

**1ª Serie**

**Programação em Dispositivos Moveis**

Ricardo Nunes - 31656

João Silvestre – 32766

15/04/2012

*Instituto Superior Engenharia de Lisboa*

Engenheiro: Pedro Pereira

Engenharia Informática e de Computadores

Semestre de Verão 2011/2012

Índice

[Introdução 4](#_Toc322290379)

[Estrutura do Trabalho 5](#_Toc322290380)

[MenuActivity 5](#_Toc322290381)

[LauncherActivity 5](#_Toc322290382)

[TwitterAsync 6](#_Toc322290383)

[TwitterAsyncTask 6](#_Toc322290384)

[Tratamento de Excepções 6](#_Toc322290385)

[YambaApplication 7](#_Toc322290386)

[LaunchModes 7](#_Toc322290387)

[Conclusão 8](#_Toc322290388)

# Introdução

Neste trabalho foi pedida a implementação de uma aplicação em Android, que serve de cliente de serviços do tipo do Twitter.

Neste documento encontra-se explicado todas as decisões e aspectos de implementação, que se considerem relevantes. Aqui encontram-se documentadas as decisões e o objectivo que se pretende atingir com a tomada destas.

# Estrutura do Trabalho

## MenuActivity

A aplicação precisa de um menu, que irá aparecer em duas actividades, dado que os passos para a criação de um menu numa actividade é igual para todas as actividades, variando apenas em pontos bastante específicos. Os pontos em que isto pode variar são:

* O estado dos *items* do menu;
* A acção dos botões.

A MenuActivity aparece então como uma forma de contrariar esta repetição permitindo, no entanto, pontos onde cada actividade possa se “ligar” de forma a poder alterar um dos pontos indicados anteriormente. Para este efeito foi usado o padrão Template Method e criado o método de *hook* doModifyMenu. Este irá permitir definir qual o estado dos botões do menu, sendo que todos os botões se encontram desactivados, usando o método auxiliar enableMenuItem. No entanto, isto não permite definir a acção dos botões, nesse aspecto foi decidido que quem pretende mudar o comportamento de algum dos botões deve fazer *override* ao método do Android SDK (onOptionsItemSelected), e definir ai o comportamento normalmente ou chamar o *super* caso não o queira mudar. Esta decisão aparece de forma a permitir que a implementação do comportamento seja feita da mesma forma, quer a actividade derive da MenuActivity ou não.

## LauncherActivity

A LauncherActivity aparece como resposta ao requisito que pede que o utilizador seja redireccionado para a actividade das preferências, na primeira utilização. Foi feita uma tentativa inicial desta redirecção ser feita no Application, e embora isto funcionasse, a redirecção apenas era feita após a actividade inicial ser criada. Isto obrigava então que a actividade tivesse que estar protegida contra a inexistência de propriedades que têm valores por omissão, sendo que na nossa opinião isto não faz sentido.

Esta actividade verifica então a existência das preferências, e com base nisso escolhe a actividade a abrir, portanto quando se quiser mudar a actividade inicial deve ser alterada aqui. De forma a que a LauncherActivity não fique no histórico, pois esta para o utilizador não existe, é necessário no manifesto adicionar o seguinte: android:noHistory=”true”.

## TwitterAsync

A comunicação com o serviço é feita através da UI, logo, toda esta interacção será executada na thread da UI. A comunicação com um serviço online deve ser sempre considerada uma acção lenta que, quando executada na thread da UI, irá dar ao utilizador a sensação de bloqueio, tornando a sua experiência de utilização má. De forma a evitar isto deve-se usar uma outra thread para executar estas acções, o Android oferece mecanismos, tais como as AsyncTask, para facilitar este processo.

De forma a tornar todo este processo transparente na UI foi criada a classe TwitterAsync que serve de proxy para a class Twitter, garantindo que todas as acções são executas de forma assíncrona. Esta permite que sejam registados *listeners* para diversas acções, sendo estes chamados na thread da UI pela *task* correspondente. Por exemplo, o *listener* StatusPublishedListener é chamado quando a tarefa acaba de enviar um *status*.

### TwitterAsyncTask

Existe um aspecto em comum partilhado por todas as *tasks,* que implementam uma determinada acção do Twitter API, e este aspecto é o facto de todas precisarem de um objecto Twitter para poderem efectuar a acção. Desta forma, a TwitterAsyncTask garante que esse objecto exista e seja o mais actual possível, pois o utilizador pode, por exemplo, mudar de utilizador e/ou serviço.

### Tratamento de Excepções

Dado que todas as acções são executadas em background é necessário permitir que a UI possa receber estas excepções, e trata-las como desejar. O mecanismo implementado, embora não sendo o ideal, permite que isto aconteça através de um *listener* especial, registado de forma estática, que é chamado sempre que uma excepção acontece.

Mas quem chama este *listener*? Esta é mais uma das responsabilidades da TwitterAsyncTask, que apanha todas as excepções e notifica quem estiver registado no *listener*, usando a thread da UI. Nesta versão da aplicação quem se encontra a receber todas as excepções é o objecto Application, que sempre que uma excepção de credenciais erradas ou de um URI inválido vai mostrar a actividade das settings. Também foi tentado que um Toast fosse mostrado a indicar qual o erro, no entanto, é necessário o contexto da actividade em *foreground* para mostrar um *toast*, sendo que nesta versão esta funcionalidade não se encontra a funcionar.

## YambaApplication

O objecto *application* não é só responsável por tratar as excepções ocorridas na interacção com o serviço, como foi explicado em tópicos anteriores. É também este que implementa os mecanismos que irá garantir que o proxy usado para comunicar com o serviço se mantém com os dados actualizados, quando o utilizador os modifica na actividade das preferências.

Para este efeito foi implementado, e registado, um *listener* OnSharedPreferenceChangedListener que se encontra à escuta de 3 preferências:

* Username
* Password
* ServiceURI

Este quando recebe uma notificação que uma destas propriedades foi mudada modifica a mesma no proxy do serviço.

## LaunchModes

O Android SDK disponibiliza diversas modos para iniciar actividades, estes permitem controlar quantas instâncias existem de uma determinada actividade. Nesta implementação foram usados dois desses modos, o singleInstance e o singleTask.

* TimelineActivity - singleTask
* StatusActivity - singleTask
* DetailsActivity - singleTask
* PreferencesActivity - singleTask
* LauncherActivity – singleInstance

A actividade LauncherActivity é singleInstance pois esta apenas serve para lançar as outras, não fazendo sequer parte da história, não havendo necessidade de existir múltiplas instâncias. A actividade DetailsActivity foi feita uma escolha de ser singleTask pois não se quer permitir que o utilizador crie demasiadas instâncias, vendo vários *tweets* sem usar o botão *back*. Caso seja necessário abrir uma nova instância pode-se sempre lançar numa nova *task*.

# Conclusão

Todos os objectivos propostos pelo enunciado foram atingidos, no entanto, algumas das soluções precisam de uma melhoria, principalmente o tratamento de excepções. A forma como o menu foi implementado também poderá apresentar alguns desafios no futuro dado que obriga a usar herança, e esta é “limitada” em Java.

No entanto, acredita-se ter-se conseguido uma solução modular e que permitirá a implementação de novas funcionalidades facilmente no futuro.