Rapport de stage

Stage effectué du 7 Aout au 7 Septembre par :

-Michel-Ange DAGRAIN

-Carla M. ST JUSTE

-Sandy THELEMAQUE

2012

Carla

Hewlett-Packard

17-Sep-12



We thank the Women’s Peer to Peer Network for allowing us to conduct this course. A special thanks to our master workshop **Ms. Tayana Etienne** for her support and guidance.

We thank **Vania** **Charles** and **Marguerite ST LOUIS** who kindly send us during these four weeks their skills and knowledge.

Finally, we thank the staff of Solutions SA we have warmly welcomed.

**SOMMAIRE**

Introduction……………………………………………………………………………………………………………………………………3

Presentation de “women peer to peer”……………………………………………………………………..……………….4

Présentation du programme de stage…………………………………………………………………………………………4

Les différentes étapes a la réalisation du projet…………………………………………………………………..5

Conclusion…………………………………………………………………………………………………………………………………………22



The internship is an essential element to our training. This is a contact that allows the student to experience socio-economic enterprises.

Housed Solutions SA has 7 August to 7 September, this internship Women Peer to Peer network has allowed us to test our ability to analyze and synthesize and make a comparison between theory and practice. Thanks to this training we were able to test and improve our knowledge, to see closely the progress of computer work, live professional reality, to face the practical problems encountered.

Women peer to peer network is an organization whose goal is first to encourage women in technology, we had a mission to develop a prototype application for recording telephone numbers of members FONKOZE.

To effectively expose these 4 weeks of internship we present in the following pages women peer to peer network which is the organization for which the training was conducted and the project on which we were brought to work. We then outline the stages in the project and difficulties and finally we will summarize the work done and we share our impressions of the course.

**Presentation of women peer to peer**

Women "peer to peer" is a program run by Rachel Levental whose main objective is the design and implementation of a communication network for Haitian women in the trade. To establish this network of Haitian women in technology will be used.

**Presentation of the training program**

The objective of the course is to develop and implement a prototype application for recording telephone numbers of members FONKOZE. This work should be carried out in four weeks from August 6 to September 7, 2012 following a work schedule from 5 per day Monday to Friday.

* Planning
  1. R th e first week:

During this week we had to do research on the tools that will be used to perform the work. At the end of the week a report on the research should be delivered.

* 1. M th e second week:

-Visit to FONKOZE

Modelling of the database

-Choice and learning technologies used.

* 1. M th e third week:

-Development of prototype

* 1. M th e fourth week:

-Prototype development and training.

**Different stages in the project:**

1. **Research and Experiences**

Before starting to implement the application we started doing research. During the internship at the end of the first week there was a report highlighting the main points of our research has been conducted primarily Frontline SMS, Google Fusion table, Python ... As part of the project the Research on Frontline SMS were very useful since we used this application for the collection of SMS, beyond the interest in this report to share our experiences with this platform.

* + 1. Experiences with Frontline SMS

On the site of Frontline SMS two versions of the application were available: version and version 1.6.16.3 ...... We downloaded and tested both. For ease of use of the first version compared to the second we opted for the first version seemed more and more complete. As we have used modem that distributed by NATCOM the E173Eu-1. We tested the sending and receiving sms through Frontline SMS to an individual and a group of people, experienced the option of defining keywords and creating external control provided by the application. We also experimented with exports and imports data Frontline SMS. Indeed, it was observed that could export all SMS messages and save in the database Frontline SMS in an Excel table. We tested the connection with a base Frontline SMS of H2 and also a database MySQL.

* + 1. Conn. ex Frontline ion with a MySQL database

Our first Frontline SMS connections with MySQL databases that were created just for testing were generated abnormalities in the functioning of Frontline SMS. Indeed, having established the connection of the latter with any MySQL database, 21 tables were automatically created in the database in question and the application is also closed automatically. It is restart the application problems have arisen, sometimes the application does not restart, sometimes it tries to restart but eventually generates errors. This made Frontline SMS unusable for a few days because the problem persisted even after uninstalling and reinstalling the application. This is what prompted us to connect to a database Frontline SMS of H2 then establish a connection between the database and the MySQL database to be used by our application to store new numbers. However after installing SQLyog Enterprise establish a direct connection between Frontline SMS and database MySQL seems to be fine so we have another version of the application only where the MySQL database is used as the data that Frontline also wishes to address are stored on this basis.

1. **Meeting in** **FONKOZE**

**FONKOZE is a non-profit organization, comprised of 60,000 members and 250,000 savings accounts, whose main mission is to help poor Haitian women improve their lifestyle through a credit and training to enable them to better manage their business.**

**A FONKOZE uses a very reliable banking software "eMerge" and several smaller databases from MS Access and MS Excel tables, each containing different information.** **No systematic technique for updating the various databases is still operational, among other they would interconnect these small databases between them.**

* + **Needs of managers FONKOZE**

**Before, when recording information on a new member of FONKOZE in a database, the phone number was not given mandatory now that everyone, regardless of economic status, either able to own a phone, officials thought it would be interesting to have this information in order to contact a member directly without going through the head of the center.** **To address this need is to say get the phone number of a member FONKOZE officials have appealed to technology.** **Officials would have liked any mobile phone a member can send an SMS to update their phone number in the database.** **Beyond a SMS platform, namely Frontline SMS is used to collect the SMS received and an application will be implemented to perform processing on SMS, and save the received telephone number in a database MySQL created mainly for this purpose .** **Officials stressed that the implemented application should have its own database that they can then connect with other existing databases.**

1. **Implementation of application**

To start we will present the format that the client should use to send FONKOZE SMS its new mobile phone number. Basically, it has the following structure: Keyword [space] IDmembre [space] number. The key word used is "fk" for F K ON ZOE. All variations are taken into account by the application, that is to say, "CF", "fK" and "Fk" are also eligible.

All members of FONKOZE has a unique identifier that identifies them. This number has the format 711 figure 6 other 01 will follow the keyword and precede the new number.

As mentioned earlier due to problems with the connection Frontline SMS with a base of MySQL we opted for a direct connection to a database H2. We used Notepad + + and phpMyAdmin for the prototype.

* Présentation des codes source

import java.sql.DriverManager;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.awt.event.\*;

import java.io.\*;

class ImportDriver{

private String link="";

private String username="";

private String passwd="";

private Connection connect;

private Statement statement;

public ImportDriver(String link, String username, String passwd){

this.link=link;

this.username=username;

this.passwd=passwd;

}

public boolean verifyConnection(){

try{

Class.forName("org.h2.Driver").newInstance();

this.connect=DriverManager.getConnection("jdbc:h2:file:"+this.link, this.username, this.passwd);

this.statement=this.connect.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE, ResultSet.CONCUR\_UPDATABLE);

return true;

}

catch(SQLException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

catch(ClassNotFoundException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

catch(InstantiationException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

catch(IllegalAccessException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

return false;

}

public ResultSet executer(String chaineSQL){

try{

ResultSet rs=this.statement.executeQuery(chaineSQL);

return rs;

}catch(SQLException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

return null;

}

public void close(){

try{

this.statement.close();

this.connect.close();

}catch(SQLException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

}

class ImportDriver1{

private String link="";

private String username="";

private String passwd="";

private Connection connect;

private Statement statement;

public ImportDriver1(String link, String username, String passwd){

this.link=link;

this.username=username;

this.passwd=passwd;}

public boolean verifyConnection(){

try{

Class.forName("com.mysql.jdbc.Driver").newInstance();

this.connect=DriverManager.getConnection("jdbc:mysql:"+this.link,this.username, this.passwd)

this.statement=this.connect.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE, ResultSet.CONCUR\_UPDATABLE);

return true;

}catch(SQLException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

catch(ClassNotFoundException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

catch(InstantiationException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

catch(IllegalAccessException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

return false;

}

public ResultSet executer(String chaineSQL){

try{

ResultSet rs=this.statement.executeQuery(chaineSQL);

return rs;

}catch(SQLException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);

}

return null;

}

public void close(){

try{

this.statement.close();

this.connect.close();

}catch(SQLException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public class FontlineSms2{

public static ResultSet rs,result;

private static String resourcePath;

public static ImportDriver1 ImD1;

public static ImportDriver ImD;

public static void main(String args[]){

System.out.println(getConfigDirectoryPath());

loadFonkozeBase();

loadSmsBase();

}

public static void loadSmsBase(){

ImD=new ImportDriver(getConfigDirectoryPath(), "", "");

if(ImD.verifyConnection()){

System.out.println("Connected to database fontlineSms");

try{

rs=ImD.executer("SELECT \* FROM message");

if(rs !=null){

while(rs.next()){

System.out.println("id: "+rs.getInt("id"));

System.out.println("Date: "+rs.getInt("Date"));

System.out.println("Expediteur: "+rs.getString("senderMsisdn"));

System.out.println("Message: "+rs.getString("textContent")+"\n\n");

String chaine=rs.getString("textContent");

String chaineTab[]=chaine.split(" ");

if (chaineTab.length!=0){

for(int i = 0; i<chaineTab.length; i++){

System.out.println("chaine("+i+")="+chaineTab[i]);

}

if(result !=null){

while(result.next()){

if (result.getString("numero\_membre")==chaineTab[0]){

result.moveToInsertRow();

result.updateString("telephone",chaineTab[1]);

result.insertRow();

break;

}

}

}

}

else{ System.out.println("the database is empty");}

}

}

else{

System.out.println("the database is empty");

}

}catch(SQLException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);

}

}else{

System.out.println("Connection to message failed !!!");

}

}

public static void loadFonkozeBase(){

ImD1 = new ImportDriver1("//localhost/fonkoze","root","");

if(ImD1.verifyConnection()){

try{

result=ImD1.executer("SELECT \* FROM membre");

if(result !=null){

while(result.next()){

System.out.println("id: "+result.getInt("id"));

System.out.println("NumeroMembre: "+result.getString("numero\_membre"));

System.out.println("Nom: "+result.getString("nom"));

System.out.println("Prenom: "+result.getString("prenom"));

System.out.println("Centre: "+result.getString("prenom"));

System.out.println("Telephone: "+result.getString("telephone"));

System.out.println("Adresse: "+result.getString("adresse")+"\n");

}

}

else{

System.out.println("the database is empty")}

}catch(SQLException ex){

Logger.getLogger(ImportDriver.class.getName()).log(Level.SEVERE, null, ex);}

else{

System.out.println("Connection failed !!!"); }

}

}

public static String getUserHome() {

return System.getProperty("user.home")+File.separatorChar+"AppData"+File.separatorChar+"Roaming"+File.separatorChar+"Microsoft"+File.separatorChar+"Windows"+File.separatorChar+"FrontlineSMS" + File.separatorChar;

}

public synchronized static String getConfigDirectoryPath() {

if(resourcePath == null) {

resourcePath = getUserHome();

}

if(resourcePath.charAt(resourcePath.length()-1) != File.separatorChar) {

resourcePath += File.separatorChar;

}

return resourcePath + "frontlinesms\_h2\_db";

}

}

Once the issue of Frontline SMS connection with a database MySQL has been resolved, we opted to use a single database instead of two. One of the tables generated by Frontline SMS (table message) in the database MySQL creates smsapp be used by the latter to store the received SMS. For this we must ensure that at all times the application that is to say, Frontline SMS is in "running", it recognizes the modem to be used for receiving SMS and that is permanently connected with the base smsapp.

When SMS is received by Frontline SMS, its contents are saved in the table field textContent message. In the same table is found in the date field, which contains the date on which the SMS has been delivered. The application we have implemented will the contents of SMS received will test it to see if it is valid to say that the keyword, membership number and format of the phone number are correct . If the received message is invalid the application ignores the message, otherwise it will check if this issue had not already been registered in the database. If the number was active it ignores the client's request who wanted to make an update, if the number was not in the database but the client had done in the past to store a number, the application number placed in the former another table (archive table), delete the old table updatenum which is the table that contains the active phone numbers and adds the new number. A presentation of the source code of the application will give a clearer idea of how it goes about updating phone numbers. Outside the main class we have two other classes which are the class and the class MySQLcli refresh. In the following lines further details will be given on the functions of its classes.

* Présentation de la classe MySQLCli

Comme IDE nous avons utilisé NetBeans 7.1.2 et comme SGBD nous avons utilisé SQLyog Enterprise.

package repertoire\_telephonique;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

public class MySQLCli {

private String dbURL = "";

private String user = "";

private String password = "";

private java.sql.Connection dbConnect = null;

private java.sql.Statement dbStatement = null;

private int countrow;

//setter

public void setlastrow (int lr)

{this.countrow=lr;}

/\*\*

\* Constructeur

\* @param url

\* @param user

\* @param password

\*/

public MySQLCli(String url, String user, String password)

{

this.dbURL = url;

this.user = user;

this.password = password;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Méthodes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* Connecter à la base de données

\* @return false en cas d'échec

\*/

public Boolean connect()

{

try {

Class.forName("com.mysql.jdbc.Driver").newInstance(); this.dbConnect=DriverManager.getConnection("jdbc:mysql:"+this.dbURL,this.user,this.password)

this.dbStatement = this.dbConnect.createStatement();

return true

}

catch(SQLException ex)

{ Logger.getLogger(MySQLCli.class.getName()).log(Level.SEVERE,null,ex);}

catch(ClassNotFoundException ex)

{ Logger.getLogger(MySQLCli.class.getName()).log(Level.SEVERE, null, ex);}

catch(InstantiationException ex)

{ Logger.getLogger(MySQLCli.class.getName()).log(Level.SEVERE, null, ex);}

catch(IllegalAccessException ex)

{ Logger.getLogger(MySQLCli.class.getName()).log(Level.SEVERE, null, ex);}

return false;

}

/\*\*

\* Executer une requete SQL

\* @param sql

\* @return resultat de la requete

\*/

public ResultSet exec(String sql)

{

try {

ResultSet rs = this.dbStatement.executeQuery(sql);

return rs;

}

catch (SQLException ex)

{ Logger.getLogger(MySQLCli.class.getName()).log(Level.SEVERE, null, ex);}

return null;

}

/\*\*

\* Executer une requête SQL

\* @param sql

\* @return le numero de la ligne

\*/

public int exec1(String sql)

{

try {

int rowcount = this.dbStatement.executeUpdate(sql);

return rowcount

}

catch(SQLException ex)

{Logger.getLogger(MySQLCli.class.getName()).log(Level.SEVERE, null, ex);}

return 0;

}

/\*\*

\* Fermer la connexion au serveur de donnees

\*/

public void close()

{

try {

this.dbStatement.close();

this.dbConnect.close();

}

catch (SQLException ex)

{Logger.getLogger(MySQLCli.class.getName()).log(Level.SEVERE, null, ex);

}

}

This class contains four methods, two of which are used to connect and disconnect to a respective base MySQL and two for the execution of SQL queries.

* Présentation de la classe refresh

package repertoire\_telephonique;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Date;

public class Refresh {

private MySQLCli M;

public Refresh (MySQLCli m)

{

this.M=m;

}

public int CheckAdd() throws SQLException

{

int NbreAjout=0;

ResultSet R=M.exec("SELECT value FROM variable WHERE ID=1");

if(R!=null && R.absolute(1))

{

int x=R.getInt("Value");

ResultSet RS=M.exec("SELECT COUNT(\*)FROM message");

If (RS!=null && RS.absolute(1))

{

int y=RS.getInt("COUNT(\*)");

if(x<y)

{

NbreAjout=y-x;

}

}

}

return NbreAjout;

}

public void treatment () throws SQLException

{

ResultSet R=M.exec("SELECT value FROM variable WHERE ID=1");

if(R!=null && R.absolute(1))

{

int lastposition=R.getInt("Value");

ResultSet Re=M.exec("SELECT \* FROM message");

if (Re!=null && Re.absolute(lastposition))

{

while (Re.next())

{

String textcontent=Re.getString("textContent");

String[] tabsplitmessage=textcontent.split(" ");

if (tabsplitmessage.length==3)

{

boolean keyvalid=Checkkey(tabsplitmessage[0]);

boolean IDvalid=CheckID(tabsplitmessage[1]);

boolean numvalid=Checknum(tabsplitmessage[2]);

if(keyvalid==true && IDvalid==true && numvalid==true)

{

String ID=tabsplitmessage[1];

String num=tabsplitmessage[2];

Date date=Re.getDate("date");

ResultSet Res= M.exec("SELECT IDmember FROM updatetab WHERE IDmember=tabsplitmessage[1]");

if(Res==null)

{

int countrow = M.exec1("INSERT INTO updatetab (IDmember, NewNumber,Dateupdate) VALUES('" + ID + "','" + num+ "','" + date + "'");

lastposition++;

}

if (Res! =null &&! Res.getString ("NewNumber").equals (tabsplitmessage [2]))

{

String IDMember=Res.getString ("IDmember");

String OldNumber=Res.getString("NewNumber");

Date DateEnable=Res.getDate("Dateupdate");

Date Date Disable=date;

int countrowar=M.exec1("INSERT INTO archive (IDMember,OldNumber,DateEnable,DateDisable)VALUE ('"+IDMember+"','"+OldNumber+"','"+DateEnable+"','"+DateDisable+"')");

int countrowup=M.exec1("DELETE FROM updatetab WHERE IDmember=IDMember"); int countrow = M.exec1("INSERT INTO updatetab (IDmember, NewNumber, Dateupdate) VALUES('" + ID + "','" + num+ "','" + date + "'");

lastposition++;

}

if (Res! =null && Res.getString("NewNumber").equals(tabsplitmessage[2]))

{

lastposition++;

}

} else {lastposition++ ;}

} else {lastposition++ ;}

}

int up=M.exec1 ("UPDATE variable SET value=lastposition WHERE ID=1");

}

}

}

/\*\*

\* Check if the keyword is valid

\* @param s

\* @return state of the keyword

\*/

public static boolean Checkkey (String s)

{

if ("fk".equals(s)||"Fk".equals(s)||"FK".equals(s)||"fK".equals(s))

{

return true;

}

else

{

return false;

}

}

/\*\*

\* Check if the IDmember is correct

\* @param s

\* @return state of the IDMember

\*/

public static boolean CheckID (String s)

{ String [] tab=s.split ("");

if(tab.length==12 && "7".equals(tab[1]) && "0".equals(tab[2]) && "5".equals(tab[3]) &&"0".equals(tab[10])&& "1".equals(tab[11]))

{ System.out.println("ID correct");

return true;

}

else

{ System.out.println("ID incorrect");

return false;

}

}

/\*\*

\* Check if the number's format is correct

\* @param s

\* @return state of the number's format

\*/

public static boolean Checknum(String s)

{

String [] tab= s.split ("");

if (tab.length==9)

{

System.out.println("Numero correct");

return true;

}

else

{

System.out.println("Numero incorrect");

return false;

}

}

}

The role of this class is to perform the processing on received messages and make updates. It contains five (5) methods:

- CheckAdd () to check if there are updates that is to say, check if there is no SMS sent by FONKOZE customers who have not yet been processed. This method returns an "int" is the number of messages that have been added to the database by Frontline SMS since the last update. A table is used to store the ID of the last row in the table come message, note that the ID of the last row of the table corresponds to the line number that contains the table.

- Checkkey (String s) tested the validity of the keyword is correct. The keyword must be used is "fk" and all formats are allowed.

- CkeckID (String s) tested the validity of the membership number.

- Checknum (String s) tests if the number sent is valid.

- Treatment () performs processing on the received SMS. It divides the message into three parts: the keyword, the identifier and number. It uses other methods to ensure that they are valid. If the message is valid, we check if the member in question had not already pre-registered number in the database. Beyond that, several cases may present themselves:

-The new issue was already in the database: In this case we do not know the client's request.

-The client already had a number in the database but that is different from the one he wants to add: In this case, the old number will be added to the archive table and deleted from the table updatetab and the new number will be inserted into the updatetab table.

it had no customer number in the database: In this case we insert the new customer number in the table updatetab.

      Each time a number is stored in the database when the update has been made will be recorded.

* La classe principale

package repertoire\_telephonique;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Repertoire\_telephonique {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

MySQLCli mysqlCli = new MySQLCli("//localhost/smsapp", "root", "");

if (mysqlCli.connect())

{

System.out.println("Connected to database smsapp");

Refresh R=new Refresh(mysqlCli);

try {

int NumberAdd=R.CheckAdd();

if(NumberAdd!=0)

{

R.treatment();

}

} catch (SQLException ex) { Logger.getLogger(Repertoire\_telephonique.class.getName()).log(Level.SEVERE, null, ex);

}

}

Ce stage fut pour nous une expérience très positives tant au niveau professionnel que relationnel.

Durant ces quatre semaines on a pu observer le fonctionnement de service d’une entreprise.

Le stage nous a donné gout à la programmation et nous a permis de nous évaluer.