Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Semestre de Inverno 2014/2015

2^a Série de exercícios PDM

Programação em Dispositivos Móveis

Eng.º Pedro Félix

Data de entrega: 24.11.2014

Trabalho elaborado por:

Pedro Lima N°. 33684 Sílvia Ganhão N.º 35418 LI51D – LEIC

Índice

1º Exercício	3	
2º Exercício	6	

1º Exercício

Este exercício teve como objetivo a adição de algumas caraterísticas à aplicação de notícias do *Thoth*, realizada na primeira série de exercícios.

Uma das caraterísticas é a existência de um *content provider* para armazenamento e gestão de informação sobre o conjunto de turmas disponíveis no *Thoth*, o conjunto de turmas selecionadas, e o conjunto de notícias das turmas selecionadas, incluindo o estado de visualização das mesmas. Para persistência da informação teve-se por base, uma base de dados *SQLite*.

Outra caraterística é a existência de um serviço para atualização periódica da informação, apenas quando existir conetividade via *WiFi*.

Para a realização deste exercício o primeiro passo, foi a elaboração de um *schema* (Figura 1) das opções a tomar, para uma melhor perceção das alterações a fazer, relativamente ao já realizado na primeira série.

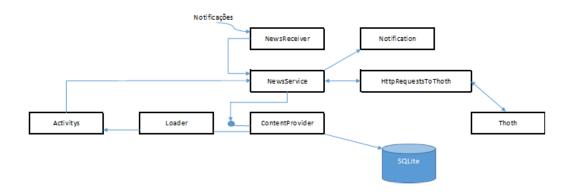


Figura 1 – schema das opções de código tomadas

A criação do *content provider* é realizada num projeto diferente com o nome *NewsClassServer*. Na classe *NewsClassOpenHelper*, incorporada neste projeto é onde se procede à criação da base de dados *thoth.db*, que irá conter as tabelas *thothClasses* e *thothNews*, também criadas a partir desta classe.

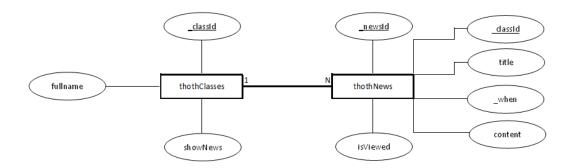


Figura 2 – Modelo EA.

Em cima apresenta-se o modelo entidade-associação referente às tabelas existentes na base de dados.

Na classe *NewsClassContentProvider*, também incorporada neste projeto, é onde se procede À criação de todas as ações que são possíveis realizar sobre a base de dados. A inserção de novos dados – *insert*, a atualização de dados já existentes - *update*, e a eliminação de dados – *delete*. Existindo ainda outro método – *query*, onde é possível realizar qualquer tipo de *query* sobre uma tabela da base de dados, tendo em conta o *Uri* que recebe, que é o que contém a informação sobre qual a tabela. Este método é chamado quando se pretende receber um *Cursor* que permita percorrer todas as linhas da tabela, para com essa informação se poder realizar alguma ação. Também nos métodos *insert*, *update*, e *delete*, é recebido um *Uri*, que permite saber em que tabela irão ser efetuadas as alterações pretendidas.

Para a interação entre as *activitys* e a atualização de dados no *content provider*, ambas as atividades (*MainActivity* – apresentação das notícias, e *SettingsActivity* - escolha das turmas), foi implementada a interface *LoaderCallbacks*<*Cursor*>, na qual são implementados os métodos *onCreateLoader*, *onLoadFinished* e *onLoaderReset*.

No método *onCreateLoader*, é criado um novo *CursorLoader* que permite percorrer a tabela presente no *content provider*. Sendo que na *MainActivity* se pretende a tabela referente às notícias, *thothNews*, e na *SettingsActivity* se pretende a tabela referente às classes, *thothClasses*.

Para se poder proceder ao lançamento de notificações a partir das ações realizadas, foram criadas algumas classes adicionais:

• NewsReceiver: Para que seja possível que assim que o wifi seja ligado exista uma atualização das notícias das turmas selecionadas foi criada esta classe que tem como função estar á "escuta" de eventos. O método desta classe, onReceive, é executado sempre que o estado do WiFi for alterado. Se o WiFi for alterado para conectado, então verifica-se se existe alguma turma nova ou notícias recentes das turmas selecionadas no Thoth.

São criados dois alarmes, *AlarmManager*, para atualizar as classes – *classesAlarm* - (1 vez por dia), e as notícias das turmas selecionadas – *newsAlarm* - (a cada 30 minutos). Mas caso o estado do *WiFi* se perca (desconectado) ambos os alarmes serão cancelados, apenas voltando a ser criados quando existir nova conexão via *Wifi*. As ações definidas nos alarmes, quando são disparadas, também são apanhadas pelo método onReceive, sendo feita a chamada ao serviço dentro deste método.

- HttpRequestsToThoth: É a partir desta classe que são realizados os pedidos HTTP ao
 Thoth, que são chamados a partir do NewsService, onde é realizada a atualização dos
 dados presentes no content provider.
- NewsService: é a classe que implementa a classe IntentService, ou seja, é o serviço que é responsável por todos os pedidos realizados ao Thoth e também por todas as alterações efetuadas ao content provider criado anteriormente. Este serviço é executado sempre que:
 - O WiFi for alterado para o estado de conectado, de forma a verificar se existem novas turmas ou novas notícias de turmas selecionadas;
 - O utilizador escolher um novo grupo de turmas, através da *SettingsActivity:*
 - As notícias das turmas que foram retiradas da seleção são eliminadas do content provider;
 - As notícias das turmas que foram agora selecionadas são obtidas através de um pedido ao *Thoth*, sendo posteriormente inseridas no content provider.
 - O utilizador lê uma notícia, de forma a colocar a indicação no content provider de que essa notícia já foi lida e assim atualizar a listview de apresentação das notícias;
 - A aplicação for iniciada e o content provider estiver vazio;
 - Um dos alarmes definidos no NewsReceiver disparar. Se for encontrada alguma nova turma ou notícia é lançada uma notificação para informar o utilizador desse acontecimento.

Tanto a *NewsAsyncTask* como a *ClassesAsyncTask* sofreram algumas alterações da primeira fase para esta. A *NewsAsyncTask* tem como função passar a informação contida no *Cursor* para um *array* de *NewItem* e retorná-lo ordenado pela data e pelo estado de visualização.

A *ClassesAsyncTask* tem como função passar a informação do *Cursor* para um *array* de *Clazz* retornando-o sem a necessidade de o ordenar, pois a informação no cursor já está ordenada pelo id de cada turma.

Foi ainda criado um novo *layout*, para apresentação das notificações que ocorrem caso haja conectividade *Wifi*, e que ao executar os alarmes de procura de novas turmas e notícias, atualizam o *content provider*.

2º Exercício

O objetivo deste exercício foi a adição à aplicação de gestão de aniversários, realizada na primeira série, da capacidade e criar notificações da proximidade de um evento. Sendo essas notificações lançadas, quando o aniversário de um contato ocorrer a menos de uma semana, ou no dia de aniversário de um contacto. Este exercício pedia ainda que a partir da notificação, fosse possível visualizar a informação do contato. Esta possibilidade já tinha sido garantida na primeira série.

Para a realização deste exercício, o primeiro passo foi a realização do *schema* (Figura 3) das opções a tomar, para uma melhor perceção das alterações a fazer, relativamente ao já realizado na primeira série.

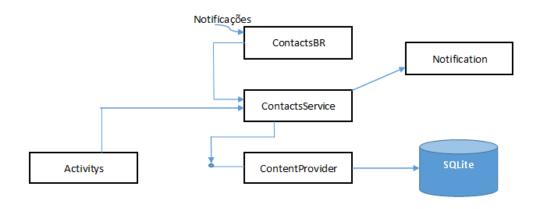


Figura 3 – schema das opções de código tomadas

Para se poder apanhar ações a partir das quais serão lançadas as notificações, foram criadas 2 classes:

• ContactsBReceiver: a partir desta classe é que serão apanhadas as ações que irão despoletar a procura de informação para posterior lançamento de notificações. Estas ações são apanhadas a partir do método onReceive(). É ainda neste método que são criados os AlarmManager que servem para que seja lançado um Service, no qual se faz a procura dos aniversários dos contatos que têm aniversário a ocorrer no espaço de uma semana, ou no próprio dia. Na procura dos contatos com aniversário a ocorrer no espaço de uma semana, o alarme é lançado uma vez por semana, a todas as segundas-feiras.

A partir do método onReceive() é onde ocorre o lançamento deste Service.

As ações que despoletma o *ContactsBReceiver* é o *boot* do aparelho (*ACTION_BOOT_COMPLETED*) e as acções definidas nos alarmes.

• ContactsService: é nesta classe, no método onHandleIntent(), que é iniciado o trabalho do Service, em que é feita uma query ao ContentProvider, que resulta num Cursor com todos os contatos existentes no dispositivo. Com este Cursor, são verificadas as datas de cada aniversário para ver em que perfil se enquadram: com aniversário a ocorrer no espaço de uma semana, ou com aniversário a ocorrer no dia presente. Esta verificação é feita no método checkDate. É também neste método que é lançada a notificação ao utilizador, apresentando o nome do contato, e o dia do seu aniversário.