*I*nstituto *S*uperior de *E*ngenharia de *L*isboa
Licenciatura em **E**ngenharia de **I**nformática e de **C**omputadores

Programação em dispositivos Móveis

Trabalho 1

Yamba Mobile

Leandro Viana N. *º 32808*
Fábio Pita N. *º 32980*
Filipe Chaves N. *º 35367*

Docente: *PauloPereira*
Semestre: 2011/2012 Verão
Data: 16/04/12

Introdução

Neste trabalho realizaremos extensões da aplicação desenvolvida da aula (aplicação Yamba).

Nesta fase vamos desenvolver quatro componentes do tipo **Activity, Status, Preferences, TimeLine e Detail.**

A aplicação também suporta internacionalização, ou seja, todos os textos apresentados devem ser especificados em Português e Inglês (idioma por omissão).

A actividade **Status** permite a publicação da mensagem de status, promovendo o envio para o serviço da aplicação servidora através do método updateStatus da classe Twitter. A interface com o utilizador inclui zona de edição da mensagem, o número de caracteres ainda permitidos (valor actualizado automaticamente) e botão para submissão da mensagem; este botão ficará indisponível (disabled) durante o envio. Lembre-se que a orientação do dispositivo poderá mudar durante o envio.

A actividade **Preferences** viabiliza a especificação dos parâmetros de configuração da aplicação, nomeadamente: username e password para autenticação no serviço, Uri base do serviço, número máximo de mensagens apresentadas na atividade **Timeline** e número máximo de caracteres admitidos em cada mensagem a publicar. Esta actividade é explicitamente através do menu ou automaticamente, no início da aplicação, caso os dados de configuração e Uri base do serviço não tenham ainda sido especificados (os restantes parâmetros assumem valores por omissão).

A actividade **Timeline** apresenta as últimas mensagens publicadas no timeline, obtidas com o método getUserTimeline da classe Twitter. Para cada mensagem é mostrada parte do (com dimensão a especificar), o nome do autor e há quanto tempo foi publicada. A aplicação mantém em memória as mensagens do timeline e esta informação é actualizada explicitamente através do menu ou automaticamente quando a actividade Timeline é aberta pela primeira vez. A actividade é lançada automaticamente no início da aplicação e explicitamente através da opção correspondente no menu, opção que ficará indisponível enquanto a actividade estiver a ser apresentada.

UserStatusActivity

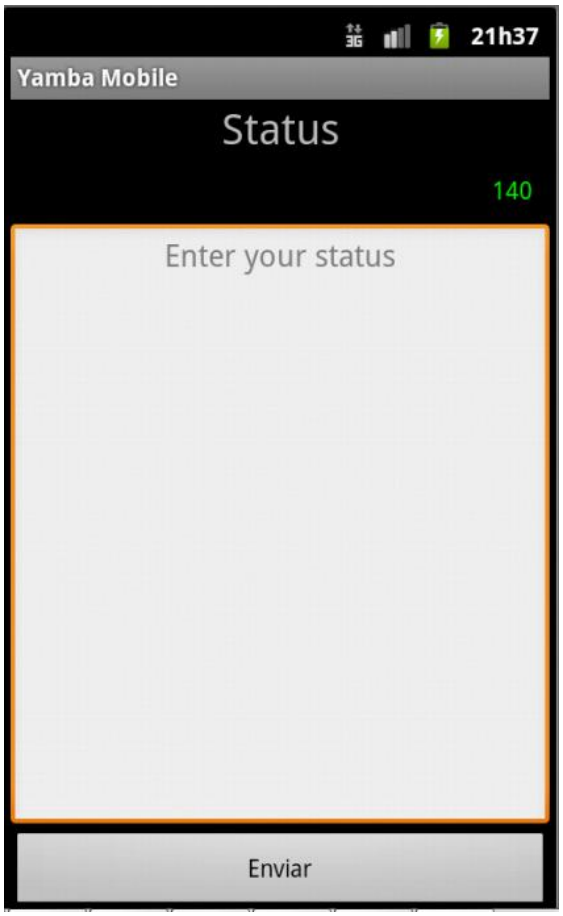

Fig01. Status

Este layout corresponde ao **StatusActivity**.

Esta **Activity** é a responsável de enviar o texto escrito para o serviço yamba, para isso foi feito uma **AsyncTask** para fazer esse envio em background e não bloquear a user interface, o contador é decrementado cada vez que é introduzido um novo caracter, não permitindo a escrita de mais caracteres quando chega ao limite, para isso foi usado um filtro (new InputFilter.LengthFilter(_maxChars)).

Este filtro não permite a escrita de mais caracteres quando os caracteres chegam ao número passado por argumento ao filtro.

Esta **Activity** , teve de implementar a interface **TextWatcher,** para que sempre fosse alterado o texto accionar um evento para actualizar o contador.

O número de caracteres é definido nas **Preferences.**

TimeLineActivity

Quando a aplicação é aberta é-nos mostrada a **Timeline** do utilizador, ou seja, a sua e a de todos os que tenha subscrito actualizações. No exemplo seguinte encontra-se a **Timeline** de uma conta que subscreveu o user "Marakana Student":

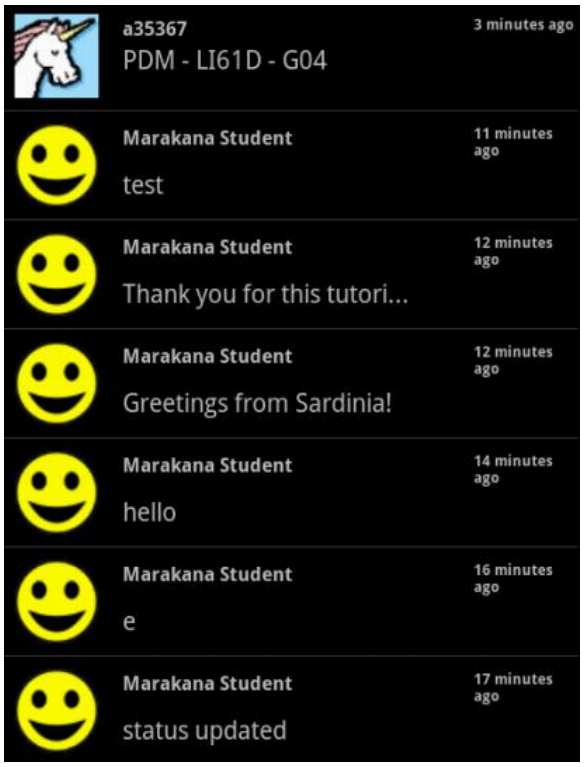

Fig02. TimeLine

Ao início pensámos criar esta **Activity** como **ListActivity**, sendo que para criar esta lista de tweets era mais cómodo, mas como tínhamos de partilhar o menu com a UserStatusActivity decidimos que o melhor era esta ser derivada de **SMActivity**.

Para apresentar os dados na lista, criámos uma *inner class* de modo a colocar os dados que queremos na lista de acordo com o xml de layout criado para servir de modelo para todas as posições da lista. Esta classe estende **SimpleAdapter** e coloca cada elemento (imagem e textos) na sua posição.

Quando o utilizador carrega numa das linhas da lista é reencaminhado para uma **DetailActivity**, que tem mais informações acerca da mensagem escrita e de quem a escreveu.

UserPreferencesActivity

Para a criação de uma página onde pudéssemos pedir informação ao utilizador utilizámos uma PreferenceActivity, da API android, que nos permite utilizar um layout predefinido com bastante facilidade:

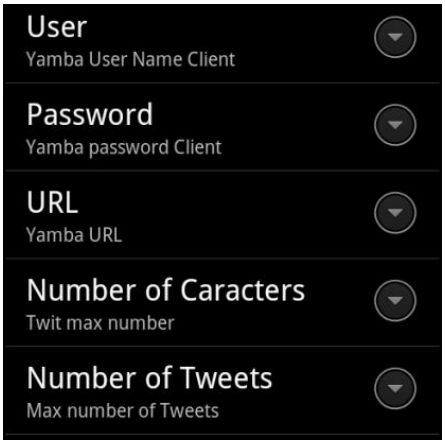

Fig03. Preferences

Esta lista é dada através do layout predefinido da PreferenceActivity e de um xml, criado por nós, com os dados que queremos que apareçam: “User”, “Password”, “URL”, “Number of Caracters”, “Number of Tweets”, como também a descrição que aparece de baixo de cada dado principal.

Para que isto seja apresentado é apenas necessário reimplementar o método onCreate() da seguinte forma:

```
15          @Override
16          public void onCreate(Bundle savedInstanceState){
17                  super.onCreate(savedInstanceState);
18                  init();
19                  addPreferencesFromResource(R.xml.prefs);
20                  Log.i(TAG, "UserPreferencesActivity.onCreate()");
21          }
```

Desta maneira temos uma lista de definições pronta para ser modificada de modo a que cada utilizador tenha as definições a seu gosto.

SMActivity

Fig04. Shared Menu

A Activity é responsável pela implementação do menu partilhado pelas Activities 'UserStatusActivity' e 'TimelineActivity' (**S**hared**M**enu**Activity**).

Esta activity extende a superclasse Activity e implementa apenas três métodos: **onCreate**, **onCreateOptionsMenu** e **onOptionsItemSelected**.

No método **onCreate** é chamado o mesmo método da superclasse e iniciada a variável de instância **MyApplication app**, *protected*, para uso nas classes derivadas.

O método **onCreateOptionsMenu** é responsável por fazer o *inflate* do menu, passando o ficheiro .xml que contém o layout do menu, menu.xml .

Este ficheiro .xml contém 4 itens:

- Carregar as preferências (UserPreferencesActivity);
- Iniciar a timeline (TimelineActivity);
- Escrever um novo tweet (UserStatusActivity);
- Fazer a actualização da timeline;

No método **onOptionsItemSelected** é onde se atribui uma acção aos diversos itens que estão no menu, ou seja, iniciar as Activities consoante seja premido o item respectivo. A acção de actualização da timeline é feita na própria Activity, que ao extender SMActivity, irá fazer *override* deste método, chamando este (método da superclasse), e adicionando uma acção a este item.

DetailActivity

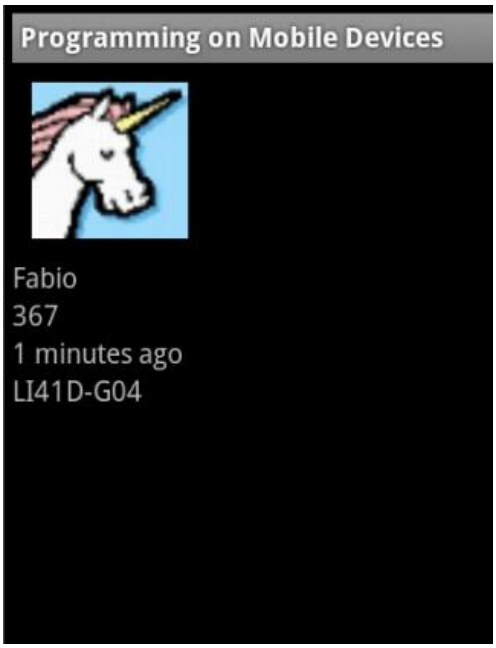

Fig05. Details

Esta Activity extende SMActivity e é bastante simples, contendo apenas dois métodos: **onCreate** e **init**, sendo o último chamado dentro do primeiro.

O primeiro método, analogamente a todas as Activities, começa por chamar o onCreate da superclasse. De seguida carrega o ficheiro detail.xml correspondente ao layout da Activity.

Este ficheiro contém:

- Um ImageView, onde será mostrada a foto do utilizador;
- Três TextView, encarregues por mostrar o nome do utilizador, a data e a mensagem em causa.

Por último é chamado o método **init**.

Este método é responsável por iniciar as Views do layout, e começa primeiro por se criar, através do método **getParcelable,** um objecto do tipo DetailsModel, que contém a informação da informação a ser mostrada. De seguida, com o método findViewById, afectam-se as variáveis de instância correspondentes às Views do layout, ou seja, um ImageView e três TextView. Após serem iniciadas as variáveis, são afectadas com os métodos, **setImageDrawable** e **setText** com os valores extraídos do objecto da classe DetailsModel.