoření aplikace pro čtení RSS a ATOM feedů. Funkce aplikace budou následující:

1. Přidávat a mazat feedy, ze kterých se budou data načítat.
2. Zobrazení nadpisů článků ze všech feedů v listu pod sebou.
3. Zobrazení detailu článku po kliknutí na položku v listu.
4. Možnost přejít z detailu článku na web, ze kterého pochází.
5. Možnost sdílet článek ostatním aplikacím.
6. Automatická synchronizace článků každých pár hodin.

Během semestru budou 3 kontrolní body, na kterých vždy předvedete zadanou část semestrální práce. Po každém kontrolním bodu dostanete ke stažení referenční implementaci, ze které si budete moci doplnit, co vám bude chybět, či opravit, co budete mít špatně.

Cílem je vytvořit funkční Android aplikaci, která má rozumnou strukturu a drží se doporučených guidelines. Tento způsob řešení by vám měl napomoci vyvarovat se zbytečných chyb, kterých se začátečníci často dopouští. Zároveň na konci semestru budete mít k dispozici rámcovou aplikaci, do které budete moci nahlédnout při tvorbě vašich dalších projektů.

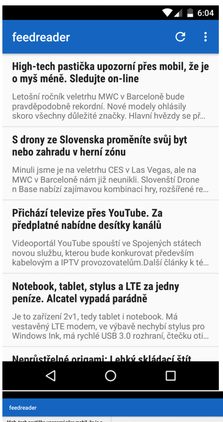
Knihovny

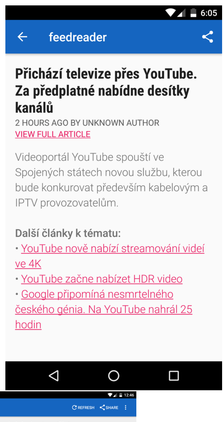
Pro parsování RSS a ATOM feedů je vhodné použít nějakou již hotovou knihovnu, než si ho implementovat sami. Oba standardy jsou poměrně rozsáhlé a i když zvládnete parsovat jejich nejjednodušší formu, obsahují spousty verzí a výjimek, se kterými nebudete počítat.

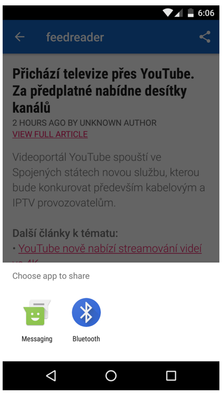
Námi zvolená a odzkoušená knihovna je port knihovny ROME pro parsování RSS a ATOM feedů pro Android. Níže uvedený zip obsahuje také port knihovny JDOM pro Android, na které ROME závisí. Obě knihovny nahrajte do adresáře **libs** v projektu. Pokud pracujete s Android Studio a používáte defaultně vygenerovaný build.gradle, měly by se vám automaticky načíst. V ostatních případech bude nutné přidat vše ručně.

feedreader-libs.zip ke stažení. Na stránkách projektu Android ROME najdete novější verzi knihovny.

Screenshoty







Formální náležitosti

- **minSdkVersion** bude nastaveno na API Level 21.
- **targetSdkVersion** bude nastaveno na API Level 28.
- **package** bude mít následující formát: cz.cvut. (např. cz.cvut.novakj1)
- Je **nutné používat Android Studio a Gradle** (případně IntelliJ IDEA). Doporučuje se použít Android Lint pro kontrolu projektu.
  - http://developer.android.com/sdk/index.html
  - Postup instalace rovněž k nalezení v informacích k prvnímu cvičení.
- Všechny **řetězce** (String) z uživatelského rozhraní **nesmí být součástí přímo zdrojového kódu** (hard-coded). Budou umístěny v Resources a bude na ně odkazováno resourceId.
- Pro klíče, které se používají pro přenos hodnoty, v rámci Bundle používejte výhradně konstanty referencované z jednoho místa. Toto platí i pro obdobné případy.
- Ikona aplikace (typicky ic_launcher) nebo další grafické prvky mohou být libovolné. Pro Refresh, Share a další typické a běžně používané prvky ideálně použít oficiální ikony přímo z SDK.
- Podoba aplikace a fukněnost bude přesně vycházet z aplikace zveřejněné na Play Store. Týká se to zejména:
  - Počtu Activit a Fragmentů a jejich rozložení pro mobilní telefony a tablety
  - Způsobu zobrazení načítání / obnovování seznamu feedů
  - Detailu článku, přidávání nových feedů a nastavení
  - Nepřidávejte nebo neupravujte nic oproti tomu, co je v ukázkové aplikaci. Není nutné pouze přidávat sekci About
  - Není nutné dodržovat přesně velikosti písma, výšky řádků, okraje a další náležitosti, které se týkají stylování
  - Aplikace je stylována podle Material Theme z AndroidX support knihovny (Theme.AppCompat.Light.DarkActionBar)
    - Activity musí v tomto případě dědit od AppCompatActivity, Fragmenty od androidx.fragment.app.Fragment, atd.

Hodnocení

Celkem bude za semestrální práci 24 bodů, které se budou udělovat postupně na jednotlivých kontrolních bodech. Minimum pro získání zápočtu je 12 bodů. Z každého kontrolního bodu je nutné získat alespoň 4 body.

Kontrolní body

Celkem budou 3 kontrolní body, vždy na začátku dané hodiny.

1. kontrolní bod v 5. týdnu, 8 bodů

- Práce bude obsahovat 2 obrazovky, kdy obě jsou Activita a uvnitř každé je vždy jeden Fragment. Na jedné obrazovce bude seznam článků. Na druhé obrazovce bude detail článku. Z první obrazovky se kliknutím na položku v seznamu dostaneme na druhou obrazovku.
  - Jako rodičovskou třídu pro Activity použijte AppCompatActivity z AndroidX support knihovny a pro Fragmenty použijte androidx.fragment.app.Fragment.
- Data budou pouze statická, tedy přímo v kódu aplikace (např. v nějaké třídě DataStorage, která bude mít pouze statické proměnné a metody. Nepoužívejte resources - XML, soubory apod.).
- Seznam položek **nevytvářejte** pomocí RecyclerView/ListView a Adapteru. Vytvořte scrollovatelný LinearLayout (seznam článků) s několika klikatelnými TextView (odpovídá jednomu článku - např. nadpis a část článku - tedy dvě TextView) pod sebou. Využijte pro vytváření řádků LayoutInflater.
- Možnost sdílet článek ostatním aplikacím z obrazovky s detailem článku.
- Do Bundle, který se bude používat pro přenos dat mezi jednotlivými obrazovkami, vkládejte vždy pouze ID článku, jež bude následně použito k načtení údaju z datového zdroje. Nikdy do Bundle nedávejte velké (textové) proměnné.
- UI pro tablety není nutné prozatím implementovat a je dobrovolné. Hodí se však již používat interface, který bude následně používan při rozšíření na tablety.
- Fragment pro seznam článků přidejte přímo do layout a druhý pro detail článku dynamicky v kódu. Použijte metody / patternu newInstance pro vytváření instance a předání parametrů (metoda setArguments)
- Kód pro sdílení článků v rámci toolbaru je obsažen pouze ve Fragmentu (musí se provolat setHasOptionMenu(true)). Kód pro sdílení tedy není vůbec obsažen Activitě (např. může ho později rozšiřovat o další položky sama). Veškerá manipulace s View od článků (detail či seznam) je rovněž vždy umístěna v kódu Fragmentu.
- LayoutInflater nesmí mít null parametr root pro metodu inflate (viz 2. přednáška)

2. kontrolní bod v 9. týdnu, 8 bodů

- Data se budou asynchronně stahovat z internetu a ukládat do databáze s pomocí databáze Room.
- Do Fragmentů se budou data načítat z Room databáze pomocí DAO a LiveData.
  - Nepřistupujte k databázi přímo z Fragmentu, ale využijte ViewModel, který vystaví LiveData - usnadníte si tak práci s ošetřením otáčení obrazovky.
- Při aktualizaci seznamu článků bude zobrazen indikátor s informací o průběhu načítání ve formě ProgressBaru v rámci ActionBaru. Informace o průběhu musí být samozřejmě zachovány i po změně konfigurace (např. otočení obrazovky) zařízení. Pro testovací účely můžete ve **vedlejším** vlákně např. použít SystemClock.sleep() k simulaci stahování většího množství dat.
- Možnost přidávat/mazat feedy. Nová obrazovka se seznamem Feedů a tlačítkem pro přidání nového Feedu. V ideálním případě budou Feedy taktéž uloženy v databázi (v samostatné tabulce).
- Seznam článků bude implementovaný pomocí RecyclerView. Detail článku načítejte opět přes DAO a LiveData.
- Možnost přejít na web, ze kterého článek pochází (na stránku s článkem).
- Při stahování dat z internetu si dejte zejména pozor na změnu konfigurace Activity (např. rotace obrazovky) a zkontrolujte, že se vše chová korektně (viz 6. přednáška).
- **Nepoužívejte** Service k žádnému účelu. Pro stahování dat využijte výhradně ViewModel, který bude v samostatné třídě.
- Pro zobrazení detailu článku můžete použít TextView a následně metodu Html.fromHtml() nebo samostatný WebView.

3. kontrolní bod na posledním cvičení (cvičení vycházející na volný den se již nepočítá), 8 bodů, finální odevzdání práce

- Stahování dat bude probíhat v rámci Job pro JobScheduler implementovaný v samostatné JobService. Síťový request nesmí běžet v hlavním vlákně.
  - Při uživatelově manuálním refreshi spusťte bezodkladně ten samý Job.
- Job se bude taktéž každých pár hodin za pomoci JobScheduleru spouštět a stahovat tak data automaticky, bez uživatelova přičinění.
  - Je nutné zajistit, aby se Job spouštěl i po restartu telefonu (tzv. persistent Job).
- UI musí reflektovat stav JobService - tedy když se začnou na pozadí sama od sebe stahovat data, v UI to musí být vidět ve formě viditelného ProgressBaru (stačí indefinite progress). ProgressBar se musí zobrazit a) když je UI viditelné a začne stahování b) když stahování již běží a UI se zobrazí.
  - K signalizaci stavu (a dotazům na stav) můžete použít LocalBroadcastManager nebo návrhový vzor Repository, které bude vystavovat LiveData se stavem service.
- UI pro tablety i pro mobilní telefony. Na hlavní obrazovce bude po levé straně sloupec s články a po pravé straně detail právě vybraného článku.
  - Po otočení obrazovky musí zůstat vybraný stejný článek.

Řešení

- Bude průběžně doplňováno na titulní stránce

Naposledy změněno: Neděle, 4. duben 2021, 09.51