



Universidade do Porto  
Faculdade de Engenharia

**FEUP**

---

# **Tumblr – Aplicação Android**

---

## *Relatório Final*

### **Sistemas Distribuídos**

3º Ano do Mestrado Integrado em Engenharia Informática e  
Computação

#### **Elementos do Grupo:**

Fábio Filipe Costa Pinho, 080509111 - ei08111@fe.up.pt  
Joana Filipa Vieira Barbosa, 080509086 - ei08086@fe.up.pt  
Nuno Machado Matos, 080509140 – ei08140@fe.up.pt  
Tiago Daniel Sá Cunha, 080509142 – ei08142@fe.up.pt

---

Porto, 5 de Junho de 2011

## Índice

1. Objectivo .....	3
2. Descrição.....	3
a) Funcionalidades.....	3
b) Estrutura do programa.....	4
• <i>Geral</i> .....	4
• <i>Autenticação</i> .....	4
• <i>Visualização</i> .....	4
• <i>Upload</i> .....	4
c) Avaliação .....	5
d) Valorização .....	6
e) Distribuição de trabalho.....	6
3. Conclusões .....	7
4. Recursos.....	7
5. Anexos.....	8
a) Exemplo da execução .....	8

# 1. Objectivo

---

No contexto da unidade curricular de Sistemas Distribuídos, foi-nos proposto o desenvolvimento e implementação de uma aplicação que permitisse ao utilizador a sua interacção com um website aplicando o paradigma de REST.

Não foram criadas quaisquer limitações relativamente à escolha do website nem às API a serem utilizadas, portanto optámos por desenvolver uma aplicação que promovesse essa interacção com o website *Tumblr*.

O objectivo desta aplicação passa, fundamentalmente, por disponibilizar ao utilizador uma ferramenta de uso simples e que lhe possibilite autenticar-se na sua conta, bem como ver e editar os conteúdos da mesma.

Considerámos um objectivo alcançável mas, ao mesmo tempo, um desafio custoso e interessante, uma vez que essa aplicação seria desenvolvida em *Android*.

## 2. Descrição

---

### *a) Funcionalidades*

A aplicação desenvolvida tem como funções a visualização e inserção de conteúdos no perfil do utilizador no *Tumblr*, bem como a autenticação dos seus dados da conta.

Através da *password* e do *username* definidos aquando da criação da sua conta, o utilizador pode efectuar o *login*, autenticando-se para, de seguida, poder usufruir da aplicação e editar a sua conta.

A visualização dos conteúdos da sua *dashboard*, passa pela apresentação de todos os ficheiros ou *posts* que estejam presentes no perfil, tais como citações, imagens, vídeos, etc. Esta funcionalidade foi implementada para o utilizador poder consultar aquilo que já detém no seu perfil, impedindo assim que faça *upload* de conteúdos repetidos.

Por outro lado, e como última funcionalidade, temos a inserção de conteúdos no perfil. Essa inserção pode ser feita de duas maneiras: *upload* de ficheiros presentes num directório ou através de um url que contenha aquilo que o utilizador pretende mostrar, nomeadamente imagens, vídeos ou áudio. Esta funcionalidade foi imposta por ser considerada essencial e por representar a melhor maneira de cumprirmos os requisitos e objectivos propostos, isto é, a interacção através do conceito REST.

Ainda relativamente à inserção de conteúdos na *dashboard*, é também possível inserir *posts* de texto normais bem como citações e conversações, que possibilitam a interacção de um utilizador com outros que estejam registados no *Tumblr*.

As funcionalidades acima descritas estão desenvolvidas para *Android*, constituindo uma agradável interface para a sua utilização e colocando o projecto num panorama bastante actual.

## *b) Estrutura do programa*

- ***Geral***

O estilo *RESTFUL* consiste numa arquitectura entre clientes e servidores em que os clientes, através de pedidos, comunicam com o servidor e este, em resposta ao pedido e mediante o protocolo sob o qual se rege, devolve uma mensagem cujo conteúdo corresponde ao pretendido pelo cliente.

Esta resposta pode ser dada em vários formatos. Neste caso, o servidor do *Tumblr* responde com uma mensagem estruturada em *XML* e *JSON*, necessitando, posteriormente, de um *parser* para separar a informação pretendida.

- ***Autenticação***

A autenticação foi feita utilizando a classe *URLConnection* do *Java*, através do envio de uma *string* data em formato GET, com a informação de login, isto é, *email* e *password* do cliente.

Como resposta, o servidor retorna uma *string* que representa um ficheiro XML onde estão descritas todas as informações da conta correspondente, nomeadamente o nome, número de *posts*, tipos de permissões, entre outros. Dessa *string* é apenas necessário retirar o nome escolhido pelo cliente para a sua conta, uma vez que este campo é utilizado posteriormente para aceder aos *posts* existentes.

- ***Visualização***

A componente de visualização dos posts contidos no *tumblr* corresponde à parte de leitura da *DashBoard* da conta do cliente. Para aceder à *DashBoard* é necessário fazer um pedido *URLConnection* ao site <http://nomecliente.tumblr.com/api/read/json>. A última componente do url descreve o tipo de resposta que pretendemos ao pedido realizado.

Neste caso, a resposta do servidor é dada através da linguagem de interpretação JSON, o que facilita em muito o *parsing* da resposta através do recurso a bibliotecas Java. A informação relativa aos *posts* é guardada num *array*, tornando-a acessível em qualquer altura sem que seja necessário refazer o pedido ao servidor.

- ***Upload***

A componente de upload, corresponde ao envio de *posts* para o *tumblr*. A ligação ao servidor é realizada através do envio de uma *string* de conteúdo variável consoante o tipo de *post*, através de *URLConnection*. Há sete tipos possíveis de *posts*: texto, imagem, citação, *link*, conversa, áudio e vídeo. Para cada *post* é então enviada para um servidor uma *string*

com os dados de autenticação do cliente, bem como a informação relativa ao *post*. O servidor devolve o número identificador do *post*, caso este tenha sido efectuado com sucesso, ou um código de erro, identificador do tipo de erro que tenha surgido. Os casos de upload de ficheiros a partir de um directório local é um caso especial em relação aos demais. O cliente pode optar por fazer upload de imagens, áudio ou vídeo (mediante os formatos aceites pelo próprio *tumblr*).

A componente de upload dos ficheiros é realizada com recurso a um *multipart-form* em Java, em tudo semelhante ao utilizado, por exemplo, nas páginas Web.

### c) Avaliação

Ao longo do desenvolvimento da aplicação, foram sendo testadas as várias possibilidades de upload de ficheiros ou informação. Primeiramente, foi testada a inserção de textos regulares com título e mensagem, confirmando o sucesso da acção.

Em seguida, testámos a inclusão de citações e conversações. As primeiras conteriam a fonte e o conteúdo e as últimas apenas a mensagem. Uma vez mais, foi confirmado o sucesso da acção.

Posteriormente, foi testado o *upload* de ficheiros por *URL* e através do directório, tendo sido escolhidas as imagens como primeiro tipo de conteúdos a serem enviados. Apesar de o *upload* pelo *url* ser uma funcionalidade relativamente simples de ser feita, o *upload* a partir do directório constituiu o desafio mais difícil, devido ao método aceite pela API do *Tumblr* representar um método com escassez de informação e bastante precário na sua objectividade, isto é, foram necessárias várias restrições ao nível da encriptação e codificação do conteúdo da informação apesar de isso não ser especificado na documentação do *Tumblr*.

Uma vez atingido o sucesso para o *upload* das imagens, o *upload* de áudio e vídeo tornou-se mais simples, pois o método era igual, mudando apenas alguma informação a ser enviada.

Relativamente ao processo de *download* e disponibilização de informação ao utilizador, tornou-se factual a sua simplificação para todos os tipos de *posts* que não fossem áudio, vídeo e imagem, uma vez que já possuíamos todo o conteúdo destes *posts* num *array*, obtido depois do *parsing* da resposta em JSON recebida.

Assim, para estes casos, apenas se tornou necessária a apresentação dos diferentes tipos de informação relativos a cada *post*, depois de os encontrar numa listagem de *posts* da *DashBoard*.

Contudo, para os outros tipos de *posts*, foram adoptados diferentes métodos. Para as imagens, foi utilizado um método que obteria a imagem propriamente dita através do *url* associado a cada *post* do tipo imagem. Foi usada, então, a classe *Drawable* e a função *ImageOperations* do *Android* para associar, depois, esta imagem a uma *ImageView*.

Por sua vez, o vídeo é reproduzido depois de sofrer o mesmo tipo de acção que a imagem, pois também é obtido através do *url* criado pelo *Tumblr*. Para a implementação em *Android*, foram utilizadas as classes *MediaController* e *Uri*.

Contudo, surgiu um imprevisto, uma vez que o emulador de *Android* onde desenvolvemos a aplicação não aceita a reprodução de um formato de vídeo para além do *3GP*, ao contrário do sistema operativo *Android* que aceita, por exemplo, o formato *MP4*. Este problema ocorreu devido ao facto de o Tumblr não aceitar o upload de vídeos no formato 3GP, por isso torna-se impossível a visualização dos vídeos que conseguimos fazer upload pela aplicação. O mesmo se aplicou aos ficheiros áudio que apesar de serem detectados por parte do Android, não eram reproduzidos, devido a incompatibilidades com o emulador.

Por fim, uma nota final para o tipo de *post Link*, em que o conteúdo da página Web relativa a este *post* é visualizada através do próprio browser do sistema operativo Android.

De um modo geral este processo de avaliação foi-nos favorável pois conseguimos testar a qualidade da nossa aplicação e corrigir os erros verificados ao longo da sua implementação.

#### *d) Valorização*

Devido à conjugação de prazos com outros projectos, resultando na falta de tempo por parte dos elementos do grupo, infelizmente não conseguimos implementar nenhum dos dois pontos registados como funcionalidades para valorização no template de especificação.

Acrescendo às justificações anteriores devemos agora referir que a API do *Tumblr* não responde a pedidos de leitura dos conteúdos de uma *DashBoard* qualquer, sem ser efectuada a autenticação previamente, e fornecendo os dados de *username* e *password* no pedido, pelo que se tornou impossível a ideia de obter o conteúdo de posts originários de outros utilizadores baseando-se em tags de pesquisa.

#### *e) Distribuição de trabalho*

A distribuição de trabalho para a execução deste projecto ficou dividida da seguinte maneira:

Fábio Pinho – *Upload* de imagens, áudio e vídeo através de *REST*. Desenvolvimento da autenticação dos utilizadores no *Tumblr*. Interpretação das respostas do servidor. Criação de páginas de leitura de *posts* de link, áudio e vídeo.

Joana Barbosa – Desenvolvimento da parte da aplicação *Android* referente à pesquisa e *upload* de ficheiros e visualização dos diferentes tipos de *posts*. Criação dos menus de navegação na aplicação e de autenticação.

Nuno Matos – *Upload* de *posts* de texto (normais/*regular*, citações/*quotes* e conversa/*conversation*). Criação de menus de navegação na aplicação android. Elaboração do relatório.

Tiago Cunha – Interpretação e organização dos *posts* recebidos em JSON. Listagem de *posts* da *Dashboard* na aplicação e criação de páginas de leitura de *posts* de texto (*normais/regular*, citações/*quotes* e conversa/*conversation*) .

## 3. Conclusões

---

Em suma, relativamente ao template de especificação entregue anteriormente, conseguimos implementar as funcionalidades a que nos propusemos, possibilitando a obtenção do produto final desejado.

Ainda em comparação com o template, verificámos também que várias ideias podiam ter sido melhor estruturadas (nomeadamente a nível das funcionalidades, uma vez que existiam pontos e aspectos que se podiam fundir numa só categoria) e que o não cumprimento das valorizações sugeridas constitui a maior frustração para os elementos do grupo.

Assim, de um modo geral, este trabalho foi bastante recompensador: cumprimos os objectivos, ganhámos experiência e aprendemos novos conceitos que, com a evolução constante destas plataformas, vão ganhar muita preponderância no quotidiano.

## 4. Recursos

---

Wikipedia - Representational State Transfer [Em linha] Porto. [Consult. 25-05-2011]. Disponível em WWW:

[URL:http://en.wikipedia.org/wiki/Representational\\_State\\_Transfer](http://en.wikipedia.org/wiki/Representational_State_Transfer)

Fernandes, José Maria - An (almost codeless) overview of Android [Em linha]. Porto [Consult. 16-05-2011]. Disponível em WWW:

[URL:http://moodle.fe.up.pt/1011/file.php/678/ACodelessOverviewOnAndroid\\_by\\_JFernandes.pdf](http://moodle.fe.up.pt/1011/file.php/678/ACodelessOverviewOnAndroid_by_JFernandes.pdf)

Maranhão, Rui - R04/06 – RESTful [Em linha]. Porto [Consult. 16-05-2011]. Disponível em WWW:

[URL:http://moodle.fe.up.pt/1011/mod/resource/view.php?id=15211](http://moodle.fe.up.pt/1011/mod/resource/view.php?id=15211)

Maranhão, Rui - Assignment #2: Your App [Em linha]. Porto [Consult. 10-05-2011 – 05-06-2011]. Disponível em WWW:

[URL:http://moodle.fe.up.pt/1011/mod/resource/view.php?id=16803](http://moodle.fe.up.pt/1011/mod/resource/view.php?id=16803)

## 5. Anexos

---

### *a) Exemplo da execução*

