Controlling and Monitoring Specific Hardware Resources

Mihaela MUNTEAN, Gabriela MIRCEA, Andreea POP
West University of Timisoara,
mihaela.muntean@e-uvt.ro, gabriela.mircea@e-uvt.ro, badau.andreea@gmail.com

Within an agile approach, a RIA application for controlling and monitoring specific hardware will be proposed. It is always a challenge to choose the suitable technologies and IT frameworks to transform the PIM model into the PSM model of the future RIA and after that to implement the prototype. The main technological aspects are presented; the approach makes direct referees to the developed tiers of the RIA prototype.
Keywords: *Rich Internet Applications (RIA), Multi-Tier Architecture, Agile Development, RIA Frameworks*

1 Introduction

Rich Internet Applications (RIAs) is an umbrella term for various applications that are combining "the media-rich power of the traditional desktop with the deployment and content-rich nature of web applications" [1]. According to Gartner, RIA frameworks can be divided in JavaScript/AJAX-based frameworks and plugin-based frameworks. The first ones are browser-based and more lightweight, the second ones are more heavy-weight with a bigger download footprint (Valdez, 2009; Busch & Koch, 2009).

Plugin-based frameworks like Adobe Flex, JavaFX and Microsoft Silverlight, offer extended support for media, are highly interactive, cross-platform, cross-device, cross-browser and desktop-like with offline and out-off-browser support [8].

A typical RIA application has a multi-tier architecture, based on data integration layer, business logic & services layer and presentation layer. Taking into consideration the new client-side capacities, the new presentation features, and the different communication flows between the client and the server, the following four phases in designing RIA applications are necessary:

- data model design,
- business logic & services design,
- presentation design, and
- communication design [12].

Communication is a cross-cutting concern related to data synchronization, business logic & services distribution and presentation. The communication must provide the binding between the presentation and the underlying data/business logic & services layers using synchronous/ asynchronous methods.

Best practices in RIA development enrich the general design considerations with specific design issues depending on the requirements and the technologies used for implementation.

With respect to the general architecture of a RIA application (*Figure 1*), a prototype for controlling and monitoring specific hardware resources was developed.

The demarche was developed within an Adobe Flex framework, a just-in-time deployment model being employed.

2 Agile Approach for Developing the RIA Prototype

Based on the author's results obtained in the agile development of portals [10], [11], the demarche was applied for developing the proposed RIA application. The main functionalities of the RIA application for controlling and monitoring specific hardware resources are described by the use cases diagram (*Figure 2*).

The agile development framework proposed in [10], [11] recommends the use of prototype technique enriched with MDA (Model Driven Architecture) specific attributes and is based on the following phases:

- Conception: the PIM model elaboration is targeted, according to the requirements;
- Design: targets the elaboration of the PSM model specific for the RIA prototype, i.e. the finalization of the architecture of this model, taking into account all details

DOI: 10.12948/issn14531305/20.4.2016.02

regarding the IT infrastructure, which must sustain the unitary, integrating vision of the PIM model; the building of the PSM model will take into account the future implementation solution of the RIA, by relating the model to certain developing technologies and frameworks;

- I.T.I (Implementing -Testing- Installation) phase has the goal to implement the RIA prototype according to the PSM model, followed by the testing of the prototype; often, as a result of testing its functionality, the prototype invalidation leads to the revision of the PSM model and aims at correcting some aspects related to technologies and implementing frameworks.

With respect to the RIA general architecture, the PIM and further the corresponding PSM models have been elaborated: model for data layer; model for business & services layer; model for presentation layer; model for communication schema; model for security schema; model for general management and maintenance of the RIA application.

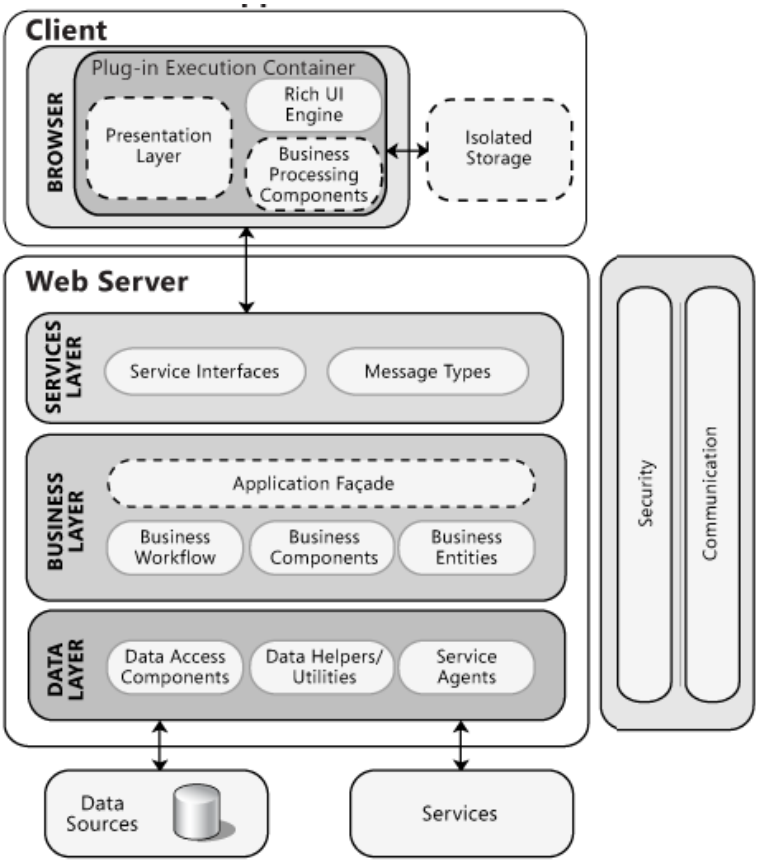

Fig. 1. RIA general architecture [14]

Both PIM and PSM models were described with the help of the UML language, al kind of specific diagrams being developed. The advantage of using UML modeling language is obvious [6], [7], and in RIA development approaches consolidates the agile development desiderata.

Functionalities like

- reading temperature, voltage, video sensors values, inputs and outputs,
- changing the sensors' configuration and properties,
- changing the current outputs values, and
- managing user accounts (*Figure 2*)

are sustained both by PIM and further by PSM models with concrete implementations within the multi - tier architecture of the RIA.

It is always a challenge to choose the suitable technologies and IT frameworks to transform the PIM model into the PSM model of the

DOI: 10.12948/issn14531305/20.4.2016.02

future RIA and after that to implement the prototype.
For implementation of the RIA prototype the following technologies and frameworks have been used: Java, Adobe Flex, Adobe BlazeDS, Apache Tomcat, Servlet technology and SQLite.

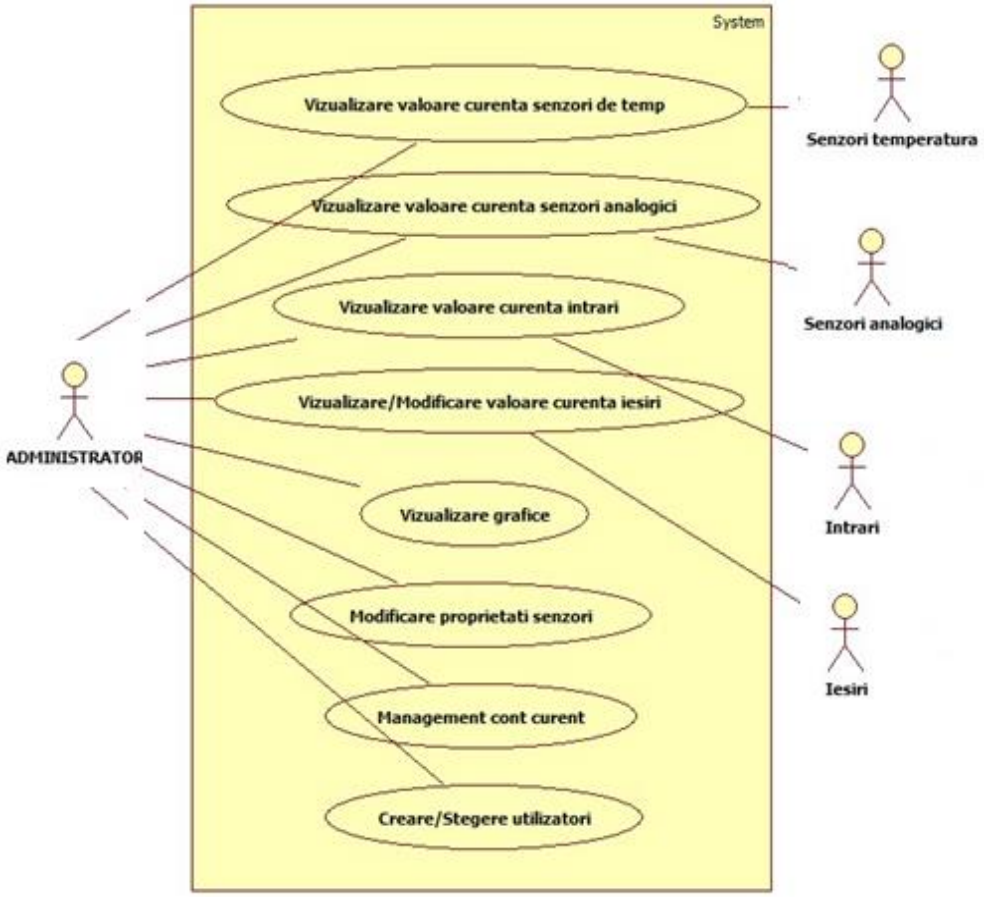

Fig. 2. Modeling the main functionalities

3 Technologies Used for Implementation of the RIA Prototype

3.1 SQLite and DAO

Managing data is a critical part of app development. The PSM model of the data layer include implementation aspects specific to SQLite databases. Best practices recommend the use of SQLite in situations like Embedded devices and Internet of Things, Websites, Data analysis, etc. The database diagram is presented in Figure 3.
The access mechanism used to access the data layer from the above layers is DAO (Data Access Object). The DAO pattern is a widely accepted mechanism to abstract away the details of persistence in an application, including in the proposed RIA prototype. The idea is that instead of having the business layer and/or service layer communicate directly with the database (*Figure 3*), file system, or whatever persistence mechanism the application uses, the business layer speaks to a DAO layer instead (*Figure 4*). This DAO layer then communicates with the underlying data layer.

DOI: 10.12948/issn14531305/20.4.2016.02

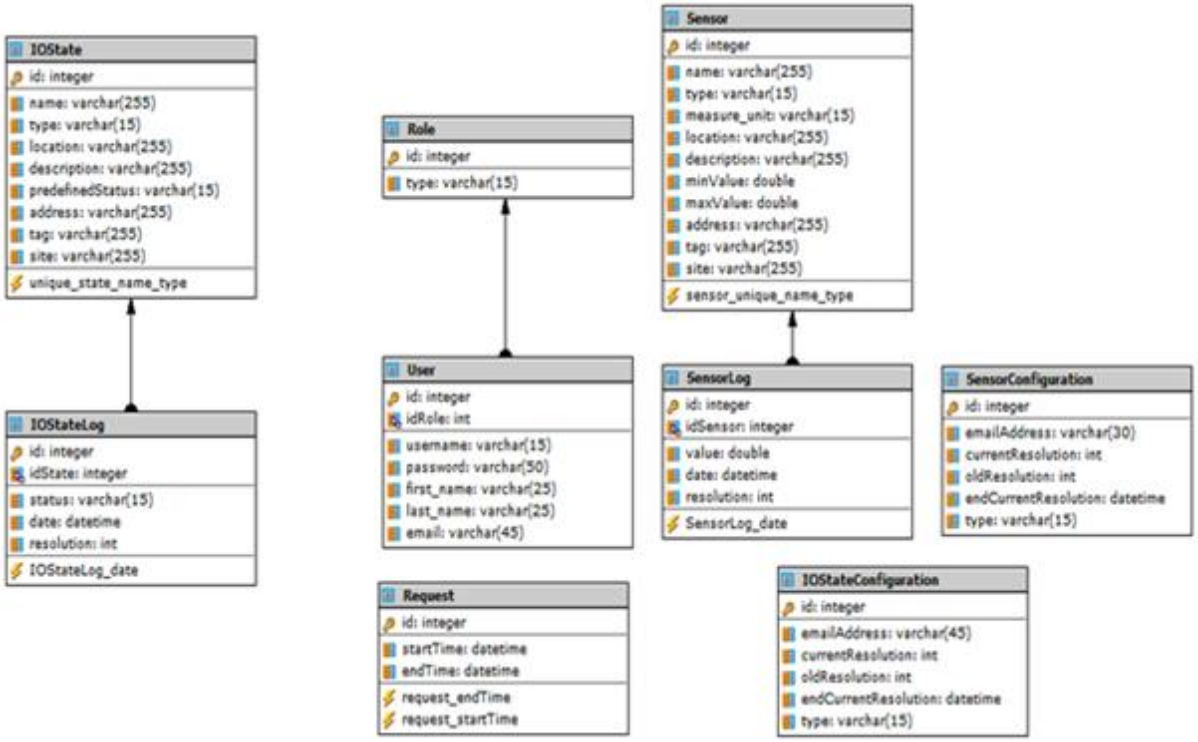

Fig. 3. Database diagram

Data Access Objects as a design concept can be implemented in a number of ways (*Code sequence 1*).

Code sequence 1. SQLManager class

```
public class SqlManager {
private Connection connection = null;

public void connect() {
String url = null;
String driver = null;
try {
driver = "org.sqlite.JDBC";
Class.forName(driver).newInstance();
url = "jdbc:sqlite:" +
SqlConfig.databaseName;
connection =
DriverManager.getConnection(url);
if(connection != null)
connection.setAutoCommit(true);
else
System.out.println("failed to connect");
} catch (SQLException sqle) {
sqle.printStackTrace();
connection = null;
} catch (Exception e) {
e.printStackTrace();
connection = null;
}
}
public void disconnect() {
try {
if (connection != null)
connection.close();
} catch (Exception e) {
System.out.println("Error: " +
e.getLocalizedMessage());
} finally {
connection = null;
}
public Connection getConnection() {
return this.connection;
}
}
public void
updateState(RpcInputOutputState state) {
synchronized (DbLock.getInstance()) {
SqlManager sqlManager = new
SqlManager();
sqlManager.connect();
Connection connection =
sqlManager.getConnection();
Statement statement = null;
try {
statement =
connection.createStatement();
statement.executeUpdate("UPDATE " +
SqlConfig.IOState.tablename + " SET " +
SqlConfig.IOState.name + "='" +
state.name + "', " +
SqlConfig.IOState.location + "='" +
state.location + "', " +
SqlConfig.IOState.description + "='" +
state.description + "', " +
SqlConfig.IOState.site + "='" +
state.site + "', " +
SqlConfig.IOState.logic + "='" +
state.logic + "' WHERE " +
SqlConfig.IOState.id + "=" + state.id);
NotificationManager.getInstance().sendEv
entToClients(new
RemoteEvent(RemoteEvent.STATE_UPDATED));
```

DOI: 10.12948/issn14531305/20.4.2016.02

```
} catch (SQLException e) {
e.printStackTrace();
} finally {
try {
statement.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
sqlManager.disconnect();
}
}
}
```

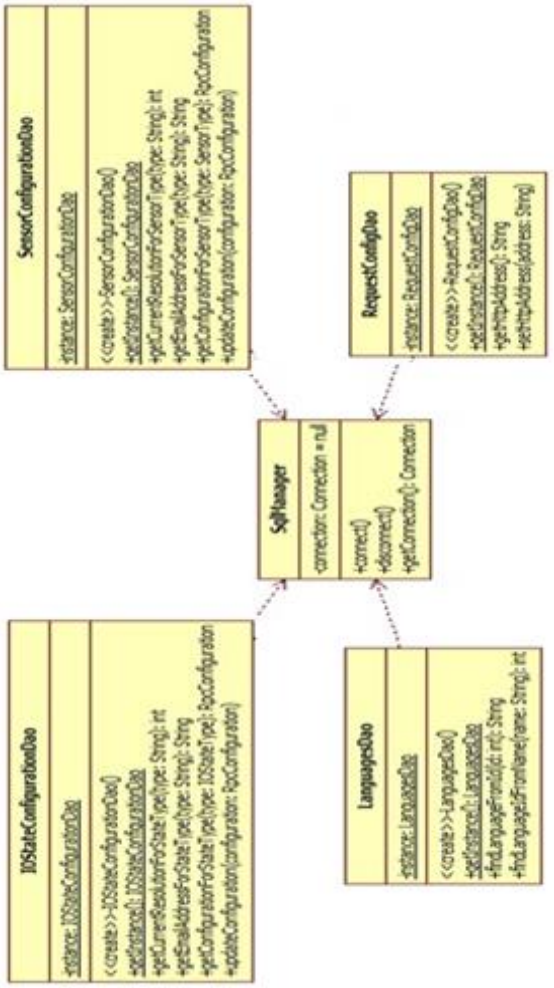

Fig. 4. DAO layer. Class diagram

3.2 Java. Multithreaded Programming and the Servlet Technology

Java is pure OOP language and provides integrated support for multithreaded programming [2], [5].

Java supports cross-platform code through the use of Java bytecode, that can be interpreted on any platform by JVM.

Implementing the business & services model of the RIA prototype involves creating classes (*Figure 5*), creating objects from those classes, and developing executable program(s) for the considered functionalities that use those objects [9].

Monitoring the specific hardware resources, meaning reading temperature, voltage, video sensors values, inputs and outputs has been implemented by using multithreaded programming. A multi-threaded program contains two or more parts that can run concurrently and each part can handle different task at the same time making optimal use of the available resources especially when your computer has multiple CPUs. Below is the code for monitoring temperature, by reading the sensor values (*Code sequence 2*). The threads have been created by extending the Thread class, approach that provides more flexibility in handling multiple threads.

The request-response programming model uses servlets (*Code sequence 3*) for extending the capabilities of the RIA application. Java Servlet technology defines HTTP-specific servlet classes.

Javax.servlet and javax.servlet.http package provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines lifecycle methods. When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services (*Code sequence 3*).

Code sequence 2. Create Thread

```
class TemperatureReaderThread extends
Thread {
private static int secondsResolution;

public static int getResolution() {
RpcConfiguration configuration =
SensorConfigurationDao.getInstance()
.getConfigurationForSensorType(SensorTyp
e.TEMPERATURE);
Date today = new Date();
if (configuration.oldResolution != 0
&& configuration.endCurrentResolution !=
null &&
configuration.endCurrentResolution.compa
reTo(today) <= 0) {
configuration.currentResolution =
configuration.oldResolution;
configuration.oldResolution = 0;
```

DOI: 10.12948/issn14531305/20.4.2016.02

```
configuration.endCurrentResolution =
null;
SensorConfigurationDao.getInstance().upd
ateConfiguration(configuration);
}
return configuration.currentResolution;
}

public static String getEmail() {
RpcConfiguration configuration =
SensorConfigurationDao.getInstance()
.getConfigurationForSensorType(SensorTyp
e.TEMPERATURE);
return configuration.emailAddress;
}

public void init() {
secondsResolution = getResolution();
}

public void run() {
while (true &&
!HardwareReaderThread.finished) {
try {
System.out.println("TEMPERATURE");
Thread.sleep(secondsResolution * 1000);
HardwareReader hardwareReader = new
HardwareReader(
secondsResolution, getEmail());
hardwareReader.readRequests();
hardwareReader.readTemperatures();
hardwareReader.deleteOldLogs();
hardwareReader.sendMail();
// at the end: read again the
resolution, maybe it has been
// changed
secondsResolution = getResolution();
NotificationManager.getInstance().sendEv
entToClients(
new
RemoteEvent(RemoteEvent.HARDWARE_OK));
} catch(OneWireException oneException) {
oneException.printStackTrace();
if(oneException.getMessage().contains("n
ot available"))
NotificationManager.getInstance().sendEv
entToClients(
new
RemoteEvent(RemoteEvent.HARDWARE_NOT_FOU
ND));
} catch (Exception e) {
e.printStackTrace();
}
}
}
}
```

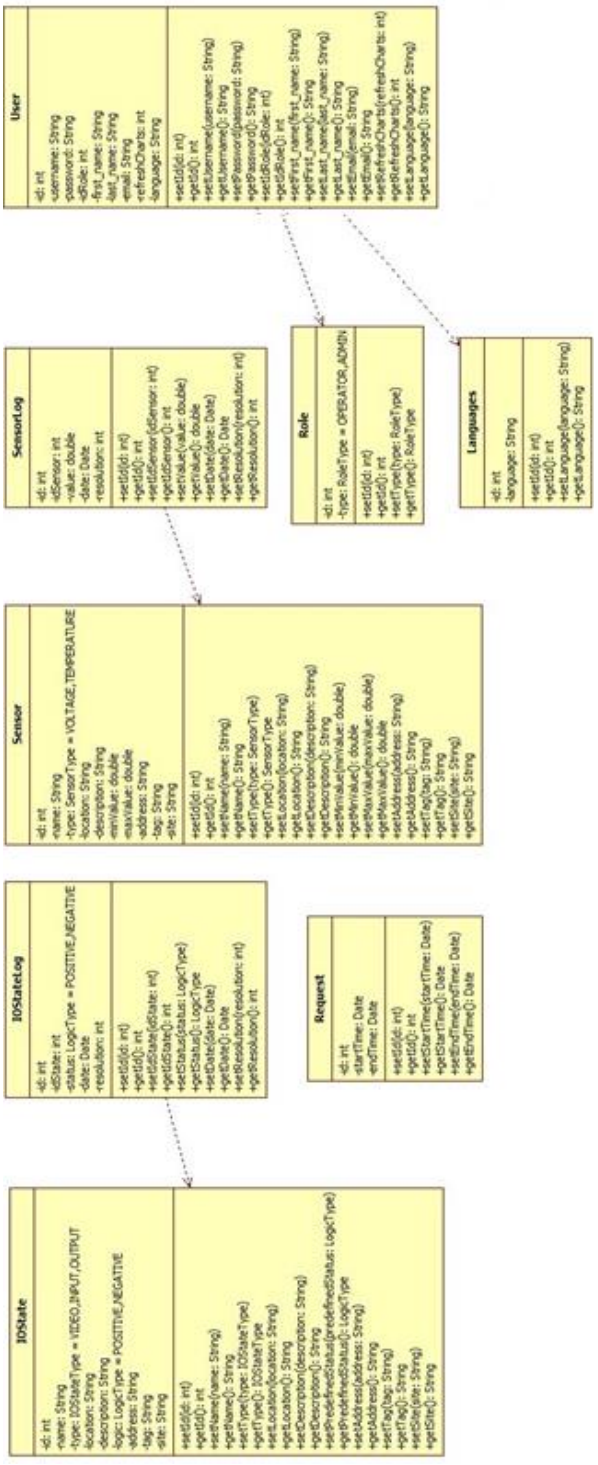

Fig. 5. Business logic. Class diagram

Cod sequence 3. Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SomeServlet extends
HttpServlet {
```

DOI: 10.12948/issn14531305/20.4.2016.02

```
public void doGet(HttpServletRequest
request, HttpServletResponse response)
throws ServletException, IOException {
...
}
public void doPost(HttpServletRequest
request, HttpServletResponse response)
throws ServletException, IOException {
... }
}
```

3.3 Adobe Flex Framework

Interactivity, responsiveness and richness are three general characteristics of RIAs. Adobe Flex provides development tools, user interface and connectivity components that simplify the development of RIA applications (Figure 6). Common used for implementing the presentation layer, the flex framework manages a good communication with the underlying layers of business logic & services [4].

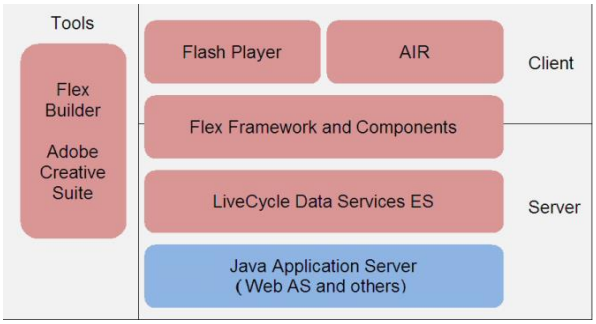

Fig. 6. Adobe Flex framework [15]

Flex can dynamically change views and send and retrieve data asynchronously to the server in the background, updating but never leaving the single application interface (similar to the functionality provided by the XMLHttpRequest API with JavaScript). The Flex framework has three remote procedure call APIs that communicate with the server over HTTP: HTTPService, WebService, and RemoteObject. Best practices in building Flex and Java client-server applications make substantial referees to Blade DS or LiveCycle Data Services (*Figure 7*).

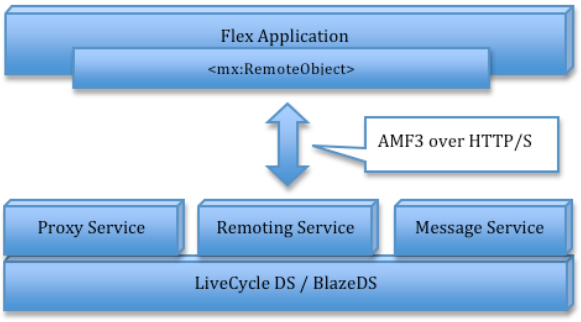

Fig. 7. Flex remoting architecture [16]

For implementing the proposed RIA application for controlling and monitoring specific hardware resources, it was necessary to configure the BladeDS and to establish, in the *<server_location>\webapps\blazeds \WEB-INF\flex\services_config.xml* file, the communication with the Java server (*Code sequence 4*).

Code sequence 4. Flex - Java server communication

```
<service id="remoting-service"
class="flex.messaging.services.Remoting
Service"
messageTypes="flex.messaging.messages.R
motingMessage">
<adapters><adapter-definition id="java-
object"
class="flex.messaging.services.remoting.
adapters.JavaAdapter"
default="true"/>
</adapters>
<destination id="StatusApplication">
<properties>
<source>com.statusApp.interface.StatusAp
plication</source>
</properties>
<channels>
<channel ref="my-amf"/>
</channels>
</destination>
</service>

<channel-definition
class="mx.messaging.channels.AMFChannel
id="my-amf" >
<endpoint
uri=http://localhost:8400/blazeds/messag
ebroker/amf
class="flex.messaging.endpoints.AMFEnd
point"/>
</channel-definition>
```

Also, in the *<server_location>\webapps\ blazeds\WEB-INF\flex\messages_ config.xml* file, the following specifications are needed.

DOI: 10.12948/issn14531305/20.4.2016.02

```
<destination
id="ServiceAdapterDestination">
<channels>
<channel ref="my-streaming-amf"/>
</channels>
<adapter
ref="NotificationManagerServiceAdapter"/
>
</destination>

<default-channels>
<channel ref="my-streaming-amf"/>
</default-channels>
```

3.4 Some User Views for Controlling and Monitoring Specific Hardware Resources

One of the main functionalities of the proposed RIA prototype is reading the temperature, voltage, video sensors values, inputs and outputs. It is an ongoing process, the values being stored into the database and concomitant displayed in different screens (*Figure 8*, *9*).

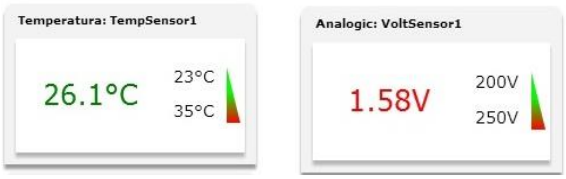

Fig. 8. Current values of temperature sensor and analogic sensor

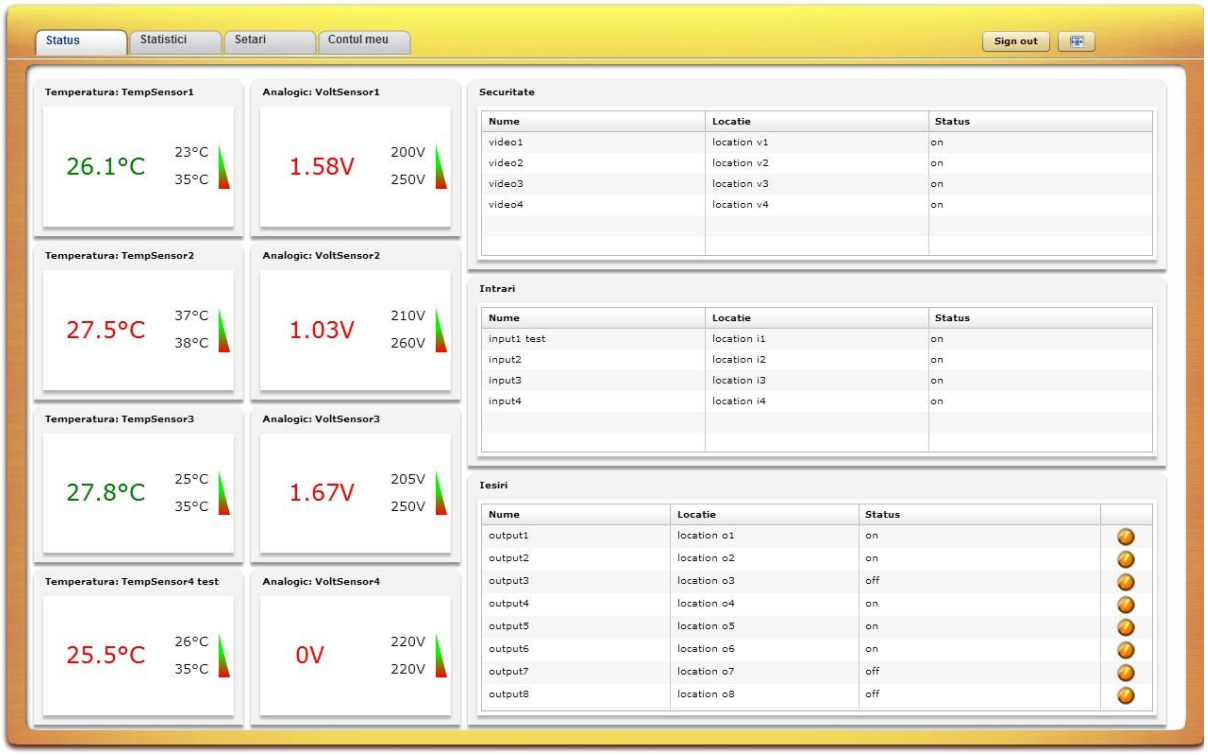

Fig. 9. Current values of the sensors, of the inputs and outputs

The second main functionality, changing the sensors' configuration and properties, allows to adapt these devices in order to optimize the controlling process (*Figure 10*). When necessary, it is possible to change the outputs values (*Figure 11*).

Managing user accounts, the fourth mentioned functionality is viable for the application administrator, that is responsible for creating accounts and controlling the user access to the system.

DOI: 10.12948/issn14531305/20.4.2016.02

Nume	Unitate de masur	Locatie	Descriere	Site	Valoare minima	Valoare maxima	Adresa	Tag	
TempSensor1	°C	location1	description 1		23	35	81000000CF45B0		
TempSensor2	°C	location2	description2		37	38	CB000000BE5E4E		
TempSensor3	°C	location 3	description3		25	35	AE0000003A0FCE		
TempSensor4 tes	°C	location 4	description4		26	35	7D000000BF10BE		

Fig. 10. Modification of the temperature sensor's characteristics

Iesiri

Nume	Locatie	Status	
output1	location o1	on	
output2	location o2	on	
output3	location o3	off	
output4	location o4	on	
output5	location o5	on	
output6	location o6	on	
output7	location o7	off	
output8	location o8	off	

Fig. 11. Current value and location of video outputs

4 Conclusions

Based on the author's experience in portal development frameworks, and the results obtained in the agile development of portals [10], [11], the theoretical considerations where transposed to the quick development of RIA applications. The proposed scenario in paragraph 2 was applied for developing an application for controlling and monitoring specific hardware resources. The PIM and PSM models were designed for each tier of the RIA multi-tier prototype: data model, business logic & services model, presentation model, communication model; security aspects were also taken into consideration.

Based on the identified requirements, the functionalities of the application were designed and were transposed into the tiers' PIMs, which have been coagulated into the unitary application PIM model. Further, the PSM model was developed.

Practically, the final version of the RIA prototype is obtained by an iterative process which regards the adjustment of the PSM, its implementation and the testing of the prototype solutions for verifying the imposed requirements.

The validation of the prototype leads to the application installation and its transfer to the users [11].

The implementation demarche is based on the following technologies: SQlite and DAO, Java multithreaded programming and the servlet technology, Apache Tomcat and Adobe Flex framework. In paragraph 3, the main technological aspects are presented; the approach makes direct referees to the developed tiers.

Finally, some controlling and monitoring screens have been exemplified.

References

[1] J. Allaire, *ColdFusion MX, JRun 4 and Rich Internet Applications* - Interview in Java Developer's Journal, 2002

[2] J. Block, *Effective Java*, 2rd Ed, Addison Wesley, 2008

[3] M. Busch, N. Koch, *Rich Internet Applications. State-of-the-Art*, Technocal Report 0902, Ludwig-Maximilian Universität München, 2009

[4] A. Cole, *Learning Flex 3. Getting up to Speed with Rich Internet Applications*, O'Reilly Media, Inc., 2008

DOI: 10.12948/issn14531305/20.4.2016.02

[5] B. Eckel, *Thinking in Java*, 4th ed., Prentice-Hall PTR, 2006
[6] M. Fowler, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999
[7] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd Edition, Addison-Wesley, 2003
[8] C. D. Granbäck, *Rich Internet Applications (RIAs). A Comparison between Adoble Flex, JavaFX and Microsoft Silverlight*, Chalmers University of Technology Ed., 2009
[9] W. Hasselbring, S. Giesecke, *Trustworthy Software Systems*, Research Methods in Software Engineering, 2006
[10] M. Muntean, *Abordări ale unor sisteme colaborative în medii bazate pe cunoaştere*, Editura Mirton, Timişoara, 2010
[11] M. Muntean, "Considerations Regarding the Agile Development of Portals," *Journal of Applied Computer Science & Mathematics*, no. 10 (5) /2011
[12] J.C. Preciado, M. Linaje, S. Comai, F. Sanches-Figueroa, *Designing Rich Internet Application with Web Engineering Methodologies*, 2008, https://www.pst.ifi.lmu.de/~kochn/preciado-et-al_icwe2008.pdf
[13] R. Valdes, *Key Issues in Rich Internet Applications Platforms and User Experience*, Gartner Inc., 2009
[14] Microsoft, *Microsoft Application Architecture Guide, 2nd Edition, Chapter 23: Designing Rich Internet Applications*, October 2009, Available at: https://msdn.microsoft.com/en-us/library/ee658083.aspx
[15] S. Laxmi Kata, *Developing rich Internet applications for SAP using Adobe Flex*, 2010, Available at: http://www.mouritech.com/documents/sap_with_flex.pdf
[16] J. Stallons, *The architecture of Flex and Java applications*, 2010, Available at: http://www.adobe.com/devnet/flex/articles/flex_java_architecture.html

With a background in Computer Science and a Ph.D. obtained both in Technical Science and in Economic Science (Economic Informatics), professor **Mihaela I. MUNTEAN** focused her research activity on topics like information technology, knowledge management, business intelligence, business information systems. Over 70 papers in indexed reviews and conference proceedings and the involvement with success in 7 multi - annual national research grants/projects are sustaining her contributions in the research fields mentioned above. Currently, professor Mihaela I. Muntean is the chair of the Business Information Systems at the West University of Timişoara and an IT independent consultant.

Gabriela MIRCEA received her degree on Informatics from the West University of Timisoara, Faculty of Mathematics and Informatics, in 1993 and her doctoral degree in Mathematics in 2003. Since 1993 she is teaching at West University of Timisoara, Faculty of Economics and Business Administration, Department of Business Information System. Her research activity is focused on web technologies, computer networks and on interdisciplinary topics implying various numerical simulations. The scientific research results have been published in eight books/chapters at national and international publishing houses and 65 articles appearing in journals and proceedings of international conferences from Romania or abroad. Her involvement can be measured in one international and eight national research grants/projects that she took part in.

DOI: 10.12948/issn14531305/20.4.2016.02

Andreea POP is a senior software developer, currently working as a development consultant for one of the largest software companies in the world. She holds a Bachelor degree in Computer Science and a Master degree in Business Information Systems. She is passionate about software design and architecture, best practices and solution architecture.

DOI: 10.12948/issn14531305/20.4.2016.02