APPLICATION ON DEMAND PORTAL USING J2EE TECHNOLOGIES

Ch. Bouras
Research Academic Computer Technology Institute, & Computer Engineering and Informatics Dept., Univ. of Patras
61 Riga Feraiou Str, Patras, GR-26221, Greece
bouras@cti.gr

A. Gkamas
Research Academic Computer Technology Institute, & Computer Engineering and Informatics Dept., Univ. of Patras
61 Riga Feraiou Str, Patras, GR-26221, Greece
gkamas@cti.gr

D. Primpas
Research Academic Computer Technology Institute, & Computer Engineering and Informatics Dept., Univ. of Patras
61 Riga Feraiou Str, Patras, GR-26221, Greece
primpas@cti.gr

K. Stamos
Research Academic Computer Technology Institute, & Computer Engineering and Informatics Dept., Univ. of Patras
61 Riga Feraiou Str, Patras, GR-26221, Greece
stamos@cti.gr

ABSTRACT
This paper describes the design and the implementation of the ASP-NG portal, which is a portal for providing the user with the necessary interface in order to access an Application on Demand (AoD) service. The ASP-NG portal is responsible for the interaction with the user of the Application on Demand service. The implementation of the ASP-NG portal is based on the Java 2, Enterprise Edition (J2EE) platform. The ASP-NG portal offers to its users the capability to select and customize the language of the user interface in order to present information in their preferred language. Moreover the ASP-NG portal offers to the portal administrator the capability to customize the look and feel of the ASP-NG portal.

KEYWORDS
J2EE, AoD, Web Services, e-business, e-marketplace

1. INTRODUCTION
Using the Application on Demand service the user can rent and use an application even if he has not purchased the complete application package. Instead, the user agrees with the ASP company to rent the application and each time the user only needs to download only the necessary modules for a specific session. The user is not able to use the application outside of the mutually agreed frame with his ASP, but he experiences the application as if he actually used the complete application package and installed it in his computer. The whole procedure is handled by the AoD system and is completely transparent to the end user.

This paper describes the design and the implementation of the ASP-NG portal of the ASP-NG project. The strategic goal of the international consortium undertaking the ASP-NG project is to research, develop

*This work is supported by the ASP-NG (ASP – New Game - IST-2001-35354) project of the IST (Information Society Technologies) program of EC (European Commission)
and implement a new middleware for Application Service Providers (ASPs), which integrates all the required functionality for running an ASP, including new methods such as streaming application, to deliver productivity tools and rich content, secure keying of IP and Digital Rights Management (DRM) with capabilities of running on different platforms and operating systems. The ASP-NG project is aimed at providing ASPs, small, medium, or large enterprises (SMEs and Telecoms), with a complete set of features required to run an ASP service.

The main role of the ASP-NG portal is to provide to the user the necessary interface in order to use the Application on Demand (AoD) service. The ASP-NG portal is responsible for the interaction with the end user. Through the ASP-NG portal the user is able to browse the available applications, rent and use them. In addition, the ASP-NG portal offers functionality such as searching for specific applications, checking already rented applications, creating “subusers”, that is, users with limited capabilities that are under the direct supervision of the parent user, credit purchase and renewal, viewing transaction history, chatting with other users of the ASP-NG system, asking questions at the ASP-NG portal’s forum, viewing already answered questions and more. For this reason it implements some functionality itself and some functionality is implemented in the other modules of the ASP-NG system, such as the Backoffice module and the CRM module.

Apart from the end user (“client”) the ASP-NG portal architecture defines a number of administrative roles. The administrators are responsible for maintaining and managing parts of the AoD system. It is assumed that the administrators are going to be persons from the ASP company that provides the AoD service. More particularly, the following administrators have access through the ASP-NG portal: 1) Portal Administrators: These administrators are responsible for configuring the ASP-NG portal. 2) Content Administrators: These administrators are responsible for application and content management (upload application metadata describing an uploaded application, create application packages, remove application from list of available applications). 3) User Administrators: These administrators are responsible for users management (create users, edit users, remove users). 4) System Administrators: These administrators are responsible for managing the rest of the administrator users (Portal Administrators, Content Administrators, User Administrators) of the ASP-NG portal.

In addition, the ASP-NG portal offers to its users the capability to select and customize the language of the user interface in order to present information in their preferred language. Initially, the ASP-NG portal supports only a small set of languages but the portal administrator has the capability to add support for more languages. Moreover, the ASP-NG portal offers to the portal administrator the capability to customize the look and feel of the ASP-NG portal.

The rest of this paper is organized as follows: In section 2, we give a brief overview of the ASP-NG system architecture with focus to the ASP-NG portal. Section 3 presents the implementation technologies of the ASP-NG portal. Finally, section 4 concludes the paper and discusses some of our future work.

### 2. ARCHITECTURE

Figure 1 presents the general architecture of ASP-NG system mainly from the ASP-NG portal point of view (This means that some modules which do not interact directly with the ASP-NG portal may not be displayed in this figure). As this figure shows, the ASP-NG portal interacts mainly with the Backoffice module and the users access the ASP-NG system services through the ASP-NG portal. The ASP-NG portal is responsible for the interaction with the user. For this reason it implements some functionality itself and some functionality is implemented on the other modules of the ASP-NG system. In addition the ASP-NG portal accesses the system database through the application server of the Backoffice module.

The Backoffice module consists of the application server, the mail server and the database management system (DBMS). The application server provides services to other modules (the ASP-NG portal is using these services) and access to the system database. The system database stores information related to the users, their credits, their accounts, the applications, which application has been rented by which user, etc., while the portal database stores different versions (different languages) of the ASP-NG portal, customizable interface, graphics, etc. The CRM module is responsible for handling customers’ relationships. The ASP-NG portal “drives” the end-user to the CRM through hyperlinks for reporting problems and incidences and get
connected with the CRM. The AoD server is responsible for providing the core Application on Demand technology.

The ASP-NG portal interacts with the Backoffice module with direct communication with the use of Enterprise JavaBeans (EJB) technology and more particularly with the use of Java RMI, which is used by the EJB technology for distributed communication. The EJB technology offers a number of alternative methods for implementing the business logic of a web application. For this reason, the developer can choose between Entity Enterprise JavaBeans, Session Enterprise JavaBeans and Message Enterprise JavaBeans. The Backoffice module is implemented using Entity Enterprise JavaBeans for managing data persistence in the system database. It also uses Session Enterprise JavaBeans in order to implement business logic and provide access to the Entity Beans. Therefore, the ASP-NG portal module does not have direct access to the Entity Enterprise JavaBeans, but it rather has access the Session Beans, which construct a layer covering the Entity Beans. This technique offers a number of benefits and is called the Session Facade ([Marinescu]).

There is not any direct interaction between the ASP-NG portal and the CRM module except that the ASP-NG portal provides links to the CRM module. Through these links the ASP-NG portal provides to the CRM module the user ID and the user preferred language. The diagnostic tools collect information about the status of the ASP-NG portal with the use of an SNMP agent and the SNMP protocol, which is installed in the web server of the ASP-NG portal. The interface between the ASP-NG portal and the AoD module is based on the use of JavaScript functionality. In addition the web browser must be compatible with ActiveX for easier interaction with the AoD client and the web server must be compatible with JavaBeans in order to access the AoD system.

In this paper we concentrate on the design and the implementation of the ASP-NG portal and the Backoffice module that serve the web-based front-end, which provides the AoD service. Both ASP-NG portal and Backoffice module are implemented using J2EE technologies.

![Figure 1: General Architecture of ASP-NG System](image)

### 3. IMPLEMENTATION TECHNOLOGIES

#### 3.1. Implementation of ASP-NG portal and Backoffice module

The implementation of the ASP-NG portal and the Backoffice module is based on the Java 2, Enterprise Edition (J2EE) platform ([J2EE]). More specifically, the ASP-NG portal makes use of the JavaServer Pages (JSP) and Servlet technologies and the Backoffice module makes use of Enterprise JavaBean (EJB) technologies. The J2EE technology provides a component-based approach to the design, development, assembly and deployment of enterprise applications, in order to reduce costs and enable faster enterprise application design and development. Its multitier distributed application model is not tied to the products and
Application Programming Interfaces (API) of any one vendor. J2EE applications are made up of components. A J2EE component is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and that communicates with other components. The web services of the J2EE specification defines client components, Servlet and JavaServer pages components that run on the server, and Enterprise JavaBeans components that run inside the EJB container.

Servlets are platform independent Java modules used inside a web server framework for extending the web server’s capabilities via a request-response programming model. JavaServer Pages is a technology that comprises of a language for developing JSP pages, an expression language for accessing server-side objects and mechanisms for defining extensions to the JSP language. A JSP page is a text-based document that describes how to process a request and construct a response. JavaServer Pages is an extension of the Java Servlet Technology, since a JSP page is translated into a Servlet that services requests and follows the Servlet life cycle. Enterprise beans are the J2EE components that implement the Enterprise JavaBeans technology. They run inside the EJB container, a runtime environment within the J2EE server that provides system-level services to its enterprise beans. Enterprise beans are the components that encapsulate the business logic of an application.

![Figure 2: ASP-NG implementation technologies](image)

The architecture of the ASP-NG portal is based on JavaServer pages that mainly contain the presentation data for the ASP-NG portal. In order to separate business logic from presentation logic, the JavaServer pages use functionality, which is implemented inside Enterprise JavaBeans, hosted by the Backoffice module (and have been implemented in the Backoffice module of the ASP-NG system). In order to provide this business logic the Backoffice module is using a number of EJBs which we call the Backoffice API. Using this approach we can also benefit from the transaction, security, performance and stability features offered by the EJB container of the Application Server. The JSP pages do not directly access the main ASP-NG database. Instead, they use Enterprise JavaBeans methods that enable the JSP pages to access the data they need. The ASP-NG portal also uses the internationalisation features offered by the JSP technology in order to implement the multilingual functionality of the ASP-NG portal.

In this paragraph we provide information about the implementation technologies of the ASP-NG portal and the Backoffice API. Initially we present the implementation technologies of the ASP-NG portal and the Backoffice API and after that we provide the implementation technologies of ASP-NG portal interfaces and the Backoffice API interfaces with the other modules of the ASP-NG system. Figure 2 presents the implementation technologies, which have been used for the implementation of the ASP-NG portal and the Backoffice API.

### 3.1.1 Implementation technologies of ASP-NG portal

The implementation of the ASP-NG portal is based on the J2EE (Java 2, Enterprise Edition) and we selected the Oracle Application Server 9i Java Edition (Oracle AS9i) ([Oracle]) as the implementation platform. Oracle AS9i is one of the most advanced and reliable J2EE compliant application servers available today at
the market. We have to mention that during the implementation of the ASP-NG portal we did not rely on any Oracle AS9i proprietary feature and we implemented a pure J2EE application. This means that the ASP-NG portal can be deployed and run on any J2EE compliant application server without (or with minimal) changes.

The ASP-NG portal consists of a J2EE web application, which runs in the Oracle AS9i. The Oracle AS9i is using as front end the Oracle HTTP server (which is a modified version of the Apache web server) and the J2EE web application runs inside the JSP/Servle container of the Oracle AS9i. The Oracle HTTP server is using as a proxy the JSP/Servle container of the Oracle AS9i. With the above approach we have the following benefits: 1) The static content (static HTML pages, graphics) is served by the Oracle HTTP server and the JSP/Servle container does not waste resources (memory, CPU) to serve the static content. 2) The Oracle HTTP is based on the well known Apache web server and as a result, it offers all the Apache security features (Apache is one of the most secure web servers) and Apache reliability. 3) Due to the fact that both Oracle HTTP server and JSP/Servle container are running “inside” the Oracle AS9i there are no delays with this approach. The Forum of the ASP-NG portal is also implemented with the use of JSP/Servle technologies and is also running in the JSP/Servle container. Like the ASP-NG portal, the Forum is using as front-end the Oracle HTTP Server.

The end user client consists of a Windows 98 or Windows NT/2000/XP personal computer with a java enabled web browser and the AoD player installed. The client of the ASP-NG portal is a java enabled web browser with JavaScript technology (for example Microsoft Internet Explorer or Netscape Navigator). The web browser accesses the HTML pages of the ASP-NG portal and forum (with the use of HTTP and HTTPS) and we use JavaScript in order to perform client side validation to web forms. The interaction between the web browser and the AoD client is based on ActiveX and Java Script technologies. In addition the end user accesses the chat feature of the ASP-NG portal with the use of a Java Applet. Figure 3 shows the ASP-NG portal.

Figure 3: ASP-NG portal screen shot

3.1.2. Implementation technologies of Backoffice API

The implementation of the Backoffice API is based on J2EE (Java 2, Enterprise Edition) and we selected the Oracle Application Server 9i Java Edition (Oracle AS9i) as the implementation platform. During the implementation of the Backoffice API module we did not rely on any Oracle AS9i proprietary features and we implemented a pure J2EE application. This means that we have for the Backoffice API the same flexibility that we also have for the ASP-NG portal: both can be deployed and run on any J2EE compliant application server without (or with minimal) changes. On the development phase, we used mostly the OC4J (Oracle Containers 4 Java) for deployment and testing of the Backoffice API for simplicity reasons. When
the versions of the Backoffice API had been tested then we deployed them on the full Oracle Application Server 9i. The difference between the OC4J and the full Oracle Application Server 9i is that the Oracle Applications Server 9i contains more modules besides the OC4J, like the HTTP Apache server for front-end, the Web Cache module etc.

The Backoffice API consists of entity and session EJBs. For the development of the Backoffice we used the JDeveloper ([Oracle]) programming environment by Oracle. The Entity beans models the system database in the Java environment and they are implemented as bean–managed EJBs, which means that all the load, store, insert methods etc. were implemented by code written by the programmer (and not automatically generated by the container). Alternatively, there is the capability to use container managed EJBs, where the EJB container has the responsibility for programming these operations. In our case, we chose the first solution in order to have the full control of the entities beans behavior. On the other hand, the session EJBs are implemented according to the Backoffice API module design and they are stateless session EJBs that expose their methods in local and remote interfaces. Generally, in the implementation of the Backoffice API we provide both local and remote interfaces, so in the future it is possible to change the communication between session and entity EJBs with minimum modifications on the source code. In addition, each entity bean corresponds to a table on the ASP-NG system database. For the communication between the entity EJBs with the tables, a JDBC connection is used. The JDBC connection can be established using JDBC driver and the JDBC API that allows accessing the database using pure Java programming language. Figure 4 show the system database.

![Figure 4: ASP-NG system database schema](image)

3.1.3. Implementation technologies of ASP-NG portal interfaces

The ASP-NG portal has the following interfaces with other modules of the ASP-NG system:

- Backoffice API: The interface between the Backoffice API EJBs and the ASP-NG portal is based on the standard EJB communication process, which is the use of Java Naming Services and RMI (Remote Method Invocation). More particularly, the application server, which hosts the Backoffice API, is using a naming service. The ASP-NG portal is using this naming service in order to obtain reference to the Backoffice API EJBs and then the ASP-NG portal can call Backoffice API calls with the use of RMI
technology. We have to mention that the ASP-NG portal and the Backoffice API can be installed to the same application server or different application servers. The communication between the ASP-NG portal and the Backoffice is based on Remote Interfaces and therefore the Backoffice EJBs and the ASP-NG portal can be running on different physical servers, in order to achieve better scalability.

- AoD module (server side): The interface between the ASP-NG portal and the AoD module is based on AoD Web and scripting Interfaces. More particularly, the web and scripting interfaces provide important AoD functionalities to the ASP-NG portal and are embedded into it. There is a function that compares the components detected on the user’s computer to the application’s requirements in the database, and a JavaBean that exports necessary functions for retrieving the directions for launching an application. There are also the following pages:
  1. Shortcut page: Redirects an application launch request from a start menu shortcut.
  2. Scheduling page: Retrieves scheduling directions to the Player on the user machine.
  3. Error Log page: Posts error events from the Player on the user machine to the client error log database table.
  4. Error Report page: Is called when the end-user sends feedback via the AoD Player. Its goal is to insert the information reported by the end-user at the error log and error report tables.

- AoD module (client side): The interface between the AoD module and the web browser is based on the AoD control, which is a web component that can run on the users computer, and can be activated from the web. The AoD control comes as an ActiveX component for use with Internet Explorer. The AoD control exports two functions. The first retrieves some information about the user’s computer and is used for example to get the installed components versions on the user’s computer (such as DirectX) and compare it to the required version by an application. The second function exported from the AoD control runs the selected application on the user’s computer. If the end user’s computer does not support the ActiveX technology, a special type of file is downloaded from the end user’s browser. This type of file, with the .rgmx extension, is recognized by the AoD player and allows the user to start the application.

- CRM module: The interface between ASP-NG portal and CRM module consists of a few web links between the ASP-NG portal and the web part of the CRM module. More particularly these web links pass to the CRM module the following information: Login of the user, Encrypted authentication information, User preferred language, User preferred look and feel.

### 3.1.4. Implementation technologies of Backoffice API interfaces

The Backoffice API has the following interfaces with other modules of the ASP-NG system:

- AoD module: The interface between the Backoffice API module and the AoD module is based on JDBC access to the AoD database. More particularly, the Backoffice module performs read-only access to the AoD database in order to retrieve information regarding the available applications on the AoD system.

- CRM module: The interface between the Backoffice API module and the CRM module consists of a few calls from the Session EJBs of the Backoffice module to the session EJBs of the CRM module. More particularly these calls are related to: User information (create new user, update user information, delete user), Application information (insert new application, update application information, delete information) and Rent application. The CRM module is implemented using J2EE technology, so the calls are made using RMI with Remote Interfaces. In addition, the CRM calls are implemented in such a way that the CRM module can be deployed in a different Application Server from the ASP-NG system.

- Mail Server: The BackOffice API uses the javax.mail and javax.mail.internet packages in order to send e-mails using an existing mail server. With these classes, the ASP-NG system creates e-mails, specifying the mail subject, sender, body and destination’s address. Those classes connect with the mail server using specific ports and then the mail server sends the corresponding emails.

- Database: The BackOffice API also interacts with the ASP-NG database as well as with the AoD database directly. The interface between the databases and the BackOffice is based on the JDBC connection polling method.

- ASP-NG portal: The interface between the ASP-NG portal and the BackOffice API is based on calls using the Remote Interfaces from the ASP-NG portal to the BackOffice API methods.

- Diagnostics module: In addition, there is an interface between the Diagnostics module and the BackOffice module, which means that the Diagnostic tools make some kind of monitoring of the server that hosts the Application Server. This interface is based on the SNMP protocol.
Chat server: Finally, the chat server was independently implemented, using a free open-source chat module and we made the appropriate changes in order to satisfy the aims and the scope of the ASP-NG portal and system. The implementation is based on the Java Programming language and a Java applet (that is the chat client) is connected directly to the chat server.

3.2. The ASP-NG Portal Database

The ASP-NG portal database supports the operation of the localization and the customization of the look and feel of the ASP-NG portal. One basic requirement of the portal database is the capability to store multilingual content, and for this reason we use UTF-8 encoding in the portal database. In order to support that service the ASP-NG portal database contains 8 tables. Figure 5 shows the portal database tables and their relationships.

When the portal administrator wants to insert a new language to the ASP-NG portal, the necessary information is inserted in the Languages table. The portal administrator can then select the page to translate for the new language. He is then presented with a form where he can translate all strings contained in the selected page. With the database tables completed (through the WWW forms that the portal administrator sees), the .properties files can be generated. A similar approach is applied in order to add new skins and create the necessary.properties files.

For the correct operation of the ASP-NG portal, during the installation of the ASP-NG system, the portal database contains (at least) all the necessary information for the operation in the default language (English) and the default skin. After the installation, the portal administrator has the capability to add new languages and new skins. Because the default language and the default skin are used as guidelines for adding new languages and new skins, it is the responsibility of the portal administrator to check that the new default skin and default language are properly installed, when he changes the default language and the default skin.

Figure 5: ASP-NG portal Database Schema

3.3. Authentication and Security Issues

The ASP-NG portal offers a built-in authentication system for registering and validating registered users. The authentication of a user is based on a login/password combination over secure HTTPS/SSL ([SSL]) connections and the users’ passwords are stored encrypted in the ASP-NG Backoffice database. With this approach even the system administrator has not the capability to read a user password (of course the system administrator has the capability to reset the user password). We have decided to base the authentication mechanism on the common approach of login/password combination and not to use more advanced mechanisms (for example user authentication with the use of certificates) for the following reasons:

- The proposed authentication system (login/password combination over secure HTTPS/SSL connections) provides a good security level, which satisfies the requirements of an e-commerce solution like the ASP-NG service.
- Most of the end-users are familiar with the use of login and passwords.
A more advanced authentication system, which needs specific technical knowledge from the user, adds overhead to the authentication mechanism and may discourage a user without the necessary technical knowledge.

In addition, the ASP-NG portal provides the capability of secure communication in the mean of secure HTTP/SSL connections. Every time the user provides personal information and passwords (for example during registration or authentication) or during renting transactions (for example rent an application) all the communication is done over secure HTTP/SSL connections. We select the approach to provide only the most sensitive parts of ASP-NG portal over secure HTTPS/SSL connections and not the whole ASP-NG portal, in order to increase the performance of the ASP-NG portal, due to the fact that the HTTPS/SSL connection includes an overhead to the overall performance of the ASP-NG portal, since it has to encrypt the whole content of all pages protected by HTTPS/SSL.

3.4. ASP-NG Portal localization Features

Today’s global Internet demands global software- that is, software that can be developed independently of the countries or languages of its users, and then localized for multiple countries or regions ([Seshadri]). The ASP-NG portal offers to its users the capability to select and customize the language of the user interface in order to present information in their language. The implementation of the ASP-NG portal multilingual user interface is based on the internationalization features of the J2EE platform. The J2EE platform provides a rich set of APIs for developing internationalized WWW portals like the Locales (class java.util.locale) and Resource Bundles (class java.util.ResourceBundle).

More specifically, for the implementation of the ASP-NG portal multilingual user interface we used locales for identifying the user language preference either indirectly by the user browser setting or directly by asking him for his language preferences. The only information we are interested in for each locale is the locale’s language. The information for each different language of the ASP-NG portal are stored in the portal database (see section 3.2 for more information) and every time an interface item (for example a string) changes or every time the portal administrator decides, the appropriate resources bundle files (.properties files) are created based on the portal database information. These resources bundle files are loaded to the application server memory during ASP-NG application server start up. With the above approach - building the resources bundle files on demand and not extracting the information of the portal database during the operation of the ASP-NG portal - we improve the performance of the ASP-NG portal. This is because when a user accesses the ASP-NG portal the interface information is stored in the server memory and the ASP-NG portal does not have to retrieve user interface information from the database, which may cause delays. In addition, the fact that the localization information is not changed very often leads us to the above-described approach.

3.5 ASP-NG Customized Look and Feel

ASP-NG portal offers to the portal administrator the capability to customize the look and feel of the ASP-NG portal. With the term “customize” we mean the capability that the portal administrator has to perform the following:

- Changing (create, edit, remove) the stylesheet (.css file), which controls the look and feel (for example fonts and colour of each element) of the ASP-NG portal.
- Changing (create, replace, remove) the graphic files of the ASP-NG portal with others with the same dimension.
- Changing the strings of the ASP-NG portal (this capability is provided through the localization features of the ASP-NG portal).

The realization of the above features of the ASP-NG portal is based on the concept of Skins ([Heumann]). Skins are a common approach to provide customized look and feel not only to WWW portals but also to other kinds of applications. With the term “Skin” we refer to a collection of interfaces objects (for example graphics), which can be used to change the user graphic interface. In the case of ASP-NG portal the skins consist of the portal stylesheet and the portal graphics.
4. CONCLUSION – FUTURE WORK

The J2EE architecture is based on a very flexible and well-structured technology that can be used in order to implement large-scale portals and web applications. Some of its most important advantages are the separation of the business logic from the data representation and the performance benefits from the object oriented programming. In particular, in case that the objective is the implementation of a demanding web portal with much content and many expected visitors, then the J2EE technology becomes a very serious contender. The J2EE technology consists of various independent components (EJBs, Servlets, JSP pages), which all together produce a powerful package for web development. Today, J2EE is supported by a large and impressive list of vendors and for most of its components there are widely approved standards. Interest in this technology is increasing and despite the emergence of new competitors like Microsoft’s .NET architecture, this trend is expected to continue in the visible future. However, there are some points that need systematic work to be finalized, as there are also rough edges and bugs in many implementations that need to be improved.

In this paper we presented the design and the implementation of the ASP-NG portal, which constitutes a web interface in order to use the Application on Demand (AoD) service. The design of the ASP-NG portal is based on current state of the art J2EE technologies and offers enhanced capabilities like multilingual interface and customized look and feel. The implementation of the ASP-NG portal offers enhanced performance and scalability based on J2EE technologies, separates the presentation and the business logic and allowing these two entities to operate independently.

Our future work includes the evaluation of the ASP-NG portal in the context of trials, which are going to take place in the context of the ASP-NG system. During these trials, the performance and efficiency of the platform and the implemented application are going to be measured. We expect that several hundred actual users are going to access the ASP-NG portal and make use of the Application on Demand service.

ACKNOWLEDGEMENT

We want to thank the European Commission (EC) for supporting this work in the context of ASP-NG IST project (Contract number: IST-2001-35354). In addition, we want to thank our partners in the ASP-NG project for their valuable cooperation. More information regarding ASP-NG project can be found in http://www.asp-ng.eu.org.

REFERENCES


