# Redes Móveis e Sem Fios

## Relatório Final

Development of Internet of Things sensor monitoring based on SigFox, Arduino and Android

Bernardo Gomes, 75573 Diogo Martins, 75462

5 de Maio de 2016

# Conteúdo

# 1 Objectivo

O objectivo do projecto é o desenvolvimento de um sistema de monitorização de temperatura.

O sistema, deverá ser baseado num sensor de temperatura associado a um dispositivo *Arduino (Akeru 3.3)*, que irá comunicar as suas medições a um servidor *SigFox*, armazenando-as na *Cloud*.

Na óptica do utilizador, irá ser desenvolvida uma aplicação em ambiente *Android*, que fornecerá os dados presentes na *Cloud* com uma apresentação *user friendly*. Pretende-se ainda que seja possível que o utilizador registe um novo dispositivo a monitorizar na aplicação, bem como definir alarmes para certos valores de temperatura.

# 2 Solução encontrada

Tal como referido na secção anterior, a monitorização da temperatura e da qualidade de medição do sensor, irá ser feita pelo utilizador com recurso à aplicação, mas tendo a *Cloud SigFox* como intermediária.

Por forma a que os dados de cada utilizador sejam independentes do dispositivo utilizado, foi construída uma base de dados adicional, que guarda informação de *login* bem como dos *devices* e os *thresholds* de alarme de cada utilizador.

A arquitectura será então a apresentada na figura 1:

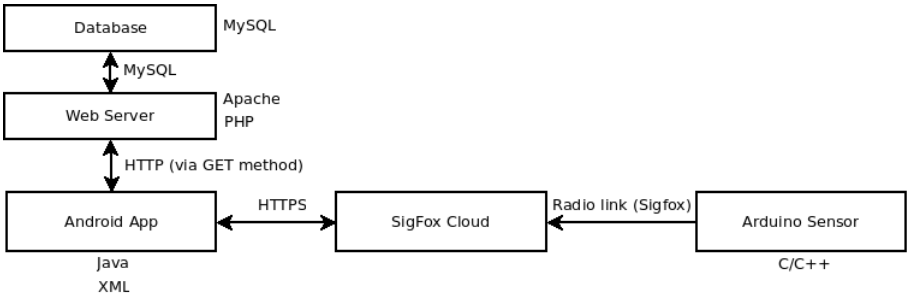


Figura 1: Arquitectura geral

## 2.1 Aplicação *Android*

A aplicação *Android*, com a qual o utilizador irá ter contacto directo, será constituída por cinco actividades:

- *MainActivity*;
- *CreateLogActivity*;
- *LogsActivity*;
- *NewAlarmActivity*;
- *AddDeviceActivity*.

As relações entre as actividades descritas, encontram-se representadas na figura 2.

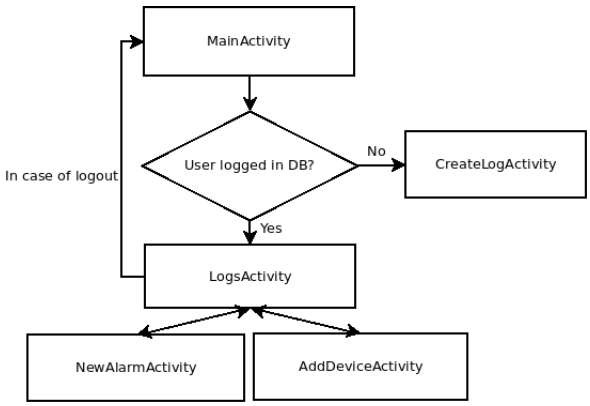


Figura 2: Arquitectura da aplicação *Android*

A *MainActivity* tem como objectivo perguntar ao utilizador o nome com o qual está registado na rede, realizando *queries* à base de dados com as informações do utilzador. No caso de existir registo prévio, as informações são armazenadas num ficheiro que irá funcionar como *cache* na aplicação e é lançada a *LogsActivity*. Caso o utilizador na tenha registo, é lançada a *CreateLogActivity* para registar o novo utilizador.

A actividade de registo de um utilizador (*CreateLogActivity*), terá apenas três campos de inserção de texto: *username, password* e *devicetype-id*. Estes parâmetros são gravados no ficheiro de texto descrito anteriormente, bem como na base de dados central. De seguida, a aplicação irá lançar a *LogsActivity*.

A actividade de visualização da informação da *Cloud* (*Logs*), é constituída por duas *threads* principais. A primeira consiste na obtenção das mensagens do dispositivo por pedidos HTTPS (GET) periódicos, de acordo com a informação de registo do utilizador e que é actividada por uma *checkbox*. Posteriormente a resposta será enviada para a *thread* principal (da API) que a disponibiliza ao utilizador. No caso de a *checkbox* não estar activa, os pedidos podem ser efectuados apenas ao clicar no botão de pedido de informação. O esquema desta actividade está descrito na figura Figura 3.

Esta actividade tem ainda a opção de registar um novo dispositivo para monitorização, bem como adicionar um novo alarme de temperatura. No caso de um *threshold* de temperatura, lido da base de dados no início da aplicação, ser ultrapassado, a *thread* que realiza o *parsing* da informação deverá lançar uma notificação ao utilizador.

À semelhança da actividade *CreateLogActivity*, as actividades *AddDeviceActivity* e *NewAlarmActivity* serão apenas compostas por campos de texto. Após o registo num ficheiro e na base de dados das informações recolhidas, estas actividades irão retornar no *stack*, voltando à actividade anterior.

No caso de o utilizador pretender fazer *logout*, é lançada a *MainActivity*, por forma a que seja possível iniciar a sessão com outro *username*.

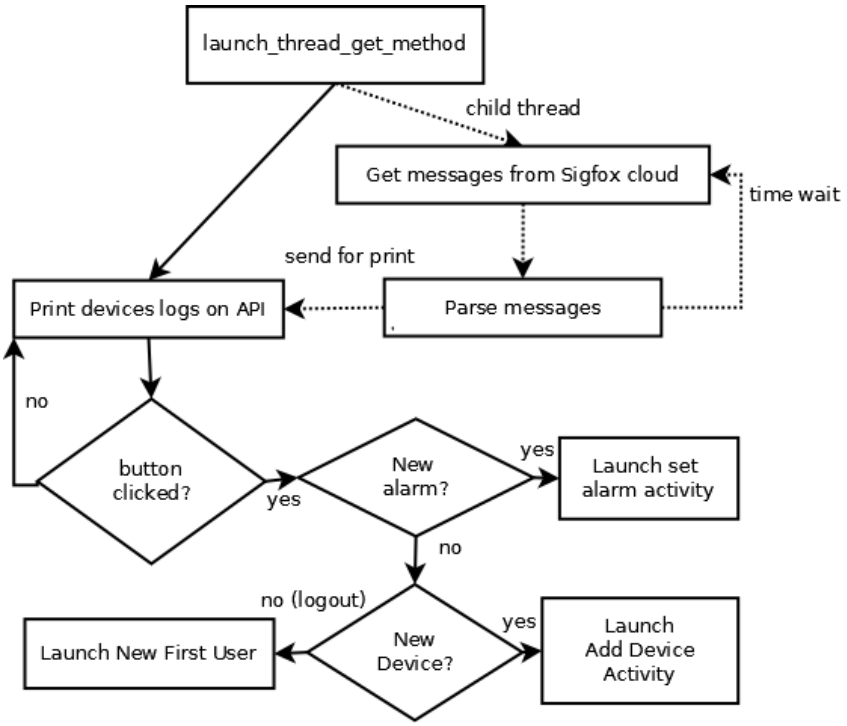


Figura 3: Arquitectura da actividade *Logs*

## 2.2 Servidor *SigFox*

O papel deste servidor é o armazenamento da informação medida pelo sensor e a recepção de pedidos por parte da aplicação *Android*. Consoante o tipo de pedido, irá realizar uma resposta em *JSON* com as informações das medições.

Em termos de implementação para o projecto, foi apenas necessário conhecer a forma como os pedidos devem ser realizados bem como o formato de resposta.

## 2.3 Sensor Arduino

O sensor deverá realizar medições periodicamente enviando-as para a *Cloud SigFox*, via rádio. Sendo possível enviar 120 mensagens para a *Cloud SigFox* por dia, envia-se 5 mensagens por hora logo 1 mensagem a cada 12 minutos.

## 2.4 Web Server

Por forma a realizar pedidos à base de dados, quer de leitura quer de escrita, é utilizado um *Web Server* que recolhe a informação do utilizador recebida pela aplicação através do método *GET* e realiza as operações respectivas na base de dados. No caso de ser realizado um pedido de informação, esta é colocado num *array* que é retornado à aplicação em *JSON*.

# 3 Detalhes técnicos

Para a implementação da solução descrita anteriormente, para os diferentes módulos recorreu-se às seguintes tecnologias:

Para a aplicação *Android*:

- **MainAcitivity, CreateLogActivity, NewAlarmActivity e AddDeviceActivity** - Nestes módulos, realizam-se pedidos a um *Web Server* relativamente aos dados de um dado utilizador. Assim, na *MainActivity*, com base nos dados recolhidos da base de dados, é escrito um ficheiro, por forma a efectuar/consultar registos (utilizador, dispositivo ou alarme) noutras actividades. Nas restantes, o objectivo de contacto com o servidor *web* é o de acrescentar informação à base de dados, sendo também o ficheiro interno actualizado.

  Para este efeito, utiliza-se a Classe *FILE* presente em *Android* para o armazenamento interno (*cache*) e por cada pedido ao servidor, é necessária a abertura de *threads* por forma a não bloquear a *User Interface* (UI), recorrendo à classe *AsyncTask*;

- **Logs** - Neste módulo, será essencial a abertura de uma *thread*, na medida em que a UI principal não permite efectuar pedidos periódicos pelo facto de estes a poderem bloquear. Assim, caso o utiizador queira realizar pedidos periódicos, esta *thread* será aberta (recurrendo à classe *Thread*). Caso contrário, será apenas necessário utilizar novamente a classe *AsyncTask* para os pedidos, que apenas serão realzados mediante a pressão de um botão na aplicação.

  Para efectuar os pedidos HTTPS, utiliza-se a classe *HttpURLConnection*. O formato destes pedidos seguem as normas descritas na REST API-Students fornecida pelo corpo docente.

  Ao receber a resposta, a *thread* é processada com *parsing* de *JSON* com recurso à classe *JsonReader*. Após o processamento da resposta, é verificado se a temperatura recebida ultrapassa algum *threshold* definido pelo utilizador. Em caso afirmativo, é gerada uma notificação escrita e sonora com recurso a um objecto *Notification* e *Ringtone*.

Para a implementação da base de dados para armazenar os registos de cada utilizador, foi seguido o seguinte modelo:

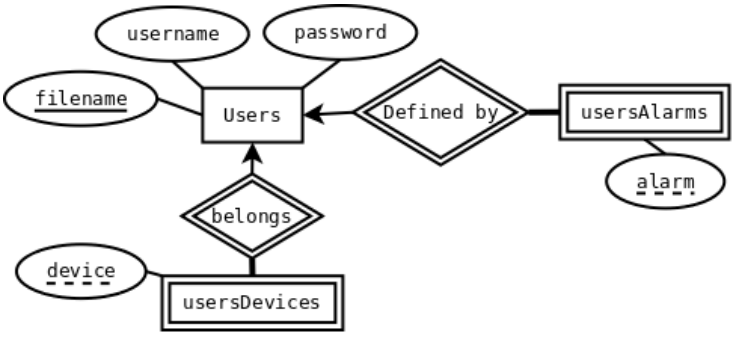


Figura 4: Arquitectura da base de dados

Como tal, a base de dados irá ter três tabelas, tal como se pode verificar no *script*, colocado na figura 5, utilizado para a sua criação.

```
drop table if exists users;
drop table if exists usersDevices;
drop table if exists usersAlarms;

create table users(
    filename varchar(40),
    userName varchar(255),
    password varchar(255),
    primary key(filename));

create table usersDevices(
    filename varchar(40),
    device varchar(40),
    primary key(filename, device));

create table usersAlarms(
    filename varchar(40),
    alarm varchar(40),
    primary key(filename, alarm));
```

Figura 5: Script utilizado para a criação da base de dados

Relativamente ao alojamento da base de dados e do servidor *web*, recorreu-se ao servidor do IST *sigma.tecnico.ulisboa.pt*. O acesso à base de dados encontra-se protegido contra *SQL Injection* utilizando-se para tal *prepared statements with variable binding*, o tráfego no entanto não se encontra encriptado pois não se considerou uma ameaça de grande nível para este tipo de projecto.

Para a implementação do sensor de temperatura, recorreu-se às bibliotecas associadas ao dispositivo *Akeru 3.3* disponibilizadas pelo *Snootlab*. Para enviar mensagens periodicamente de 12 em 12 minutos utilizou-se a função *millis()* presente em ambiente *Arduino*.

# 4 Verificações da solução encontrada

Do planeamento atrás referido, foi já adiantado parte do trabalho referente à programação da aplicação, nomeadamente as actividades *Welcome Screen, New First User, Set Alarm e Add Device*. De seguida são apresentadas algumas capturas de ecrã destas actividades em versão *beta*.

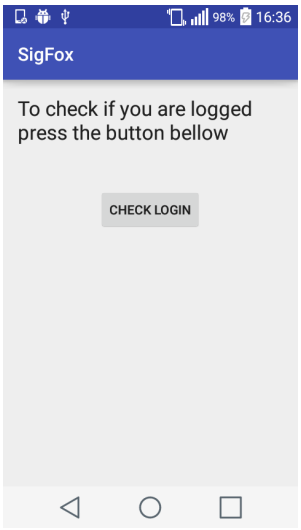


Figura 6: Welcome Screen

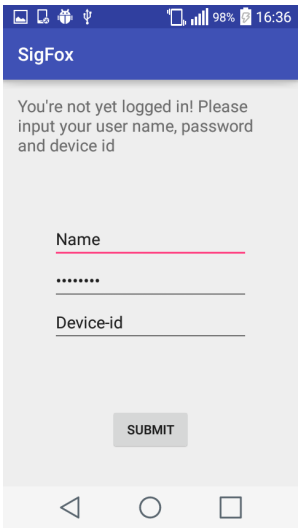


Figura 7: Registo de um novo utilizador

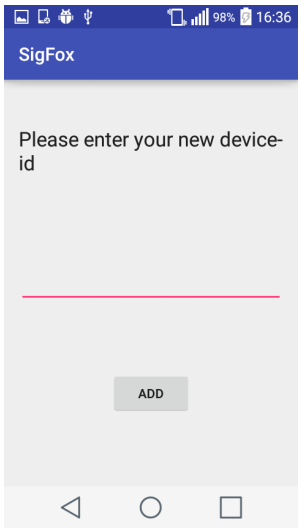


Figura 8: Registo de um novo dispositivo

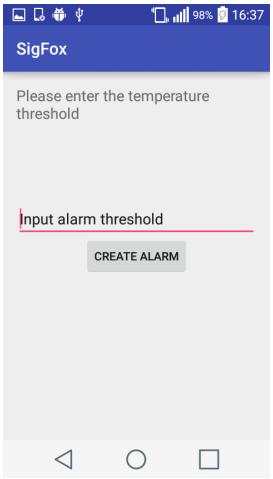


Figura 9: Registo de um novo alarme

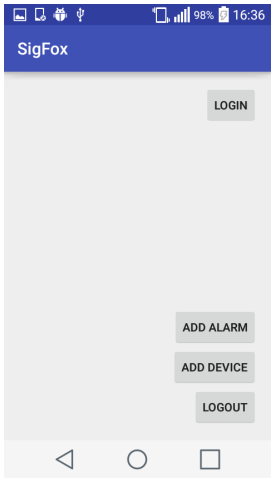


Figura 10: Actividade *Logs*

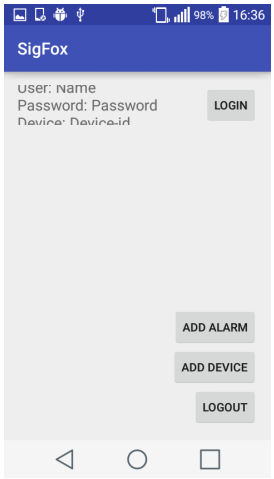


Figura 11: Actividade *Logs* com *login*

Relativamente à obtenção das leituras do sensor de arduíno a partir da *Cloud Sigfox*, foi realizado um programa à parte por forma a testar apenas esta funcionalidade, tal como representado nas figuras abaixo. O programa, além de recolher os dados, realiza o *parse* do *JSON*.

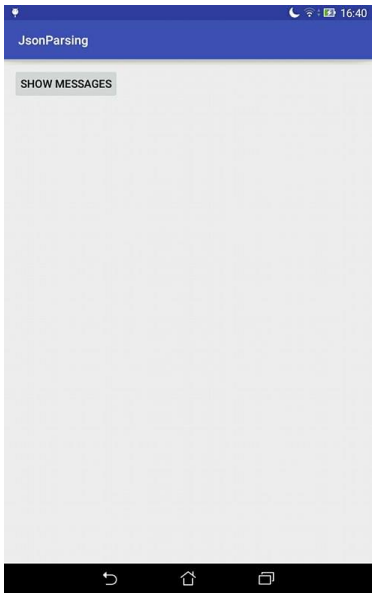


Figura 12: Botão para obtenção da informação da *Cloud*


Figura 13: Recolha da informação com *parse*

Será agora necessária a interligação dos dois programas e a activação das notificações, por forma a que a aplicação fique terminada.

Relativamente ao sensor, foi apenas realizado o trabalho de pesquisa, sendo iniciada a implementação quando a aplicação estiver concluída.

# 5 Alterações face ao planeamento inicial

# 6 Pontos críticos

# Referências

[1] [FILE16] `http://developer.android.com/reference/java/io/File.html`, Março 2016

[2] [THREAD16] `http://developer.android.com/reference/java/lang/Thread.html`, Março 2016

[3] [HTTPURLCONNECTION16]

`http://developer.android.com/reference/java/net/HttpURLConnection.html`, Março 2016

[4] [URLCONNECTION16]

`http://developer.android.com/reference/java/net/URLConnection.html`, Março 2016

[5] [JSON16] `http://developer.android.com/reference/android/util/JsonReader.html`, Março 2016

[6] [NOTIFICATION16]

`http://developer.android.com/training/notify-user/build-notification.html`, Março 2016

[7] [AKERU16] `https://github.com/Snootlab/Akeru`, Fevereiro 2016