Instituto **S**uperior de **E**ngenharia de **L**isboa

Licenciatura em Engenharia Informática e de Computadores

Semestre de Inverno 2014/2015

Projeto Final
PDM
Programação em Dispositivos Móveis

Eng.º Pedro Félix

Data de entrega: 19.01.2015

Trabalho elaborado por:
Pedro Lima
Nº. 33684
Sílvia Ganhão
N.º 35418
LI51D – LEIC

Índice

Introdução 3
IselAppServer 4
IselApp 6
Utils 6
Entidades 6
CustomAdapters 6
AsyncTasks 7
ViewModels 7
ClassesActivities 7
NewsActivities 7
ParticipantsActivities 7
WorkItemsActivities 7
Fragments 8
ListModels 8
Handlers 8
Receivers 8
Services 8
SyncAdapters 9
Funcionamento Geral 10
Notícias 10
Turmas 12
Trabalhos 14
Participantes 16
Esquema Global 18

Introdução

O objetivo deste trabalho é a implementação de uma aplicação Android que interage com a ferramenta de ensino Thoth, através da sua API. As funcionalidades presentes na aplicação são:

- A escolha de turmas;
- O visionamento das notícias e dos trabalhos associados a cada uma das turmas escolhidas;
- E ainda é possível ver os participantes de cada turma.

Para a melhor utilização desta aplicação foram criados dois projetos, o IselAppServer e o IselApp, para que o utilizador possa aceder à informação sem ser necessário ligação à Internet.

As soluções utilizadas para solucionar os diversos desafios encontrados durante a implementação da aplicação são descritas ao longo deste relatório.

IselAppServer

Esta aplicação foi criada de forma a implementar o *Content Provider* e a criar a base de dados que dá suporte à informação disponibilizada ao utilizador. É nesta base de dados que a informação fica armazenada para que o utilizador possa aceder-lhe sem que seja necessário fazer qualquer tipo de conexão à Internet. Criou-se uma base de dados *SQLite*, de nome **IselApp.db**. Para fazer a gestão desta base de dados é necessário implementar um *Content Provider*.

Na base de dados foram criadas três tabelas:

- ➔ A tabela **classes**, que armazena a informação relativa às turmas existentes no Thoth;
- ➔ A tabela **news**, que armazena a informação relativa às notícias das turmas selecionadas pelo utilizador;
- ➔ A tabela **workItems**, que armazena a informação relativa aos trabalhos associados a cada turma escolhida pelo utilizador.

O modelo Entidade-Associação destas tabelas está definido na Figura 1.

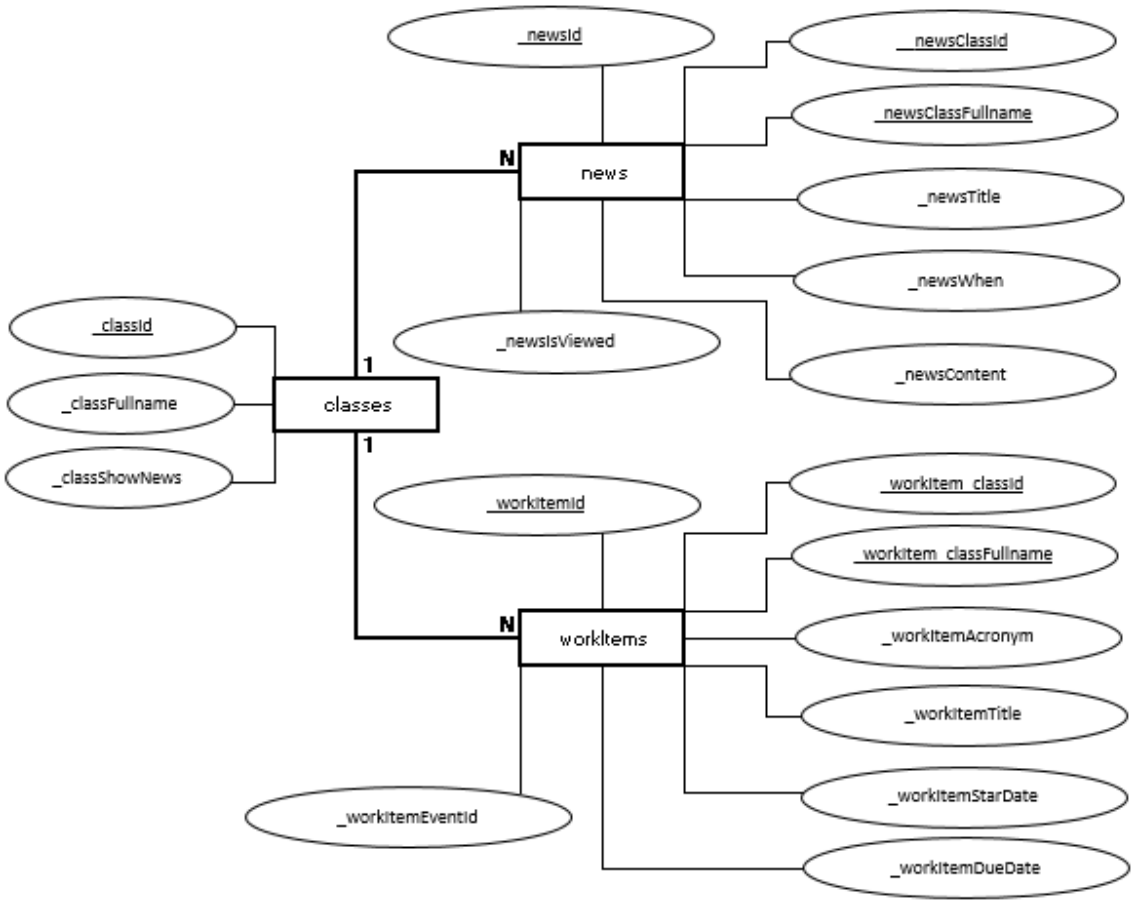

Figura 1- Modelo EA

Para criar a base de dados **IselApp.db** foi definida a classe **IselAppOpenHelper**. De modo a que a classe tenha os requisitos necessários para criar a base de dados é preciso que estenda de **SQLiteOpenHelper**, classe que permite a criação e a gestão de bases de dados. É nesta classe que é criado o ficheiro da base de dados assim como as tabelas que suportam o modelo de dados obtido a partir do Thoth.

Para além desta classe, foram ainda definidas mais duas classes: a classe **IselAppContentProviderContract** onde está definido o valor do campo *Authority* associado a este *Content Provider*, que é utilizado pela outra classe implementada, a classe **IselAppContentProvider**. Esta última classe referida é responsável por fazer toda a gestão da base de dados da aplicação (*inserts*, *deletes*, *updates* e *querys*) através de uma instância da classe **IselAppOpenHelper**, do campo *Authority* e de um *Uri* que identifica univocamente cada uma das possíveis tabelas presentes na base de dados. A combinação destes elementos possibilita a correta gestão da base de dados da aplicação.

Na classe **IselAppContentProvider**, para além da implementação dos métodos *insert*, *delete*, *updates* e *query*, estão também implementados dois métodos que permitem fazer operações de forma transacional. Estes métodos são:

- **applyBatch**: recebe um conjunto de operações e efetua-as todas dentro de uma transação de forma a garantir que todas as operações sejam feitas com sucesso e que caso haja uma exceção todas as operações já executadas sejam anuladas.
- **bulkInsert**: método semelhante ao anterior mas apenas executa operações de *insert*, também de forma transacional.

IselApp

É neste projeto que está feita toda a implementação das funcionalidades tidas como objetivo desta aplicação. Para o correto e eficiente funcionamento desta aplicação foram implementadas diversas soluções, recorrendo aos mais variados objetos disponibilizados pelo Android (*AsyncTask*, *Service*, *Fragments*, …).

Para melhor organização do projeto, foram criados vários packages, de forma a separar as classes por funcionalidades.

Utils

Sendo necessário realizar algumas operações auxiliares, foi criado este *package*, onde se encontram diversas utilidades:

- **RequestsToThoth** – realiza os diferentes tipos de pedidos ao Thoth (turmas, notícias, trabalhos, participantes);
- **NotificationLaunch** – realiza o lançamento das notificações;
- **CalendarEvents** – auxilia na adição de eventos ao calendário dos trabalhos (*WorkItems*) a serem entregues.

Entidades

De forma a guardar a informação obtida através do *Content Provider* ou de um pedido HTTP para auxiliar na manipulação dos dados ao longo da execução da aplicação, foram criadas quatro classes representando cada uma delas uma entidade diferente. As classes são:

- **ClassItem** - armazena os dados associados a uma turma;
- **NewsItem** – armazena os dados associados a uma notícia;
- **WorkItem** - armazena os dados associados a um trabalho;
- **ParticipantItem** – armazena os dados associados a um participante.

Todas estas classes implementam a interface **Serializable**.

CustomAdapters

De forma a relacionar a *view* de cada tipo, com a informação conseguida, foram criados *Adapters* para cada entidade existente:

- **ClassesCursorAdapter** – relaciona a view das turmas com a informação das mesmas;
- **NewsCustomAdapter** – relaciona a view das notícias com a informação das mesmas;
- **ParticipantsCustomAdapter** – relaciona a view dos participantes com a informação dos mesmos;
- **WorkItemCustomAdapter** – relaciona a view dos trabalhos com a informação dos mesmos.

AsyncTasks

De forma a passar a informação detida pelo cursor, obtido através de um *CursorLoader*, para um *array* para depois ser usado por um *CustomAdapter* foram criadas diferentes *AsyncTasks*:

- **NewsAsyncTask** – passa a informação do cursor para um *array* de *NewItem* e retorna esse *array*;
- **ParticipantsAsyncTask** – passa a informação do cursor para dois *arrays* de *ParticipantItem*, um que contém os participantes que são professores e outro que contém os participantes estudantes. O retorno é uma *ArrayList* que contém estes dois últimos *arrays* criados;
- **WorkItemsAsyncTask** – passa a informação do cursor para um *array* de *WorkItems* e retorna esse *array*.

ViewModels

Para associar os *widgets* das *views* de cada tipo, com a informação necessária a ser apresentada, foram criados os seguintes *ViewModels*:

- **ClassesViewModel** - disponibiliza os *widgets* da *view* referente às turmas;
- **NewsViewModel** – disponibiliza os *widgets* da *view* referente às notícias;
- **ParticipantViewModel** – disponibiliza os *widgets* da *view* referente aos participantes;
- **WorkItemViewModel** – disponibiliza os *widgets* da *view* referente aos trabalhos.

ClassesActivities

Contém a classe **SettingsActivity** onde é implementada toda a funcionalidade referente ao lançamento da atividade onde é possível o utilizador selecionar as turmas.

NewsActivities

Contém as classes **MainActivity** e **NewsItemActivity**, onde são implementadas as funcionalidades referentes ao lançamento das atividades que mostram a lista de todas as notícias de uma turma, e cada notícia individualmente, respetivamente.

ParticipantsActivities

Contém as classes **ParticipantsActivity** e **ParticipantItemActivity**, onde são implementadas as funcionalidades referentes ao lançamento das atividades que mostram a lista de todos os participantes de uma dada turma e cada participante individualmente, respetivamente.

WorkItemsActivities

Contém as classes **WorkItemsActivities** e **WorkItemLinkActivity**, onde são implementadas funcionalidades referentes ao lançamento das atividades que mostram a lista de todos os trabalhos de uma determinada turma, e a página de internet do trabalho em si, respetivamente.

Fragments

Para apresentação das informações das notícias e participantes foram criados fragmentos, para visualização dos itens por lista e individualmente:

- ➔ Para as notícias:
  - **NewsItemListFragment** – Fragmento a partir do qual é apresentada a lista das notícias de uma dada turma;
  - **NewsItemFragment** – Fragmento a partir do qual é apresentada um notícia individualmente;
- ➔ Para os participantes:
  - **ParticipantItemListFragment** – Fragmento a partir do qual é apresentada a lista dos participantes de uma dada turma;
  - **ParticipantItemFragment** – Fragmento a partir do qual é apresentado um participante individualmente.

ListModels

Para obtenção das informações das notícias e participantes de uma dada turma, de forma que essa informação seja passada aos fragmentos, existem dois modelos auxiliares, que suportam a obtenção da mesma: **NewsListModel** e **ParticipantListModel**.

Handlers

O processo de obtenção da imagem do participante é algo que requer um cuidado especial, pois devido ao número de imagens que são necessárias carregar, associado ao facto de por cada imagem se efetuar um pedido HTTP, torna o processo muito demorado o que impossibilita que seja feito na *UI thread*. Devido a este problema foram criadas três novas classes de forma a implementar uma solução que recorre a *Loopers* e *Handlers*:

- ➔ **ImageHandlerThread** - Possibilita a criação de uma nova *thread* que possui um *Looper*, onde é efetuado o processo de obtenção das imagens;
- ➔ **ImageHandler** - Associada ao l*ooper* originado pela **ImageHandlerThread** tem como função receber as diversas mensagens e processá-las;
- ➔ **SetViewHandler** - Associada ao *mainLooper*, tem como única responsabilidade associar a imagem obtida do pedido HTTP ao *ImageView* presente no *layout* correspondente.

Receivers

Contém a classe **IselAppReceiver**, que tem como função ficar alerta quando houver mudanças no estado da rede, e assim lançar o *SyncAdapter*, para atualização da informação existente no *Content Provider*.

Services

Contém a classe **IselAppService**, onde é feita a atualização da informação no *Content Provider*, após serem selecionadas ou desseleccionadas turmas, atualizando as notícias e os trabalhos da mesma.

SyncAdapters

O componente *SyncAdapter* foi criado para encapsular o código de transferência de dados entre a aplicação e o *Content Provider*. Para tal foi necessária a criação de algumas classes:

- **IselAppSyncAdapter** – Onde se encontra o código de atualização da informação, obtida da plataforma Thoth, existente no *Content Provider*;
- **IselAppSyncService** – Componente que permite que a *framework* do *SyncAdapter* corra o código existente na classe **IselAppSyncAdapter**.
- **IselAppAuthenticator** - A framework *SyncAdapter* assume que é necessário uma conta com acesso *login* para poder haver transferência de informação, daí a existência de uma classe *Authenticator*. Embora nesta aplicação não seja necessário a existência de uma conta, este componente tem de ser criado na mesma.
- **IselAppAuthenticatorService** – Para que o *SyncAdapter* tenha acesso ao *Authenticator* é necessário criar um *bound Service*. Este retorna um objeto Android *binder* que permite ao *SyncAdapter* chamar o *Authenticator* e assim fazer transferência de informação.

Funcionamento Geral

Notícias

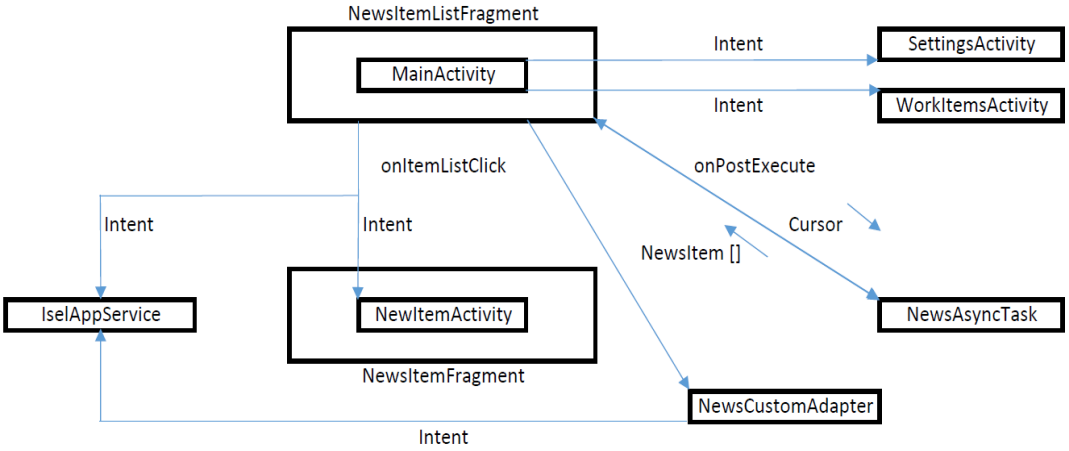

Figura 2 - Schema referente à atividade das notícias

A atividade dentro da aplicação que disponibiliza o visionamento das notícias das turmas selecionados pelo utilizador é a **MainActivity**. É sempre esta atividade que é executada quando a aplicação é lançada. As notícias são mostradas ao utilizador em forma de lista ordenada primeiro pelas notícias visionadas e não visionadas e depois pela sua data. Uma notícia considera-se vista sempre que o utilizador clicar numa notícia presente na lista e vir o seu detalhe.

As notícias são obtidas através de um *CursorLoader*, que faz um pedido ao *ContentProvider* para obter todas as notícias presentes na base de dados. Através do cursor obtido é lançada a **NewsAsyncTask** que coloca a informação detida pelo cursor num *array* de *NewItems*, de forma a ser possível criar uma instância do **NewsCustomAdapter** que preenche a *ListView* presente no *layout* com base nesse *array*. A partir deste ponto todas as notícias estão presentes na lista para o utilizador visionar. Sempre que o utilizador selecionar uma notícia ainda não vista a aplicação lança um serviço, **IselAppService**, que irá alterar a informação na base de dados relativa a essa notícia de forma a alterá-la para visionada.

O visionamento detalhado de uma notícia pode ser feito de duas formas dependendo do tamanho do ecrã do dispositivo que o utilizador está a usar:

- ➔ Se o tamanho do ecrã for inferior a 600dp é lançada a atividade **NewItemActivity** que é responsável apenas por mostrar ao utilizador o detalhe de cada notícia.

➔ Se o tamanho do ecrã for igual ou superior a 600dp (Tablet) o detalhe de cada notícia é disponibilizado num segundo fragmento disponibilizado no *layout* da **MainActivity** não sendo necessário lançar a atividade **NewItemActivity**.

A partir desta atividade é ainda possível lançar as atividades que permitem visionar os trabalhos de cada turma e selecionar as turmas pretendidas, através do menu existente no canto superior direito do ecrã.

Figura 3 – Exemplo de layout das notícias.

Figura 4 – Exemplo de layout das notícias com o menu aberto.

Turmas

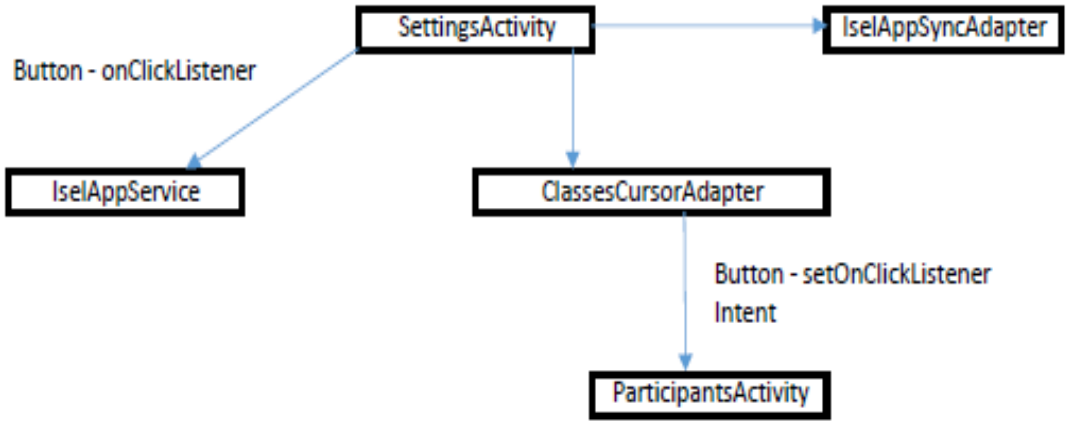

Figura 5 - Schema referente à atividade das classes

A atividade que permite o visionamento e seleção das turmas assim como o lançamento de uma atividade que permite ver quais os participantes de cada turma é a **SettingsActivity**. Esta atividade pode ser lançada a partir da **MainActivity** ou da **WorkItemsActivity,** portanto quando for finalizada pode retornar para uma delas dependendo de onde foi lançada. As turmas existentes são disponibilizadas ao utilizador através de uma lista, onde cada item desta lista contém o nome da turma e dois *widgets*, uma *CheckBox* e um *Button* em forma de uma imagem. A *CheckBox* permite ao utilizador selecionar as turmas das quais pretende ver as notícias e os trabalhos e o *Button* vai disponibilizar ao utilizador o visionamento dos participantes daquela determinada turma.

As turmas são obtidas através de um *CursorLoader*, que faz um pedido ao *Content Provider* para obter todas as turmas presentes na base de dados. O resultado deste pedido é um cursor que é utlizado pelo **ClassesCursorAdapter** que disponibiliza a informação, contida neste cursor através da *ListView* presente no *layout*.

No entanto, no caso da primeira execução da aplicação em que o cursor não tem nenhuma informação, pela base de dados ainda estar vazia, é necessário fazer a sincronização entre a aplicação e o Thoth. Para fazer a sincronização é necessário fazer um **syncRequest** que irá executar assincronamente o método **onPerformSync** da classe **IselAppSyncAdapter** que tem como função efetuar o sincronismo entre a aplicação e o Thoth. Depois de efetuada a sincronização, a tabela relativa às classes na base de dados é preenchida e o cursor obtido no *CursorLoader* é notificado de alterações na base de dados, e assim é repetido o processo de obtenção do cursor, já com a informação nova, e assim disponibilizadas as turmas ao utilizador.

O visionamento dos participantes de uma turma é feito através do lançamento da atividade **ParticipantsActivity** que é provocado através de um *click* no botão existente em cada item da lista das turmas. Esta opção apenas está disponível quando existe ligação à Internet pois esta informação não é guardada na base de dados da aplicação.

Para finalizar a seleção das turmas o utilizador necessita de carregar no botão “OK” que vai originar duas operações:

- ➔ Lançamento de um serviço (**IselAppService**) que vai sincronizar com o Thoth as notícias e os trabalhos das turmas selecionadas para que o utilizador tenha acesso à informação e eliminar da base de dados todas as notícias e trabalhos correspondentes às turmas que o utilizador tinha selecionadas, mas já não tem. É atualizado também, na base de dados, na tabela das turmas, o campo que define se a turma está ou não selecionada pelo utilizador.

- ➔ Finalização da atividade **SettingsActivity** retornando assim para a atividade que a lançou (**MainActivity** ou **WorkItemsActivity**).

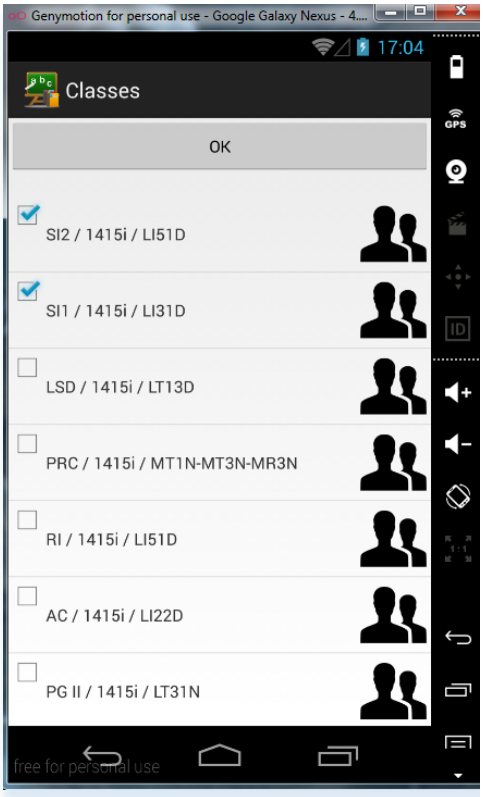

Figura 6 – Exemplo de layout das turmas.

Trabalhos

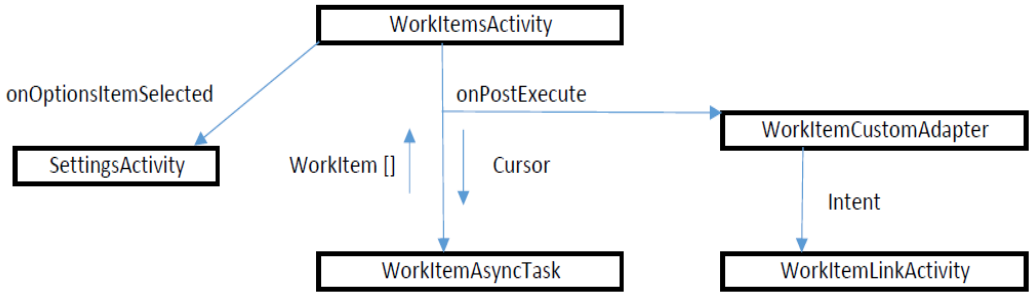

Figura 7 - Schema referente à atividade dos trabalhos

A atividade que permite o visionamento dos trabalhos de uma turma é a **WorkItemsActivity**. Esta atividade pode ser lançada a partir da **MainActivity**. Através do botão "*See Work Items*" presente no menu no canto superior direito.

Os trabalhos de uma turma são disponibilizados ao utilizador através de uma lista, onde cada item dessa lista contém três *TextView*, referentes ao nome da turma, ao título do trabalho e à data limite de entrega.

Os trabalhos são obtidos através de um *CursorLoader*, que faz um pedido ao *Content Provider*, para obter os trabalhos presentes na base de dados. Através do cursor é lançada a **WorkItemsAsyncTask**, que coloca a informação detida pelo cursor num *array* de *WorkItems*, de forma a ser possível criar uma instância do **WorkItemsCustomAdapter**, que preenche a *ListView* presente no *layout* com base nesse *array*.

A partir deste ponto todos os trabalhos estão presentes na lista para o utilizador visualizar. Sempre que o utilizador selecionar um trabalho, é lançada uma nova atividade, **WorkItemLinkActivity**, que abre a página na internet correspondente ao *link* do trabalho.

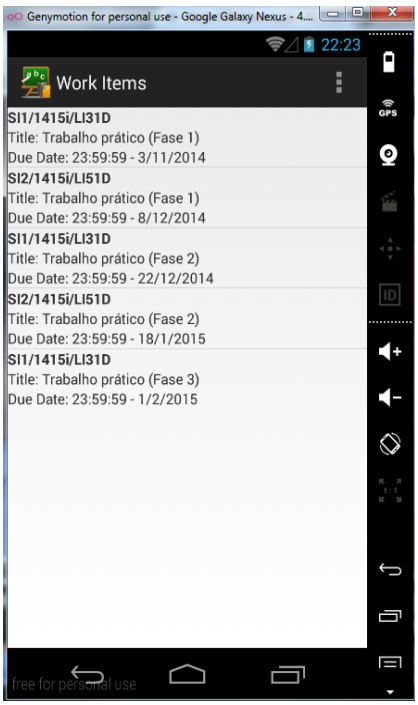

Figura 8 – Exemplo do layout dos trabalhos.

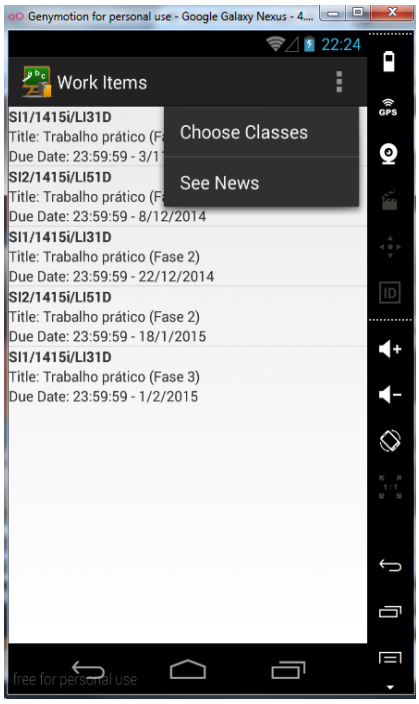

Figura 9 – Exemplo do layout dos trabalhos com o menu aberto.

Participantes

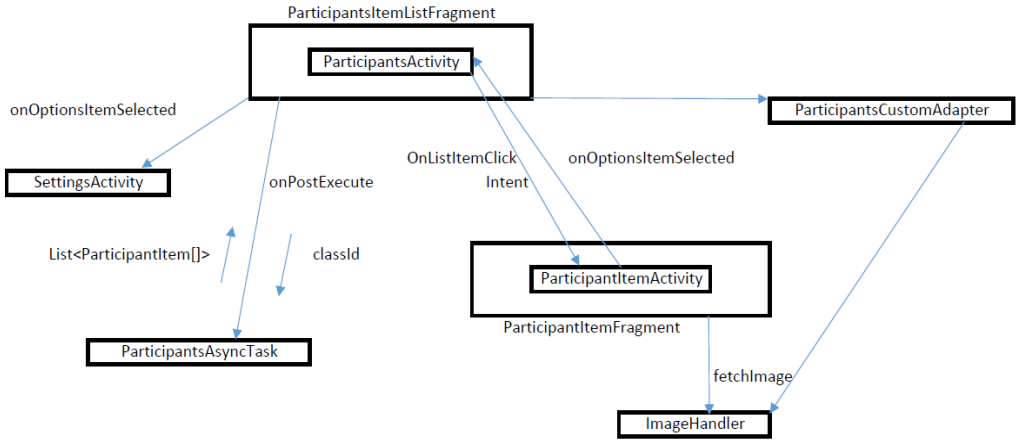

Figura 10 - Schema referente à atividade dos participantes

A atividade que permite o visionamento dos participantes de cada turma é a **ParticipantsActivity**. Esta atividade é lançada a partir da **SettingsActivity**. O visionamento dos participantes é feito de forma diferente de todas as outras listas devido ao facto de um participante poder ser um professor ou um estudante. Desta forma a informação é disponibilizada através de dois separadores, um que mostra a lista dos participantes da turma que são professores e outro que mostra os participantes estudantes.

Existe ainda a possibilidade de ver a informação de cada participante de forma detalhada clicando no item pretendido da lista de participantes.

A informação de todos os participantes de uma turma é obtida através de um pedido HTTP ao Thoth, passando nesse pedido o identificador da turma de que se pretende ver os participantes. O resultado deste pedido é um *array* com todos os participantes que é depois passado para a **ParticipantsAsyncTask** que vai dividir este *array* em dois, um só com os professores e outro só com os estudantes. No retorno da *AsyncTask* é passado o *array* dos professores ou dos estudantes ao **ParticipantsCustomAdapter** dependendo do separador selecionado, de forma a preencher a *ListView* presente no *layout*.

Associado a cada participante existe sempre uma imagem, cujo link que permite fazer o download da imagem, é obtido através do pedido ao Thoth. O processo de obtenção da imagem do participante é algo que requer um cuidado especial, pois devido ao número de imagens que são necessárias carregar, associado ao facto de por cada imagem se efetuar um pedido HTTP, torna o processo muito demorado o que impossibilita que seja feito na *UI thread*. Para solucionar este problema recorreu-se a *Loopers* e *Handlers*, funcionalidades disponibilizadas pelo Android. Utilizou-se um *handler* para obter a imagem, ou seja, fazer o pedido HTTP e depois

é utilizado um outro *handler* para associar a imagem ao *widget* do item correspondente da lista de participantes, sendo todas estas operações feitas assincronamente.

No entanto, mesmo com esta solução verifica-se que ao percorrer a lista de participantes, a imagem de cada participante continua a levar algum tempo a ser carregada o que causa um efeito visual não muito agradável ao utilizador.

De forma a minimizar os recursos gastos necessários para o correto funcionamento desta tarefa, sempre que uma imagem associada a um participante for carregada, apenas no primeiro carregamento é feito um pedido HTTP de forma a obter a imagem. Depois de obtida a imagem, esta é guardada no armazenamento externo, o que garante uma maior rapidez na obtenção da imagem sempre que esta for necessária.

As imagens são guardadas numa pasta pública na memória externa, pois como as imagens são públicas não é necessário guardá-las numa pasta privada e assim podem ser utilizadas por outras aplicações existentes no Android. Cada imagem é guardada com o número associado ao participante, para que seja usada sempre a mesma imagem no caso de o participante existir em mais de uma turma.

Foi definido que o máximo tamanho possível desta forma de fazer cache é de 3 MB para que não ocupe muito espaço na memória externa. Sempre que estiver quase a ficar cheia são apagadas metades das imagens contidas na memória de forma a guardar novas imagens.

Esta forma de *caching* apenas é possível se o acesso à memória externa estiver disponível e com acessos para escrita e leitura, se não, é necessário carregar as imagens sempre através de pedidos HTTP.

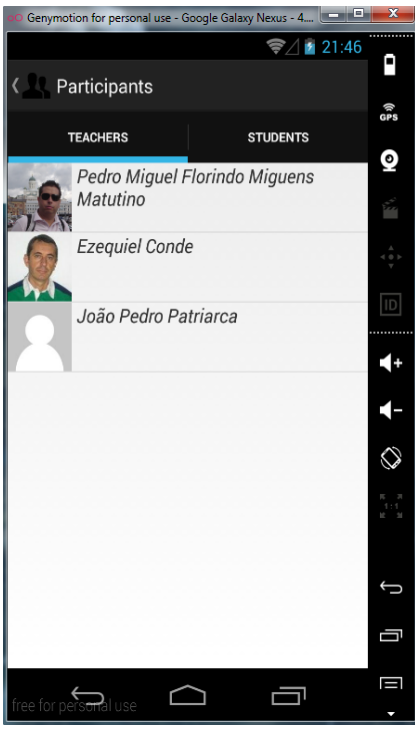

Figura 11 – Exemplo do layout dos participantes com o separador dos professores selecionado.

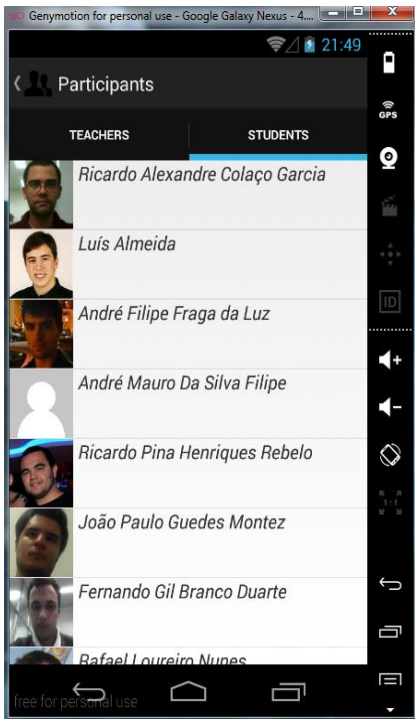

Figura 12 - Exemplo do layout dos participantes com o separador dos estudantes selecionado.

Esquema Global

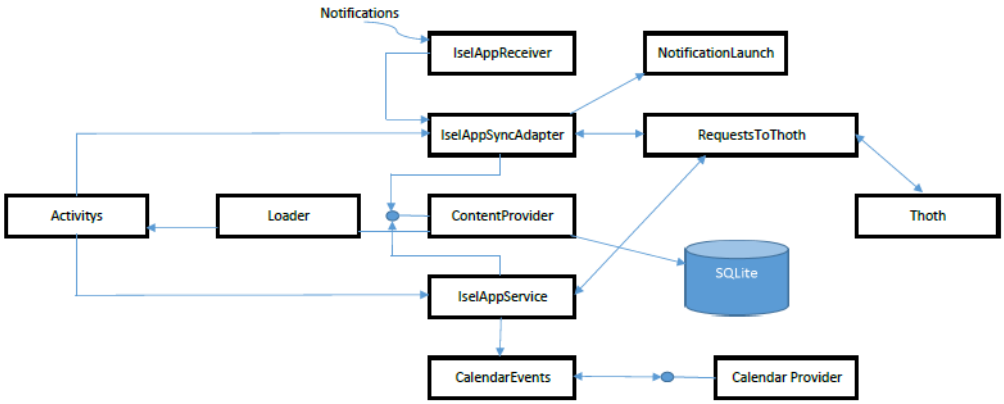

Figura 13 – Esquema Global