# Byce

## Progetto di Internet Of Things di Alberto Morini (141986)

### **Anno accademico:** 2021/2022

\pagebreak

## Indice



## Byce, a battery logger

Il progetto si suddivide in due parti: Byce e il server.
Byce è un'app, progettata con l'obiettivo di rilevare il livello della batteria dei dispositivi Android e in seguito, inviare i dati al server.

L'applicazione può essere estesa anche al sistema operativo iOS, poiché è stata sviluppata attraverso Apache Cordova, quindi non si tratta di un'app nativa bensì ibrida.
Per l'esame è stata realizzata solamente la versione per Android.

Il server è rappresentato da qualsiasi personal computer in grado di eseguire il linguaggio NodeJS e ospitare un database MySQL.
L'obiettivo è quindi quello di rimanere in ascolto delle informazioni ricevute dai vari telefoni/tablet, memorizzare i dati nella base di dati e poi rappresentarli attraverso il software Grafana.

### Study case:

L'idea è nata da una necessità: un ristorante che utilizza dei tablet per ordinare il cibo e sfogliare il menù; alla fine del servizio un cameriere deve controllare tutti i tablet per verificare quale di questi debba essere ricaricato.

Invece, con questa soluzione il cameriere può in pochi secondi ottenere una panoramica dello stato di tutti i devices connessi alla rete.

L'applicativo si può utilizzare anche per scopi personali, ad esempio ricevere una notifica quando il proprio device ha raggiunto la carica completa oppure se è sia stato scollegato dalla presa di corrente.

\newpage

## L'app byce

### Struttura generale

L'applicazione deve ottenere la percentuale di carica e comunicare con il server.

In seguito, è necessario che il dispositivo sia riconoscibile, in altre parole il server deve sapere chi ha mandato tali dati.

Per tale obiettivo vi sono due soluzioni: utilizzare il MAC address oppure sfruttare il nome del dispositivo (eg. "Alby's iPhone"/"Samsung S5"/"Tablet21"). Quest'ultima idea è la soluzione ottimale, poiché poiché non necessita l'implementazione di un registro associativo tra MAC address e un'ulteriore nome; inoltre molti dispositivi (se si pensa a quelli personali) hanno già un nome personalizzato e quindi riconoscibile dall'utente.

Il cambiamento dello stato di carica consiste nella variazione della percentuale o nella variazione dell'alimentazione da corrente (collegato/scollegato). Ad ogni cambiamento, l'applicazione scatena un evento il quale sarà ascoltato da un'apposita funzione che si occuperà quindi di rilevare i dati precedentemente elencati, calcolare un timestamp e in seguito inviare una POST request al server in ascolto.

In fine, l'app deve essere in grado di continuare la sua esecuzione anche in background.

## Cordova Apache

Cordova è un framework Javascript sviluppato da Nitobi (acquisita poi da Apache).
Per eseguire Cordova è fondamentale aver già installato NodeJS e NPM.

Quindi `$ sudo npm install -g cordova` (viene utilizzato il comando "sudo" poiché alcuni sistemi richiedono i requisiti amministratore).

Al fine di implementare tutte le funzionalità richieste occorre installare dei plugin aggiuntivi:

- cordova-plugin-battery-status -> si interfaccia con la batteria del device
- cordova-plugin-device-name -> ottiene il nome del dispositivo
- cordova-plugin-background-mode -> permette l'esecuzione in background
- cordova-plugin-advanced-http -> consente l'invio di POST request via HTTP

### Il motore

Il cuore dell'applicazione è rappresentato principalmente da 3 file:

- www/index.html -> l'interfaccia grafica
- www/js/index.js -> le funzioni che vogliono eseguire
- config.xml -> file di configurazione generica, quindi il nome dell'app e altre informazioni

all'interno del file Javascript, è necessario dichiarare l'ascoltatore

```
document.addEventListener('deviceready', onDeviceReady, false);
```

il quale andrà a chiamare la funzione definita (onDeviceReady in questo caso) non appena l'applicazione sarà in esecuzione.

## Android

Per la piattaforma Android si richiede di installare alcuni tool di sviluppo, i quali:

- Android studio `$ sudo snap install android-studio --classic`
- Gradle `$ sudo apt-get install gradle`
- Un device virtuale (il quale deve essere acceso), installabile tramite Android Studio

Eseguendo il comando `$ cordova requirements` si otterrà una lista dei requisiti e se questi siano soddisfatti o meno (molto probabilmente servirà una specifica versione di Android SDK ottenibile attraverso Android Studio).

Successivamente bisogna configurare Android SDK nel proprio terminale, quindi modificando il file ".bashrc" aggiungendo:

```
export ANDROID_SDK_ROOT=$HOME/Android/Sdk
export PATH=$PATH:$ANDROID_SDK_ROOT/tools/bin
export PATH=$PATH:$ANDROID_SDK_ROOT/platform-tools
export PATH=$PATH:$ANDROID_SDK_ROOT/emulator
export ANDROID_HOME=$Home/Android/
```

in seguito, si deve ricaricare il file di preferenze attraverso `$ source .bashrc`

Per costruire l'APK occorre aggiungere la piattaforma Android al progetto `$ cordova platform add android` e poi `$ cordova build` si occuperà di realizzare il pacchetto di installazione per ogni piattaforma aggiunta.

Vi è bisogno di apportare alcune modifiche al file AndroidManifest.xml dove si dichiara il comportamento e i permessi richiesti dall'app.

> E' stato incluso il file generato alla creazione dell'APK utilizzata, solitamente il file si colloca in "*byce/platforms/android/app/src/*" dopo la costruzione dell'applicativo

Una volta imporata l'APK nel dispositivo Android, è richiesto di abilitare il permesso di installare applicazioni da fonti sconosciute, poiché l'APK non è firmata.

## Il pacchetto

Il pacchetto inviato è di Content/Type: JSON.
esempio di pacchetto/JSON

```
{
    "batteryLevel": 89,
    "inCharge": true,
    "name": "AlbysAndroid",
    "date": "2022/03/09",
    "time": "23:01:46"
}
```

JSON rappresenta un enorme vantaggio, poiché i dati vengono manipolati attraverso un 'dizionario' Javascript, linguaggio sul quale si basa sia l'applicazione che il server (NodeJS).

Utilizziamo `cordova.plugin.http.setDataSerializer('json');` per sfruttare tale codifica nei messaggi HTTP.

\newpage

# Il server

## struttura generale

Il server si avvia attraverso il comando `node server/server.js` quindi istanzia una socket con indirizzo IP 10.0.0.3 e porta 8124.

Il processo alla ricezione di un messaggio estrapolerà le informazioni contenute, le serializzerà in un file con formato CSV (utile per il debug) e in seguito, instaurerà una connessione con il database MySQL dove processerà una query di inserimento.

Il server non è tenuto a rispondere ai client, anche perché i client non sono stati programmati per elaborare la ricezione di richieste. Tuttavia si invia comunque un messaggio di conferma di ricezione (utile nel debug e in caso per sviluppi futuri).

## NodeJS

Node consente attraverso l'engine V8 di Chromium di eseguire script javascript al di fuori di un browser.
L'installazione può avvenire attraverso il gestore di pacchetti di Linux, quindi `$ sudo apt-get instal nodejs`

Il programma avvierà un server http in ascolto sulla porta `8124` all'indirizzo IP del computer sul quale avverrà l'esecuzione (per comodità si è ricorso all'utilizzo dell'IP statico `10.0.0.3` ).

Per implementare i vari obiettivi, è necessario importare alcuni moduli (librerie), installabili attraverso NPM (Node Package Manager) `$ npm install pacchetto`

Una volta aggiunti i pacchetti richiesti, bisogna includerli nel file server.js

```
var http = require('http'); //module for the creation of the server
var fs = require("fs"); //store a CSV with the messages recived, useful for debug
var mysql = require('mysql'); //module to connect with MySQL
```

verrà utilizzata la funzione `JSON.parse(pacchetto)` per convertire il messaggio JSON ricevuto in un 'dizionario' javascript.

## Il database

Il database è stato realizzato attraverso il sistema di gestione MySQL di Oracle.

Per l'installazione basta eseguire `$ sudo apt install mysql-server`

Successivamente si effettua il login `$ mysql -h localhost -P 3306 -u root -p` e si cambia la password default dell'utente root (è possibile anche aggiungere un nuovo user nel caso lo si desideri)

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'YourPsw'
```

La creazione del database avviene tramite le seguenti query:

```
CREATE DATABASE byce;
CONNECT byce;
--Si crei la tabella 'Dataset' dove verranno inseriti i dati
create table Dataset(
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    battery INTEGER NOT NULL,
    incharge BOOLEAN NOT NULL,
    name VARCHAR(256) NOT NULL,
    data DATE NOT NULL,
    tempo TIME NOT NULL
);
```

L'inserimento dei dati nella base di dati avviene attraverso la parte NodeJS del server

```
let jsonPack = JSON.parse(pacchetto);
....
var con = mysql.createConnection({...});

con.connect(function(err) {
    ....
    var sql = "INSERT INTO Dataset (battery, incharge, name, data, tempo) VALUES (" + jsonPack.batteryLevel+"

    //execute the query
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("1 record inserted");
```

```
    });
});
```

## Grafana

Grafana è un software che consente di estrapolare informazioni da un database e rappresentarli attraverso grafici interattivi e vari indicatori.

L'installazione (sistemi Debian) avviene eseguendo i seguenti comandi nel terminale:

```
sudo apt-get install -y adduser libfontconfig1
wget https://dl.grafana.com/enterprise/release/grafana-enterprise_8.4.3_amd64.deb
sudo dpkg -i grafana-enterprise_8.4.3_amd64.deb
```

Successivamente, si avvia il processo Grafana tramite `$ sudo systemctl restart grafana-server` il quale sarà disponibile presso la porta `3000`.

Una volta effettuato l'accesso e cambiata la password di default, si dovrà aggiungere il plugin MySQL per consentire a Grafana di interrogare la base di dati.
A configurazione terminata, si può creare una Dashboard dedicata dove si potrà aggiungere vari "pannelli" i quali forniranno una rappresentazione grafica della query che si desidererà inserire.

# Conclusioni

## Problematiche

Dalla versione 9 di Android, il sistema operativo interrompe totalmente l'esecuzione di un'app in background da più di 5 minuti circa.

Per ovviare questo probelma è stata implementata una scappatoia, costituita dal portare il processo in "foreground" e nuovamente in "background".

Purtroppo questa soluzione vede il display del dispositivo accendersi (poiché si porta in primo piano l'app) ogni 5 minuti.

La scelta di impostare un timer ogni 5 minuti è stata voluta solamente per ottenere una monitorazione continua dei devices; nell'adozione del sistema in un mondo reale il timer può essere tranquillamente impostato a un determinato delta tempo

```
setInterval(()=>{
    cordova.plugins.backgroundMode.unlock(); //is like moveToForeground, but works even if the phone is locke
    cordova.plugins.backgroundMode.moveToBackground();
}, 240000); //4min
```

## Sviluppi futuri

1. Estendere l'applicativo anche alla piattaforma iOS
2. Consentire al client di specificare l'indirizzo IP del server, in modo da permettere una configurazione semplificata del sistema.
3. Rilevare ulteriori risorse come utilizzo della CPU e RAM, memoria utilizzata etc.

## Tecnologie utilizzate

- Lista device:

    - Samsung Galaxy S5 con Android 9 (AlbyAndroid nel database)
    - Qualcosa con Android 10 (SS88)

- Ubuntu 21.10, come server.

    - Java JDK 1.8.0
    - NodeJS v16.14.0
    - NPM v8.5.2

- Rete privata con indirizzi di classe A ( `10.0.0.0/24` )