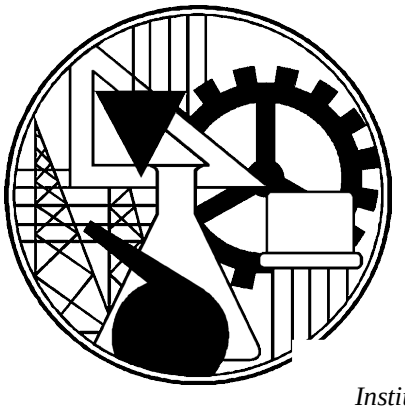

ISEL

Instituto Superior de Engenharia de Lisboa

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Programação em Dispositivos Móveis

Relatório do 1º Trabalho

Docente: Pedro Pereira

Discentes: Ana Correia e Diogo Cardoso

Abril 2012

Conteúdo

Introdução 3

Integração de Serviços 4

Started Services 4

Implementação 5

Bounded Service 7

Conclusão 8

Introdução

Este relatório aborda as soluções adotadas na realização do segundo trabalho da unidade curricular *Programação em Dispositivos Móveis*. O objetivo do trabalho era a familiarização com o componente *Service* pertencente à plataforma *android*.

Devido à solução desenvolvida na primeira fase do trabalho, o foco deste trabalho passou apenas pela integração com da solução anterior com os serviços da plataforma. Esta alteração foi realizada sobre a abstração criada anteriormente para acesso ao serviço *Yamba*.

Integração de Serviços

A solução deveria usar três serviços:

- StatusUploadService - serviço que trata de realizar o *upload* de um *status*.
- TimelinePullService - serviço que verifica periodicamente se existem alterações na *feed* de status do utilizador.
- UserInfoPullService - serviço que sabe obter a informação sobre o utilizador.

Destes três serviços dois deles são do tipo *StartedService* (StatusUploadService e TimelinePullService) e um do tipo *BoundedService* (UserinfoPullService), os serviços do tipo *Started* executam-se no mesmo processo que a aplicação enquanto que o serviço *Bounded* executa-se num processo isolado.

Started Services

A implementação do comportamento dos serviços foi bastante simples já que o código utilizado no último trabalho foi reaproveitado, a maior diferença está no comportamento desde que a *activity* inicia a operação assíncrona até que é reportado o resultado dessa operação.

Anteriormente o fluxo de execução era o seguinte:

- *Activity* regista um *handler* na instancia de *TwitterServiceClient*, para ser chamado quando uma operação assíncrona tenha acabado.
- *Activity* inicia a operação assíncrona.
- Uma *AsyncTask* é lançada para realizar a operação.
- A *AsyncTask* retorna e chama o *handler* previamente registado pela *activity*.

Com a integração dos serviços o fluxo de execução é:

- *Activity* regista um *handler* na instancia de *TwitterServiceClient*, para ser chamado quando uma operação assíncrona tenha acabado.
- *Activity* inicia a operação assíncrona.
- A instancia de *TwitterServiceClient* inicia o serviço passando como parâmetro um *handler* de forma a que este possa notificar quando a sua operação tiver completa.

- O serviço arranca e realiza a operação de forma assíncrona.
- O serviço chama o *handler* de *TwitterServiceClient* de forma a que este notifique a *activity*.
- A instancia de *TwitterServiceClient* trata de garantir que o *handler* registado pela *activity* é chamado no contexto da *thread main*.

Este sistema faz com que toda a infraestrutura seja *event driven* tornando-a flexível e simples de utilizar.

Implementação

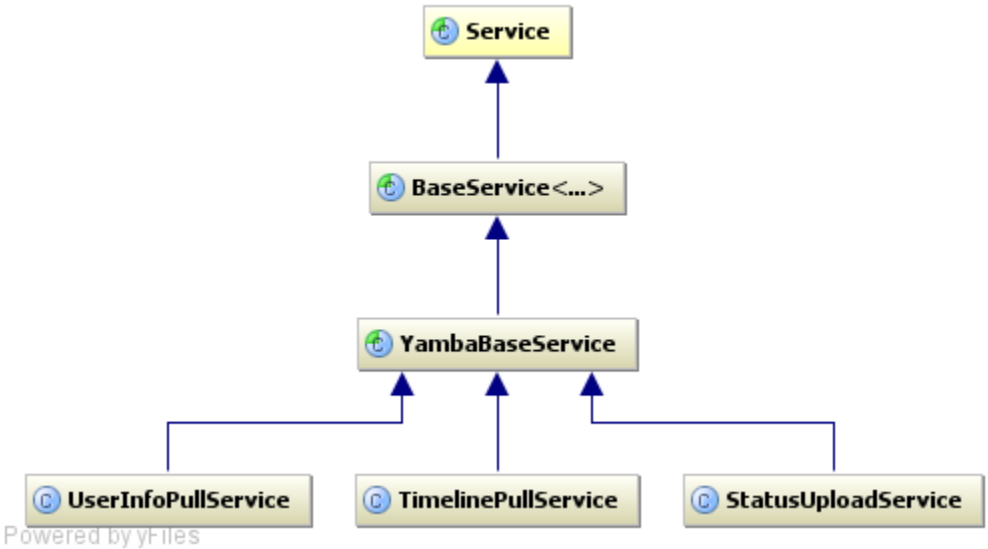

Figura 1 - Diagrama dos serviços implementados.

A Figura 1 ilustra os três serviços implementados no âmbito deste trabalho. As classes *BaseService* e *YambaBaseService* são classes de extensão à infraestrutura *android* contendo neste momento apenas o acesso tipificado à instancia de *Application* e o acesso à instancia de *NavigationMessenger*.

Dos dois serviços do tipo *StartedService* o único que mereceu especial atenção foi o *TimelinePullService*, devido à natureza da sua funcionalidade. O serviço *TimelinePullService*

deverá periodicamente verificar se existem novas *status* presentes na *timeline* do utilizador, para tal é necessário utilizar algo que seja chamado periodicamente. A solução passou por utilizar uma *TimerTask* que implementa o comportamento de chamar um objeto função de forma periódica.

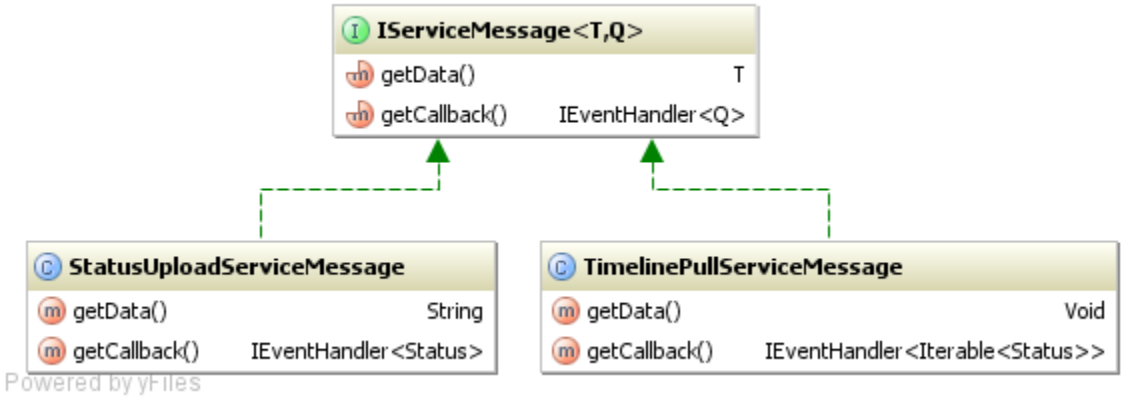

Figura 2 - Diagrama de uma mensagem de serviço.

A passagem de dados do fluxo referido anteriormente é realizado através instancias de *IServiceMessage* ilustrada na Figura 2. Neste é possível observar a presença de um *callback* com o tipo concreto do retorno do serviço bem como o tipo concreto dos parâmetros do serviço. Uma vez que os serviços do tipo *Started Service* se executam no mesmo processo que a aplicação cliente foi utilizado o *NavigationMessenger* criado no desenvolvimento do trabalho para passar instancias do tipo *IServiceMessage*.

Bounded Service

A implementação e consumo do serviço do tipo *bounded* foi ligeiramente diferente, pelo facto de não ter sido usada a infraestrutura construída até aqui, seja para o envio de informação para o serviço seja para receber o seu retorno, isso deve-se maioritariamente à natureza deste serviço especifico já que este irá ser executado num processo diferente ao processo que corre a aplicação.

A implementação concreta do serviço *UserPullService* foi realizada através de *Messengers* e *Handlers*. Esta opção deveu-se ao facto da utilização destes mecanismos serem por si só assíncronos não necessitando de começar novas *threads* de forma a que a *thread main* não bloqueie. A alternativa seria usar serviços do tipo *AIDL* que pela sua natureza bloqueante seria necessário gerir mais *threads* para o invocar caso fosse necessário o fazer dentro da *thread main*.

Conclusão

Ao concluir este trabalho foi notório que o esforço realizado no trabalho anterior foi bastante proveitoso. A infraestrutura *event driven* criada encaixou perfeitamente com este novo passo dos serviços, tornando a sua integração rápida e simples. Apesar do cliente do serviço *bounded service* ter sido realizado dentro da *activity* que o consume com a infraestrutura que foi construída é possível que este seja abstraído na próxima fase do trabalho.