

**2ª Serie**

**Programação em Dispositivos Moveis**

Ricardo Nunes - 31656

João Silvestre – 32766

14/05/2012

*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 alterada a componente de comunicação da aplicação Yamba, recorrendo a serviços. Foi também implementado uma nova componente da aplicação, responsável por mostrar os dados do utilizador.

Neste documento encontram-se explicadas as implementações, e a forma como estas foram integradas com a aplicação.

# Estrutura do Trabalho

## TwitterAsync

Houve uma pequena alteração na implementação desta classe, todos os métodos passam agora a necessitar de saber o contexto (Android Context) em que foram chamados, para que se possa fazer o startService.

## StatusUploadService

A responsabilidade deste serviço é de fazer o upload do estado do utilizador, este serviço foi implementado com base na classe Service. Foi criada uma AsyncTask, muito parecida com a já existente mas que para o serviço quando concluí o envio, sendo que o acto de parar o serviço é feito num bloco finally para garantir que mesmo em casos de erro o serviço é terminado.

Na integração, com a aplicação, bastou apenas alterar a implementação do método updateStatusAsync, sendo que agora em vez de iniciar uma AsyncTask este vai começar o serviço, fazendo startService sobre o contexto passado no parâmetro.

## GetTimelineService

Este serviço foi implementado recorrendo a uma TimerTask, usando um Timer, esta permite uma forma fácil de agendar tarefas para serem executadas de uma forma recorrente.

Dado que este serviço é dependente de algumas preferências, este encontra-se à escuta de modificações nestas. Se a actualização automática for activada/desactivada é feito um cancelamento da tarefa que se encontra agendada, é feito um novo agendamento tendo em conta a escolha do utilizador. Quando o tempo entre actualizações é mudado, caso a actualização automática esteja activa, é também feito um novo agendamento da tarefa.

A integração com a aplicação foi feita da mesma forma que a usada no StatusUploadService.

# Conclusão

A estrutura de comunicação, com o serviço, implementada na fase anterior permitiu que estas alterações fossem realizadas de uma forma fácil e quase transparente para a aplicação. No entanto, a forma como o bounded service foi implementado carece de cuidados, e espera-se conseguir uma melhor integração deste, com o resto da aplicação, na próxima fase.