Troisième partie

Annexes

A

Codes sources - Classes JAVA

Listing A.1 – DataSource

```
package CoucheSonde;

import java.sql.*;
import java.sql.Connection;

import ca.edbc.jdbc.*;
import ca.ingres.jdbc.*;
import com.mysql.jdbc.*;
import oracle.jdbc.driver.*;

/**
 * classe JavaBean permettre la gestion des parametres via une application WEB
 *
 */
public class DataSource {

  public DataSource(String url, String user, String password, String name,
      String driverClasse) {
    super();
    this.url = url;
    this.user = user;
    this.password = password;
    this.name = name;
    this.driverClasse = driverClasse;
  }

  public DataSource() {
    super();
  }

  private String url;

  public String getUrl() {
    return this.url;
  }
```

```
private String user;

public String getUser() {
  return this.user;
}

private String password;

public String getPassword() {
  return this.password;
}
private String name;



public String getName() {
  return name;
}

public void setName(String name) {
  this.name = name;
}

public String driverClasse;

public String getDriverClasse() {
  return this.driverClasse;
}

public void setUrl(String url) {
  this.url = url;
}

public void setUser(String user) {
  this.user = user;
}

public void setPassword(String password) {
  this.password = password;
}



public void setDriverClasse(String driverClasse) {
  this.driverClasse = driverClasse;
}

/**
 *   * retourne une instance de connection selon les attribus de cette classe
 */
public Connection getConnetcion() {
  Connection con = null;
  try {
    Class.forName(driverClasse).newInstance();
```

```
    } catch (ClassNotFoundException e1) {
      // TODO Auto-generated catch block
      e1.printStackTrace();
    } catch (InstantiationException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (IllegalAccessException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    try {

      con = DriverManager.getConnection(url, user, password);

    } catch (SQLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    return con;

  }
}
```

Listing A.2 – LogFileSource

```
package CoucheSonde;




public class LogFileSource
{

    private String path;
    public String getPath ()  {
        return this.path;
    }


    private String name;
    public String getName () {
        return this.name;
    }
  public void setPath(String path) {
    this.path = path;
  }
  public void setName(String name) {
    this.name = name;
  }
  public LogFileSource(String path, String name) {
    super();
    this.path = path;
    this.name = name;
```

```
  }

}
```

Listing A.3 – DBCollectorManager

```
package CoucheSonde;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;

import xml.XMLTools;


public class DBCollectorManager {







  private static volatile DBCollectorManager instance = null;

  private DBCollectorManager(){
    super();
      requests=new ArrayList<DBRequest>();
      dbsupporteds=new ArrayList<DBSupported>();
      datasources=new ArrayList<DataSource>();

  }
  public static synchronized DBCollectorManager getInstance() {

    if (instance == null)
      instance = new DBCollectorManager();

    return instance;
  }



  public static ArrayList<DBRequest> requests;
  public static ArrayList<DBSupported> dbsupporteds;
  public static ArrayList<DBRequest> getDBRequests() {
    return requests;
  }



  public static ArrayList<DataSource> datasources;


  /**
   * retourne la liste des DBRequests
```

```
 */
public   ArrayList<DBRequest> getAllDBRequests() {
  requests = (ArrayList<DBRequest>) XMLTools
      .decodeFromFile(DBCollector.CONFIGEFILE);

  return requests;

}

public static ArrayList<DBSupported> getDbsupporteds() {
  return dbsupporteds;
}

public static ArrayList<DataSource> getDatasources() {
  return datasources;
}
public static void setDatasources(ArrayList<DataSource> datasources) {
  DBCollectorManager.datasources = datasources;
}
/**
 * retourne la liste des DataSources contenant les parametre des bases de
 * donnees
 */
public   ArrayList<DataSource> getAllDataSources() {
  datasources = (ArrayList<DataSource>) XMLTools
      .decodeFromFile(DBCollector.DATASOURCEFILE);

  return datasources;

}

/**
 * Sauvegade une liste de datasource retourn 1 si la sauvegarde est bien
 * passe retourne 0 dans le cas contraire
 */
public    int saveAllDataSources() {
  try {
    XMLTools.encodeToFile(datasources, DBCollector.DATASOURCEFILE);
  } catch (Exception e) {
    return 0;
  }
  return 1;

}

/**
 * Sauvegade une liste de DBRequests, retourn 1 si la sauvegarde est bien
 * passe retourne 0 dans le cas contraire
 */
public   int saveAllDBRequests() {
  try {
    XMLTools.encodeToFile(requests, DBCollector.CONFIGEFILE);
  } catch (Exception e) {
    return 0;
```

```
    }
    return 1;

  }

  /**
   * retourne la liste des base de donnees supportees
   *
   * @return liste des DBSupported
   */
  public ArrayList<DBSupported> getAllDBSupporteds() {
    return (ArrayList<DBSupported>) XMLTools
        .decodeFromFile( DBCollector.DBSUPPORTEDFILE);

  }

  /**
   *
   * @param dbsupporteds
   *              ArrayList des DBSupported
   * @return un entier egal a 1 si la sauvegarde c'est bien passe egal 0 sinon
   */
  public int SaveAllDBSupported() {
    try {
      XMLTools.encodeToFile(dbsupporteds, DBCollector.DBSUPPORTEDFILE);
    } catch (Exception e) {
      return 0;
    }
    return 1;
  }

    public void addDBRequest(String name,String sqlRequest,String description,
     String dataSourceName)
    {
      DBRequest r=new DBRequest(name,sqlRequest,description,dataSourceName);
      requests.add(r);
    }


    public void addDataSource(String url, String user,String password ,
    String name,String driverClasse)
    {
      DataSource ds=new DataSource(url,user,password,name,driverClasse);
      datasources.add(ds);
    }


    public void addDBSupported()
    {

    }


    public void updateDBRequest(int index,String name,String sqlRequest,
```

```
    String description, String dataSourceName)
    {
      DBRequest r=new DBRequest(name,sqlRequest,description,dataSourceName);
      requests.set(index, r);

    }


    public  void updateDataSource(int index,String url, String user,
    String password ,String name,String driverClasse)
    {
      DataSource ds=new DataSource( url,user,password,name,driverClasse);
      datasources.set(index,ds);
    }


    public  void updateDBSupported()
    {
    }


    public  void deleteDBRequest(int index)
    {
      requests.remove(index);
    }


    public  void deleteDataSource(int index)
    {
      datasources.remove(index);
    }


    public   void deleteDBSupported()
    {
    }
}
```

Listing A.4 – LogColllectorManager

```
package CoucheSonde;

import java.util.ArrayList;

import org.w3c.dom.Document;
import org.w3c.dom.Node;

import xml.XMLTools;




public class LogColllectorManager
{
```

```
  private static  volatile  LogColllectorManager instance = null;

  private LogColllectorManager() {
    super();
      logrequests=new ArrayList<LogRequest>();
       logfilesources =new ArrayList<LogFileSource>();
  }
  public static synchronized LogColllectorManager getInstance() {

    if (instance == null)
      instance = new LogColllectorManager();

    return instance;
  }


  public static ArrayList<LogRequest> logrequests;
  public static ArrayList<LogFileSource> logfilesources;




    public ArrayList<LogFileSource> getAllLogFileSource()
    {
       logfilesources =(ArrayList<LogFileSource>) XMLTools
    .decodeFromFile(LogFileCollector.LOGSOURCEFILE);
      return logfilesources ;

    }


    public int saveAllLogFileSource()
    {
    try {
      XMLTools.encodeToFile(logfilesources, LogFileCollector.LOGSOURCEFILE);
    } catch (Exception e) {
      return 0;
    }
    return 1;
   }


    public ArrayList<LogRequest> getAllLogRequest()
    {
      logrequests=(ArrayList<LogRequest>) XMLTools
    .decodeFromFile(LogFileCollector.CONFIGEFILE);
//       for(int i=0;i<logrequests.size();i++){
//         Document doc=LogFileCollector.getInstance().Collect(
      logrequests.get(i).getName(), new ArrayList<Params>());
//         Node element=doc.getElementsByTagName("element").item(0);
//         String str="";
//         for(int j=0;j<element.getChildNodes().getLength();j++)
//           str+=element.getChildNodes().item(j).getNodeName()+",";
```

```
//          logrequests.get(i).setResultFields(str);
//          System.out.println(str);
//        }
      return logrequests;

    }


    public int saveAllLogRequest()
    {
    try {
      XMLTools.encodeToFile(logrequests, LogFileCollector.CONFIGEFILE);
    } catch (Exception e) {
      return 0;
    }
    return 1;
    }


    public void addLogFileSource(String path, String name)
    {
      LogFileSource lfs=new LogFileSource(path,name);
       logfilesources.add(lfs);
    }


    public void addLogRequest(String logFileSourceName, String name,
    String description, String startMotif, String endMotif)
    {
      LogRequest r =new LogRequest(logFileSourceName, name,description, startMotif,
      endMotif);
      logrequests.add(r);
    }


    public void updateLogFileSource(int index,String path, String name)
    {
      LogFileSource lfs=new LogFileSource(path,name);
       logfilesources.set(index, lfs);

    }
    public void deleteLogFileSource(int index)
    {

       logfilesources.remove(index);

    }


    public static ArrayList<LogRequest> getLogrequests() {
    return logrequests;
  }
  public static ArrayList<LogFileSource> getLogfilesources() {
    return logfilesources;
```

```
  }
  public void updateLogRequest(int index,String logFileSourceName,
  String name,String description, String startMotif, String endMotif)
    {
      LogRequest r =new LogRequest(logFileSourceName, name,description, startMotif,
      endMotif);
      r.setFields(logrequests.get(index).getFields());
      logrequests.set(index,r);


    }


    public void deleteLogRequest(int index)
    {
      logrequests.remove(index);
    }


    public void deleteField(int index,int idexfield)
    {
      logrequests.get(index).getFields().remove(idexfield);
    }
    public void addField(int index,String name, String motif, String delimetter)
    {
      Field f=new Field(name,motif,delimetter);
      logrequests.get(index).getFields().add(f);
    }
    public void updateField(int index,int indexfield,String name, String motif,
    String delimetter)
    {
      Field f=new Field(name,motif,delimetter);
      logrequests.get(index).getFields().set(indexfield,f);
    }

}
```

Listing A.5 – EICalculator

```
package CoucheIndicateurs;


import java.lang.reflect.Method;
import java.util.List;
import java.util.ArrayList;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import xml.XMLTools;

import CoucheSonde.CollectorFactory;
```

```
import CoucheSonde.DBRequest;
import CoucheSonde.ICollector;
import CoucheSonde.Params;


public class EICalculator
    implements ICalculator
{
  private static volatile EICalculator instance = null;

  private EICalculator() {
    super();

  }

  public static synchronized EICalculator getInstance() {

    if (instance == null)
      instance = new EICalculator();

    return instance;
  }
    public static ArrayList<EIndicator> eindicators =new ArrayList<EIndicator>();

    public List<EIndicator> eIndicator = new ArrayList<EIndicator> ();
    public List<EIndicator> getEIndicator () {
        return this.eIndicator;
    }

  public void Init () {
    eindicators = (ArrayList<EIndicator>) XMLTools
    .decodeFromFile(EIndicator.CONFIGUREFILE);

  }

  public Document calculate(String id) {
    EIndicator eindicator=null;
    for(int i=0;i<eindicators.size(); i++)
    {
      if(id.equals(eindicators.get(i).getId())){
        eindicator=eindicators.get(i);
        break;
      }
    }
    /** etape 1 : collecter les donnees*/
ICollector collector=CollectorFactory.getInstance().
    createCollector(eindicator.getRequestType());
    collector.Init();
    Document doc=null;


    for(int i=0;i<eindicator.getRequestNames().size();i++){
      if(i==0)
        doc=collector.Collect(eindicator.getRequestNames().get(i),
```

```
        eindicator.getParams());
      else
      {
        NodeList nl=collector.Collect(eindicator.getRequestNames().get(i),
        eindicator.getParams()).getElementsByTagName("element");
        Element root=doc.getDocumentElement();
        for(int j=0;j<nl.getLength();j++){
        Node node=doc.createElement("element");

        for(int k=0;k<nl.item(j).getChildNodes().getLength();k++)
        {
        Element n=doc.createElement(
        nl.item(j).getChildNodes().item(k).getNodeName());
        n.appendChild(
        doc.createTextNode(
        nl.item(j).getChildNodes().item(k).getTextContent()));
        node.appendChild(n);
        }
        root.appendChild(node);
        }
      }


    }
    /** etape 2 : executer la formule*/
    Class c= Operation.class;
    int index1=0,index2=0;

    while(index2>=0){
      index2=eindicator.getFormula().indexOf(";", index1+1);
      if(index2>0){
      String[] str=eindicator.getFormula().substring(index1,index2).split("#");
      String methode=str[0];
      String[] args=str[1].split("\\|");
      Class[] cl=new Class[args.length+1];
      cl[0]=Document.class;
      for(int i=1;i<cl.length;i++)
        cl[i]=String.class;
      try {
        Method m= c.getMethod(methode,cl);
        Object[] o=new Object[cl.length];
        o[0]=doc;
        for(int i=1;i<cl.length;i++)
          o[i]=args[i-1];

        doc=(Document)m.invoke(Operation.getInstance(),o);
        //System.out.println("++++++++++++++++++++++Resultat Indicateur+++
        ++++++++++++++++++++++++++
        \n"+XMLTools.getDocumentAsXml(doc)+"+++++
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++");
      } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
```

```
        }
        index1=index2+1;
      }
      return doc;
    }

}
```

Listing A.6 – FLICalculator

```
package CoucheIndicateurs;


import java.util.List;
import java.util.ArrayList;

import org.w3c.dom.Document;

import xml.XMLTools;

import CoucheSonde.Params;


public class FLICalculator
    implements ICalculator
{
  private static volatile FLICalculator instance = null;

  private FLICalculator() {
    super();

  }

  public static synchronized FLICalculator getInstance() {

    if (instance == null)
      instance = new FLICalculator();

    return instance;
  }

    public List<FLIndicator> flindicators = new ArrayList<FLIndicator> ();
    public List<FLIndicator> getFLIndicator () {
        return this.flindicators ;
    }

  public void Init() {
    flindicators = (ArrayList<FLIndicator>) XMLTools.
    decodeFromFile(FLIndicator.CONFIGUREFILE);

  }

  public Document calculate(String id) {
    // TODO Auto-generated method stub
```

```
    return null;
  }

}
```

Listing A.7 – IndicatorManager

```
package CoucheIndicateurs;

import java.util.ArrayList;

import xml.XMLTools;
import CoucheSonde.DBCollector;
import CoucheSonde.DBRequest;
import CoucheSonde.Params;


public class IndicatorManager
{
  public static ArrayList<EIndicator> getEindicators() {
    return eindicators;
  }

  public static void setEindicators(ArrayList<EIndicator> eindicators) {
    IndicatorManager.eindicators = eindicators;
  }

  public static ArrayList<FLIndicator> getFlndicators() {
    return flndicators;
  }

  public static void setFlndicators(ArrayList<FLIndicator> flndicators) {
    IndicatorManager.flndicators = flndicators;
  }

  private static volatile IndicatorManager instance = null;

  private IndicatorManager() {
    super();

  }

  public static synchronized IndicatorManager getInstance() {

    if (instance == null)
      instance = new IndicatorManager();

    return instance;
  }

  public static ArrayList<EIndicator> eindicators=new ArrayList<EIndicator>();
  public static ArrayList<FLIndicator> flndicators=new ArrayList<FLIndicator>();

    public ArrayList<EIndicator> getAllEIndicators()
```

```
    {
      eindicators = (ArrayList<EIndicator>) XMLTools
    .decodeFromFile(EIndicator.CONFIGUREFILE);
    return eindicators;
    }
    public int saveAllEIndicators()
    {
      try {
      XMLTools.encodeToFile(eindicators, EIndicator.CONFIGUREFILE);
    } catch (Exception e) {
      return 0;
    }
    return 1;
    }
    public void addEIndicator(String id, String designation,String requestType,
    String formula,ArrayList<String> requestNames,ArrayList<Params> params)
    {



      eindicators.add(new EIndicator( id,designation,requestNames , requestType,
      formula,params));
    }
    public void updateEIndicator(int index,String id, String designation,
    String requestType, String formula,ArrayList<String> requestNames,
    ArrayList<Params> params)
    {


      eindicators.set(index,new EIndicator( id,designation,requestNames , requestType,
      formula,params));
    }
    public void deleteEIndicator(int index)
    {
      eindicators.remove(index);
    }
//    public void addRNIndicator(int index,String RequestName)
//    {
//      eindicators.get(index).getRequestNames().add(RequestName);
//    }
//
//    public void deleteRNIndicator(int index,int RNindex)
//    {
//      eindicators.get(index).getRequestNames().remove(RNindex);
//    }
//    public void addParamIndicator(int index,String name,String value)
//    {
//      eindicators.get(index).getParams().add(new Params(name,value));
//    }
//
//    public void deleteParamIndicator(int index,int Paramindex)
//    {
//      eindicators.get(index).getParams().remove(Paramindex);
//    }
```

```
//

}
```

Listing A.8 – FLogic

```
package fuzzy;

import net.sourceforge.jFuzzyLogic.FIS;

public class Test {

  /**
   * @param args
   */
  public static void main(String[] args) {
    String fileName = "./data.FCL";
    FIS fis = FIS.load(fileName, true);
    // Error while loading?
    if (fis == null) {
      System.err.println("Can't load file: '" + fileName + "'");
      return;
    }

    // Show
    fis.chart();


    // Set inputs
    fis.setVariable("nb_bugs", 25);
    // Evaluate
    fis.evaluate();

    // Show output variable's chart
    fis.getVariable("nb_bugs").chartDefuzzifier(true);
    System.out.println(fis.getVariable("nb_bugs").defuzzify());

    // Print ruleSet
    System.out.println(fis);

  }

}
```

B

Dossier de conception

Dans cette annexe nous exposons les différents diagrammes de séquence utilisés pour la représentation de l'ensemble des scénarios d'utilisation du système AMS. Nous avions définis, dans la phase d'analyse et de conception, neuf cas d'utilisation du système. Les diagrammes de séquence associés aux cas d'utilisation « Gérer les sondes sur les fichiers LOG » et « Gérer les indicateurs flous » ont été présentés dans le chapitre 6 de cette thèse ; cette annexe est donc dédiée aux sept autres diagrammes.

Diagrammes de séquence du système AMS

1. Diagramme de séquence nominal associé au cas d'utilisation "Gérer les sources sur les bases de données"

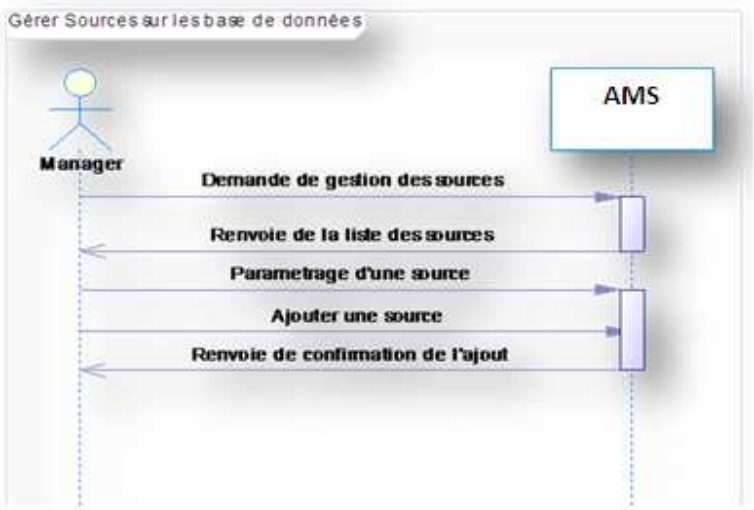


Fig. B.1 – Diagramme de séquence nominal du UC : Gérer les sources sur les bases de données

2. Diagramme de séquence nominal associé au cas d'utilisation : Gérer les sources sur les fichiers LOG

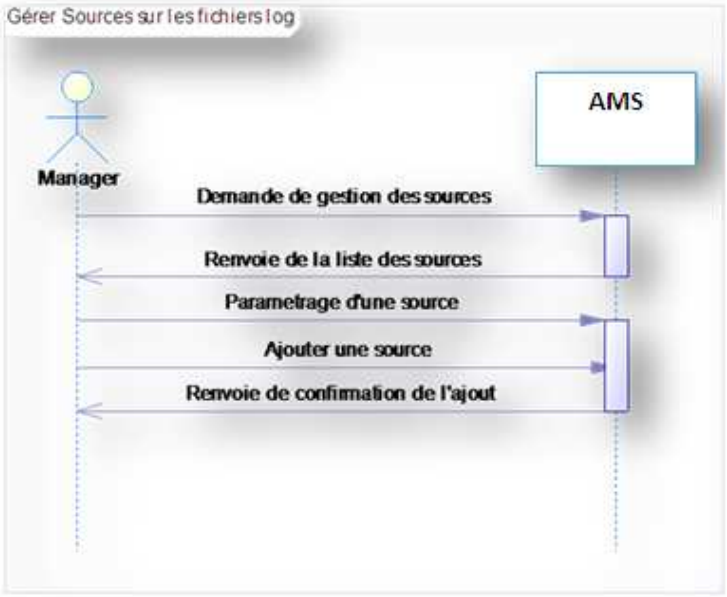


FIG. B.2 – Diagramme de séquence nominal du UC : Gérer les sources sur les fichiers LOG

3. Diagramme de séquence nominal associé au cas d'utilisation : Gérer les sondes sur les bases de données

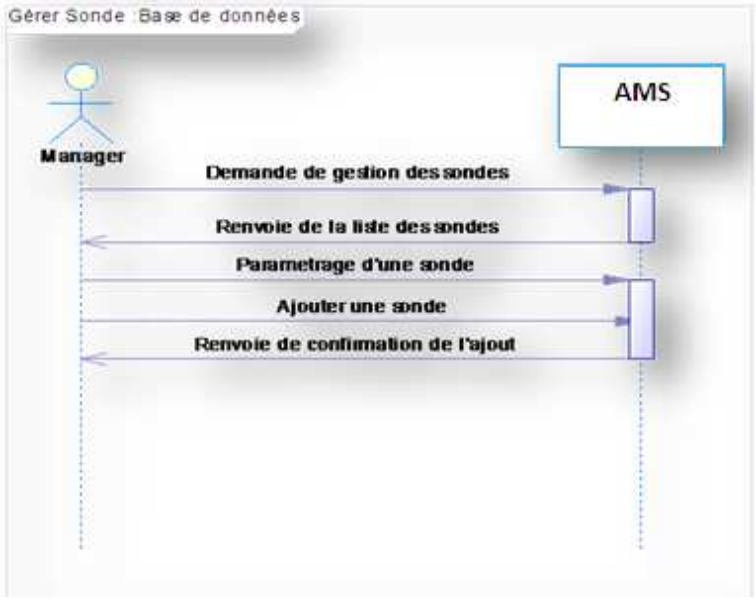


FIG. B.3 – Diagramme de séquence nominal du UC : Gérer les sondes sur les bases de données

4. Diagramme de séquence nominal associé au cas d'utilisation : Gérer les indicateurs élémentaires

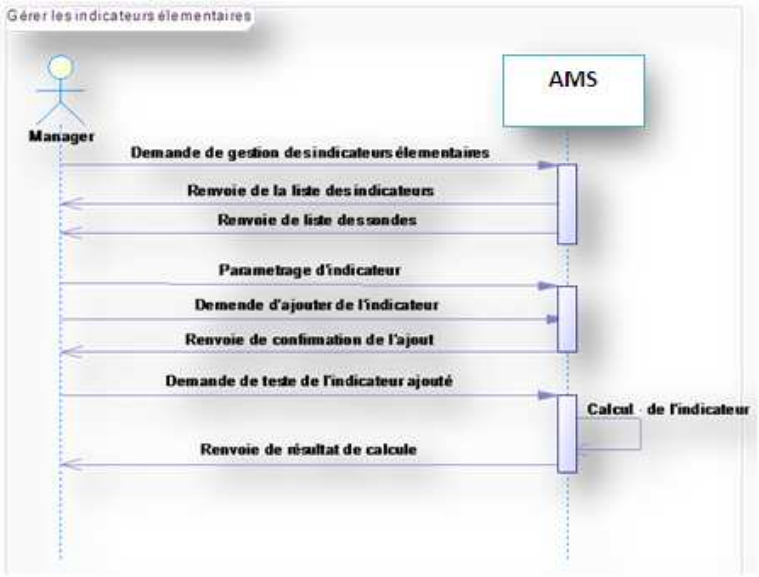


FIG. B.4 – Diagramme de séquence nominal du UC : Gérer les indicateurs élémentaires

5. Diagramme de séquence nominal associé au cas d'utilisation : Gérer les tableaux de bord

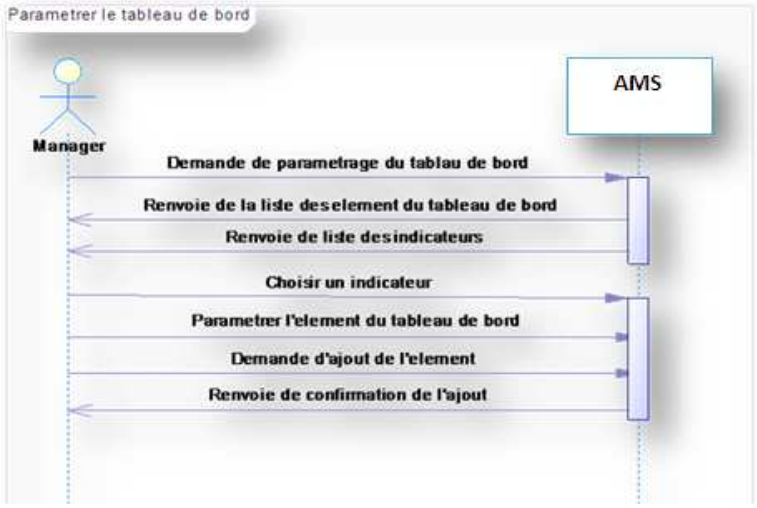


FIG. B.5 – Diagramme de séquence nominal du UC : Gérer les tableaux de bord

6. Diagramme de séquence nominal associé au cas d'utilisation : Visualiser les tableaux de bord

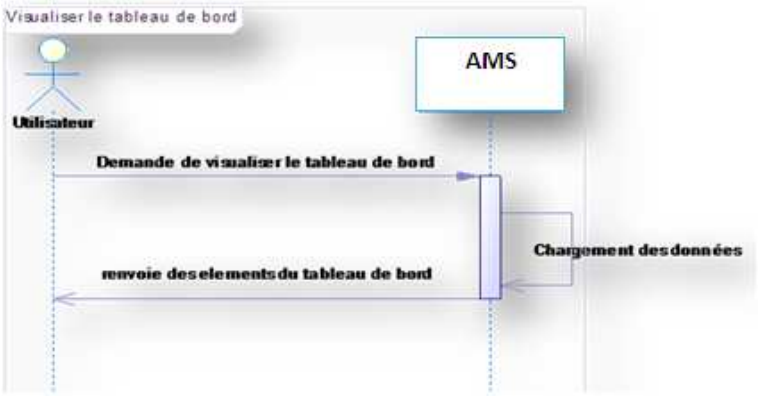


FIG. B.6 – Diagramme de séquence nominal du UC : Visualiser les tableaux de bord

7. Diagramme de séquence nominal associé au cas d'utilisation : Gérer les actions de notification

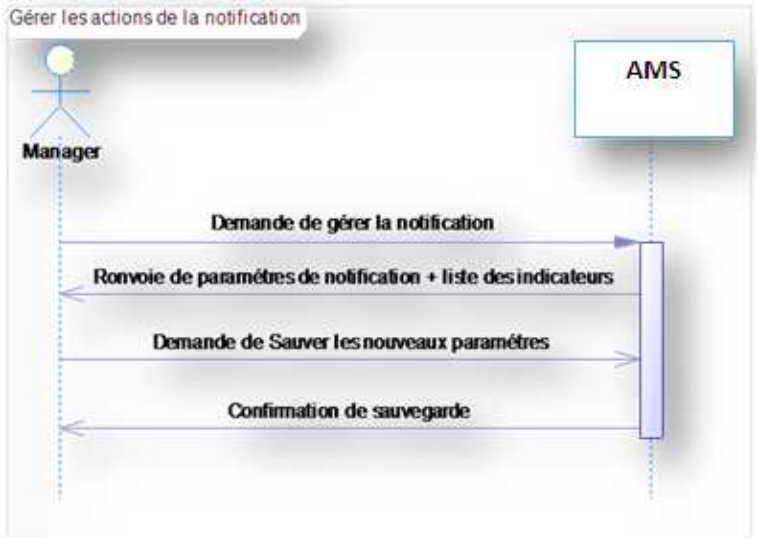


FIG. B.7 – Diagramme de séquence nominal du UC : Gérer les actions de notification