EGI-InSPIRE

APPLICATIONS DATABASE ACTIVITIES WORK PLAN (MAY-OCTOBER 2011)

(TNA3.4)

Document identifier:	**AppDBWorkplanY2A-v9**
Date:	**09/06/2011**
Activity:	**NA3**
Lead Partner:	**IASA/GRNET**
Document Status:	**FINAL**
Dissemination Level:	**PUBLIC**
Document Link:	**https://documents.egi.eu/document/510**

Abstract

This document presents the objectives, responsibilities and work plan of the "EGI Applications Database" subtask of the "TNA3.4 Technical Services" task of the EGI-InSPIRE project. It starts with a summary of the activities accomplished within the subtask in the second six months (1-Nov-2010 to 30-Apr-2011) of the project, followed by a work plan for the 1-May-2011 to 31-Oct-2011 six month long period. The subtask is responsible for the development and operation of the EGI Applications Database (AppDB) that is available at http://appdb.egi.eu.



I. COPYRIGHT NOTICE

II. DOCUMENT LOG

Issue	Date	Comment	Author/Partner
1	28/04/2011	First draft	W. V. Karageorgos / IASA,GRNET M. Chatziangellou / IASA
2	06/05/2011	Second draft	W. V. Karageorgos / IASA,GRNET N. Zarife / IASA N. Theofilopoulos / IASA
3	07/05/2011	First complete version	W. V. Karageorgos / IASA,GRNET A. Nakos / IASA,GRNET N. Zarife / IASA N. Theofilopoulos / IASA
4	07/05/2011	Review of first version	M. Chatziangellou / IASA A. Nakos / IASA,GRNET
5	08/06/2011	Finalisation	G. Sipos / EGI.eu

III.

IV. DOCUMENT AMENDMENT PROCEDURE

Amendments, comments and suggestions should be sent to the authors. The procedures documented in the EGI-InSPIRE "Document Management Procedure" will be followed: https://wiki.egi.eu/wiki/Procedures

V. TERMINOLOGY

A complete project glossary is provided at the following page: http://www.egi.eu/about/glossary/.

VI. PROJECT SUMMARY

To support science and innovation, a lasting operational model for e-Science is needed – both for coordinating the infrastructure and for delivering integrated services that cross national borders.

The EGI-InSPIRE project will support the transition from a project-based system to a sustainable pan-European e-Infrastructure, by supporting 'grids' of high-performance computing (HPC) and high-throughput computing (HTC) resources. EGI-InSPIRE will also be ideally placed to integrate new Distributed Computing Infrastructures (DCIs) such as clouds, supercomputing networks and desktop grids, to benefit user communities within the European Research Area.

EGI-InSPIRE will collect user requirements and provide support for the current and potential new user communities, for example within the ESFRI projects. Additional support will also be given to the current heavy users of the infrastructure, such as high energy physics, computational chemistry and life sciences, as they move their critical services and tools from a centralised support model to one driven by their own individual communities.

The objectives of the project are:

1. The continued operation and expansion of today's production infrastructure by transitioning to a governance model and operational infrastructure that can be increasingly sustained outside of specific project funding.
2. The continued support of researchers within Europe and their international collaborators that are using the current production infrastructure.
3. The support for current heavy users of the infrastructure in earth science, astronomy and astrophysics, fusion, computational chemistry and materials science technology, life sciences and high energy physics as they move to sustainable support models for their own communities.
4. Interfaces that expand access to new user communities including new potential heavy users of the infrastructure from the ESFRI projects.
5. Mechanisms to integrate existing infrastructure providers in Europe and around the world into the production infrastructure, so as to provide transparent access to all authorised users.
6. Establish processes and procedures to allow the integration of new DCI technologies (e.g. clouds, volunteer desktop grids) and heterogeneous resources (e.g. HTC and HPC) into a seamless production infrastructure as they mature and demonstrate value to the EGI community.

The EGI community is a federation of independent national and community resource providers, whose resources support specific research communities and international collaborators both within Europe and worldwide. EGI.eu, coordinator of EGI-InSPIRE, brings together partner institutions established within the community to provide a set of essential human and technical services that enable secure integrated access to distributed resources on behalf of the community.

The production infrastructure supports Virtual Research Communities (VRCs) − structured international user communities − that are grouped into specific research domains. VRCs are formally represented within EGI at both a technical and strategic level.

VII. EXECUTIVE SUMMARY

This document presents the objectives, responsibilities and work plan of the "EGI Applications Database" subtask of the "TNA3.4 Technical Services" task of the EGI-InSPIRE project. AppDB is the descendant of the EGEE Applications Registry [R 2] portal, which was initially developed by the IASA regional coordination team during the course of the EGEE-III project. It provides a catalogue of applications that have been ported, or are being ported, within the infrastructure [R 1] and provides information on reusable application developer tools and application developer profiles. As such it enables new communities to discover and reuse EGI applications and tools, and to find experts with relevant expertise. By the reuse of ported applications and/or reusable tools one of the main barriers of grid adoption is eliminated.

During the second six months of the project, about 86% of the work items from the previous work plan [R 3] have been completed. The unaccomplished tasks are still work-in-progress due to issues involving third parties. Moreover, additional effort has been invested in work items which arose during this period from new requirements capturing, mainly through the dedicated AppDB support unit in the EGI Helpdesk and through the EGI Requirement Tracking (RT) system.

Tasks within the new work plan fall under the key areas of Quality of Information, Information Retrieval, Dissemination, Compatibility, and Architecture.

TABLE OF CONTENTS

1 INTRODUCTION

The EGI Applications Database (AppDB) is developed and provided in the context of the "TNA3.4 Technical Services" task of the EGI-InSPIRE project. The AppDB stores tailor-made computing tools for scientists and NGI User Support Teams to use and stores contact information about the developers of these tools. The applications and tools filed in AppDB are finished products, ready to be used on the European Grid Infrastructure. By storing pre-made applications and tools, the AppDB aims to

- alleviate the need for scientists and NGI User Support Teams to spend time developing their own software
- avoid duplication of effort across the EGI community

This document provides a summary of achievements from the past six-month period (November 2010 – April 2011) driven by the previous work plan [R 3], and describes a new work plan for the next six-month period (1-May-2011 to 31-Oct-2011). Development effort will be focused on the following key areas:

- Quality of information: providing mechanisms by which the community can provide feedback on the stored items, identify, correct and/or revoke problematic application and tool profiles.
- Information retrieval: Improved search and browse capabilities based on tagging and new database structure.
- Notification / Dissemination: Implementation and integration of notification mechanisms (email and RSS feeds) into the system, so community members can register for and receive notifications about changes in AppDB content.
- Cross-browser compatibility: Study the compatibility problems that exist in the current system with Internet Explorer 9 and work towards a fully compatible version.
- EGI User Support Platform: Define additional AppDB gadgets for communities to integrate custom AppDB views into their own portals. Work towards the implementation of these gadgets and study the integration of multiple gadgets in an "EGI User Support Platform".
- Architecture: Speed up and scale up the AppDB by migrating it to PostgreSQL database and provide support for new types of clients and client devices with a restructured codebase.

2 ACTIVITY SUMMARY

The activities of the past six months were mainly focused on service integration. As such, effort was allocated towards interfacing the AppDB with the EGI Operations Portal and the GOCDB on one hand, and on the other, on providing a web-API for other services to interface with the AppDB. A web-gadget was also provided as a proof-of-concept of what can be done through the said web-API. The rest of the activities mainly concerned new features or enhancements to the GUI, and addressing bugs and requests from the EGI Helpdesk support unit and the EGI RT system respectively.

2.1 Service integration

Integration with the EGI Operations Portal [R 4] has been successfully completed, thus providing VO data through the AppDB, with respect to application and tool entries. Data is retrieved via an XML export facility of the Operations Portal, and is cached on the AppDB server. If more than one hour stale at the time of a relevant user request, then the data is refreshed. Relevant server-side Active Record objects have been added to the object model, in order to manipulate the VO data and index references in the AppDB Relational Database Management System (RDBMS). It should be mentioned that the Operations Portal development team is working on a SOAP web-service based on the Lavoisier framework [R 6], in order to deliver VO and NGI data. At the time of integration, this service was not readily available, but switching to it, instead of using the XML export functionality is to be evaluated at a later point, after the service has been successfully deployed.

Similar effort has been allocated to NGI data integration through the GOCDB PI [R 7], meaning that the relevant RBMS structures and server-side Active Record objects have been prepared. Nevertheless, integration has been stalled, due to the fact that the GOCDB stores obsolete EGEE data (e.g. ROC entries) along with new EGI data (NGI entries), without providing any kind of discrimination. Moreover, country relations to NGIs are not provided either, thus making the mapping to the AppDB data model impossible without ad-hoc data injections. The GOCDB development team has been contacted regarding these issues, but has expressed no intent of implementing the needed modifications in the near future, due to excess workload [R 13]. The alternative of using the Operations Portal Lavoisier web-service instead will be evaluated once the service is available, in cases it better matches the AppDB data model requirements.

Furthermore, large effort has been allocated towards providing and properly documenting a prototype RESTful web-API, so that other services or portals in general may interface with the AppDB. The said API provides unauthenticated read-only data, in detail about applications and tools, related publications, and researcher profiles, as well as in reference about VOs, middleware, countries, and (sub)disciplines. Documentation about the API can be found in the EGI NA3 wiki pages [R 8], and XML schemata are provided from the dedicated FQDN which provides the API itself [R 9]. Currently, the API is in beta status- until sufficient feedback about its functionality has been gathered from the community - and plans about extending it with authenticated read/write access are being investigated.

Finally, a web-gadget (portlet) has been developed, as a proof-of-concept, of what can be achieved through the web-API. This web-gadget is already in use by various regionalized sites [R 10], and an on-line editor is available through the AppDB site [R 11]. Through this web-gadget, third parties can provide on-line embedded detailed information to filtered entries about applications, pertaining for example to a certain VO, or country. The possibility of providing more gadgets, or extending the existing one with more functionality, such as write-access -- provided the API is also thus extended -- is scheduled to be investigated in the next six months.

2.2 Features and enhancements

The rest of the terms effort, development-wise, has been mainly attributed to work related to new features and enhancements of the portal. More than 15 GGUS and RT tickets have been addressed, along with the issues that were communicated over e-mail. Examples of such actions include a preliminary reporting tool for managers, major speed improvements of the portal, support for non-UMD middlewares, enhancements to the data export functionality, addition of more details in application and people entries, and overall usability of the portal's user interface. Of such actions that were defined in the previous work plan, the one pertaining to IE8 compatibility is still work-in-progress, since the constant addition and modification of features greatly impaired the ability to thoroughly test. With the public release of IE9, new tests have been planned, and compatibility is expected to improve.

3 WORK PLAN

As it has been agreed within TNA3.4, a new release is expected every six months from each of the ”Technical Services” of TNA3.4 (Ideally one release before the EGI User Forum and one before the EGI Technical Forum). New releases must be prepared according to development plans endorsed by the UCB, based on the requirements collected from user communities. A summary of the work plan for the next 6 months (26 weeks) is presented in Table 1.

One issue that has to be mentioned is that in Table 1, items S13 & S14 (in red - also marked with a dagger) are sprints that our team will try to accomplish in the current six months work-cycle, by involving more than one engineer to work in parallel with the first person who works on S1-S12. But this cannot be guaranteed, as the participation of the second engineer could be realized only if our involvement in other WPs allow this. In this way S13 & S14 could be finished within the next 6 months as well, despite of the fact that the overall effort required to follow the extended work plan, exceeds the upcoming six months timeframe by 8 weeks (given one person is allocated at any given time).

Sprint	Task	Duration	
		Weeks	Persons
Architecture			
S1	Migrate DBMS to PostgreSQL	4	1
S2	Audit and re-factor existing codebase	4	1
Information Retrieval			
S3	Re-implement searching/filtering mechanism	3	1
S4	Applications entry tagging mechanism	2	1
Quality of Information			
S5	User comment/ranking system	3	1
S6	Entry problem reporting system	1.5	1
S7	Broken link detection notification system	1.5	1
S8	Application revocation mechanism (only for Managers)	2	1
S9	Detect and promote or remove non-finished apps based on status	1	1
S10	Development of mechanism to ensure future application/tool name uniqueness	1	1
Dissemination			
S11	Notification services (email/RSS)	2	1
Cross-browser Compatibility			
S12	Investigation and estimation of Internet Explorer 9 (IE9) compatibility	1	1

EGI User Support Platform			
S13	Develop new/extend existing gadget with write access to the AppDB	4†	1
S14	Provide write-access through API	4†	1
	Total	**26 (34†)**	

Table 1 AppDB six-month work plan

Ongoing Effort

Apart from the items mentioned in the above work plan, there should also be mention about the effort that will be allocated to tasks such as bug-fixing, support for new requirements from the users, and maintenance of the project in general. This effort is estimated to be c. 25% of the term, i.e. about 6.5 person-weeks, and will take place in parallel to the rest of the work-items, by allocating more than one persons in average.

3.1 Architecture

3.1.1 S1: Migrate DBMS to PostgreSQL

The existing RDBMS of choice, namely MySQL with an InnoDB engine, suffers from well-know limits with regards to scalability, performance, and query language expressiveness. During the first year of the project, where design and development was done through RAD processes, this was a good choice due to the simplicity of set-up, portability, and small learning curve, since most professionals are already familiar with it. Nevertheless, as the projects matures, and requirements rise, this choice fails to meet the augmenting demands for power.

On the other hand, a more advanced, yet open RDBMS such as PostgreSQL, now seems to be more suitable for the project's needs. Its superior query language expressiveness, paired with better performance and advanced features can deliver greater data transformation abilities inside the data tier. This automatically translates to better optimized code within the existing codebase architecture, and thus a better overall performance in general.

In order to migrate to the new RDBMS, the server-side Active Record object model must be adapted to account for SQL discrepancies between the two RDBMS's, despite the framework's abstraction layer and use of the same data model. Moreover, in-house code such as stored procedures, views, triggers, etc. will have to be re-written to a great extent, since said discrepancies are great, with the de facto lack of an abstraction layer. Finally, members of the project that are not familiar with PostgreSQL, will have to face a steep learning curve. All of the above lead to a much optimistic 4 week sprint for the migration process.

3.1.2 S2: Audit and re-factor existing codebase

Currently the project is based on a tree-tier client-server architecture, where web content is delivered to the presentation tier through a server-side Model-View-Controller (MVC) design. In this scheme, controllers in the business tier handle user requests by asking for data from objects of the

Active Record model in the data tier , which they then pass on to the views in the presentation tier for rendering. Rendering is mainly done through server-side code (PHP), with minor post-presentation client-side handling through Javascript, mainly due to the extensive use of AJAX (Image 1).

This combination of architecture and design patterns has many well-know benefits, such as good modularity. For instance, migrating to a much different RDBMS, as described in the previous paragraph, would be rather unfeasible without it. It also serves very well in terms of abstraction in the presentation tier, providing for ease in rendering changes, yet it can only serve one kind of client type transparently -- in this case, the portal's GUI.

This has worked well so far, however, with the introduction of the web-gadget, and much talk about more gadgets, or ever iOS clients, such as mobile phones or tablets, this design suffers from duplication of effort in the presentation tier. In plain words, each client type must provide most of the rendering code from scratch. However, introducing a second, client-side MVC design inside the presentation tier would provide the framework to integrate all client types into a unified rendering codebase. This is readily feasible by exploiting the newly deployed RESTful API. With the proposed scheme, a user request is again handled by a controller, who this time simply provides the view with a template setup with little or no calls to any model. The view then renders the content asynchronously in the client through AJAX, by calling the API instead, which in a second iteration performs the full server-side MVC cycle to deliver the relevant data (Image 2).

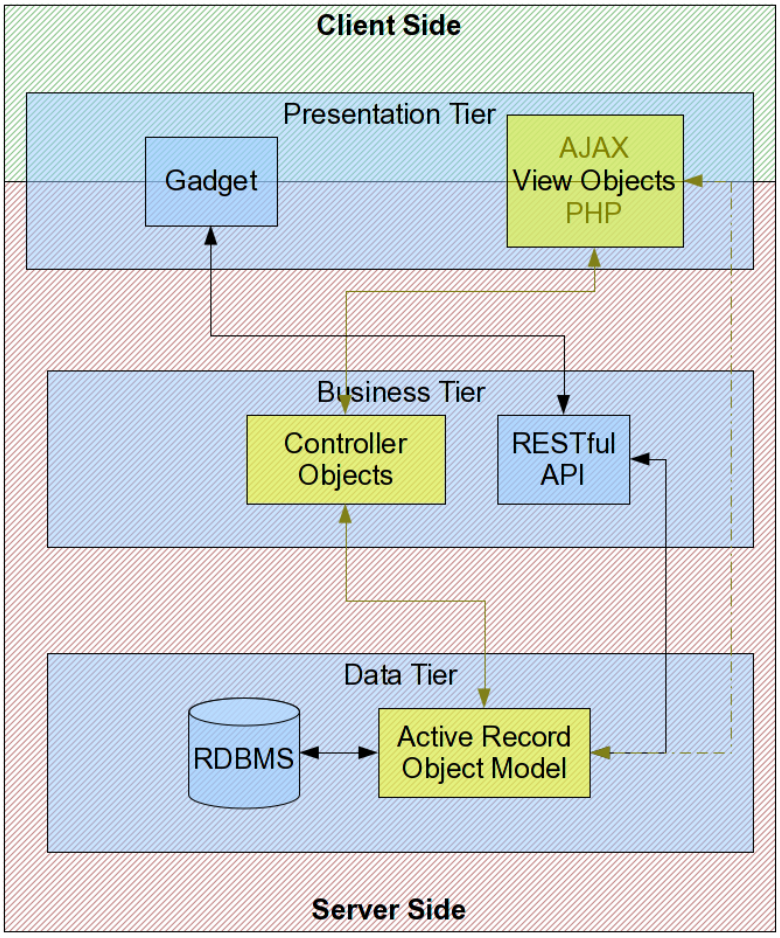

Image 1: Existing architecture. The server-side MVC design can be seen in yellow.

Such a re-factoring will require almost completely rewriting the presentation tier code, translating to major workload. Yet, the advantage is obvious, since time spent will be earned back in the future, by centrally coding, maintaining, and debugging presentation logic for all client types.

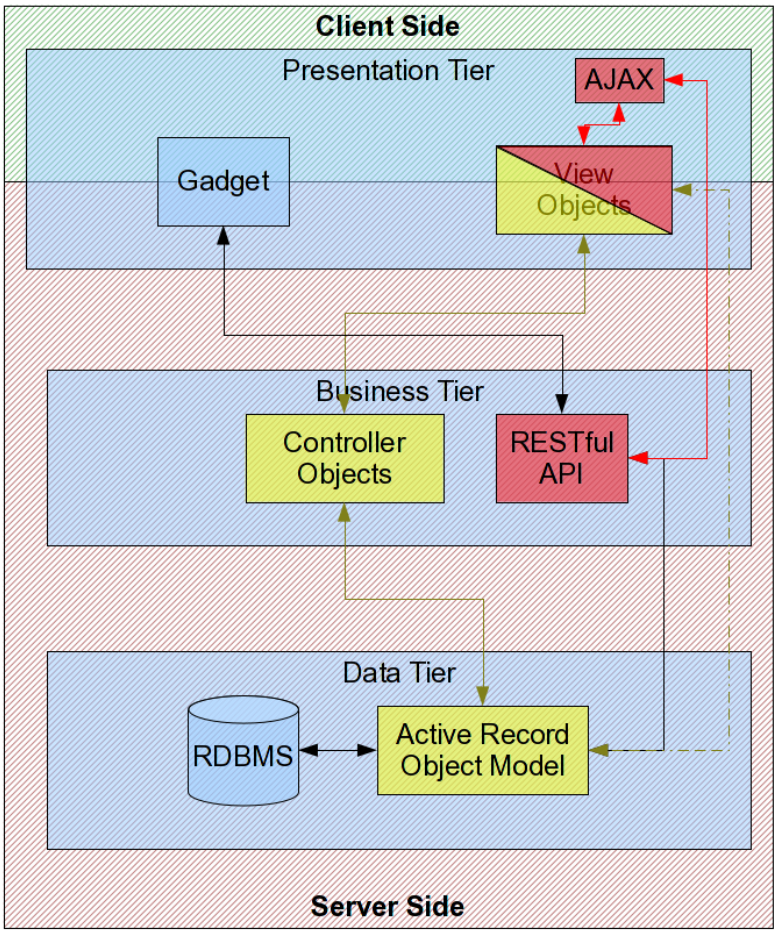

Image 2: Proposed architecture. The server-side MVC design can be seen in yellow, whereas the new client-side MVC can be seen in red.

3.2 Information Retrieval

3.2.1 S3: Re-implement searching/filtering mechanism

In its current state, data searching and filtering through the portal is limited to one entry type at a time (e.g. applications XOR people XOR VOs, etc.), and provides for no nesting nor field-wide search for a given entity (e.g. search for the string "test" in either name or description). The reasons for this are due to the fact that at the time the filtering mechanism was designed, there weren't many entity types, and also to the fact that data transformation in the data tier was limited, given the RDBMS of choice, if one makes the sensible decision of constricting oneself to viable queries with regards to computational time.

The migration to PostgreSQL is expected to alleviate computational time for such complex queries, and allow for the design of an extended filtering mechanism, which will meet the criteria described above.

3.2.2 S4: Applications Entry Tagging Mechanism

As information stored in the system rapidly grows larger in volume, finding what one seeks becomes a non-trivial task. Even though such a task may greatly benefit from a powerful filtering mechanism, giving the ability to the user to tag existing information with custom metadata can help fill in the gaps and provide for better searching, especially since such a feature can be carried out by people, for other people who search for the same type of information to benefit from. We plan on providing a set of predefined tags attributed automatically by the system to certain entry types (e.g. in order to distinguish between native entries, and entries imported by other similar systems), as well as free-string tags which can be attributed by the users on demand. Tags can also facilitate the integration of information from the AppDB with information from other sources – relation between information segments can be established through tags.

3.3 Quality of Information

3.3.1 S5: User Comment/Ranking System

Quality of information can be a sensible subject in large data stores. One of the actions planned in order to help ensure content quality is the provision of quality related metrics, such as comments and rankings. These metrics should be provided by authenticated end users through the course of time, and should be visible by everyone. Nevertheless, UCST members (managers) may need to have stronger rights upon them (e.g. adding special flags such as “hidden” or “to be improved”).

3.3.2 S6: Entry Problem Reporting System

There have been several cases during the past six months, where a certain application entry raised disputes concerning the “ownership” of the referenced application, or the use of data, such as logos or names, which belonged to other projects. Moreover, given the fact that it is fairly easy to obtain an EGI SSO, it is not beyond the realm of possibilities that we be faced with malevolent entries, of the “spam” type. Therefore, users will be provided with the ability to “report” a certain application entry, providing a short text about the type of infringement, so that appropriate measures may be taken on by the UCST, or the submitter of the entry. Both UCST and the entry owner will be notified in e-mail about such reports.

3.3.3 S7: Broken Link Detection Notification System

One of the requests made by the UCST pertaining to Quality of Information, is the detection of application entries with broken links in their URLs section. Such entries should be corrected by the submitter's team (who should be notified via e-mail), or be considered for deletion by the UCST, in case of no response withing a sensible time frame. To this purpose, a web spider will be developed and deployed, which will periodically scan URLs and perform the appropriate actions.

3.3.4 S8: Application Revocation Mechanism

As a catch-all case for application entries that present some kind of problem not identified by any of the above cases, a mechanism for temporal removal of said entries from class of presentable information should exist. This means that in such cases, said entries should not appear in index pages, nor search results, not should they be accounted for by statistics and export functionalities, etc.). Therefore, multiple modifications should be made in all code that relates to fetching application entries in the data tier.

3.3.5 S9: Detect and promote or remove non-finished apps based on status

During the EGEE era, application entries stored in the AppDB featured a status property, which represented their development status as fas as gridification was concerned. However, since the EGI era, it has been deemed that only completed applications (i.e. fully gridified and in production) should exist in the database. Therefore, the status field is planned to be removed, and a list of application entries that are not marked as completed should be compiled and forwarded to the UCST, in order to contact the respected development teams, and either promote their status, or remove them from the database.

3.3.6 S10: Development of mechanism to ensure future application/tool name uniqueness

A recurring issue which has been raised during the past term is related to the naming of application entries. There have been several cases where users did not check for the existence of an entry for a particular application, and thus re-added it. Moreover, there have been other cases where users created entries referring to new applications, which nevertheless used the same name with an existing yet different application, or with a fork thereof. While the later cases are “legitimate” for all matters, they do seem to create turmoil and disorientation. Therefore, in order to avoid future problems of this nature, it has been decided that a mechanism will be implemented which will impose name uniqueness to the new entries. Existing entries sharing the same name are few, and a comprehensive list should be compiled and forwarded to the UCST in order to separated them. Once uniqueness is imposed, entries that would otherwise use the same name, could instead be deployed with a diacritic suffix. Users registering new entries will be thus prompted in such cases.

3.4 Dissemination

3.4.1 S11: Notification Services (email/RSS)

Apart from information retrieval on-site, delivery of selected information off-side is equally important. Therefore, subscription services are to be developed, providing mailing lists and RSS feeds to the end users, based on criteria of their choice. Furthermore, separate privileged lists should exist in order to notify the UCST or NGI representatives about certain events such as new entry additions, role verification requests, etc.. Such privileged lists could also be used to inform the UCST about event concerning action falling under § (Quality of Information)

3.5 Cross-browser Compatibility

3.5.1 S12: Investigation and estimation of Internet Explorer 9 (IE9) compatibility

Microsoft Internet Explorer 8.0 compatibility is still a work-in-progress, mainly due to the fact that the constant addition of new features during the past term greatly impaired out ability to thoroughly test. The recent release of MSIE 9.0 means that effort should now be focused on this new version, which might prove to be more compatible with fewer modifications to the codebase. Moreover, compatibility of mobile browsers, such as the native Android browser, Mozilla Fennec 4.0, etc. should also be considered, time permitting, especially under the context of § (Audit and re-factor existing codebase).

3.6 EGI User Support Platform

3.6.1 S13: Develop new/extend existing gadget with write access to the AppDB

As with the case of read-only access, write-access could also be provided via gadgets -- as an out-of-the-box solution -- provided that the API is extended first. This could be done by either extending the existing gadget, or by providing a different dedicated gadget.

Depending on input from other NA3 partners, regarding their respective services, data exchange schemes between gadgets might be considered, towards forming a common framework for the EGI User Support Platform. The goal of related information exchange between partners would be an internal document, describing technical issues pertaining to the development of the gadgets. This document should finally be discussed within TNA3.4, to ensure a common roadmap.

3.6.2 S14: Provide write-access through API

One prospective feature that has been much discussed, both internally (NA3) and otherwise, is the provision of write-access though the web-API. This has been requested for cases where

- another site wants to maintain its own database, yet integrate with the AppDB, by presenting our entries, and synchronizing its entries with the AppDB
- another site wants to provide a customized view of the AppDB contents, without maintaining a separate database of its own, yet procure its users with the ability to create/modify entries on-site (i.e. without redirecting to the AppDB)
- another site wants to permanently migrate/hand over its data over the AppDB.
- variations and combinations of one or more of the above

All of the above cases obviously require some kind of write-access to the AppDB data tier, and even though alternatives to a write-enabled API have been investigated (e.g. migration tools, redirection workflows, peer pull-pull access, etc) -- each with their own pros and cons -- the verdict weights towards the former. This has been made much clear during the EGI User Forum 2011, held in Vilnius, Lithuania, where questions made by the users after the AppDB presentation mainly regarded this issue. Moreover, discussions made afterwards, with representatives from initiatives such as GISELA [R 12] and IGI [R 13] for instance, also favoured the write-enabled API.

Our team, therefore, will try to allocate effort into extending the existing RESTful API prototype with write-access, time permitting, according to Table 1. If that is not possible, then this work-item will be transferred to the next term (Y2B). Important Issues that are to be considered while redesigning the prototype include

- security implementation: authorization may be based on existing SSO design, including non-personal (i.e. site-wide) accounts, or extended with certificated features. Delegation of credentials should also be addressed, when operations are non-personal.
- synchronization: the case of interacting sites, which maintain their own database as well, will require a synchronization scheme/protocol, in order to avoid collisions and loss of data. This issue would most probably be address through an internal technical document proposing the scheme, for prospective partners to agree upon.

Nevertheless, the realization of this sprint is strongly dependant upon the outcome of S13, meaning that it will only be implemented if it has been agreed upon by TNA3.4 that it is required in order to complete S13, since - up to this date - there exist no formal requirements for all other prospective cases, such as an RT ticket.

4 CONCLUSION

The AppDB subtask has used the second six-month term of the project to:

- Provide a read-only RESTful web-API
- Integrate with the Operations Portal to provide VO data
- Provide a web gadget for other sites to integrate with the AppDB
- Implement various improvements requested by users

The plan for the next six-month term includes:

- Quality of information: providing mechanisms by which the community can provide feedback on the stored items,identify, correct and/or revoke problematic application and tool profiles.
- Information retrieval: Improved search and browse capabilities based on tagging and new database structure.
- Notification / Dissemination: Implementation and integration of notification mechanisms (email and RSS feed) into the system, so community members can register for and receive notifications about changes in AppDB content.
- Cross-browser compatibility: Study the compatibility problems that exist in the current system with Internet Explorer 9 and work towards a fully compatible version.
- EGI User Support Platform: Define additional AppDB gadgets for communities to integrate custom AppDB views into their own portals. Work towards the implementation of these gadgets and study the integration of multiple gadgets in an “EGI User Support Platform”.
- Architecture: Speed up and scale up the AppDB by migrating it to PostgreSQL database and provide support for new types of clients and client devices with a restructured codebase.

5 REFERENCES

R 1	http://appdb.egi.eu
R 2	https://na4rs.marie.hellasgrid.gr/
R 3	https://documents.egi.eu/document/276
R 4	http://operations-portal.egi.eu/
R 5	http://grid.in2p3.fr/lavoisier/
R 6	https://wiki.egi.eu/wiki/GOCDB/PI/Technical_Documentation
R 7	https://wiki.egi.eu/wiki/TNA3.4_Technical_Services
R 8	http://appdb-pi.egi.eu
R 9	https://wiki.egi.eu/wiki/AppDB_Gadget_Editor#Success_Stories
R 10	http://appdb.egi.eu/gadgets/editor
R 11	http://www.gisela-grid.eu/
R 12	http://www.italiangrid.org/communities/applications/porting/igi_application_database
R 13	https://rt.egi.eu/rt/Ticket/Display.html?id=1016