

Comunicação

Para este protótipo, além da implementação do grafismo, da base de dados e das funções básicas da lista de contactos e do Moods em si, decidimos enveredar também por uma tentativa de comunicação entre o Android e o servidor.

O primeiro passo foi a definição de objectivos e procura de exemplos: foi acordado iniciar este módulo pelas maneiras mais simples de enviar e receber dados entre o Android e o servidor. Para isso, procurámos enviar uma série de pares nome-valor num URL e criar do lado do servidor um script simples, usando a linguagem ASP, aprendida na disciplina de Laboratório Multimédia 5, cuja função seria ler estes pares e alterar uma BD de teste.

Este passo revelou-se relativamente simples de aprender, visto que o Java, linguagem em que o Android funciona, contém algumas funcionalidades que lidam directamente com o envio de URLs.

Decidimos então, visto que já conseguíamos assegurar um envio de dados mínimo entre o Android e o servidor, subir um pouco a fasquia, e preparar toda a parte de comunicação de modo a que esta fosse o mais definitiva possível, exigindo o menor número de alterações posteriores. Tendo em conta que teremos de trabalhar com múltiplos utilizadores, isto levou-nos a alterar o modo de envio destes dados: passou a ser pretendido um método HTTP POST (para não enviar os parâmetros por URL, considerando as limitações do número de caracteres que cada url pode ter, para além da segurança acrescida que este método proporciona). A linguagem do servidor foi também alterada: PHP pareceu-nos a escolha óbvia, pelas facilidades de comunicação com MySQL, e pelo facto dos webservices em SOAP relativos ao SAPO funcionarem com esta linguagem, permitindo uma mais fácil adaptação posterior.

Em relação ao HTTP POST, encontramos diversos exemplos espalhados pela web: recolhemos alguns, para compreensão do seu funcionamento e adaptação de algumas funções ao nosso serviço. Foi-nos posto então um grave problema: com a publicação do último SDK do Android várias funcionalidades foram retiradas, sendo que todos os exemplos que conseguimos encontrar simplesmente não funcionavam, impossibilitando-nos de compreender o seu funcionamento. Uma nova pesquisa revelou-se também infrutífera, obrigando-nos a alterar a relevância deste método, passando-o para segundo plano.

Para a recepção de dados, o servidor passou a ter a funcionalidade de criar XMLs com dados que o Android teria de ser capaz de ler e atribuir aos campos correctos (para mais informação sobre os comportamentos do servidor, basta ler o post seguinte).

Isto criou diversos novos problemas para a parte do Android: para além de ser necessário encontrar forma de conseguir que o nosso programa lesse um XML, teria de

conseguir reconhecer os diferentes campos, reconhecer os campos da BD a que estes correspondem, e atribuir os valores correctos a cada campo. Isto tendo em mente que a aplicação suportaria mais do que os dois utilizadores estabelecidos para o protótipo.

Tendo em conta que os leitores RSS se baseiam em ficheiros XML, estabelecemos uma analogia com o que desejávamos fazer e conseguimos reunir diversos exemplos de leitores RSS já criados para o Android. A maior parte dos que encontrámos utilizava livrarias externas ou não disponibilizava o seu código de forma livre, pelo que optámos apenas pelo único que encontrámos e que não apresentava nenhuma destas limitações.

Este leitor funciona com base numa livraria disponibilizada no próprio Android chamada SAXParser, que contém diversas funções criadas a pensar especificamente na leitura de XMLs.

O leitor em si funciona da seguinte forma: abre um URL que contém um ficheiro XML, e utiliza o SAXParser para ler todas as tags de início e fim, e obter os caracteres entre elas, utilizando variáveis definidas pelo programador de forma a indicar ao programa dentro de que tag se encontram os dados.

Enquanto a forma de processar o XML se aproxima ao desejável, a forma de apresentação e tratamento destes dados não: o programa apenas concatena todos os dados numa string e apresenta-os no ecrã.

Decidimos então não só adaptar a forma de tratamento destes dados, de modo a que estes fossem enviados para os campos respectivos na BD do Android, mas também fazer isto de maneira a preparar o suporte para múltiplos utilizadores. A opção óbvia tornou-se em fazer funções de envio e recepção, pensadas de forma a funcionar com múltiplos campos e muita informação, sendo estas usadas para obter os parâmetros necessários nas suas diversas aplicações .

Após muitas horas de trabalho a tentar criar uma solução que cumprisse o idealizado, foram detectados diversos problemas graves que impossibilitaram o desenvolvimento desta solução de forma a suportar multi-utilizador, pelo menos nesta fase:

O primeiro destes problemas tem a ver com o número de campos e de utilizadores lidos no XML. Ao tentar criar uma solução universal, pensámos em guardar a informação num array. Deparámos então com o primeiro problema: em Java, ao declarar um Array, é necessário indicar o seu tamanho antes de o utilizar, não sendo possível alterar este tamanho após a sua criação. É portanto fulcral saber a quantidade de utilizadores e de campos antes de iniciar o parsing do XML. O passo seguinte passou pela decisão de criar um header que viria do servidor contendo o número de campos, os seus nomes e o número de contactos a ler. Foram testadas várias soluções, até termos descoberto que o SAXParser torna impossível parar o seu processamento

durante a leitura para executar outras funções, deixando-nos apenas com 2 soluções: ou teríamos de ler o XML duas vezes, ou dividir em 2 XMLs a informação, contendo um o header e o outro os campos e os contactos. Decidimos optar pela segunda.

Outro problema tem a ver com o passamento de parâmetros entre funções. Pelo facto de termos dividido toda a leitura e envio em várias funções, por vezes estas têm de ser chamadas entre si, e cada uma necessita de diferentes parâmetros para se iniciar e devolver, o que causou algumas incompatibilidades.

Os diversos problemas que tivemos levaram-nos a optar por reverter as nossas ambições em termos de comunicação para um nível mais baixo: decidimos optar por voltar para apenas 2 utilizadores, dando uma ideia geral da comunicação, sendo que todo o trabalho que desenvolvemos neste sentido, da parte do Android, será reaproveitado e melhorado no próximo módulo da disciplina. Da parte do servidor, este encontra-se totalmente operacional, tendo uma base de dados avançada de forma a rentabilizar estas funcionalidades no futuro.

A nossa solução para a comunicação entre os dispositivos passa então por fazer com que o Android faça pedidos ao servidor, através de uma página PHP, que vai invocar funções para cada acção que o utilizador queira executar.

Claro que este método traz alguns problemas, sendo que o maior se centra no perigo de o telemóvel ficar dependurado a tentar estabelecer a conexão com o servidor.

Seguindo este modelo de conectividade, desenhamos então uma pequena arquitectura de serviços, específica para o protótipo, mas pensada de maneira a poder ser reutilizada mais tarde, baseada num modelo de pedido-resposta.

Este modelo processa-se então da seguinte forma. Um dispositivo móvel Android, com a aplicação Sapó Moods executa uma função. Essa função despoleta uma acção no servidor, através de uma string URL. O servidor executa a acção, e dá uma resposta ao Android.

Vejamos por exemplo a função de alterar o meu estado.

1. O dispositivo móvel Android executa a função "ChangeStatus".
2. Essa função despoleta uma acção, que envia, numa string URL, os parâmetros ID_Status, StatusMotive e ID_Moods para uma página *.php;
3. Essa página, por sua vez, lê esses parâmetros enviados no URL e executa uma função "UpdateStatus", que faz um "update" aos campos "ID_Status" e "StatusMotive" da tabela MOODS, para um determinado "ID_Moods".

// exemplo do ficheiro PHP que executa as funções

```
<?php
/* primeiro incluimos o ficheiro que contem as funções */
include "functions.php";

/* em segundo lugar chamamos do URL os seguintes pares
nome=valor */
$Moods = $_GET["ID_Moods"];
>Status = $_GET["ID_Status"];
$Motive = $_GET["StatusMotive"];

/* depois invocamos as funções que estão definidas na
página php que incluimos no inicio deste ficheiro */

UpdateStatus($Moods,$Status,$Motive); /* chama a função
que vai escrever na base de dados os valores enviados
enviados pelo Android */

WriteXML(); /* chama a função que vai escrever os ficheiros
XML para cada utilizador Moods (demonstrado na
imagem anterior)*/

?>
```

// exemplo do ficheiro PHP que contém as funções

```
<?php

/* criamos a função de "update" dos estados na BD */
function UpdateStatus($Moods,$Status,$Motive) {

    /* criamos a ligação com o servidor */
    if(!$dbconnect = mysql_connect('localhost','username',
'password')) {
        echo "Connection with the server failed.";
        exit;
    }

    /* criamos a ligação com a BD */
    if(!mysql_select_db('username_db')) {
        echo "Cannot connect to database.";
        exit;
    }

    /* fazemos update aos campos "ID_Status" e
    "StatusMotive" para o "ID_Moods" que é passado na
    função */
    $table1 = 'MOODS';
    $query1 = "UPDATE $table1 SET `ID_Status` = '$Status',
'StatusMotive' = '$Motive' WHERE $table1.`ID_Moods` =
'$Moods'";
    $dbresult = mysql_query($query1, $dbconnect);

    /* fechamos a ligação com a BD */
    mysql_close($dbconnect);
}

/* criamos a função de criação/update dos ficheiros XML */
function WriteXML() {
    ...
    ...
}

?>
```

4. Depois de resolvida esta função, a página invoca uma outra função, "WriteXML",

que escreve ficheiros *.XML para todos os utilizadores, com os contactos dos mesmos, sempre que existe uma alteração do estado ou do motivo.

A solução é ainda bastante simples, mas serve os propósitos do protótipo. Estes ficheiros XML são criados para que, depois de devidamente lidos pelo Android, estes permitam a actualização dos campos Estado e Motivo para todos os contactos de cada utilizador do Android.

// exemplo de um ficheiro XML

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  /* para cada utilizador que pertence à lista de contactos
  deste mood existe uma tag "main", dentro da qual estão
  os filhos ('ID_Moods', 'ID_Status' e 'StatusMotive') */
  <main>
    <ID_Moods>"username"</ID_Moods>
    <ID_Status>"Estado"</ID_Status>
    <StatusMotive>"Motivo"</StatusMotive>
  </main>
  /* a tag "main" é assim replicada tantas vezes quantos
  forem os ID_Moods associados a um utilizador */
  <main>
    ...
  </main>
</root>
```