Deliverable No. 8.1.2

Design and prototype implementation of the p-medicine portal

Grant Agreement No.:	270089
Deliverable No.:	D8.1.2
Deliverable Name:	Design and prototype implementation of the p-medicine portal
Contractual Submission Date:	31/07/2012
Actual Submission Date:	31/07/2012

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	***p-medicine***
Project Full Name:	From data sharing and integration via VPH models to personalized medicine
Deliverable No.:	D 8.1.2
Document name:	Design and prototype implementation of the p-medicine portal
Nature (R, P, D, O)[1]	R
Dissemination Level (PU, PP, RE, CO)[2]	PU
Version:	1.00
Actual Submission Date:	31/07/2012
Editor: Institution: E-Mail:	Fatima Schera Fraunhofer IBMT fatima.schera@ibmt.fraunhofer.de

ABSTRACT:

This deliverable describes the development of the *p-medicine* portal which provides *p-medicine* users an entry point to collaborate, share data and expertise and use tools and services to improve personalized treatments of patients. The document includes an analysis of the user requirements for the portal and the functionality of the portal as well as a definition of the portal user groups and their roles in the portal. Furthermore, the development process of the *p-medicine* portal based on the Liferay framework is introduced. The approaches for the development and integration including examples as well as the way how to integrate particular p-*medicine* tools and services are described. Finally, the current state of the *p-medicine* portal including several integrated *p-medicine* tools and services is given.

KEYWORD LIST: portal, enterprise portal framework, Liferay, portlet, plugin, integration, deployment, p-medicine tools and services, data mining, ObTiMA, layout, community, user groups, role, template

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 270089.

[1] **R**=Report, **P**=Prototype, **D**=Demonstrator, **O**=Other

[2] **PU**=Public, **PP**=Restricted to other programme participants (including the Commission Services), **RE**=Restricted to a group specified by the consortium (including the Commission Services), **CO**=Confidential, only for members of the consortium (including the Commission Services)

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	10/05/2012	First draft	Fatima Schera
0.2	27/06/2012	Improved draft	Gabriele Weiler
0.3	28/06/2012	Extended and improved	Fatima Schera
0.4	02/07/2012	Extended and improved	E. Neri, D. Szejnfeld, U. Schwarz, A. Anguita, S. Sfakianakis, H. Kondylakis, G. Zacharioudakis, S. Kiefer, F. Schera
0.5	16/07/2012	Extended and improved	E. Neri, A. Maghsoodi, F. Schera
0.6	17/07/2012	Improved	F. Schera, G. Weiler
1.0	24/07/2012	Improved	H. Stenzhorn, F. Schera

List of contributors

- Fatima Schera, Fraunhofer IBMT
- Gabriele Weiler, Fraunhofer IBMT
- Stephan Kiefer, Fraunhofer IBMT
- Elias Neri, CUSTODIX
- Dawid Szejnfeld, PSNC
- Ulf Schwarz, IFOMIS
- Alberto Anguita, UPM
- Stelios Sfakianakis, FORTH
- Giorgos Zacharioudakis, FORTH
- Haridimos Kondylakis, FORTH
- Benjamin Jefferys (UCL)
- Georgios S. Stamatakos (NTUA)
- Aisan Maghsoodi (PHILLIPS)
- Holger Stenzhorn (UdS)

Contents

1 Executive Summary

The *p-medicine* portal shall provide clinicians, patients and researchers a platform to collaborate, share data and expertise, and use tools and services to improve personalized treatments of patients. The aim of this document is to explain why we need a portal solution for the *p-medicine* project and to describe how the portal for *p-medicine* has been developed and how the users involved into the project can use the portal. The document analyses user requirements for the portal and defines the portal user groups and their roles in the portal. Furthermore, several technical solutions that seem appropriate to fulfil the complex user requirements for the *p-medicine* portal are evaluated and the decision for using the Liferay framework for the *p-medicine project* is explained. We also provide information about the *p-medicine* services and tools that will be directly or indirectly integrated into the portal as well as the community based architecture of the portal. In addition, we describe how the *p-medicine* security framework is implemented into the portal. Finally, we present the current state of the portal with the tools and services that are already accessible through it and how they can be used.

2 Introduction

p-medicine aims at developing new tools, an IT infrastructure and VPH models to accelerate personalized medicine for the benefit of the patient. Therefore a platform that shall support clinicians in the decision making process and more individualized treatments of their patients based on services, tools and models will be developed in *p-medicine*.

For the *p-medicine* platform it is crucial to provide a user-friendly common access point to the tools and services of the integrated platform to be realized as a portal solution.

In this document we explain why a portal solution is needed for the *p-medicine* project and describe a portal framework including its architecture and components. Furthermore, we define the main requirements for the *p-medicine* portal and analyse which user groups will use the portal. Based on the user requirements for the *p-medicine* portal we evaluate and compare several portal solutions and explain why we have decided to use the Liferay framework for the *p-medicine* project. We also show how we have designed the *p-medicine* portal and describe how to integrate the *p-medicine* tools and services developed for the portal. Additionally, we show some tools and services that have already been integrated into the portal and describe how the users of the *p-medicine* portal can use them.

The deliverable contains the following chapters:

- Chapter 3: **Requirements for the *p-medicine* portal**

  This chapter analyses the requirements for the *p-medicine* portal, identifies the user groups of the *p-medicine* portal and their roles as well as the functionality of the tools and services that will be provided for *p-medicine* users via the portal.

- Chapter 4: **Evaluation of existing portal frameworks**

  This chapter contains important technical details about portals such as architecture, components and standards used for portals. We also investigate some available technical solutions for the development of a portal. Finally, we give some reasons for our decision to use the Liferay framework as the *p-medicine* portal.

- Chapter 5: **Using the Liferay framework for the *p-medicine* project**

  In the first section we describe the functionality of the Liferay[3] Portal framework used for the *p-medicine* portal. In the next two sections the approach for the integration of the *p-medicine* tools and services as well as some details for the development of these tools as plugins for the portal are provided. In the following, we describe the community-based structure of the portal. In addition, we show details about the integration of the *p-medicine* tools and services into the portal. This section contains also information about the design and layout of the portal. Finally, we introduce the main components of the user interface of the *p-medicine* portal.

- Chapter 6: **Current development state of the *p-medicine portal***

  In this chapter we describe the actual available functionality of the *p-medicine* portal. First, some details about the server hosting the portal are provided. In the next section we introduce the *p-medicine* functionality that is available as plugins of the portal. In addition we describe the integration of the *p-medicine* security framework as well as the Data Mining tools and ObTiMA into the portal. Finally, we show the communities that are already included in the portal. We also describe a developed template that simplifies the creation of new communities in the *p-medicine* portal.

- Chapter 7: **Conclusion**

[3] http://www.liferay.com/

This chapter concludes the deliverable by summarizing the current development of the portal and describing how the portal can be extended in the future with new features.

3 Requirements for the *p-medicine* portal

The clinicians, researchers and patients involved in the *p-medicine* project have the need for an integrated platform to collaborate, share data and expertise, as well as use tools and services to improve personalized treatments of patients. One method to improve the user experience and improve overall user productivity is to aggregate these various *p-medicine* applications into a single portal.

3.1 Analysis of requirements for the *p-medicine* portal

The *p-medicine* portal is the common access point to the sharing and collaboration platform for the *p-medicine* users. First, we identify the user groups and user roles specific to *p-medicine*. Then, we describe general user requirements for the *p-medicine* portal and describe in particular the tools and services which shall be accessible in the *p-medicine* environment via the portal. The description of the user requirements for the entire project is given in the Deliverable D2.2.

The following initial list of the *p-medicine* portal users and their roles in the p-medicine portal is based on discussions with end users:

Role in the portal	User groups
CLINICIAN	clinicians, physicians
CLINICAL TRIAL LEADER	clinical trial chairman, clinical trial designers
SCIENTIST	researchers from the scientific community (mathematicians, physicists, bioinformaticians, computer scientists, etc)
PATIENT	patients and their relatives
DATA MANAGER	study nurses, documentalists
ADMINISTRATOR	technical administrator of the *p-medicine* portal (responsible for installations, configurations, backups, database maintenance, etc.)
PMED ADMINISTRATOR	administrator of the *p-medicine* portal (responsible for user management, user permissions, community management, etc.)
COMMUNITY ADMINISTRATOR	administrators of the communities created in the *p-medicine* portal
DEVELOPER (IT)	developers of tools and services for *p-medicine*
LEGAL AND ETHICAL ISSUES	people dealing with legal and ethical issues
GUEST	not authenticated portal users

However, in the future new roles may arise. Different roles giving permissions for using different tools developed under p-*medicine* (e.g. OBTIMA USER) could be created. Therefore it needs to be possible in the *p-medicine* portal that new roles in the portal can be created dynamically.

3.1.1 User requirements for the *p-medicine* portal

The *p-medicine* portal shall be a secure unified access point to the tools, services and data shared in the *p-medicine* environment in the form of a web-based user interface. A short subset of important requirements is provided below which has to be considered during the portal development:

- Manage individual users and user groups (e.g. clinicians, researchers, patients, etc.) in the *p-medicine environment.*
- Manage user roles and user permissions to the resources available in the *p-medicine* environment. The roles and rights management system regulates which data and tools, services and models an end-user can access and work with. Therefore a role based access, based on the legal and ethical framework developed in the project shall be developed.
- Manage multiple *p-medicine* specific documents and applications. Therefore aggregation, integration, personalization and permissions for the *p-medicine*-specific content shall be available in the portal.
- The portal shall provide a unified and comfortable user interface that supports the end user to achieve his/her aims efficiently, effectively and satisfactory way.
- Use the available *p-medicine* tools and services in the *p-medicine* scenarios described in D2.2 (e.g. to manage clinical trials, to execute scientific workflows with data mining tools, to manage biomaterial, to retrieve clinical and trial data for patients from the data warehouse).
- Download *p-medicine* tools (e.g. flash tutorials as well as eLearning tools which will be developed in the WP16) and to share patient related data and contents (documents, pictures) collected in *p-medicine* in the portal.
- The access to the portal shall be secure. Therefore the data protection and data security framework which provides a role driven access to the *p-medicine* resources, based on the *p-medicine* legal and ethical framework, developed for the project (see the Deliverable D5.1), shall be integrated in the portal.
- The Single-Sign-On functionality shall be supported: it should not be needed for the *p-medicine* portal users to sign in to every application separately. With Single Sign-On the login procedures of different applications shall be centralized in one system, accessible with only one password. After entering the right password, only the applications that the user is authorized to use shall become available.
- Collaboration, social networking and other “Web 2.0” features shall allow to build virtual communities of users to promote collaboration for research and education. Therefore the integration of collaboration tools (message boards, calendar, chat, e-mail, blogs, etc.) could be very comfortable and helpful for the end users.
- Different project partners will contribute to the development of the *p-medicine* tools and services for integrating them into the portal. Therefore the developing of the tools and services shall be based on standards, used for portal components (e.g. The Java portlet specifications[4] JSR 168 and JSR 286). Common guidelines on how to develop *p-medicine* compliant portlets needs to be provided that guarantee e.g. a common look and feel. It shall be possible for the developers of the *p-medicine* tools to register and to publish new developed *p-medicine* tools.

Based on the requirements listed above and on the Deliverable D2.2 "Definition on scenarios and use cases and report on scenario based user needs and requirements" the tools and services for the initial integration into the *p-medicine* portal have been defined. In the next

[4] http://developers.sun.com/portalserver/reference/techart/jsr168/

sub-section the tools and services are presented which will establish the *p-medicine* functionality available in the portal.

3.1.2 *p-medicine* tools and services

Below we briefly show the *p-medicine* tools and their functionality. We expect that the list of the tools and services will be extended in the future and the newly developed tools and services will be integrated into the *p-medicine* environment dynamically.

3.1.2.1 *p-medicine* Security Framework Services

As defined in Deliverable D5.1 and further specified in Deliverable D3.4 *p-medicine* provides a lightweight dynamic security architecture. It consists of modular re-usable components which deal with security problems such as authentication, authorisation, auditing and de-identification. Those components are based on commonly used industry standards such as SAML and XACML. More details about the security framework and its integration into the *p-medicine* environment are given in the Deliverable D8-6-1 "Integration guidelines and monitoring of tools and services".

3.1.2.2 *p-medicine* Workbench

The *p-medicine* workbench shall be an application that aggregates the *p-medicine* tools and services and provides to the end users various management possibilities for publishing, discovering, evaluating, combining and invoking these tools. Clinicians will be able to use these tools that lack specific user interface or are not integrated into the portal framework through the *p-medicine* workbench in order to expedite various tasks of their daily practice. The tools and services of *p-medicine* will be registered in a tool repository, where an archive will be kept of their metadata and annotations, their quality characteristics, their documentation and their invocation endpoint. The *p-medicine* workbench will be the portal compliant proxy to this tool repository but it will also support the “behind-the-scenes” invocation of the integrated tools and the retrieval of their results.

An important task of the workbench will be to make sure that the annotations and metadata descriptions of the tools are compliant with the VPH standards (whenever applicable) and also to be in line with the HDOT (if feasible) for seamless, semantic integration with the data management services and the provided data. The VPH standards consist of standards for ontology, data, modelling, and infrastructure interoperability, reflecting the multi-faceted arena of VPH research which have been developed by the VPH-NoE's standards working group (SWG). More information about the VPH standards is provided in the D3.1 "State-of-the-art report on standards".

3.1.2.3 Data Warehouse Access Tools

A secure and scalable data warehouse will be built as a central research repository of *p-medicine* with respective services for collecting, sharing and further elaborating annotated anonymized clinical data and other research relevant data from diverse heterogeneous sources such as in particular clinical trials and electronic patient records from hospital information systems. The data warehouse will store and manage large data sets in an efficient manner and provides the main resource for new knowledge discovery, VPH modelling and simulation. A push concept will be implemented, which allows owners of data (i.e. clinicians, trial chairmen) to annotate and upload their data to the data warehouse, in order to make so far unexploited data resources available to research.

The *p-medicine* Data Warehouse will store data from a large set of heterogeneous databases (clinical trial databases, public repositories, etc). All these data are homogenized

into a common format (provided by HDOT). The homogenization process is supported by the **Data Translation Service** developed by UPM, which employs advanced algorithms in order to automate this task and provides a data upload service, which shall be able to invoke the Translation service when required. The Data Translation Service does not offer any graphical interface for users to interact with (the process is completely transparent). This service is invoked by the Data Warehouse for each dataset pushed.

3.1.2.4 OpenStack Object Storage

Storage services such as cloud storage are highly distributed, scalable and virtualized environments that offer availability, accessibility and sharing of data at low cost. Cloud storage in the *p-medicine* environment is intended to be used by the Data Warehouse as a storage backend for files. The other scenarios are related to data mining workflows that can use cloud storage for intermediate computation results, and oncosimulator application executed in a dedicated cluster environment. The *p-medicine* Cloud Storage System is built based on open source cloud storage technology OpenStack Object Store (Swift[5]) technology. It provides access to reliable storage space taking into account requirements from different end user scenarios: long term data preservation on the one hand, as well as fast access to application data in the workflow execution.

The OpenStack Object Storage API level layer is extendable by WSGI (Web Server Gateway Interface) components to conform to different standards like Cloud Data Management Interface (CDMI[6]) or de facto standards like Amazon S3. The Amazon compatible API component is given out of the box and seems to be stable but the CDMI support is on the way and being developed currently so it needs time and probably will cover only required but no optional features. One of the missing features so far is the search functionality using metadata attributes and this possibly could be realized during the project lifetime to support external services while searching for data. The security subsystem has to be extended and integrated with the *p-medicine* federated authentication mechanism using SAML and a central identity service. Security extension is a must be feature to allow project wide integration of Data Warehouse and portal tools together in a consistent system. It means that OpenStack Object Storage service will be used by other services in the project and beside this a portlet for portal end users will be introduced to support end user data management in the cloud. End users will be able to store individual files in the cloud. This portlet will be a very simple data management application to upload and download private files related to work done during interaction with *p-medicine* portal like some output analysis from some data mining tools existing in the portal or relevant reports in excel or word files etc.

3.1.2.5 Services for Storing of Medical Images

DICOM (Digital Imaging and Communications in Medicine) is the de facto standard for handling, storing, printing, and transmitting information in medical imaging. A typical DICOM file server fulfils most of the requirements for storing medical imagery.

As for the generic file store, images must be referred to by URI, since many federated image stores may exist. The image store should offer direct, secure access to images through the standard DICOM image access protocols. The need for data to only be accessible to those with permission means that the security mechanism for the DICOM server must be integrated with that of *p-medicine*. It must refer to the Custodix identity management system for identity information, and to information stored in a central data warehouse system on who is permitted to access what data. Therefore, an image storage solution must be chosen which can be adapted in this way. Unlike the generic file store, the metadata infrastructure of

[5] http://swift.openstack.org/
[6] http://www.snia.org/cdmi

DICOM will be used and maintained in parallel with the data in the structured data store. Close integration between these elements is essential, and a syncing service will be necessary to ensure data is consistent. The DICOM server will be integrated directly with the OpenStack Object Store service to extend storage space for large and numerous medical images which could be added to the system. Such a solution allows to archive big or less used data to a large and reliable cloud storage whereas local DICOM server fast disk storage will be used for recent images and used as a cache for often accessed data. Image annotations, provenance and history information added to the warehouse directly (rather than imported with images) will be primarily maintained outside of the file store, but synced with the DICOM server metadata where appropriate. These services are described in more detail in the D7-1 "Report on overall system design including VPH-Share D2.2 and indicating its impact"

3.1.2.6 ObTiMA

ObTiMA[7] is an ontology-based clinical trial management system intended to support clinicians in both designing and conducting clinical trials. The design phase is facilitated by the Trial Builder in which all aspects of a clinical trial can be specified: a trial chairman can define the outline and metadata of a trial in a master protocol to describe, e.g., trial goals or administrative data. He can further setup treatment plans for guiding clinicians through individual patient treatment where events, e.g., surgery or chemotherapy, can be defined with all necessary information. Also, the particular treatment order can be freely setup on a timeline as well as treatment stratifications and randomizations to be applied for a patient. A Case Report Form (CRF) can be assigned to each treatment step to collect documentation data.

The ontology-based creation of CRFs in the Trial Builder is one of ObTiMA's major functionalities. A graphical user interface allows defining content, navigation, and layout of CRFs to capture all patient data during a trial, e.g., medical findings or diagnostic data. The resulting descriptions are based on ontology concepts for each CRF item along with metadata, e.g., data type and measurement unit, and used to setup the trial database. So ObTiMA’s internal database should be HDOT compliant or the proper mappings to the HDOT should be made available and used.

ObTiMA will also integrate push services that will be able to push selected data into the Data Warehouse. The data and images uploaded to the Data Warehouse should be anonymized by CATS before storing into the Data Warehouse.

ObTiMA will include a trial biomaterial manager - one of the main parts of the biobank access framework (see Deliverable D 10.1). The trial biomaterial manager will enable management of biomaterial data in clinical trials and sharing selected biomaterial data. This component is described in more detail in the next section.

3.1.2.7 Biobank Access Framework

The *p-medicine* Biobank Access framework aims to provide access to different kind of human biomaterials and related data for research purposes. The framework harmonizes the data according to a standard biobank data set. In particular for the *p-medicine* biobank framework the following three main components will be developed to support the main functionalities of the framework:

- **p-BioSPRE,** the *p-medicine* Biomaterial Search and Project Request Engine that will be a metabiobank for providing researchers the possibility to search for and request biomaterial that is offered within his communities and which fits their research purposes.

[7] http://obtima.org/

- **p-BioBank Wrappers** are tools to support biobank owners to offer their biomaterial and related data within an open or closed research community, which can be stored in any biobank information system, in p-BioSPRE and manage associated requests to order biomaterial. Furthermore, a p-Biobank Wrapper comprises push services, which enable a biobank owner to push selected biomaterial data into the *p-medicine* data warehouse and share for the integration with other biomedical data sources.
- **ObTiMA Trial Biomaterial Manager** will be developed as a component of ObTiMA to enable users of ObTiMA to manage biomaterial data in clinical trials and to share selected biomaterial data. The following main functionality will be available in the Trial Biomaterial Manager:
  - An interface to manage biomaterial data in clinical trials according to the standard biomaterial data set. For this purpose predefined biomaterial CRFs will be provided in the ObTiMA CRF repository that can be adjusted to the user needs.
  - An import service that enables users to upload their legacy biomaterial data into ObTiMA.
  - A search interface to get an overview about the available biomaterial. The search interface will allow linking clinical data and biomaterial data within clinical trials and across trials for further analysis.
  - An interface to select the biomaterial data that the biobank owner wants to share, and to upload the data to p-BioSPRE. For this purpose upload services will be integrated, which take care that during the upload process the data is anonymized.

The detailed description of the *p-medicine* Biobank Access framework is provided in the Deliverable D 10.1.

3.1.2.8 Oncosimulator

An integrated predictive multiscale software system “the *p-medicine* Oncosimulator” will be developed to simulate the response of clinical tumours to several treatment schemes and/or schedules in the patient individualized context as well as a number of mutually compatible detailed models of specific tumour biomechanism aiming at enhancing our understanding of the natural phenomenon of cancer. The ultimate target of both actions is to optimize individualized cancer treatment.

The integrated system will be able to suggest the optimal treatment scheme/schedule out of a number of candidate simulated schemes/schedules for the individual patient based on their own multiscale data and the simulation outcomes. The system is envisaged to support in silico cancer research, optimization of clinical trial design and patient individualized treatment optimization.

3.1.2.9 Data Mining Tools

The data mining tools developed for *p-medicine* should translate data mining techniques into practical solutions for the analysis and prediction of patient data. To facilitate a seamless transition of methods from data mining research to medical research, a pattern-based approach will be followed. A data mining pattern can be viewed as template for concrete data mining workflows. It acts as a formalized common language between data mining experts and medical researchers/bioinformaticians. The name "data mining pattern" means a re-usable workflow template plus a description of the requirements and steps that are necessary to apply the generic data mining solution to a specific problem. It thus bridges the gap between a data mining algorithm, and a tool/workflow that applies the data mining algorithm to answer a specific research question on a given data set.

Data mining patterns will be defined for the cases of the analysis of very large data sets, the analysis of privacy-sensitive data, data mining to support collaborative systems, and predictive literature mining.

For enabling the re-usability and standardization of generic data mining scenarios in the analysis of clinical data the data mining patterns will be developed which extend the concept of data mining workflows in the sense that next to a workflow template they also include a representation of the manual work that needs to be done in order to successfully apply a workflow to a specific problem (such as checking of requirements or manual configuration of components). A formal process that describes all necessary steps for the instantiation of a data mining pattern to a specific task, i.e. the creation of an executable workflow from a generic pattern will be developed. Furthermore, tools that support the creation, management, and instantiation of data mining patterns in the *p-medicine* architecture will be provided.

The major requirement in *p-medicine* on data mining in particular from the point of view of the bioinformatician and the biostatistician (see Appendix 2 in *p-medicine* Deliverable D2.2) is to support different tools for analysis and flexibility in adding more tools. In particular, Taverna[8] as tool for creating workflows and statistical software such as SAS[9] and R[10] are required. Supporting and integrating existing workflow environments will also enable the *p-medicine* users to profit from the existing scientific workflows shared on public workflow repositories such as MyExperiment.org and others.

3.1.2.10 Push and Sync Services

In order to make use of the clinical data stored in the electronic clinical records in heterogeneous hospital information systems (HIS) and other biomedical databases *p-medicine* will provide **push services** for the seamless integration of data retrieved from HIS into the *p-medicine*'s data warehouse. These services integrate the data semantically by means of the HDOT ontology and the tools of the semantic layer so that the data in the data warehouse can be stored in a form compliant with this ontology. The data warehouse supports the secondary use of the data by providing an SPARQL interface that allows querying the data in terms of the HDOT ontology. This interface supports the seamless reuse of the data by the various analysis tools and services of the *p-medicine* platform, as e.g. data mining, simulation and decision support tools. In general this approach requires the export of the data from their original sources, their transformation to the necessary format, and their upload to the Data Warehouse.

Sync services will be developed to allow reusing data stored in the electronic clinical records in hospital information systems in running clinical trials in the ontology-based trial management system ObTiMA. The retrieved data will be stored in the Clinical Record Forms (CRF) of a running clinical trial in ObTiMA. This shall partially avoid entering data manually in CRFs when they are already included in the HIS. The resource annotation tools developed by WP4 will be used for the sync services.

The sync services will retrieve the data not directly from the hospital information systems but from the *p-medicine* Data Warehouse, since this data is already integrated compliant to HDOT.

A detailed description of the pushing and sync approaches is given in Deliverable D4.2.

3.1.2.11 Ontology Aggregator

The Ontology Aggregator tool will be developed in WP4. It will be a tool for aggregating semantic resources in an (semi-)automated fashion under HDOT. The goal is to explore

[8] http://www.taverna.org.uk/
[9] http://www.sas.com/
[10] http://www.r-project.org/

ways of creating HDOT-modules automatically from a set of pre-selected semantic resources. The purpose of the ontology aggregator tool is to allow users to find and select relevant parts of semantic resources and semi-automatically integrate those parts under HDOT, i.e. users can re-use pre-existing semantic resources and extract classes or terms from them to design specific HDOT modules for their specific needs. Additional details, characteristics and features of this tool will be provided later in the project.

3.1.2.12 Ontology Annotator

The **Ontology Annotator** is a graphical tool aimed at end users (specifically data managers) to allow them creating the necessary annotations for the translation of data into HDOT format. The tool provides a graphical representation of the schema of the database to be integrated in the *p-medicine* platform and HDOT. By clicking, selecting and dragging elements, users will be able to define semantic equivalences that enable the data to be automatically translated by the Data Translation Service. Focus will be put on usability so users are not forced to have high skills on RDF models and ontologies.

3.1.2.13 Clinical Decision Support Tools

Clinical Decision Support (CDS) tools will be developed in WP13 to enable the clinical specialists to efficiently access data and infer knowledge necessary to reach the most accurate decision for the best patient outcome. CDS tools integrated with the EHR (electronic health record) systems provide a tool set to ensure the right information is available where, when and how clinicians need it and that clinicians follow the proper clinical processes.

A second main goal is to support the clinicians to prevent or identify early in the treatment potentially serious side effects to treatments and drugs, and the patients most susceptible to develop serious side effects.

From a clinic point of view this tool will allow users to input biomarker information and using such information the tool will formulate a prediction about how a patient will respond to different treatment regimes. The tool will then suggest the best treatment for an individual patient. Eventually, the tool should be able to combine various inputs; biomarkers, pathological markers, imaging data etc. We envision a tool that can integrate several variables following a multi-dimensional approach.

In fact, the tool should be able to integrate data and information stemming from three dimensions, namely, clinical information, genomic information and psychological information. This will form the basis for a multi-dimensional analysis of a patient's predictive outcome to clinical trials.

The CDS scenarios to support more efficient execution of the trial protocols, patient stratification with respect to risk of relapse and adverse events, and early detection and management of serious adverse events for Nephroblastoma, ALL and Breast Cancer are given in D13-1.

3.1.2.14 Patient Empowerment Tools

Patients are typically seen as the recipients of care. An important ideal of personalized medicine is to better enable patients themselves to be participants and guides in their own health care. The role of patients will be strengthened in *p-medicine* by allowing them to decide at any time what kind of research is allowed to be done with their data and their own biomaterial.

In order to cope with the previous challenges *p-medicine* proposes the Interactive Empowerment Service (IEmS).The IEmS architecture combines the progress of technology and biomedical research with patients' personal needs. Patients and doctors can access the

Interactive Empowerment Service through the *p-medicine* portal. Then, questionnaires and intelligent profiling mechanisms are used in order to construct patient profiles. Thanks to collaboration among physicians, psychologists and ITs, the patient's profile can be combined with the Patient's Health Record System (PHR). PHR is one of the critical components of the IEmS, where clinical information is patient-tagged and combined with his/her psychological, social and cognitive characteristics (patient's profile). Its importance is double. Firstly, it is a container of all the information related to the patient, such as information on diagnosis and biobank data as well as information on the possible clinical trials. Secondly, the patient his/herself has the opportunity to have access to such information. In the PHR the patient can follow the journey of his/her data and based on it begin the decision process.

Furthermore, by having this information, the patient has the possibility to manage the consent for clinical trials as well as following what happens to any biological material. Providing patients with „consent management" offers a dual benefit: first of all, there is the direct empowerment aspect of controlling one's own data; secondly, it facilitates interaction between patients and doctors when a request for a new consent (e.g., for new trials) is needed, by increasing efficiency and again by involving the patient actively.

More details about the Patient Empowerment Tools are provided in chapter 8 of D2.2.

3.1.2.15 Interaction with the VPH Toolkit

One of the goals of the *p-medicine* project is to build up an interactive collaboration with the Virtual Physiological Human (VPH) Network of Excellence (NoE)[11] to harmonize tools, methods and services available on the VPH Toolkit[12]. The VPH ToolKit is a technical and methodological framework to support and enable VPH research - the collaborative investigation of the human body as a single complex system. Integration of tools from the VPH ToolKit will be implemented within the development of the *p-medicine* workbench (see above) in the Task 8.6 where a tool repository will be provided. In the *p-medicine* workbench users will be able to discover the most applicable tools for analyzing data, combine these tools into complex workflows, and share results for reusing by other users. The VPH tools can be included into the analysis workflows of *p-medicine*. To ease the integration of VPH and *p-medicine* tools, in p-medicine the tool and dataset annotations from VPH NoE should be reused and combined with semantic resources developed in *p-medicine*, as HDOT. The VPH NoE community can reuse *p-medicine* tools in their portal, and can build up a mechanism to pull tools from the *p-medicine* platform to the VPH NoE.

The details of the interaction between *p-medicine* and the VPH Toolkit are the subject of Deliverables 8.1.1, 8.6.1 and 8.6.2.

As shown in this section, the portal is an ideal choice for providing the *p-medicine* functionality. In order to discover the best technical solution for *p-medicine* we have evaluated several technical portal frameworks described in the next chapter.

[11] http://www.vph-noe.eu/
[12] http://toolkit.vph-noe.eu/

4 Evaluation of existing portal frameworks

In this chapter we describe several technical solutions that seem appropriate to fulfil the complex user requirements for the *p-medicine* portal. For our evaluation we have chosen three established enterprise portal frameworks: Liferay[13], GateIn[14] and Apache JetSpeed[15]. Furthermore, we review the technical solutions that have been chosen for the portals of the biomedical frameworks ACGT[16] (GridSphere[17]) and the content management system Joomla!,[18] on which the VPH NoE Portal is based. Last, we will have a look at WSO2 Mashup Server[19], which provides a powerful yet simple and quick way to tailor Web-based information to the personal needs of individuals and organizations. In the following, for each of the described technical solutions we list the main features, describe the license under which it is provided and describe the drawbacks.

4.1 Enterprise Portal Frameworks

Enterprise portal frameworks have been developed to allow companies or organizations to build web based interfaces to integrate their knowledge, processes and applications[20].

Such frameworks provide a single web-based environment from which all of a user's applications can run. These applications are integrated together in a consistent and systematic way. They support various features: they enable users e.g. to build web pages and websites, they integrate content and document management systems and collaboration software and provide an advanced user management and security solution.

Enterprise portal frameworks allow administrators and end-users to build web pages without coding by reusing existing "portlets", which are pluggable user interface components that are managed and displayed in a web portal. To enable interoperability of portlets a Java standard that is called Java Portlet specification JSR 168 has been defined. The Java portlet specification JSR 168 defines a set of APIs for the interaction between the portlet container and the portlets and therefore it formalizes the creation and deployment of portlets that can operate in any compliant portal framework. Built-in permission system in the portal allows defining permissions to each portlet. The current Java portlet specification JSR 286 enables better AJAX support and portlet coordination through an improved events management.

Modern portals have added multiple features that make them the best choice for a wide array of web applications. Some common uses for a portal in the *p-medicine* project include:

- Anonymous Pages and Authenticated Pages: Portals make it easy to build websites that show different content depending on whether or not the person is logged in. For example, a *p-medicine* website may feature a set of pages describing its goals, tools, services, and contact information that are accessible to all; however, after a *p-medicine* user login, additional content may be available such as tools and services that could be executed, some announcements, etc.
- Role-Based Content Delivery: Portals additionally simplify the development of websites that display different data depending on a user's role. For instance, the *p-medicine* website may feature anonymous and authenticated (logged-in) pages but can additionally

[13] http://www.liferay.com
[14] http://www.jboss.org/gatein/
[15] http://portals.apache.org/jetspeed-2/
[16] http://www.eu-acgt.org/
[17] http://www.gridsphere.org/
[18] http://www.joomla.org
[19] http://wso2.com/products/mashup-server/
[20] Description of portal platforms: http://www.liferay.com/products/what-is-a-portal/web-platform

have different pages available for various user types. A standard account can have basic services and pages, while users with administrative rights can have additional pages defined.

- Integration Platform: One of the earliest portal uses was to integrate various existing applications into a single unified user experience. Portals enabled enterprises to pull together information and applications into one website where users based on roles would have quick access to all content specific to their role.
- Portals can also serve as a repository for documents. Similar to content, documents can be added to the repository and made available through the web interface or website. For example, individuals can publish documents into a central repository and have them be made available to portal users in a central library upon their login.
- Community based: portals can collects users who have a common interest in communities. Portal users can join and leave communities whenever they want. Users can share data and content collected in repositories available for the community members.

In the following sections we describe the enterprise portal frameworks Liferay, GateIn and Jetspeed as well as the GridSphere portal.

4.1.1 Liferay framework

The Liferay[21] framework is a free and open source enterprise portal framework that has originally been developed in 2000 to provide a portal solution for non-profit organizations. Currently Liferay is maintained by the Liferay Company, which claims that Liferay is the leading open source portal for enterprises.

Below are the most important features of the Liferay framework:

- It allows simplified development of the user interface: users can build web pages and websites without coding by reusing existing portlets and gadgets. They can also define who can access their pages.
- It supports of multiple integration methods, including SOAP, REST, RSS, as well as proprietary API's.
- It includes more than 60 out of the box portlets, including a content management system and collaboration tools (wikis, message boards, blogs, polls and shared calendars).
- It provides a single log in credential for multiple systems (Single Sign-On).
- Administrators can customize, edit, add, and change user attributes, pages, web content, blogs entries, documents, images, bookmarks, forum messages, calendar events and wiki pages directly from the portal, without the need to modify their database.
- It provides a central platform for determining enterprise content policy, including who can edit and publish content, files, and applications.
- Web content, documents, message board threads can be tagged and categorized.
- Liferay supports more than 30 languages. Additional languages can be easily added.
- It includes a full document and image repository allowing users to store assets, tag them with key words, lock them, search for and leverage them in web pages, or download them for offline use.

[21] http://www.liferay.com

4.1.2 GateIn

The GateIn Portal[22] is a free and open source enterprise portal project originated as a result of merging the JBoss Portal 2.7 and the eXo Portal 2.5 that produced GateIn 3.0, and also the related projects GateIn Portlet Container, eXo JCR, and JBoss Portlet Bridge

Below are the most important features of the GateIn Portal:

- It provides authentication options including LDAP and SSO (incl. Central Authentication Service (CAS), JOSSO and OpenSSO frameworks) integration.
- In addition to creating standards-based portlets, developers can build and display Mashups[23] & OpenSocial[24] gadgets.
- provides personal dashboards allowing users to display their preferred content. Dashboards can also be shared, to make gadgets and content available within a team.
- It supports the following standards:
  - for CMS: Content Repository API (JSR 283)
  - Java portlet specification (JSR 168, JSR 286)
  - Google Gadgets Specification: OpenSocial is a standard that defines a common API for social applications across multiple websites. Built from standard JavaScript and HTML, developers can create apps with OpenSocial that access a social network's friends and update feeds. By using a common API, developers can extend the reach of their applications more quickly, yielding more functionality for users.
  - Portlet Bridges: Supports Java Server Faces (JSF) via the JSR 301 Portlet Bridge
- It provides rich user interface, wizards for creating new pages, supports advanced layout and customization of pages (language and GUI's look and feel).
- It provides functionality for management of pages and permissions.
- It provides role based access to the portal resources for users.
- It provides simplified user management.
- It supports additionally Arabic, Hebrew and Persian languages.
- It has flexible deployment architecture, provides pluggable services.

The drawbacks of this framework are:

- Installation is very difficult because there is only limited documentation.
- The first start takes several minutes
- Development and debugging is not easy (in spite of the hot deploy).
- The standard settings are not feasible and need to be modified.
- The available CMS contains several bugs.
- It provides a limited set of available portlets.
- Only a small set of features could be evaluated because of lack of documentation
- The GateIn portal[25] is still in development, but maintenance is insufficient: the last bug was solved by September 2010.

[22] http://www.jboss.org/gatein/

[23] Mashup is a web application hybrid: combination, visualization, and aggregation

[24] OpenSocial is a public specification that defines a component hosting environment (container) and a set of common application programming interfaces (APIs) for web-based applications

[25] https://issues.jboss.org/browse/GTNPORTAL

4.1.3 Apache Jetspeed-2

Apache Jetspeed[26] is an Open Portal Platform and Enterprise Information Portal, written entirely in Java and XML and based on open standards. The access to the portal is managed through a robust portal security policy. Within a Jetspeed portal, individual portlets can be aggregated to create a page. Each portlet is an independent application with Jetspeed acting as the central hub making information from multiple sources available in an easy to use manner.

A portal based on Jetspeed can make applications, database information and other data sources available to end-users through a single web site. Jetspeed provides a security infrastructure so that the information and functions made available to each user can be customized on basis of the user or a role that the user has. The user can access the portal via a web browser, WAP-phone, pager or any other device supported by the servlet engine.

Below is a short overview of Jetspeed features:

- It provides a large set of sample and administrative portlets., dynamic web components based on standards (incl. Deployment Jetspeed Portlet, Page Skins (Decorators), CSS Components, configurable CSS Page Layouts).
- It provides scalable component architecture based on Spring framework.
- It supports the following application servers: Tomcat 6: Jetty, WebSphere 6.0, Geronimo, JBoss, WebLogic.
- It integrates Velocity Macro Language for Skin and Layout Components.
- It includes XML Import/Export Utility for all Jetspeed database data to support data migration over versions.
- It provides bridges to other Web Frameworks: Wicket, JSF, Groovy, Struts, PHP, Perl, and Velocity.
- It provides rules-based Profiler for page and resource location.
- It integrates with most popular databases including Derby, MySQL, MS SQL, Postgres, Oracle, and DB2.
- It supports Pocket PC.
- It supports 12 Languages.
- It provides full text search of resources with Lucene[27].
- It is fully compliant with Java Portlet API Standard 1.0 and 2.0 (JSR 168 and JSR 286).
- It supports security standards: JAAS[28], JAAS DB Portal Security Policy, Single Sign-On, and LDAP[29] Support for User Authentication.
- It provides the Jetui AJAX Customization Engine with Dockable Toolbox, Drag and Drop and new navigation features.

The drawbacks of this framework are:

- Even the last version contains many bugs.
- The GUI is uncomfortable.
- End users need to have different technical skills (Web design, JSPs).
- Maintenance is limited.

[26] http://portals.apache.org/jetspeed-2/
[27] http://lucene.apache.org/core/
[28] http://en.wikipedia.org/wiki/Java_Authentication_and_Authorization_Service
[29] http://en.wikipedia.org/wiki/LDAP

- It uses an own language for portlet registry, PSML (Portlet Structure Markup Language). PSML is not a standard portlet specification.
- It uses an own template language, called Velocity, as an alternative to JSPs.
- It provides poor documentation

4.1.4 GridSphere (ACGT)

The portal for the European Grid infrastructure ACGT[30] has been developed based on the GridSphere[31] portal framework. GridSphere is an open-source portlet based Web portal. It enables developers to quickly develop and package third-party portlet web applications that can be run and administered within the GridSphere portlet container.

The GridSphere Portal Framework offers the following features:

- It supports easy development and integration of new portlet applications.
- It provides a higher-level model for building complex portlets using visual beans and the GridSphere User Interface tag library.
- It includes flexible XML based portal presentation description can be easily modified to create customized portal layouts.
- It supports role based access control (RBAC) separating users into guests, users, administrators and super users.
- It implements portlet service model that can encapsulate reusable portlet logic into services that may be shared between many portlets.
- It supports persistence of data provided using Hibernate JDO/OQL for database support
- It enables "portlet-izing" of Struts applications using the Portals Struts Bridge.
- It provides Single-Sign-On functionality using X.509 certificates or Shibboleth.
- It includes a set of various core portlets.
- It supports French, English, Spanish, German, Dutch, Czech, Polish, Hungarian, Italian and even Arabic, Japanese and Chinese languages.

The drawbacks of this framework are:

- No support for the current portlet specification (JSR 286).
- Is not maintained any longer - the last update has been on 11.11.2010 (version 3.2).

4.2 WSO2 Mashup Server

The WSO2 Mashup Server[32] is a powerful yet simple and quick way to tailor Web-based information to the personal needs of individuals and organizations. It is platform for acquiring data from a variety of sources including WebServices, HTML pages, feeds and data sources, and process and combines it with other data using JavaScript with E4X XML extensions. The result is then exposed as a new Web service with rich metadata and artefacts to simplify the creation of rich user interfaces.

Mashup products and frameworks all have several pieces in common. They provide ways to acquire data, they provide a central abstraction and tools for working with that data, and they provide ways to publish the result to the consumer. The WSO2 Mashup Server uses Web

[30] http://www.eu-acgt.org/
[31] http://www.gridsphere.org/
[32] http://wso2.com/products/mashup-server/

Services as the mechanism for acquiring data, XML as its central abstraction, JavaScript (with E4X XML extensions) for manipulating that data, and again Web Services as the primary way to re-publish the new information stream, with bridges to HTML, RSS, and other output mechanisms.

Although the WSO2 Mashup Server is an open source platform, some of the supported issues are commercial for customers.

The WSO2 Mashup Server offers the following features:

- It has an enhanced user interface for managing the mashups and to bundle a custom user interface for the mashups.
- It provides an extensible server administration framework.
- It provides a separable frontend & backend where a single frontend server can be used to administer several backend servers and clustering (inherited from WSO2 Carbon Platform[33]).
- It provides access to REST and WS-web services, feeds, and scraped web pages with data scripted together for creating a new service, or a web page, gadget, email or instant message.
- It provides the ability to secure hosted mashups using a set of commonly used security scenarios and the ability to upload mashups with its resources as a zip.
- It supports both recurring and longer-running tasks and service lifecycles.
- It provides monitoring, configuration of security and quality of service settings such as throttling.
- It implements simple file based deployment model: hosting of mashup services written using JavaScript with E4X XML extensions: JavaScript annotations to configure the deployed services
- It provides for automatically generation of metadata and runtime resources for the deployed mashups:
  - JavaScript stubs that simplify client access to the mashup service
  - code templates for developing rich HTML or Google Gadget interfaces
  - TryIt functionality to invoke the mashup service through a web browser
  - WSDL 1.1/WSDL 2.0/XSD documents to describe the mashup service
- It provides many useful JavaScript Host objects that can be used when writing mashups.
- It provides Equinox P234 based provisioning support.
- It provides periodic task scheduling.
- It allows E-mailing, Instant Messaging (IM).
- It supports development, production, provides professional services and turnkey packages.
- It uses JavaScript language with E4X XML extensions for composition of mashups.

The drawbacks of this framework are:

- The TryIt service doesn't work for existing services and for newly created services.
- Several bugs exist in the user management.
- Not all of the supported issues are free of charge.

[33] http://wso2.com/products/carbon/
[34] http://wiki.eclipse.org/Equinox_p2

4.3 Joomla! (VPH NoE)

The Virtual Physiological Human (VPH) ToolKit[35] is a web portal to support VPH researchers. This portal is developed by the VPH Network of Excellence (NoE) project based on the Content Management System Joomla![36]. The portal provides information about available specific tools, methods and services and it also invites researchers to submit tools, methods, and services for inclusion in the database to share knowledge with the community.

Joomla! is a free and open source content management system (CMS) for publishing content on the World Wide Web and intranets and a model–view–controller (MVC) Web application framework that can also be used independently.

Joomla! includes features such as page caching, RSS feeds, printable versions of pages, news flashes, blogs, polls, search, and support for language internationalization.

Joomla! has the following drawbacks:

- It is only content management tool. Integration and execution of web based applications/portlets not possible
- It supports only the MySQL database

4.4 Basic comparison of portal frameworks

In the following, we summarize the features of the evaluated portal frameworks. Please note, that this comparison is based on first internal experiments with the frameworks and on a basic literature research that also includes resources as forum articles[37]. Therefore the comparison is partly subjective and results may be different, when the frameworks are tested extensively.

Name	**Apache Jetspeed (version 2.0)**	**GateIn**	**Liferay**	**WSO2 MashUp Server**	**Joomla**	**Grid-Sphere**
Stability	even the last version has many errors	some errors detected	very robust in first experi-ments	errors in user manage-ment	good	--
Installation	easy	installation is difficult because there is not	easy, very good documen-ted process	easy, good documen-ted process	easy, good documen-ted process	--

[35] http://toolkit.vph-noe.eu/

[36] http://www.joomla.org

[37] http://ecmarchitect.com/archives/2009/10/01/1051, http://sadsoftware.blogspot.com/2009/02/how-to-select-open-source-java-portal.html, http://sadsoftware.blogspot.com/search/label/Liferay, http://www.liferay.com/de/community/forums/-/message_boards/category/243728, : http://www.cmscritic.com/interview-with-bryan-cheung-ceo-of-liferay/, http://www.manageability.org/blog/stuff/open_source_portal_servers_in_java, http://java-source.net/open-source/portals, http://www.oio.de/open-source-portale.htm, http://www.liferay.com/de/community/forums/-/message_boards/statistics, http://www.liferay.com/de/web/bryan.cheung/blog/-/blogs/saving-smart

		enough documen-tation				
GUI	uncomfor-table	standard	well-arranged and very comfortable	standard	standard	standard
Application Severs	Tomcat, Jeronimo, JBoss, Jetty, Web-Sphere, Weblogic	JBoss	Tomcat, Jeronimo, JBoss, Glassfish, Jetty, JOnAS, Resin	WSO2 Mashup Server	Apache-Webserver	IBM's WebSphere
Database	DB2, Derby (default), MS SQL, My SQL, Postgre-SQL, Oracle (9i or higher), SapDB	supports many databases including hsqlDB, MySQL, Postgres	support all major databases including db2, derby, firebird, informix, mysql, oracle, postgresql, sql-server, sybase	Apache derby, mySQL, postgreSQL	only MySQL	supports many databases including hsqlDB, MySQL, Postgres
Standards	JSR 168, JSR 286	JSR 168, JSR 286	JSR 168, JSR 286	-	--	JSR 168
Documen-tation	good documen-tation	few documen-tation	the best documen-tation	few documen-tation	very good documen-tation	few documen-tation
Available portlets, themes, layouts	large collection (over 40 portlets)	small collection (13 portlets)	very large collection (over 60 portlets)	not available, only JavaScript (with E4X XML extensions) based mashups	not available	small collection (10 portlets)
Deployment	hot deploy-ment	--	hot deployment	simple file based deployment model	--	--
SSO	yes	yes	yes	no	yes	yes
User and user groups management	yes	yes	yes, hierarchical, very advanced	yes, with errors	yes	yes

License	Apache Software License[38]	Affero General Public Licence[39]	GNU Lesser General Public License LGPL[40]	Apache Software License	GNU General Public License GPL[41]	GridSphere Open License GOL[42]
Community	Mailing lists, Wiki	Active Forum, Wiki	Very active Forum, Wiki, user groups in different countries	Sharing Community Portal, mailing lists, forums	Very large commu-nities of users in different countries	Mailing lists, Wiki

4.5 Conclusion – The Liferay Framework as a solution for the *p-medicine*

Based on our evaluation we have selected the Liferay framework as the technical base of the *p-medicine* portal. Our decision was based on the following features provided by Liferay:

- According to the literature and first tests the stability of the portal is very robust.
- Very good documentation material for administrators and for developers is provided.
- The installation process and first tests were easy and very well documented.
- The GUI of the portal is well arranged and very comfortable.
- The Java portlet specifications JSR 168 and JSR 286 are supported.
- A very large collection of out-of-box portlets are available.
- Deployment of customer tools and services is straightforward (hot deployment).
- A large set of collaboration tools can be integrated seamlessly.
- The framework is developed in Java.

All of the other evaluated frameworks have some drawbacks that disqualify them as candidates for the *p-medicine* portal:

- Joomla! is a content management system that does not support to integrate web-based applications or portlets.
- GridSphere does not support the newest Portlet standard and is not any longer maintained.
- JetSpeed does not provide a comfortable user interface and the end user needs technical skills.
- GateIn is not well documented and not as stable as Liferay. Already during the installation process several bugs occurred.

[38] http://www.apache.org/licenses/
[39] http://de.wikipedia.org/wiki/GNU_Affero_General_Public_License
[40] http://de.wikipedia.org/wiki/Lesser_General_Public_License
[41] http://de.wikipedia.org/wiki/GNU_General_Public_License
[42] https://ncisvn.nci.nih.gov/svn/c3pr/branches/c3prv2_elaboration_iteration4/c3prv2/codebase/projects/portal/gridsphere-2.2.8/LICENSE.txt

- WSO2 Mashup Server does not provide as extensive documentation as Liferay and seems from the first tests and according to literature not as stable as Liferay.

5 Using of the Liferay framework for the *p-medicine* project

In the following sections we describe how we have prepared the Liferay framework for using as the *p-medicine* project. The first part introduces some functionalities of the Liferay framework used in the *p-medicine* portal. In the next sections several ways for the integration of the *p-medicine* tools and services as well as a description of different portal components (plugins) are given. We show how the Plugins SDK as a part of the Liferay's development tools can be used to develop plugins for the portal using the industry standard Portlet API. In the following, the community based structure of the *p-medicine* portal is described. Finally, details about the integration of the specific *p-medicine* tools and services are given.

5.1 Functionality of the Liferay framework used for the *p-medicine* portal

The Liferay framework is an ideal choice for building the *p-medicine* web site. Using the unique constructs the platform provides, we can design a site that can handle any situation needed for the *p-medicine* portal. The Liferay framework also offers a platform for building *p-medicine* web applications as well as a large set of applications that are already implemented.

It comes with over 60 portlet applications included. These applications cover about all of the standard functionality the *p-medicine* users are likely to need in a web site: content management, forums, wikis, blogs, and much more. We only need to implement the features specific to the *p-medicine* project.

Liferay is based on the JSR-286 portlet standard. It includes utilities such as a Service Builder to automatically generate interfaces to databases.

Since it provides a multitude of tools and utilities for increasing developer productivity, the *p-medicine* portal can be developed more quickly.

p-medicine can use a powerful paradigm given by Liferay for organizing users and giving them access to the content they want to see. We can use communities, organizations, roles, and user groups to make sure the right content gets to the right people and that restricted content is protected in order that only the proper users can view it.

The collaboration environments provided by Liferay can be used among *p-medicine* user groups: the framework provides applications for document sharing and communicating of portal users. The **Document Library** portlet allows the users to create discussion threads next to the documents they need to talk about. This application provides a facility for sharing documents with the *p-medicine* users. It keeps a complete version history of all documents and is integrated with the portal's permissions system. This integration allows the owner to grant access to shared documents or prevent some of the portal users from accessing sensitive documents.

Below are some other important Liferay's built in applications that can be used in *p-medicine*. They already work with user-management and security features:

- The **Message Boards** application is a complete implementation of a web-based discussion forum software.
- The **Wiki** application is a full-fledged wiki which can be used for whatever purpose suits you. It is integrated with the Message Boards application, because it borrows functionality from that application to provide comment threads at the bottom of Wiki articles. Those threads use the users' profile information (including pictures) to uniquely identify them in a consistent way throughout the portal, which is yet another level of the integration.

- The **Calendar** application is totally integrated as e.g. with email notifications and more. It is a full-fledged calendar application that supports export and import of calendar data from other applications. This portlet can be used for either individual *p-medicine* portal users or for user groups.
- The **Activities** application can capture and display activities unique to the portal. When the portal user shares a document with a community or adds the Wiki application to a page, the Activities application can report on what the user did (and even provides an RSS feed of activities).
- *p-medicine* users can use the applications of the portal's collaboration platform as e.g. **chat**, **e-mail**, and **blogs**.
- **Tags** and **Categories** are important tools that can be used to help organize information on the *p-medicine* portal and make it easier for the users to find content that they're looking for. Tags are keywords that can be attached to a piece of web content in the portal in order to help portal users find content. Categories are a hierarchical organization of content whose structure can be defined by administrators. The tags and categories assigned to the *p-medicine* portal content define its metadata.. This can be used by the portal's search engine to score the results of a search, enabling *p-medicine* users to find content that is most relevant to their search. Tags can be created on the fly or they can be selected from the existing library of tags by the creator of the content, and it is important to tag the content whenever it is created. For the categories in the portal concepts from HDOT can be used. For navigating in the portal the *p-medicine* users can use two portlets: the **Tag Cloud** and **the Tag Navigation.**

The described functionality is built in to the Liferay framework. *p-medicine* users can pick the applications and drop them onto their pages.

5.2 Integration approach

The *p-medicine* portal is a single web-based environment from which most of the *p-medicine* applications introduced in section 3.1.2 can run. These applications will be integrated together in a consistent and systematic way.

The *p-medicine* portal users will not need to sign in to every application separately. With a Single Sign-On functionality the login procedures of different applications will be centralized in one system, accessible with only one password. After entering the right password, only the applications that the user is authorized to will become available. Within each application the internal permission system will be used. The Single Sign-On solution will be based on the Security Assertion Markup Language (SAML) 2.0[43] with the open source Shibboleth[44] System as identity provider implementation (see section 5.5.1).

The integration of tools and services developed for *p-medicine* can be performed on different ways:

- A tool for *p-medicine* can be provided as a portlet which is a web application that runs in a portion of a portal's web page (see section 4.1). This way is preferred for the *p-medicine* tools which should be developed for the project (e.g. Oncosimulator; Data Warehouse).
- For tools that offer a programmatic interface (e.g. based on Web services, REST, etc) and have been semantically annotated and registered with the tools repository, the *p-medicine* workbench portlet can be used to present a tool specific user interface, invoke them, and retrieve their results.

[43] SAML is an OASIS standard that defines an XML-based protocol, making it possible to exchange authorisation and authentication data between one or more security domains.
[44] http://shibboleth.net/

- For already existing *p-medicine* tools available as web applications (e.g. ObTiMA) several integration possibilities exist:
  - It is possible to re-write the application as a portlet, if it is rational.
  - It is possible to create a *middleware* portlet for interacting with the application (better using web service).
  - It is possible to wrap the application as an *OpenSocial* gadget for displaying it in an *iframe*[45].
  - It is possible to create a portlet that integrates the application either using an *iframe* or an *HTTP proxy* (e.g. using Liferay's *WebProxy portlet*). *iframe* is a portlet which makes it possible to embed another HTML page inside the current page. Furthermore, the user can navigate through the embedded page without loosing the context of the portal page. The iframe portlet uses the HTML iframe tag that is part of HTML 4 and is supported in all major browsers. The iframe portlet will adjust to the size of the HTML page if that page is hosted in the same server. The browser will add scrollbars if the embedded page does not fit within the size of the iframe. In this case a solution to transfer the authentication between the portal and the application is needed.
  - If the existing application is a JavaEE application, a Liferay's technology called Web Application Integrator that allows prototyping the integration by deploying the WAR file of the web application (e.g. by using the control panel or by copying it to the deploy directory) can be used. As a result the portal framework will automatically create a portlet that integrates the application using an *iframe*.

For developing or adapting the *p-medicine* tools as portlets, the developer has to use the ANT[46] based *Plugin SDK* provided by Liferay.

Developers can write *JavaScript* code in the portal by using the *JavaScript* libraries *jQuery, Dojo, YUI, Sencha* (former *ExtJs*), *Sproutcore, AlloyUI* (from Yahoo) etc.

For achieving more consistent look and feel for the applications integrated into the portal we recommend to use the colour scheme used for the *p-medicine* project page[47] by implementing of the styles in the new developed portlets: white text on dark red background (colour codes: #8B0303 and #C1132B). We have utilized this colour scheme by the development of the *p-medicine* portal layout (see section 5.6).

PHP and *Ruby* developers can use the *plugins SDK* to deploy the PHP and Ruby applications as portlets into the portal (examples are in the public Liferay *SVN* in the folder *plugins/trunk)*.

A built-in permission system in the portal allows defining permissions to each portlet. Therefore the tools and services which will be directly integrated into the portal (see the section 5.5.2) should be provided as portlets or *OpenSocial* gadgets for the integration. Portlets should be represented as *.war* files, OpenSocial gadgets - as *xml* files.

The role assigned to a *p-medicine* portlet developer allows him to deploy the developed portlet into the portal (role *DEVELOPER* - see section 3.1.1). He can do it by using the management area of the portal or by copying the developed portlet to the *deploy* directory of the portal. All related software modules needed for running of the developed portlet in the portal should be installed on the portal server by the developer or the developer should

[45] iframe is provided by Liferay as a portlet
[46] http://ant.apache.org/
[47] http://www.p-medicine.eu/

provide the detailed description of the installation steps to the technical portal administrator of the portal (IBMT).

The *p-medicine* portal will aggregate the set of portlets representing the *p-medicine* tools that are to appear on any particular page and display them properly to the *p-medicine* user. The portal administrator can arrange the *p-medicine* tools on the portal's pages in the way that works best for the portal users.

5.3 Development of plugins for the portal

Liferay offers a very high level of customization: we can modify anything in Liferay, from simple functionality changes all the way to make an own product out of it.

To add more functionality to the portal we have to develop plugins for the portal. Portlets, themes, layout templates, hooks, and web modules are plugins for the portal.

- It is possible to write own **portlets** so that the *p-medicine* applications can be added seamlessly to the portal web site's pages in a way that is indistinguishable from the built-in portlets. Portlet plugins written according to the Java standard are deployable on any portlet container.
- Liferay's **layout templates** can be customized for the *p-medicine* to customize the page layouts be for the *p-medicine* project.
- **Hooks** are the portal components, which can be used to modify portal properties or to perform custom actions on start up, shutdown, login, logout, session creation and session destruction.
- **Ext plugins** provide the possibility to override anything in Liferay with own functionality. It is not recommended to write Ext plugins for users without advanced skills. Even though Ext plugins are deployed as plugins, the server must be restarted for changes to take effect. For this reason, Ext plugins should not be combined with other types of plugins.

  In the *p-medicine* portal this approach has been used for the implementation of the *p-medicine* security framework.
- With **themes**, Liferay's look and feel can be adapted for the *p-medicine* project. Themes are custom HTML and CSS applied to the page. Using this approach we have developed a theme for the *p-medicine* look and feel.

Plugins can be developed using the Plugins SDK with Ant and an IDE (e.g. Eclipse[48] or NetBeans[49]) or a text editor.

Most of the plugins can be hot deployed (that is, deployed while the server is running) and are available immediately as portlets, layouts and themes. This prevents any server downtime for deployments. Only for deploying of ext- and hook-plugins the server needs to be restarted.

Examples for the development and deployment of portal plugins are available in the Appendix 2.

5.4 Community based structure of the *p-medicine* portal

At its most basic level the *p-medicine* portal has users, and these users can be grouped together by various ways providing a powerful mechanism for the portal administrator to configure portal resources and security in a consistent and robust manner. We have decided to use the community based structure of the portal. Portal users belong to Communities that

[48] http://www.eclipse.org/
[49] http://netbeans.org/

have a common interest (e.g. different portal users are members of a community called *SIOP Community* that has a common interest in the nephroblastoma diseases). Communities and Roles are described below. Other possible structures in the portal (e.g. Organisations, User Groups and Teams) could be also used in the project if needed.

Communities are collections of Users who have a common interest. By default, users can join and leave communities whenever they want, although the portal administrator can change this so users are assigned to communities (or invited) by community administrators. Membership in communities gives users access to the pages in the communities of which they're members. Each community can have a specific layout and an own set of available pages containing different *p-medicine* tools.

There are three types of Communities:

- An **open** (the default) community allows portal users to join and leave the community whenever they want to
- A **restricted** community requires that users are added to the community by a community administrator
- A **hidden** (private) community is just like a restricted community, but it is not visible in the list of communities. It is visible only for the community administrator which can manage the community users (add, delete)

We suggest having only communities with restricted membership in the *p-medicine* portal.

Communities for specific domains (e.g. ALL, SIOP, Breast Cancer) or specific user collections (patients, scientists, developers) will be created for the *p-medicine* portal dynamically. The portal framework provides additional default communities:

- One community for all portal users - it is a default main portal community, we can call it the *p-medicine* portal community. All portal users will be automatically members of this community. This main community will have
  - public pages (e.g. 'Welcome' page) which are visible also for not registered users.
  - private pages (visible only for logged in users)
- Communities for each portal user (default user communities) - each user is automatically a member of the default user community (called *my community*) which contains only one user where his own public and private pages are stored.

Roles described in section 3.1.1 can appear inside communities. This means that though each community in the portal has this role with its configured permissions, membership in this role is different for each community. There are two kinds of roles:

- Portal Roles
- Community Roles

Roles are used to define permissions across their scopes: across the portal or across a community.

Details about participation in and management of communities are provided in the sections 5.6.1 and 5.6.3.

5.5 *p-medicine* components for integration into the portal

We have already described the basic functionality of the *p-medicine* tools and services in section 3.1.2. Below we describe the integration of these components into the portal in more detail. First of all the integration of the domain and scenario unspecific basic tools which will be used for the whole project (e.g. security framework components) is explained and then

the integration of the domain specific tools, models and services (e.g. tool for management of clinical trials) is provided.

The domain specific tools and services which should be integrated into the portal are divided into the following two groups:

- *p-medicine* tools that can be integrated **directly** into the portal as plugins as described above
- *p-medicine* tools that are used as parts of other tools accessed from the portal for performing user workflows which are therefore **indirectly** integrated into the portal.

5.5.1 Security framework components used in the portal

Major **authentication** components in the *p-medicine* security architecture are:

- The Identity Provider (IdP) which is a service provider within a federation responsible for authentication. It provides identity assertions to other service providers.
- An Identity Consumer which is a software component that is part of a service provider. It consumes the assertions provided by the Identity Provider. It will verify the received assertion and pass it to the service provider's application layer.
- A User Enrolment & Management Service where users can be enrolled, revoked, edited, etc. (part of the user and access management in Figure 1).
- A Secure Token Service responsible for issuing identity and delegation tokens for REST clients.

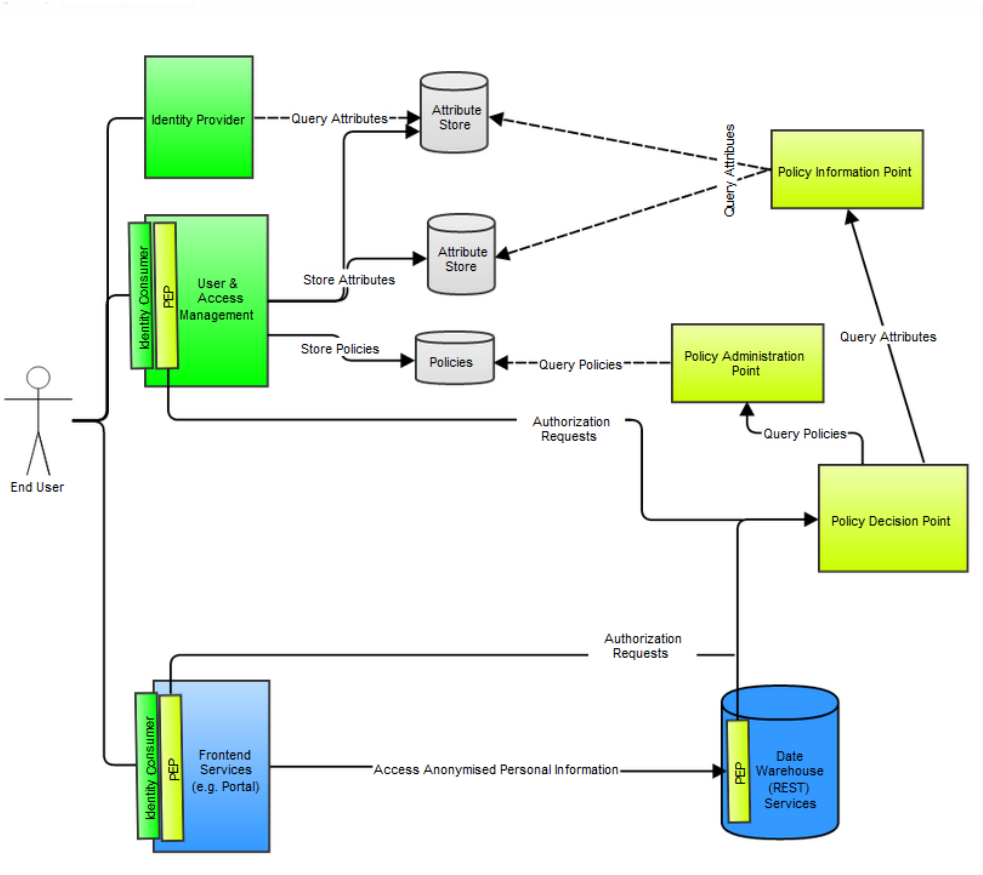

Figure 1: *p-medicine* security architecture overview

Major **authorisation** components in the architecture are:

- A Policy Enforcement Point (PEP) is a software component which requests and enforces authorisation decisions.
- A Policy Decision Point (PDP) is an entity that makes authorisation decisions. A PDP accepts authorisation requests and will make a decision based on policies fetched from a Policy Administration Point (PAP).
- A Policy Information Point is an endpoint which provides missing information to a PDP i.e. attribute information. For example if a policy requires information on a specific attribute which has not been provided with the authorisation request, a PDP might request a PIP for information on that attribute.
- A Policy Administration Point is an endpoint which manages policies. It will provide a PDP with all policies required to produce an authorisation decision.
- An Authorisation Rule (Policy) Management Service where authorisation rules can be configured generating authorisation policies. It is a part of the user and access management (Figure 1).

The central *p-medicine* authorisation framework is mainly targeted to backend services which access sensitive (patient) data. Therefore the initial focus of Liferay is to integrate with the *p-medicine* authentication components. Liferay will hereby act as an identity consumer of the *p-medicine* identity provider. The *p-medicine* user enrolment and management services will be integrated into Liferay to provide a seamless user experience.

5.5.1.1 Identity Provider and Consumer

To integrate with the *p-medicine* identity provider Liferay needs to act as a *p-medicine* identity consumer. To be able to register itself as a service provider (SP) to the *p-medicine* identity provider (IdP), Liferay must, as described in D3.4:

1. support at least the following bindings of the SAML Web Browser SSO Profile:
   a. for the authentication requests from Liferay to IdP the HTTP redirect and HTTP POST binding must be supported.
   b. for the authentication responses from IdP to Liferay the HTTP POST and HTTP artefact binding must be supported.
2. support at least the HTTP redirect and SOAP bindings of the Single Logout Profile for both IdP- and Liferay-initiated single logout:
   a. for IdP-initiated single logout the HTTP redirect and SOAP binding.
   b. for Liferay-initiated single logout the HTTP redirect and SOAP binding.
3. must publish a SAML metadata file which describes the bindings Liferay supports, its endpoints, certificates and keys

Once Liferay adheres to the above requirements it can register itself as service provider to the *p-medicine* IdP after which users can use a *p-medicine* identity to authenticate themselves on Liferay.

The first time a user authenticates on Liferay, after the identity consumer has verified the token's validity, a new Liferay user will be created locally in the Liferay database with the identity information extracted from the SAML token (e.g. first name, last name, email address, roles, and organization membership). As the IdP acts as authoritative source of identity information on subsequent authentication requests of the user with a SAML token, the information stored in Liferay is updated with the updated identity information received from the token.

5.5.1.2 User management in the portal

The user enrolment and management service is an important component of the *p-medicine* security framework. It is responsible for user enrolment and user identity and credential management. The service encompasses

1. **an administration** site where user administrators can register new users, manage existing users, manage and create organisations, manage organisation membership and enable, disable and remove users.
2. **a user profile** page where users can manage their own identity attributes such as first name, last name, email address.
3. **a public registration page** where users can register themselves. A user who registers himself through the public registration pages first needs to be accepted by a user administrator before he can create credentials.
4. **an activation page** where users can choose a username and password. After a user is registered and, in case of public registration, accepted by a user administrator, the user receives an activation mail. Through a link in this mail the user is guided to the activation page where he can chose his username and password.
5. **credential management pages** where users can request username or password recovery.

This user management functionality is integrated into Liferay through direct links (e.g. for username/password recovery) and iframe portlets. As user management and organisation management will then be handled by those portlets which render specific functionality of the *p-medicine* user management service the corresponding already existing functionality in Liferay should be disabled. As explained in 5.5.1.1 all changes to the central *p-medicine* identity are automatically propagated to Liferay during authentication.

5.5.1.3 CATS

The CATS (Custodix Anonymisation Tool Services) is a service responsible for the de-identification of clinical data files (including imaging data). CATS de-identifies data files based on their mime type and a set of pre-configured transformation rules (privacy profiles).

CATS scans for patient identifying information. This identifying information can be cleared or replaced by a pseudonym. The reference between the patient and the pseudonym can be stored locally or on PIMS. PIMS (Patient Information Management System) is a tool that stores patient/pseudonym references and matches patient records coming from different sources. CATS can also encrypt (parts of) the result file to keep sensitive information confidential.

This application can be started from the portal but it runs locally at the data source (e.g. the clinician's laptop). There the data is pseudonymized a first time before uploading it to the CATS server[50] and TTP for a second round pseudonymisation.

5.5.2 *p-medicine* tools and services which will be directly integrated into the portal

In the table below we introduce the p-medicine tools and services which will be directly accessed from the portal interface.

Tool/service	Integration Details
p-medicine	The *p-medicine* workbench application will be integrated as a portlet

[50] see D5.1 or D3.4 for information on data import flow

Workbench	into the portal. The *p-medicine* tools and services managed in the workbench application will be registered in the Tools Repository, a back-end service with tight integration with the *p-medicine* workbench, which will also provide REST interfaces for any other service that might need to interact with it. The *p-medicine* workbench acts as a front-end of the Tool repository and it provides a logical repository, where the tools can be categorized based on their functional groups, metadata about their methods and data types can be retrieved, and relevant architectural elements can be suggested based on their functional profile and their accompanying metadata. The workbench will also be used as a monitoring tool of the *p-medicine* architectural elements (AEs), by providing measurements and indications of the quality characteristics of each AE, by incorporating whenever feasible the testing clients of the AEs.
ObTiMA	ObTiMA provides a web based user interface for managing clinical trials and storing clinical information. ObTiMA will integrate push services for pushing selected data into the Data Warehouse. The data and images uploaded to the Data Warehouse should be anonymized by CATS before storing into the Data Warehouse. The integration of ObTiMA is described in the section 6.2.4.
p-BioSPRE	The metabiobank p-BioSPRE will provide a search interface that will enable authorized users to search for biomaterial according to the standard biobank data set. According to the legal requirements the biomaterial data provided in p-BioSPRE is anonymized. For request the biomaterial p-BioSPRE provides request forms that enable the user to specify the amount of biomaterial he needs and to define his research project in detail. P-BioSPRE forwards the request to the biomaterial owner, who can then contact the researcher and decide about the request. p-BioSPRE will be a web application which will be accessed via the *p-medicine* portal. The application's management of user roles and rights will be compliant to the *p-medicine* security framework.
User data management in cloud services and OpenStack Object Storage	The *p-medicine* Cloud Storage System provides REST interfaces for managing file storing in the cloud environment and it is built based on the open source cloud storage technology OpenStack Object Storage (Swift) technology. It provides access to reliable storage space taking into account requirements from different end user scenarios: long term data preservation on the one hand, as well as fast access to application data in the workflow execution. The OpenStack Object Storage is ideal for cost effective, scale-out storage. It provides a fully distributed, REST API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. It provides redundant, scalable object storage using clusters of standardized servers capable of storing petabytes of data. Every type of file system supporting extended

	attributes (xattr) could be used below to provide storage space so there is a straightforward possibility to integrate other services below. Object Storage is not a traditional file system (block storage), but rather a distributed storage system for static data such as virtual machine images, photo storage, email storage, backups and archives. Having no central "brain" or master point of control provides greater scalability, redundancy and durability. Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally simply by adding new servers. Should a server or hard drive fail, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used.

Oncosimulator	This tool will contain different modules for the integration into the portal:
	• Three components for the three cancer branches mainly addressed in the *p-medicine* project which will offer the user the opportunity to download the appropriate binaries in order to execute them locally and possibly supporting material (documentation, execution instructions etc.):
	  ○ OS-BRCA - the breast cancer branch of the Oncosimulator
	  ○ OS-WT - the Wilms tumour (nephroblastoma) branch of the Oncosimulator
	  ○ OS-ALL - The acute lymphoblastic leukemia branch of the Oncosimulator
	• Web-Oncosimulator - It is a web integrator of OS-BRCA, OS-WT, OS-ALL. The component will offer the user the possibility to run any of the above versions using the *p-medicine* resources (one possibility is for those resources to be located at ICCS).
	• OncoRunner (or OncoAccelerator) - It is a single portal component or a simple web link (will be later decided by PSNC) to the front end of a web application running models on a dedicated cluster and possibly returning visualized results. This single portlet will make use of a service which will communicate with the dedicated cluster, and will retrieve and present the execution results. The codes that will run on the dedicated cluster(s) will be the codes of OS-BRCA, OS-WT and/or OS-ALL which might have been adjusted to the specific cluster environment. This portlet will include functionality of the modules Oncosimulator service, Optimized Oncosimulator application and Oncosimulator environment described in the D15.1.
	• Special Molecular Biomechanism Models - these models can be plugged into the *p-medicine* Oncosimulator model (at a later stage)
	• Special Cellular Biomechanism Models - Details of these models will be defined at a later stage of the project.

Data Mining Tools	The Data Mining environment is a collection of services that can be accessed uniformly from the *p-medicine* portal interface. The services reuse algorithms encoded in the statistical language R, and a workflow

	execution engine allows the users to execute the workflows shared on public workflow repositories. According to the description of the Data Mining services in D11.1 a data-mining task in *p-medicine* can be described as workflow or as workflow pattern: • A data-mining **workflow** is executable in the *p-medicine* and can invoke any of the *p-medicine* data mining services and computational execution environments. • A **workflow pattern** is not executable and serves as a template data mining workflows. The Data Mining tools are already initially integrated into the *p-medicine* portal. Details about their integration and usage are provided in section 6.2.3.
Data Warehouse Access Tools	These tools will provide an HTML-based user interface integrated into the *p-medicine* portal for browsing and querying one or more instances of the *p-medicine* data warehouse. The interface will allow browsing of data in a data warehouse structured data stores based upon terms from the HDOT ontology, and advanced querying of the structured data using SPARQL. The audience for this will be mostly technically literate, such as auditors and *p-medicine* model developers. Therefore, the interface will be fairly basic in terms of user experience
Push Services	Push services will seamlessly integrate the data retrieved from HIS into the *p-medicine*'s data warehouse. They integrate the data semantically in terms of the HDOT ontology by using the tools of the semantic layer so that the data in the data warehouse can be stored in a form compliant to this ontology. The data warehouse supports the secondary use of the data by providing an SPARQL interface that allows querying the data in terms of the HDOT ontology. This interface supports the seamless reuse of the data by the various analysis tools and services of the *p-medicine* platform, as e.g. data mining, simulation and decision support tools. In terms of the origin of the data two kinds of scenarios will be supported: • Pushing data from external data sources into the DW: owners of external data sources will be able to upload data into the data warehouse. A user-friendly portlet for the *p-medicine* portal will be implemented to support the upload-process. • Pushing data from ObTiMA into the data warehouse. ObTiMA as part of the *p-medicine* platform is integrated into the portal (see section 6.2.4). A detailed description of the pushing approach is given in the Deliverable D4.2.
Ontology Aggregator	The tool will allow users to find and select relevant parts of semantic resources and semi-automatically integrate those parts under HDOT, i.e. users can re-use pre-existing semantic resources and extract classes or terms from them to design specific HDOT modules for their

	specific needs. Use cases for this are tightly connected to the use case for the annotation tool (see below), e.g. users will be able to invoke the aggregator tool if they attempt to annotate their data (schema) and do not find appropriate annotation values in HDOT or HDOT modules, so that they can invoke the ontology aggregator tool and integrate those needed for their specific purpose in an semi-automated way under HDOT. This will lead to continuous extension of the domain coverage of HDOT and the resulting enriched HDOT modules are made available for future use. Additional details, characteristics and features of this tool will be provided later in the project. Details, characteristics and features of this tool as well as its integration approach into the *p-medicine* portal will be provided later in the project.
Ontology Annotator	The Ontology Annotator is a graphical tool aimed at end users (specifically data managers) for easily creating annotations for the databases integrated in the Data Warehouse. This annotation contains the necessary information for automatically translating registers pushed into the Data Warehouse into an HDOT compliant form. These annotations consist of pairs of semantically equivalent views from the database and from HDOT. The pair of views is completed with links of equivalent elements of the views. There exists an XML format for describing these annotations, but this is far too complicated for end users to learn and it would take too long to annotate databases by hand. The Ontology Annotator allows creating annotations in a simplified and efficient manner. The tool provides graphical representations for both the database being integrated into the *p-medicine* platform and HDOT, and allows building pairs of semantically equivalent views by clicking and dragging elements. Creating annotations for databases not only involves defining equivalent views. In order to make this information really useful, further annotations are needed. Natural language descriptions of the equivalences are useful for subsequent editing and curation of the annotations. In addition, the construction of the individual views requires some sort of comprehension about RDF models. One priority of the Ontology Annotator is to hide all this complexity from users. Hence, the tool will be designed with frames and expect inputs that any average user is familiar with. The tool will include video tutorials describing the view construction process with examples and tooltips.
Patient Empowerment Tools	The following patient empowerment modules will be integrated to the portal: • **Questionnaire Application:** When a patient logins to the portal he will be able to fill a questionnaire. This questionnaire will be a short and easy to fill out instrument to measure four broad areas: perceived health state, psychological aspects, psycho-social as well as cognitive aspects, all investigated by different sub-dimensions. • **PHR:** The PHR system will store all information related to the patient, such as information on diagnosis and biobank data and will display information for the possible clinical trials. Using the PHR the patient can follow the journey of his/her data and based on it begin the decision process.

	• **Profiling Service:** This service will get as input the answers to the questionnaire and user navigation details to the PHR. Then intelligent profiling mechanisms will produce a patient profile which will be used to personalize the navigation of the user in the PHR. • **E-Consent**: This module is responsible for managing consent and re-consent. A patient will be able to provide, withdraw and manage consent for clinical trials. Moreover, since the consent might be changed or resubmitted after some time, temporal information about consent signing will be stored as well. Finally, according to the answers to consent the specific patient's data will be used or not at the analysis. • **Recommendation Service (Trial Discovery):** This service will combine profiling information with trial descriptions from the data warehouse to identify trials that a patient could be possibly enrolled. This service will enable the shared decision making among patients and doctors. From the security point of view: when a patient gives his consent on the use of his medical data (or subsets of this medical data) for a specific trial (possibly with added limitations concerning exportability, duration, etc.), a consent policy will be generated and stored on the Policy Administration Point. This policy will allow access to the patient's data for the specific trial within the limitations as given by the patient. The Policy Decision Point will then, when an access request is made, fetch the authorization and consent authorization policies. This way access is given if the user, who requests access, has sufficient rights and if the patient accessed has given his consent.

5.5.3 Tools and services which will be indirectly integrated into the portal

Not all tools and services developed or used in the *p-medicine* environment will be directly integrated into the portal. Some of them will be used as parts of other tools accessed from the portal for performing user workflows and therefore indirectly integrated into the portal.

The description of these tools is provided in section 3.1.2. Below some details are given about their integration into the portal:

Tool/Service	Details
Data Translation Service	The Data Warehouse used in the *p-medicine* technological platform is responsible for storing the data that the different portal users will eventually access and analyze. These data will include results from clinical trials, records from private institutions databases and even information available from public biomedical repositories. In order to provide a homogenized access over all these data, all handled records will be transformed into an HDOT compliant form at the moment of loading them in the DW. This task must obviously be performed in an automated manner. The Data Translation Service is in charge of performing this operation. This service is invoked whenever any data is pushed into the Data Warehouse, and performs a dynamic translation of all registries being pushed. This service will take into account coherence of data in order to correctly realize merging of similar data and transformation of literals. The Data Translation Service operations are performed in a transparent

	manner, hiding the details from the users that perform the data push. Translation of data is a computationally expensive process. Data pushes are expected to involve large amounts of records. Therefore, efficiency will be a priority in the design and development of this tool. The process will involve multiple queries to the Data Warehouse in order to assess record similarities. Minimizing these accessions is a key to avoid high latency. Hence, the service will implement some sort of intelligent data treatment that reduces this penalty and allows users to efficiently push their data into the portal.
Sync Services	The sync services are developed as a module of ObTiMA. The sync services will retrieve the data not directly from the hospital information systems but from the *p-medicine* Data Warehouse, since this data is already integrated compliant to HDOT.
p-BioBank Wrappers	Details about the integration of these tools will be defined in the later period of the *p-medicine* project development
ObTiMA Trial Biomaterial Manager	Details about the integration of this tool will be defined in the later period of the *p-medicine* project development
dcm4che[51] DICOM server (see the part "Services for storing of medical images" in the section 3.1.2)	It is the most promising candidate for the DICOM server. This is a JEE and JMX system which is deployed within the JBoss Application Server. It includes a DICOM archive and image manager and a PACS, when coupled with a viewer such as OsiriX, K-PACS[52], ClearCanvas. It is implemented in Java, making it fairly easy to integrate it with any Cloud technology and provide data archiving in the Cloud instead of local storage. It will be integrated with *p-medicine* security system using SAML and central identity management service. Acting as an archive, dcm4chee is able to store, query, and retrieve any type of DICOM object. In addition, support is included for MPPS (Modality Performed Procedure Step), GPWL (General Purpose Worklist), MWL (Modality Worklist), Storage Commitment, Instance Availability Notification, Study Content Notification, Output Content to CD Media, Hanging Protocols, and more. More details about the server and available modules and interfaces for the server is provided in the D7-1 "Report on overall system design including VPH-Share D2.2 and indicating its impact"
DrEye	Dr Eye is a flexible and easy-to-use DICOM cancer image analysis and visualisation platform for quick and precise identification and delineation of tumours in medical images. The design of the platform is clinically driven in order to ensure that the clinician can efficiently and intuitively annotate large number of 3D tomographic datasets. Tumour regions can be segmented automatically or annotated, labeled, deleted, added and redefined. A plug-in SDK mechanism allows any user to develop tools related

[51] http://www.dcm4che.org
[52] http://www.k-pacs.net/

	to tomographic image analysis/visualization and seamlessly incorporate them to DrEye. Within *p-medicine* the tool is being extended to analyse contrast enhanced MRI and diffusion MRI data and compute candidate biomarkers related to perfusion. It is possible to use this tool in all the cancer studies that participate in *p-medicine* in order to compute candidate imaging biomarkers. Details about the integration of this tool will be defined in the later period of the *p-medicine* project development
Data Warehouse Programmatic Interface	The programmatic interface for a data warehouse (see section 3.1.2) will be linked to that particular warehouse. Since *p-medicine* may ultimately refer to many different data warehouses, it is not appropriate to make the programmatic interface as a portlet within the portal. We will aim to develop the programmatic interface as a standard Java portlet, however, to allow such integration in the future.
Workflows and Pipelines for Genomics Data	One of the emerging tools to create workflows that is particularly suited to, and contains useful tools for, next-generation sequencing studies is Galaxy[53]. This is a web–based tool that was designed to allow the user to implement, reproduce, and share analysis workflows. Importantly it allows the user to set up local or server-based instances, and the first integration of Galaxy and Taverna workflow environments[54] have been realized recently. Workflows and pipelines for genomics data in *p-medicine* are the Taverna/Galaxy workflows run by Data Mining portlets integrated into the *p-medicine* portal.
Decision Support Tools	A Clinical Decision Support System (CDSS or CDS) is an interactive computer software system designed to assist physicians and other health professionals with decision making tasks, as determining diagnosis of patient data. CDS tools serve physicians' needs to improve decision efficacy through the use of electronic systems.. Details, characteristics and features of these tools are introduced in more detail in D13-1. Implementation of the tool will be realized as a desktop application (off-line format). An integration approach into the *p-medicine* portal is not yet defined, but it could be possible to provide them (incl. their source code) for downloading in the portal.

5.6 User interface of the *p-medicine* portal

In this chapter we introduce the user interface of the *p-medicine* portal and describe how the portal can be used from the users' point of view.

[53] http://genome.cshlp.org/content/15/10/1451.full
[54] http://www.taverna.org.uk/documentation/taverna-galaxy

For the *p-medicine* portal we have developed a look and feel plugin which has replaced the original Liferay layout. Details about the plugin and it usage in the portal is described in more detail in section 6.2.1. The start page of the *p-medicine* portal is shown on the Figure 2.

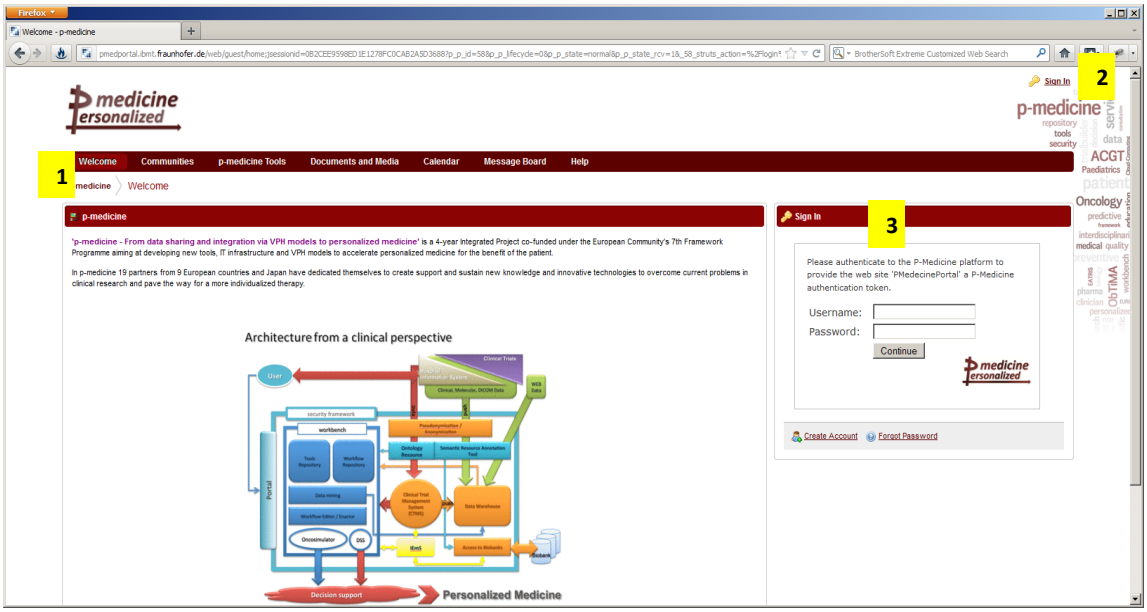

Figure 2: Start page of the *p-medicine* portal

5.6.1 Main p-medicine navigation menu

The main navigation menu (1, Figure 2) for the *p-medicine* users contains the pages:

- *Welcome* - a page with the information about the *p-medicine* project. The *Login* portlet is shown on the right side of the page (3, Figure 2). Here more portlets can be placed (e.g. with important announcements for the *p-medicine* users as well as with information about news and events). This page is visible for everybody (incl. guests).
- *Communities* - a page with (a) a list of communities where the user is member and (b) with the available user communities. This menu unit is explained below. This page is visible for everybody (incl. guests).
- *p-medicine Tools* - this is a set of pages for *p-medicine* tools and services (1, Figure 3). These pages will be displayed only after a user is successfully logged in into the portal. By selecting of these pages the user can use the tools and services included as portlets to the pages to perform the *p-medicine* scenarios described in the D2.2.
- *Documents and Media* - a page with the available *p-medicine* documents and media content (pictures, video, audio files...). This page is visible for everybody (incl. guests), but only the files according to user's permissions are displayed. The files can be opened for reading or for downloading.
- *Calendar* - a page where a *Calendar* portlet is presented. Authorized users can manage events for displaying to the portal users. This page is visible for everybody (incl. guests), but only the events according to user's permissions are displayed.
- *Message Board* - a page with the *Message Board* portlet where authorized portal users can create and manage announcements for the portal users. This page is visible for everybody (incl. guests), but only the messages according to user's permissions are displayed.

- *Help* - on this page we want to provide some help information for the portal users (e.g. user manuals). The portal users can open to read or download the information provided on this page.

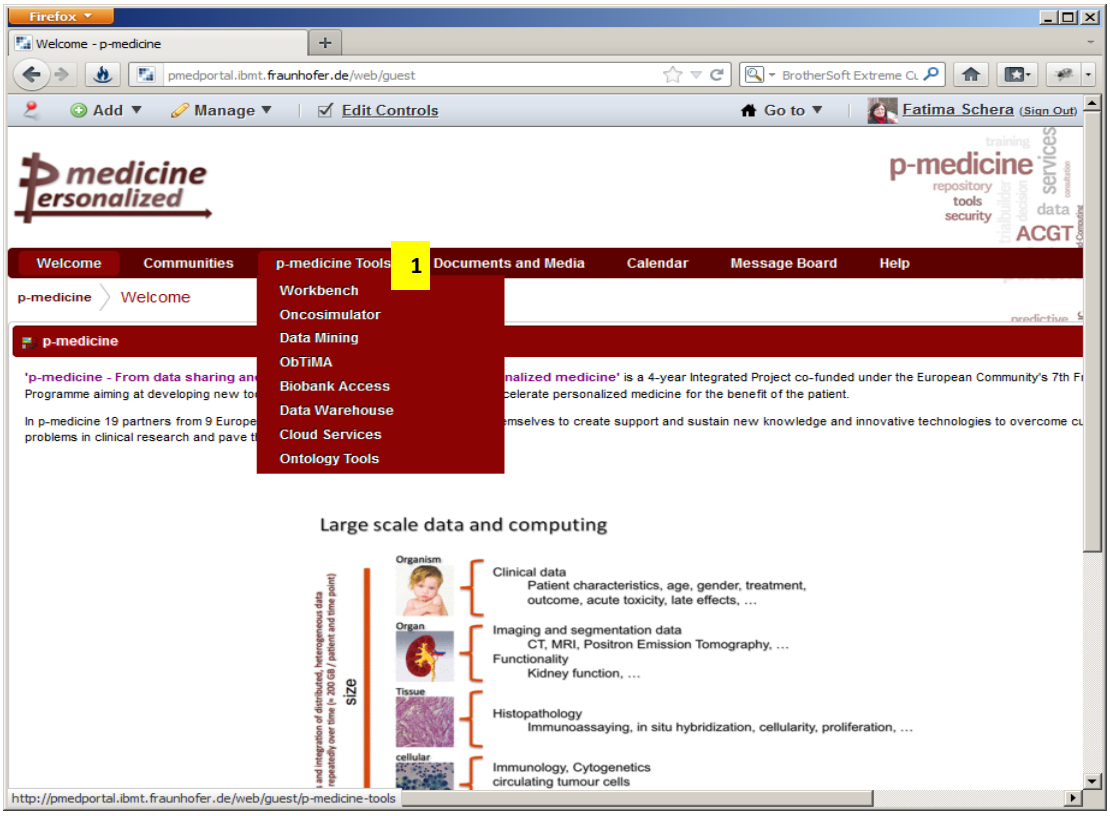

Figure 3: Main *p-medicine* navigation menu

Users get the experience of being able to sign in to the *p-medicine* site once and then navigate to the content they have access to. The Login portlet shown in Figure 2 can be also accessed by using the *Sign in* link (2, Figure 2).

5.6.1.1 Participation in the p-medicine communities

According to the community based structure of the *p-medicine* portal the users can participate or leave available communities. For doing this a user has to select the *Communities* menu where the *My Communities* portlet is displayed. The user can select either the *My Communities* register (1, Figure 4) displaying the communities where the user is member or the *Available Communities* register (2, Figure 4) which shows all *p-medicine* communities. In the first case the user can select a community where he participates for leaving the community by using the link *Leave* (3, Figure 4) or for switching to the selected community site using the link *Go to Public Pages*.

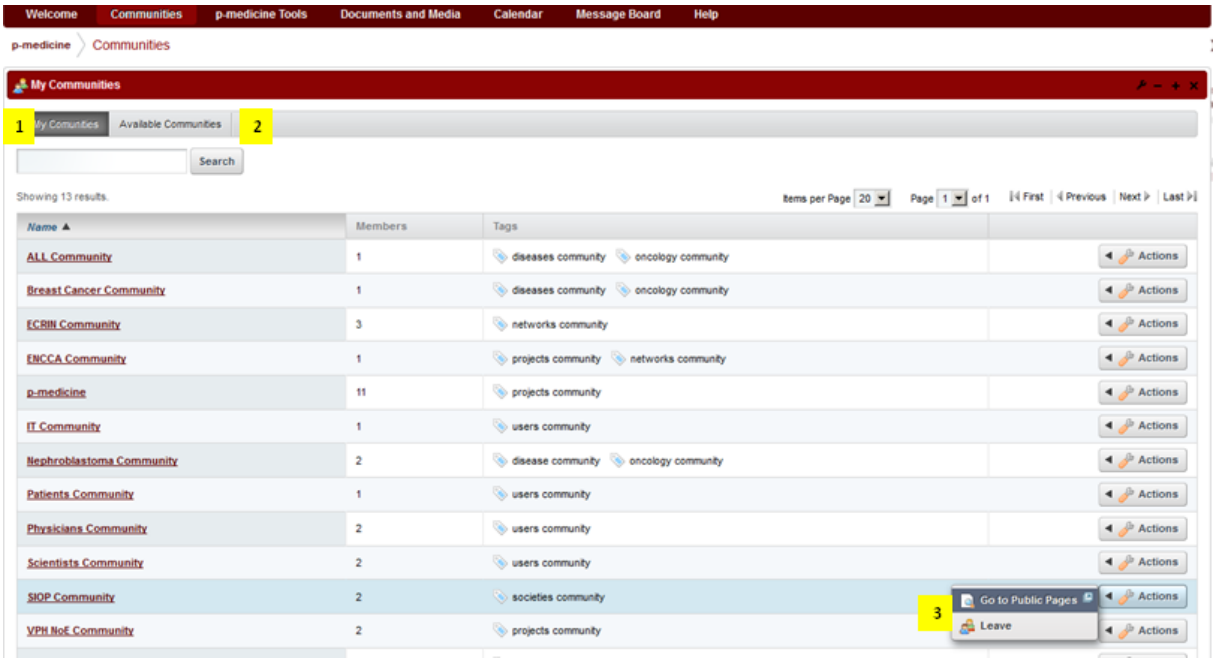

Figure 4: Page for displaying user's *p-medicine* communities

By selecting the *Available Communities* register (2, Figure 3) the list of the available communities is displayed. The user has three possibilities:

- join a community if the community has an open membership (3, Figure 5)
- request a membership if the community has a restricted membership (1, Figure 5)
- leave a community (2, Figure 5)

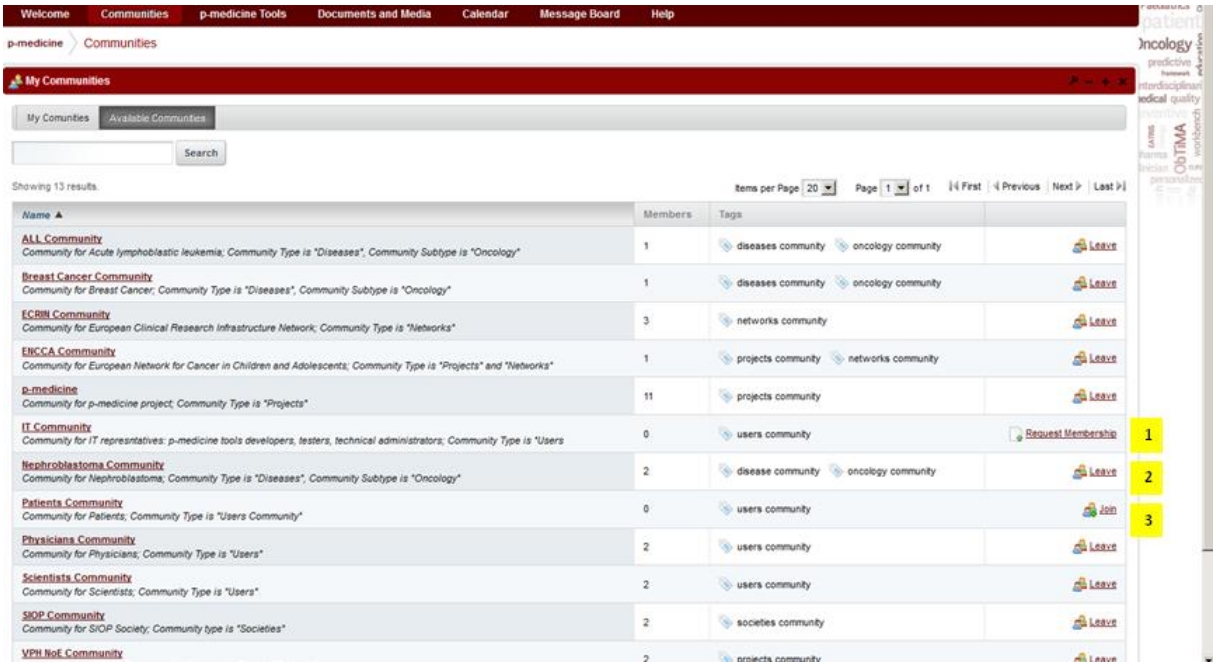

Figure 5: Page for displaying available *p-medicine* communities

After a login the portal displays the Dockbar, the second user navigation menu at the top of the page, which gives the user access to several other functions (1, Figure 6). The scope of the visible menu item depends on the user role in the portal.

5.6.2 Dockbar

There are different points on the Dockbar (see the picture below).

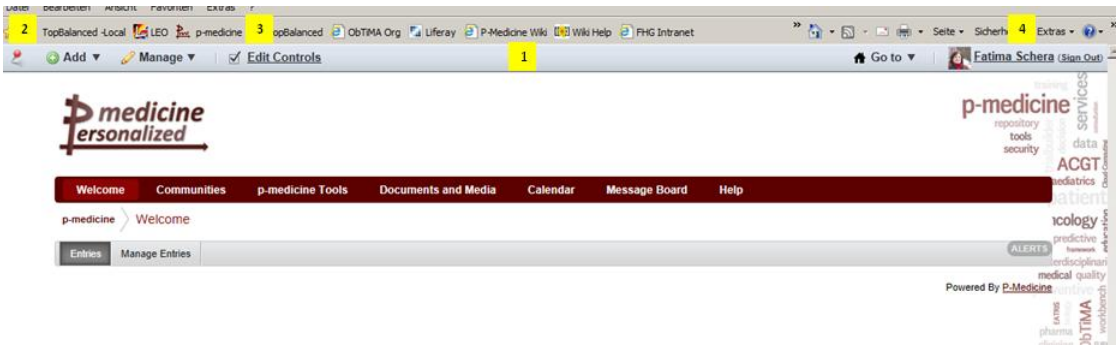

Figure 6: Dockbar - navigation menu bar

Here is a short explanation of the Dockbar's menu components:

- At the far left is a *pin icon* (2, Figure 6), which pins the Dockbar to the screen so that no matter how far down the user scrolls, it stays at the top of the screen. This can be helpful if the user is working with long pages and needs to use the Dockbar's functionality to add portlets to the bottom of the screen. This is a toggle switch, so the user can unpin the Dockbar by clicking the icon again.
- *Add* menu - we explain this menu in more detail below
- *Manage* (we explain this menu in more detail below)
- *Edit Controls* (3, Figure 6) - this function in the Dockbar isn't a menu; it's a toggle for the edit controls on the portlets. The portal (or community) administrator gets to see some icons in the title bars of the portlets on a page: an icon for closing a portlet, for minimizing it, and for the configuration menu you've already seen (you used this to change the scope of the Wiki portlet). If we're composing a page and would like to see something that more closely resembles what the portal users will see, we can use the Toggle Edit Controls link to turn off these controls.
- *Go To* (we explain this menu in more detail below)
- The final link in the Dockbar includes an access point to the user account information in the *Control Panel*. User Account menu item opens a page where the user can change his name and email address, upload a profile picture, and maintain all information about him. He can also sign out of the portal from here

Below we introduce the functionality of the main Dockbar components: *Add*, *Manage* and *Go To*.

5.6.2.1 Adding Applications

On the Figure 7 we show applications, which can be accessed from the *Add* menu.

Most of the functionality of this menu is for adding applications to the page. It can add pages, too. If we choose *Add > Page*, a new page is added next to the page we're on, and a field appears, allowing us to name the page.

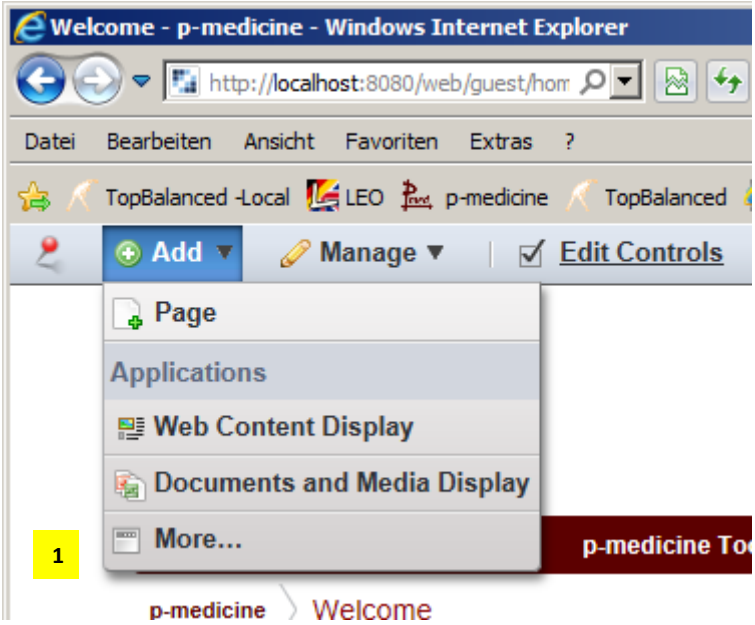

Figure 7: *p-medicine* menu for adding applications

Commonly used applications appear directly in the menu, but if the user wants to see the whole list, he has to choose the *More* submenu (1, Figure 7). Doing so pops up a fully searchable categorized view of all the applications that have been installed in the portal by default (1, Figure 8). The *p-medicine* applications will appear in this list, e.g. Data Mining Tools (2, Figure 8).

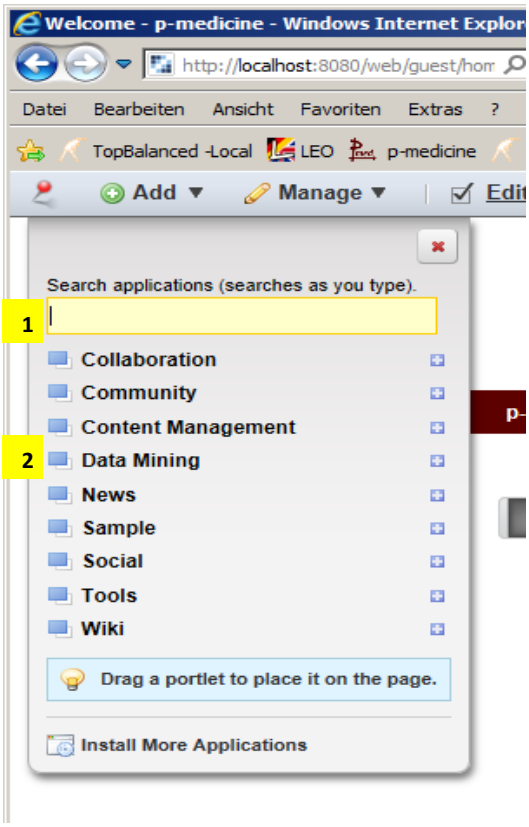

Figure 8: p-medicine menu for adding more applications

The user can add an application to a specific column of the shown portal page by dragging the application off the Applications window and dropping it into the appropriate column.

5.6.2.2 Manage Pages, layout and more

The *Manage* menu is used to manage pages, page layouts, and more. This is where the user gets access to the interface that lets him group pages in the order he wishes - as well as nest them into subpage levels. He can also apply themes to entire layout sets or to single pages. Finally, if the user is a community administrator, he can change the logo for a community.

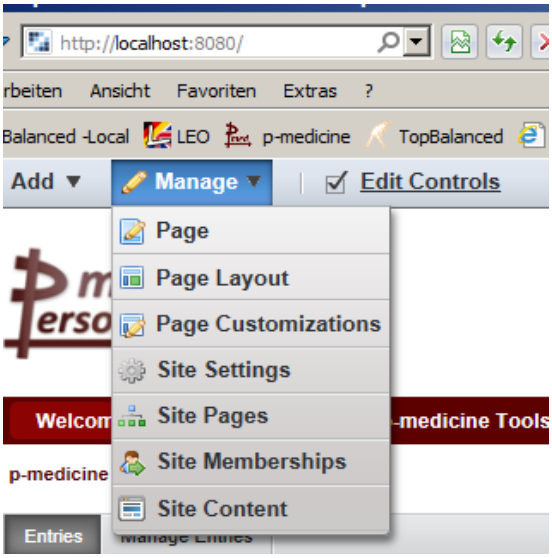

Figure 9: Dockbar menu "Manage"

5.6.2.3 Accessing the control panel and user communities

The *Go To* menu shown on Figure 10 is used to navigate to the various community and organization pages to which the user has access. Each community or organization name appears, along with its public and private layout sets, if they have them.

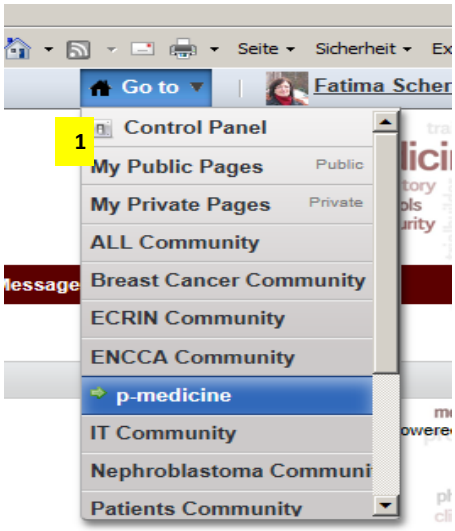

Figure 10: Dockbar menu "Go To"

One of the items in the *Go To* menu is the Control Panel (1, Figure 10), which is the central location where just about everything can be administered.

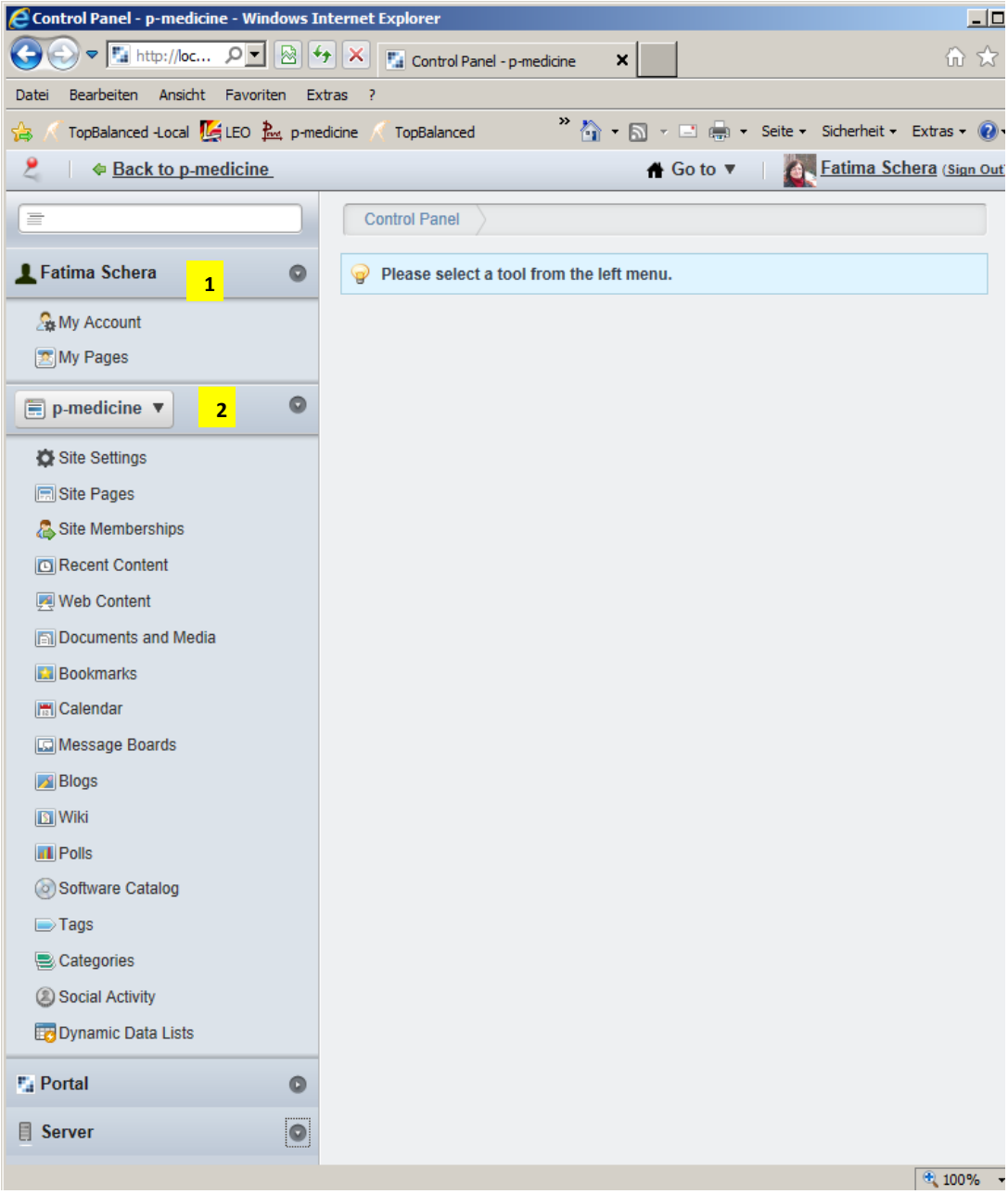

Figure 11: Control Panel (1)

The Control Panel is easy to navigate. On the left side is a list of headings with functions under them. The headings are in alphabetical order, but the functions are in a logical order. Figure 11 and Figure 12 show the Control Panel, which purposefully uses a different theme from the default pages.

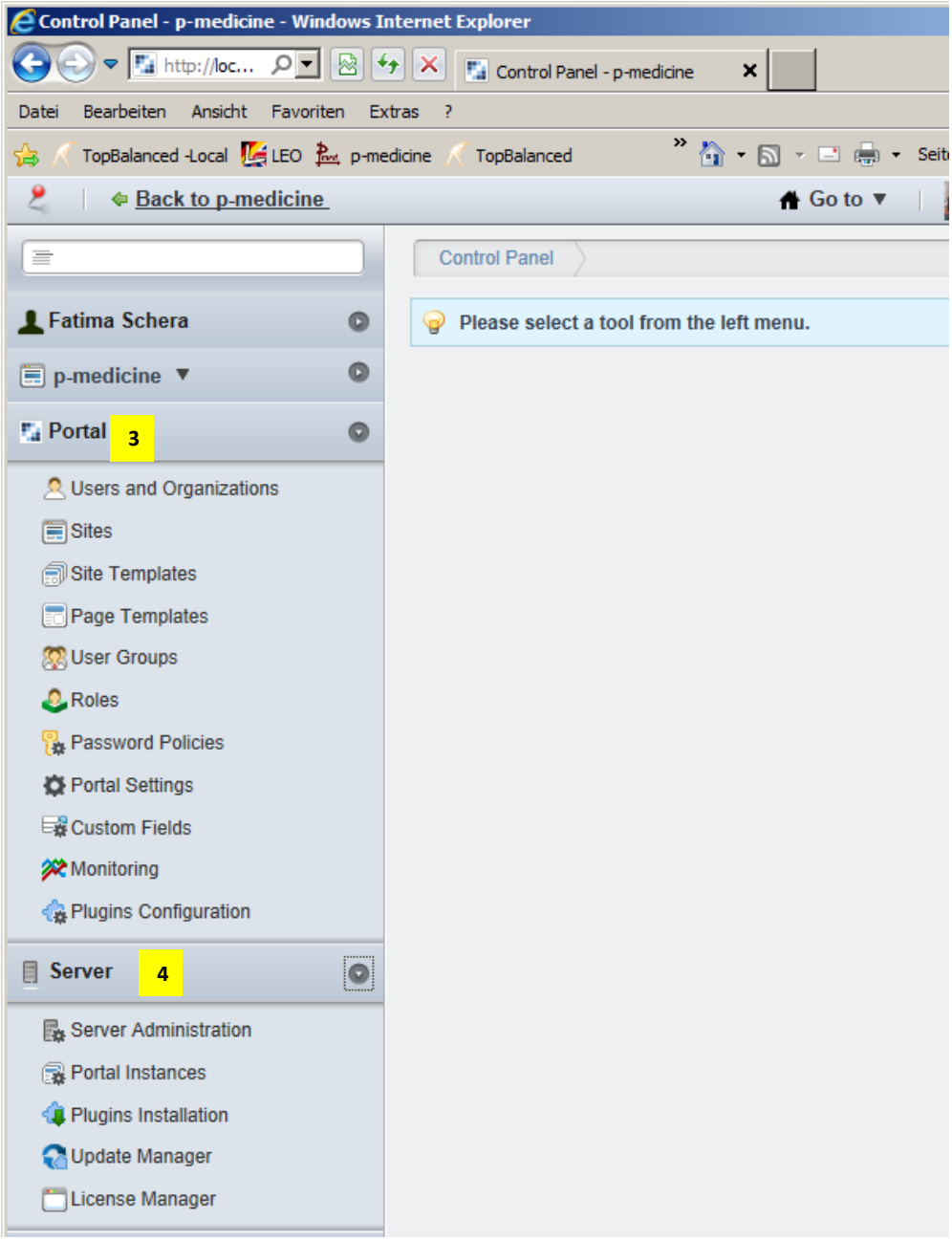

Figure 12: Control Panel (2)

The Control Panel's sections are as follows:

- The first heading **User Name** (1, Figure 11) is for the logged-in and is used to manage the user's personal space. Here the user can change the account information and manage the own personal pages.
- The **Content** section (2, Figure 11) contains links to all of Liferay's content management functions. The user can maintain web content, documents, images, bookmarks, and a calendar; administer a message board; configure a wiki; and more. These links are scoped for the particular community from which the user navigated to the Control Panel, but this can be changed using the select box.
- The **Portal** section (3, Figure 12) allows portal administrators to set up and maintain the portal. This is where he can add and edit users, organizations, communities, and roles as well as configure the settings of the portal.

- The **Server** section (4, Figure 12) contains administrative functions for configuration of the portal instances, plugins, and more.

5.6.3 Management of p-medicine communities

A new community can be created only by the portal administrator. For creating a new community the portal administrator has to go to the *Go To --> Control Panel* in the *Dockbar* (Figure 13).

On the left side of the Control Panel he has to choose the *Sites* function under the *Portal* heading (1, Figure 13) and in the displayed content on the right side he has to select *Add --> Blank Site* (2, Figure 13).

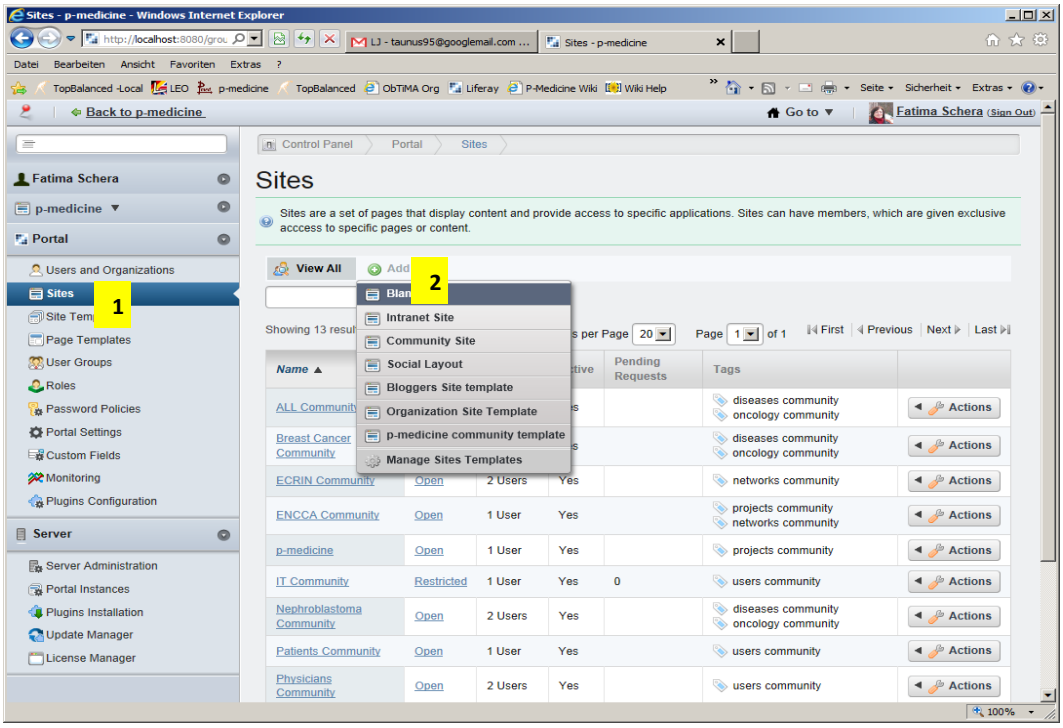

Figure 13: Communities view on the Control Panel

The portal administrator has to fill the field for the community name (1, Figure 14):

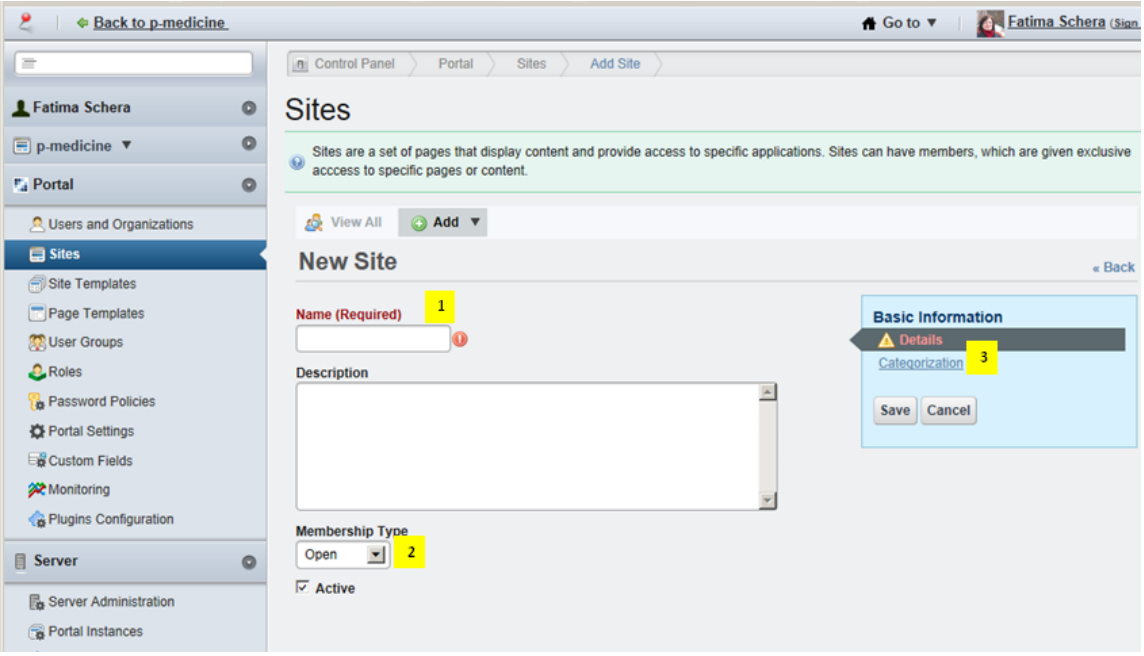

Figure 14: Creation of a new community

He also can define the membership type for the new community (open, restricted, private) in the displayed view (2, Figure 14). Under the *Categorization* option (3, Figure 14) the user can define metadata about the community by assigning a tag or a category to the community description. Furthermore, the portal administrator has to define the community administrator who can manage the community.

The community administrator manages the community membership; he defines the community layout and creates content for the community members. Creating content with a large amount of applications available in the *p-medicine* portal is a time intensive task. In order to make this work more comfortable and time-effective for the community administrator, we have developed a template for the standard *p-medicine* community with the predefined layout and contents.

For the community template we have preselected the layout designed for *p-medicine* (Figure 17). On the top of the template page the name of the template is displayed (1, Figure 15). The *p-medicine* community template currently contains six pages including access to different *p-medicine* services and tools (2, Figure 15) for community members:

- Home page - includes a place holder for the "welcome" - information about a community and a portlet for displaying announcements
- *p-medicine* Tools - includes access to the *p-medicine* tools and services (Ontology Tools, Data Warehouse, *p-medicine* Workbench, Oncosimulator, ObTiMA, Data Mining tools, Biobank Access, Cloud Services)
- Documents and Media - here can community members share documents and media files
- Calendar
- Message Boards
- Help - here we want to put some helpful information for the portal users (e.g. user manuals)

If the portal administrator selects this template as a basis structure during the creation of a new community, the new community will be created with the content provided in

the template. The community administrator can afterwards easily adjust the default content according to the community purposes: he can modify the layout of the community site pages; he can add new pages and applications as well as remove some pieces of the content if they are not necessary for the community. He can also replace the default *p-medicine* logo with the logo representing the community (3, Figure 15).

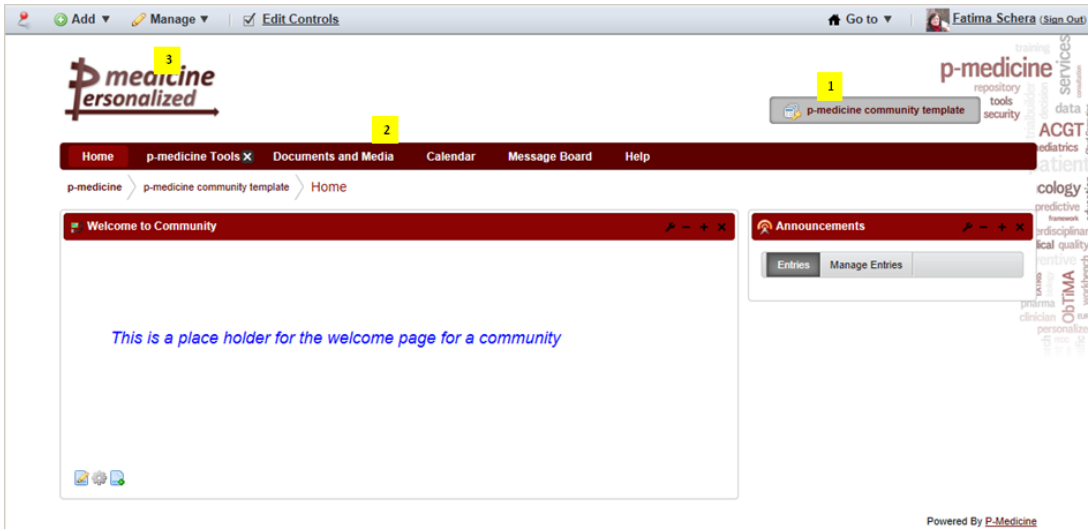

Figure 15: *p-medicine* template for creation a community

6 Current development state of the *p-medicine* portal

In this chapter we describe the current development state of the *p-medicine* portal. In the first section some details about the server which hosts the *p-medicine* portal are given. The next section contains information about and usage of the *p-medicine* components already integrated into the portal.

6.1 Installation, configuration

The *p-medicine* portal is hosted on the server *pmedportal.ibmt.fraunhofer.de* located by the Fraunhofer Institute Biomedical Engineering in St. Ingbert (Germany). Here are some facts about the server:

- the operational system is SUSE Linux -Version 12.1
- the used java virtual machine is based on the Sun Java SDK, version 1.6
- The *p-medicine* database is created on the open source object-relational database management system Postgres, version 9.1.
- Liferay framework version 6.1. is provided as a standard .war file - only around 125 MB in size - which can be installed on any application server, or as a bundle, preinstalled in any open source application server of choice. We have decided to use a bundle with the included application server Tomcat, version 7.0. Installing Liferay is as easy as unzipping an archive and editing a text file to point it to the *p-medicine* database. Furthermore, we have configured the mail server for using the mailing system included into the portal.

  The full installation guide is stored on the *p-medicine* project Wiki page in *http://atlas.ics.forth.gr/pMedicine/wiki/index.php/Installation_steps_for_the_p-medicine_portal_server*

6.2 Available plugins

There are currently some plugins already available in the portal:

- *p-medicine* theme-plugin for the *p-medicine* layout (see sections 5.6.1 and 6.2.1)
- Plugins for the *p-medicine* Security Framework
  - Identity Consumer ext-plugin (for Single Sign-On)
  - Login portlet
- Plugins for Data Mining tools
- ObTiMA
- Template site for creation of a default *p-medicine* community (see section 6.2.5)

Below we describe these components in more detail.

6.2.1 Plugin for the *p-medicine* layout

Liferay provides a standard theme for the look and feel of the portal shown below:

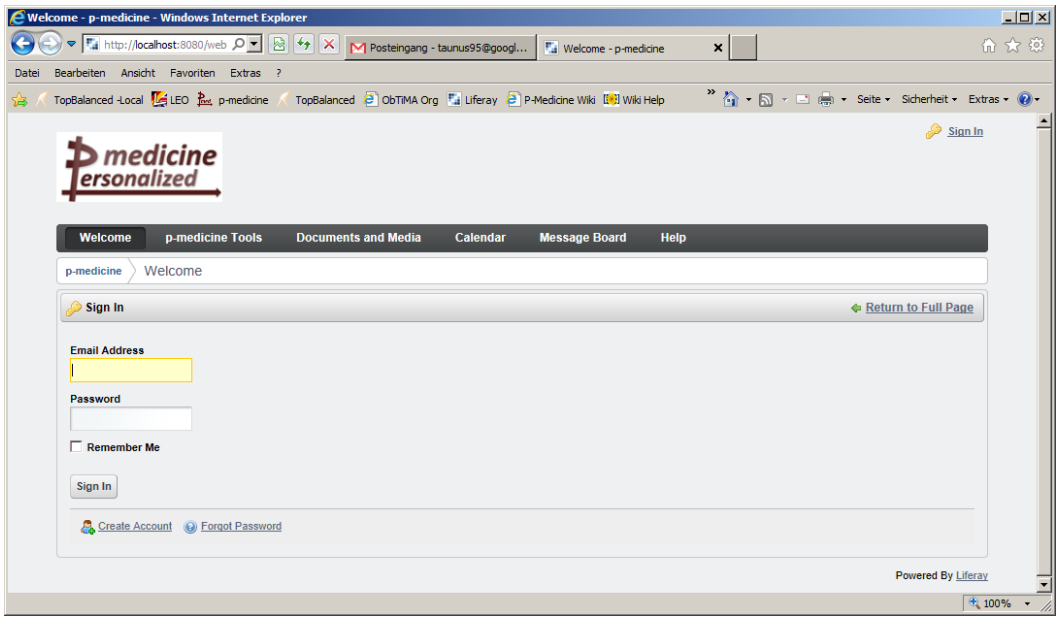

Figure 16: Original Liferay look and feel

We have developed a theme plugin for the portal responsible for the p-medicine look and feel. With the developed plugin we have replaced the original Liferay layout shown on the Figure 16. Below we show the developed p-medicine layout (Figure 17):

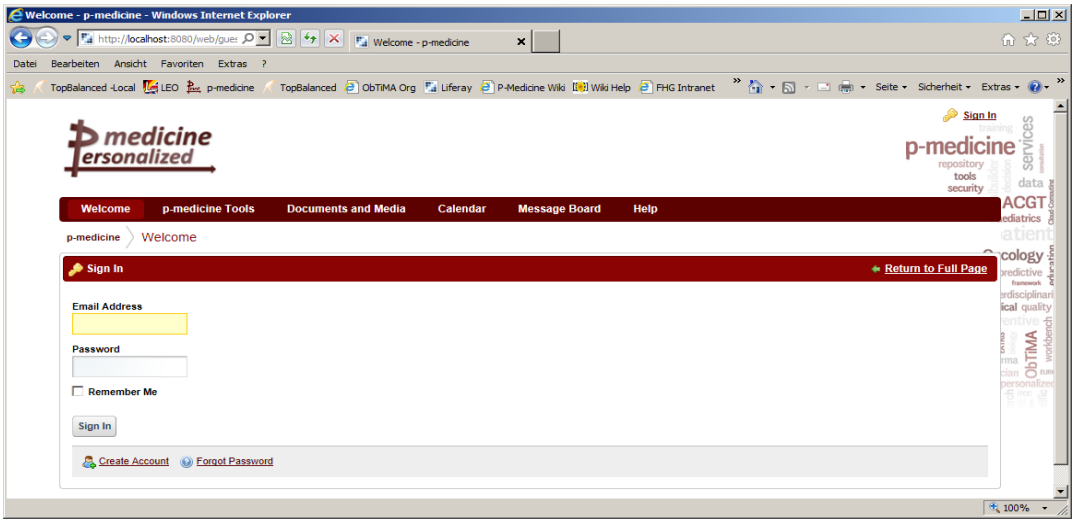

Figure 17: *p-medicine* look and feel

This theme can be selected from the list of available themes for using for the layout of the p-medicine communities. For doing this the user has to select the menu *Manage --> Site Pages* (1, Figure 18) on the Dockbar which is one of the user's navigation menu bars in the portal (see section 5.6.2):

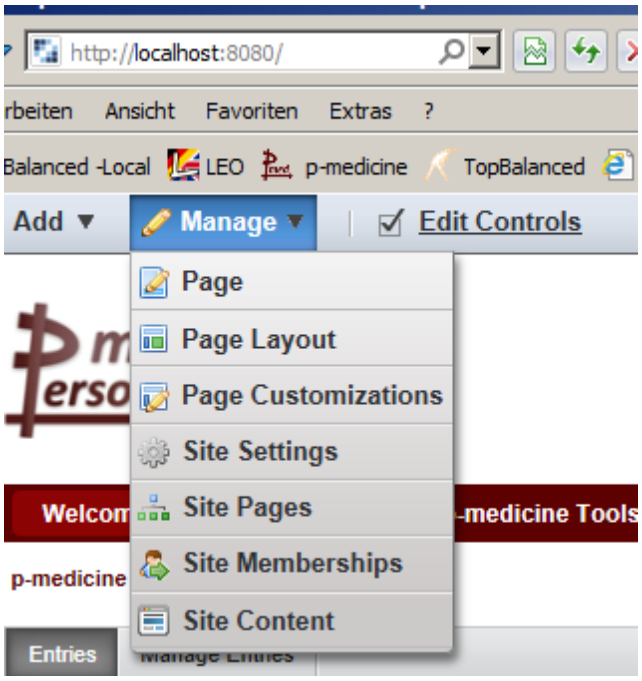

Figure 18: Dockbar menu "Manage" --> "Site pages"

On the opened view the user can select either the layout developed for the *p-medicine* project (1, Figure 19) or the original Liferay layout (2, Figure 19).

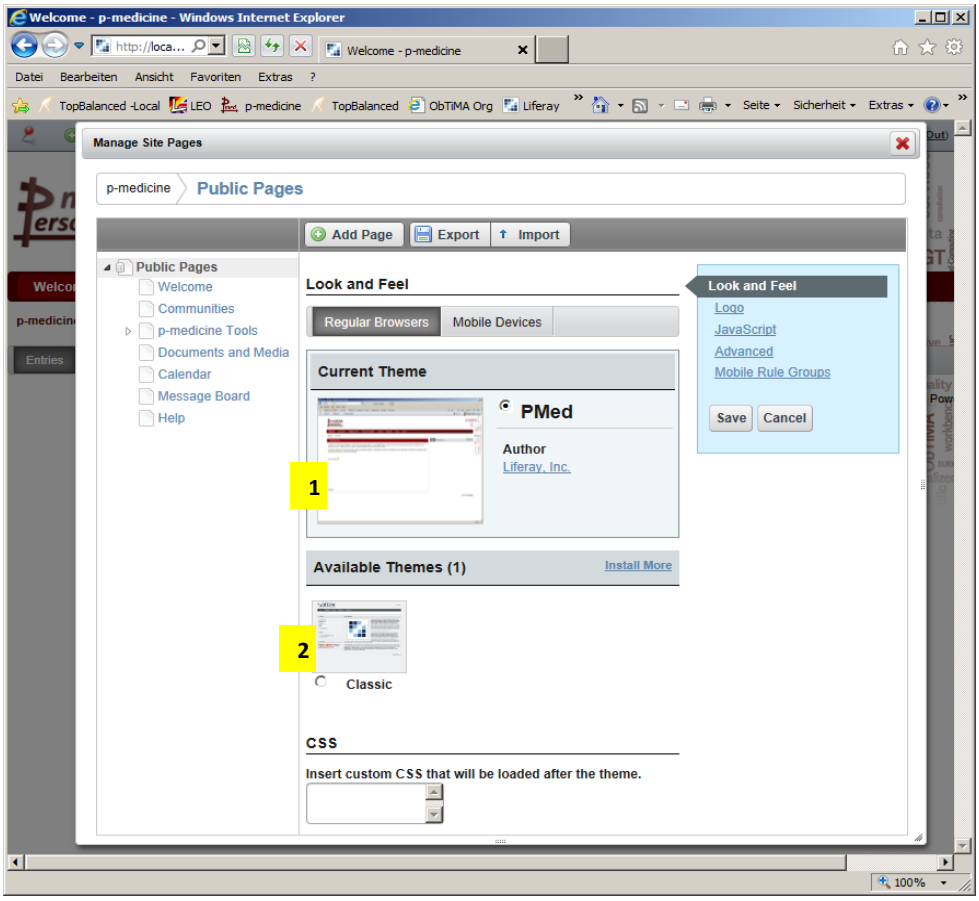

Figure 19: Selection of *p-medicine* look and feel

6.2.2 Plugins for the *p-medicine* Security Framework

There are two components initially integrated into the portal: Identity Consumer extension plugin for the Single Sign-On functionality and a portlet for a user login into the *p-medicine* environment (to the Identity Provider) which replaces the standard login portlet provided by Liferay.

6.2.2.1 Identity Consumer extension

This extension adds the following functionality to Liferay:

- a SAML service provider endpoint through Spring Security
- creation of an user from the provided SAML tokens automatically. When a user authenticates with a SAML token to the portal for the first time, a portal user is created for him based on identity information extracted from the provided SAML token.

D3.4 describes in chapter 9 the implementation details of the Liferay identity consumer extension as a sample implementation of the *p-medicine* security architecture.

6.2.2.2 Login portlet

p-medicine authentication is integrated into Liferay by using a standard Liferay iframe portlet which renders the '*p-medicine* authentication iframe'.

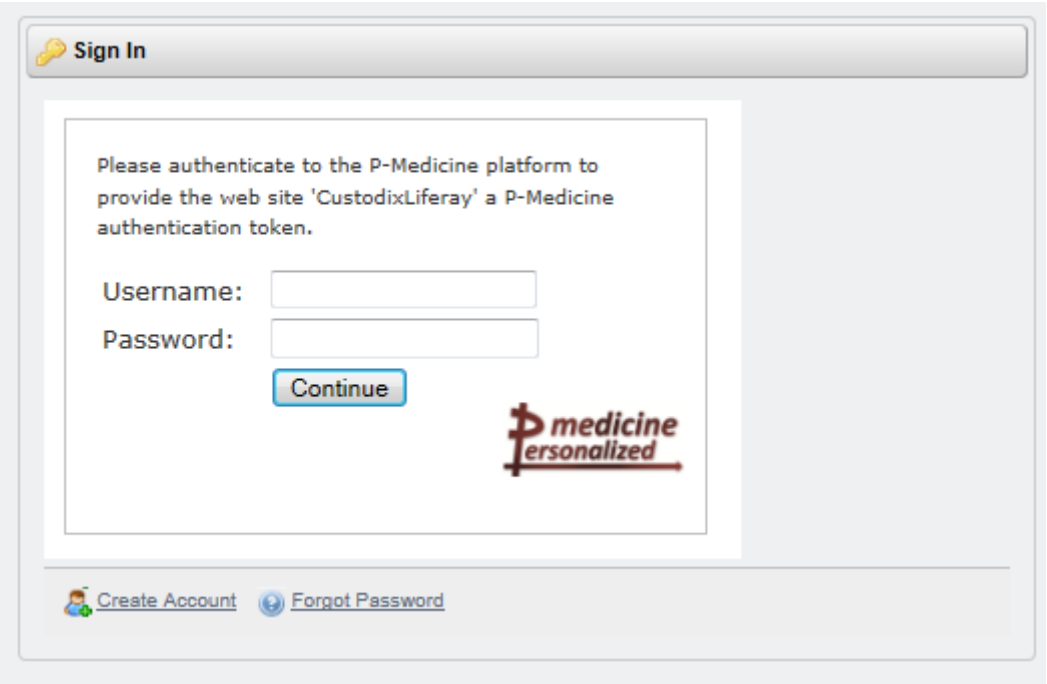

Figure 20: Login portlet

The portlet is configured to render the Liferay SAML entry point URL provided by the Identity Consumer plugin. If there is no active local security context the entry point creates a SAML authentication request and sends the user to the IdP through e.g. HTTP redirect. If the user has no active SSO session the IdP renders a login form in the iframe. Once authenticated a SAML identity assertion is created and sent back to Liferay (by redirecting to the URL of the Liferay identity consumer). If there was an active SSO session, the user isn't requested to authenticate him but instead the assertion is automatically generated. This redirect still occurs within the '*p-medicine* authentication iframe'. The identity consumer then verifies the

token and if deemed valid the whole page (not just the portlet) is refreshed rendering the home page of the authenticated user.

6.2.2.3 User management

The user management is an external site for which an attempt will be made to integrate it into Liferay by using iframes (e.g. organisation management iframe, user management iframe, etc.).

6.2.3 Plugins for Data Mining tools

This chapter introduces the integration of the data mining tools into the *p-medicine* portal. With the data mining tools the *p-medicine* users will be able to execute scientific workflows, to set up input parameters and to view results after completion. Workflows are the standard way to organize specific steps of the data-mining process (for example pre-processing, feature generation, training, parameter optimization, validation) into a series of logical and standardized steps. As required by the users, *p-medicine* will support the execution of workflows defined by workflow environments, which are de-facto standards in scientific and biomedical computing. In particular, the execution of Taverna workflows will be supported. The integration is build with the open source Workflow Management System Taverna[55].

The structure of the data mining environment installed on the same host where the *p-medicine* portal runs is represented in the Figure 21.

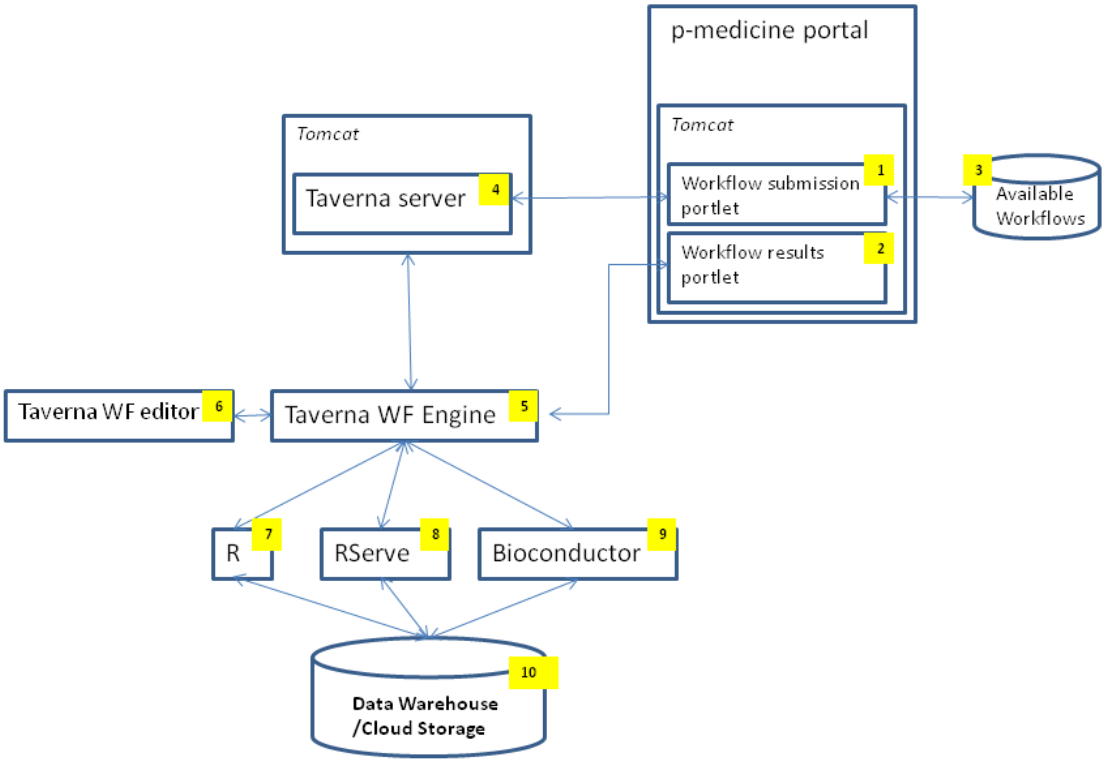

Figure 21: Overview of the data-mining services architecture

6.2.3.1 Components of the Data Mining environment

In this section the description of the Data Mining environment components (Figure 21) integrated into the portal is given.

[55] http://www.taverna.org.uk

- There are two Data Mining portlets integrated into the *p-medicine* portal:
  - a portlet for submission of a workflows (1). The portlet is connected with the public repository for reusable common workflows (3)
  - a portlet for showing results of performed workflows (2)

  Both portlets are open source and have been modified for *p-medicine* purposes.
- Taverna server (4) is the open source Workflow Management System that allows users to build complex scientific workflows. Taverna offers a huge number of services to support scientific experiments including Rshell, which allows executing R code within workflows. Furthermore, it is connected to myExperiment, which provides the user with a public database of common workflows (3). Workflows in Taverna are saved into a Taverna own file format called ".t2flow".
- New workflows can be created with the Taverna Editor (6).
- Data-Mining Workflow Execution Environment, e.g. Taverna Freefluo Engine (5) is an execution engine for the Data-Mining executable steps.
- R[56] is a programming language and an open source software environment for statistical computing and graphics (7). R is widely used for t development of the statistical software and data analysis. R is used in the *p-medicine* test scenario to implement certain sub workflows for the data analysis and graphics. The following modules have been installed together with R:
  - The *p-medicine* test scenario uses external open source R-libraries which are called Bioconductor[57] (9). Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language.
  - The *p-medicine* test scenario uses Rserve[58] to execute R Code (8). Rserve is an open source TCP/IP server which allows other programs to use facilities of R from various languages without the need to initialize R or link against R library. Every connection has a separate workspace and working directory. Client-side implementations are available for popular languages such as C/C++, Java and also Taverna Workbench. Using the Data-Mining Webapp, R scripts can be wrapped into data-mining workflow executions and be uploaded to the Data-Mining Webapp.
- Data Warehouse / Cloud Storage (10) - Cloud storage in *p-medicine* environment is intended to be used by the Data Warehouse as a storage backend for files. The other scenarios are related to data mining workflows that can use cloud storage for intermediate computation results, and oncosimulator application executed in a dedicated cluster environment. The *p-medicine* Cloud Storage System is the lowest level component in the data management architecture of *p-medicine*. It provides REST interfaces for managing file storing in the cloud environment and is built based on open source cloud storage technology OpenStack (Swift) technology. It provides access to reliable storage space taking into account requirements from different end user scenarios: long term data preservation on the one hand, as well as fast access to application data in the workflow execution.

Further data-mining services can be added, given that they provide a web-based service interface.

[56] http://www.r-project.org
[57] http://www.bioconductor.org/
[58] http://rosuda.org/Rserve/

6.2.3.2 Usage of the data mining tools in the portal

As described in section 6.2.3, the portal users can find the data mining portlets integrated into the portal by using the menu item *Add --> More* in the Dockbar (Figure 22):

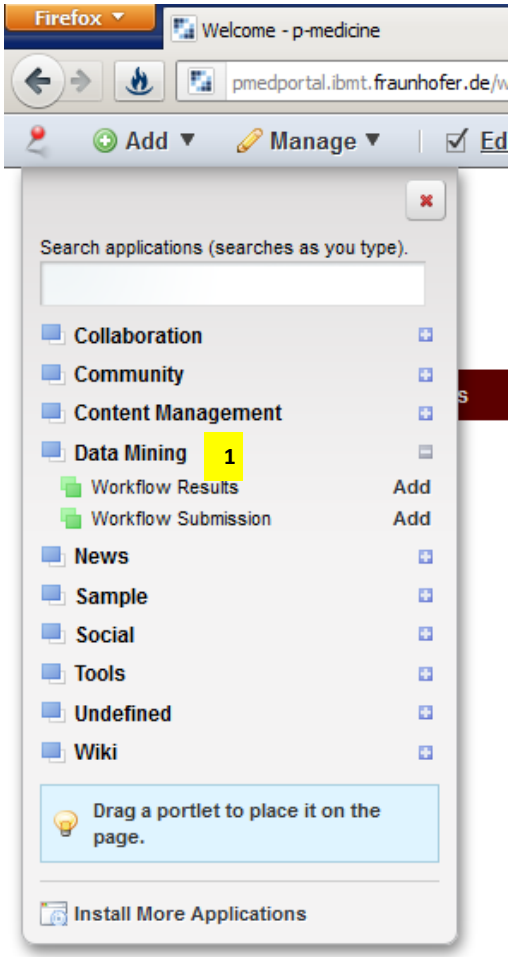

Figure 22: Data mining portlets integrated into the portal

The data mining portlets provide a solution for executing (enacting) scientific workflows, the possibility to set up input parameters and to view results after completion. They can be added to a page of the *p-medicine* portal by using the *Add* links next to the *Workflow Submission* and *Workflow results* portlets for data mining (1, Figure 22). In Figure 23 the result of adding both data mining portlets to the page which can be achieved by using the menu item *p-medicine Tools --> Data Mining* in the main *p-medicine* navigation menu is shown (Figure 3):

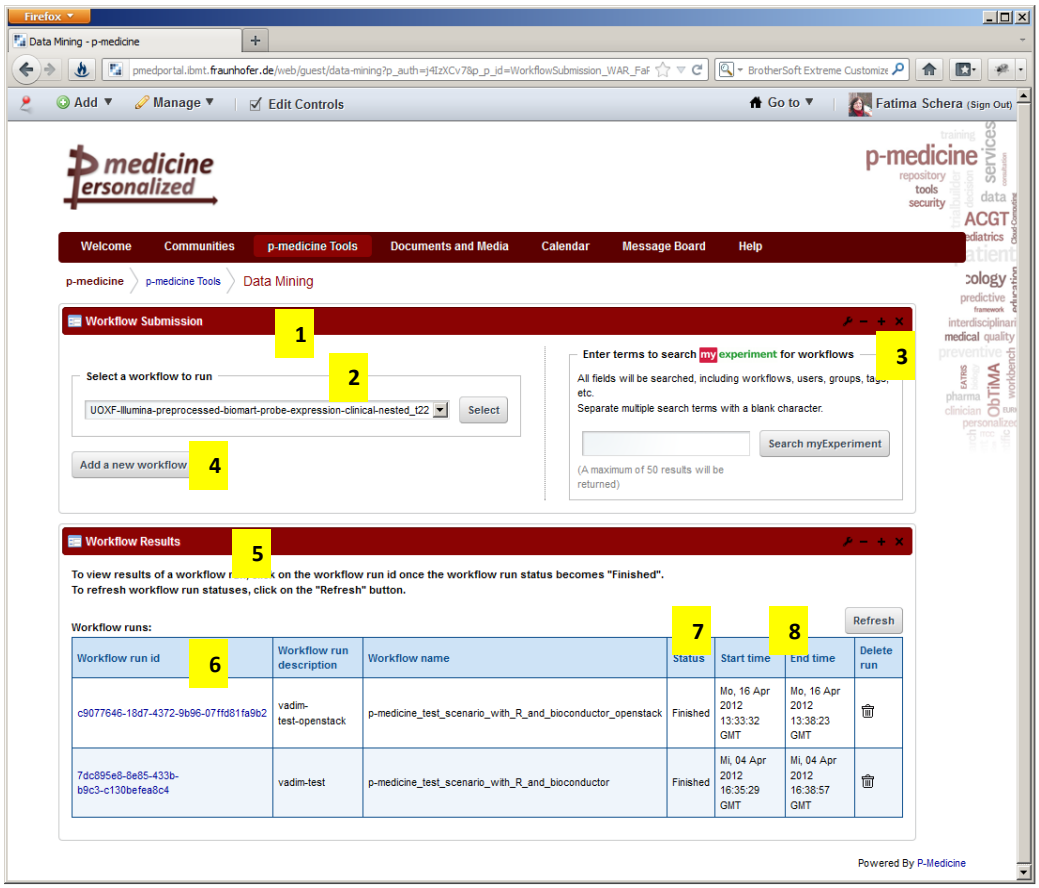

Figure 23: Data mining portlets added to the portal page

The *Workflow Submission* portlet (1, Figure 23) accesses a repository with known and available Taverna workflows. To enact a workflow, the user needs to select a workflow and create a Workflow run description. Users can either select one of the workflow existing in the repository using the drop-down list (2, Figure 23) or upload new files with workflow definitions into that repository by using the *Add new workflow* button (4, Figure 23) and then select this new workflow from the drop-down list. For selection a workflow for execution the user can also use the *myExperiment* tool (3, Figure 23) which provides the user with access to a public database of common workflows.

After selecting a workflow, the user is obliged to set up input parameters if necessary (1, Figure 24). For running the selected workflow the user has to push the *Run workflow* button (2, Figure 24). The status of the workflow execution (*Finished* or *Running*) as well as the start and end time are displayed in the table row for the selected and executed workflow (7 and 8, Figure 23).

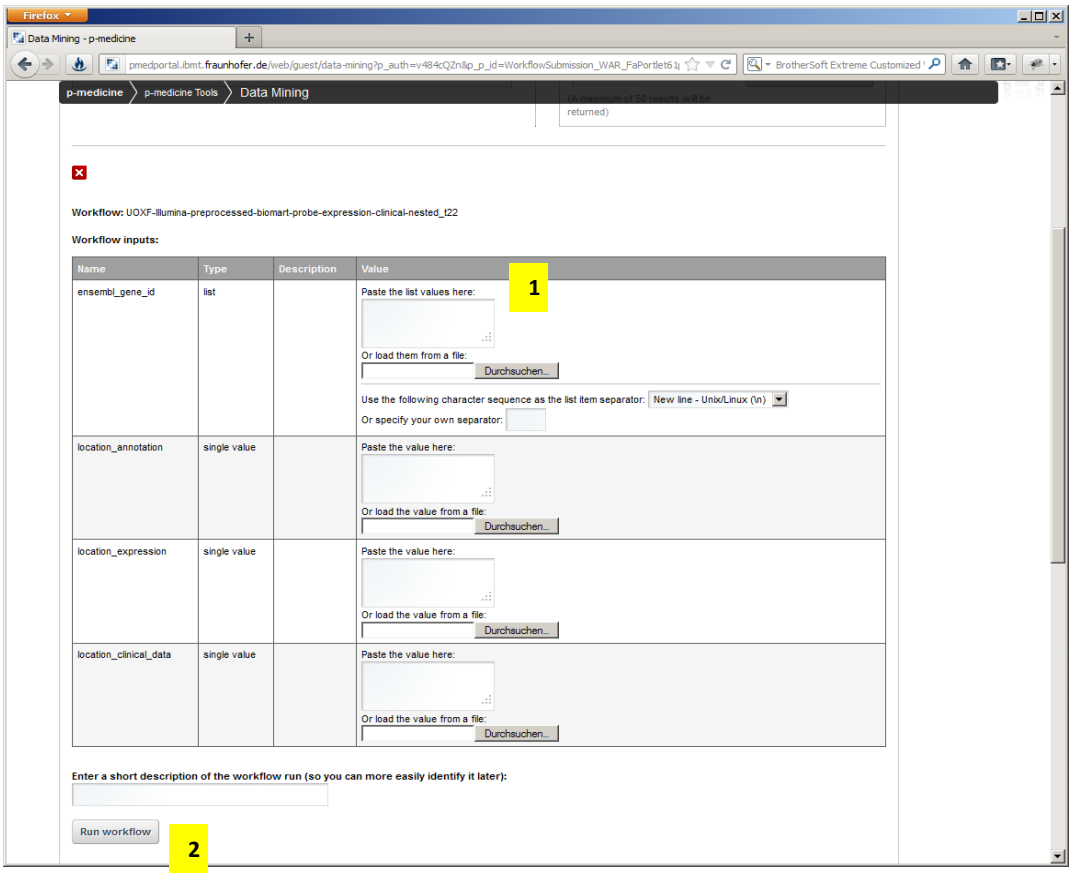

Figure 24: Input parameters for a workflow

The *Workflow Submission* portlet sends the selected workflow through the Taverna Server REST API to the Taverna Freefluo Engine for enactment. The results of the workflow enactment are saved in a customizable system path which can be accessed by the *Workflow Results* portlet or any other software or human.

Each workflow run is represented with a unique workflow ID which is automatically assigned (6, Figure 23). After finishing the workflow running (the status will have the value *Finished*) the text with the workflow id will be available as a link for selecting (1, Figure 25). A list with the result values will be shown if the workflow in the *Workflow Results* portlet is selected (2, Figure 25).

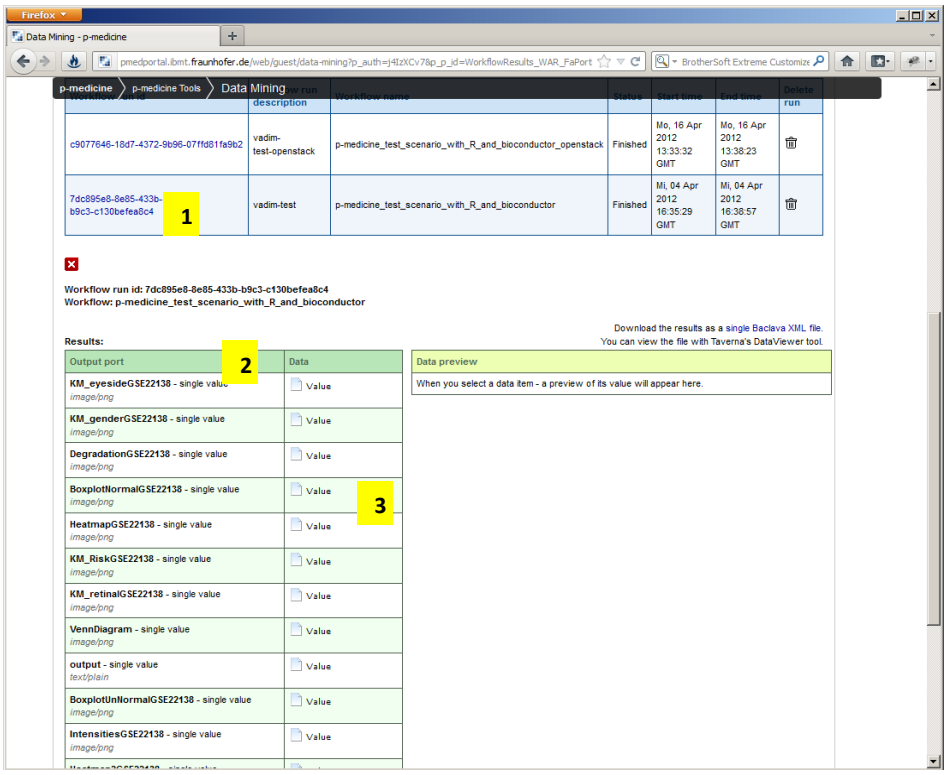

Figure 25: Results of the executed workflow

Each value can be viewed by selecting the link *Value* (3, Figure 25). The preview of the value will be shown on the right side of the window (1, Figure 26):

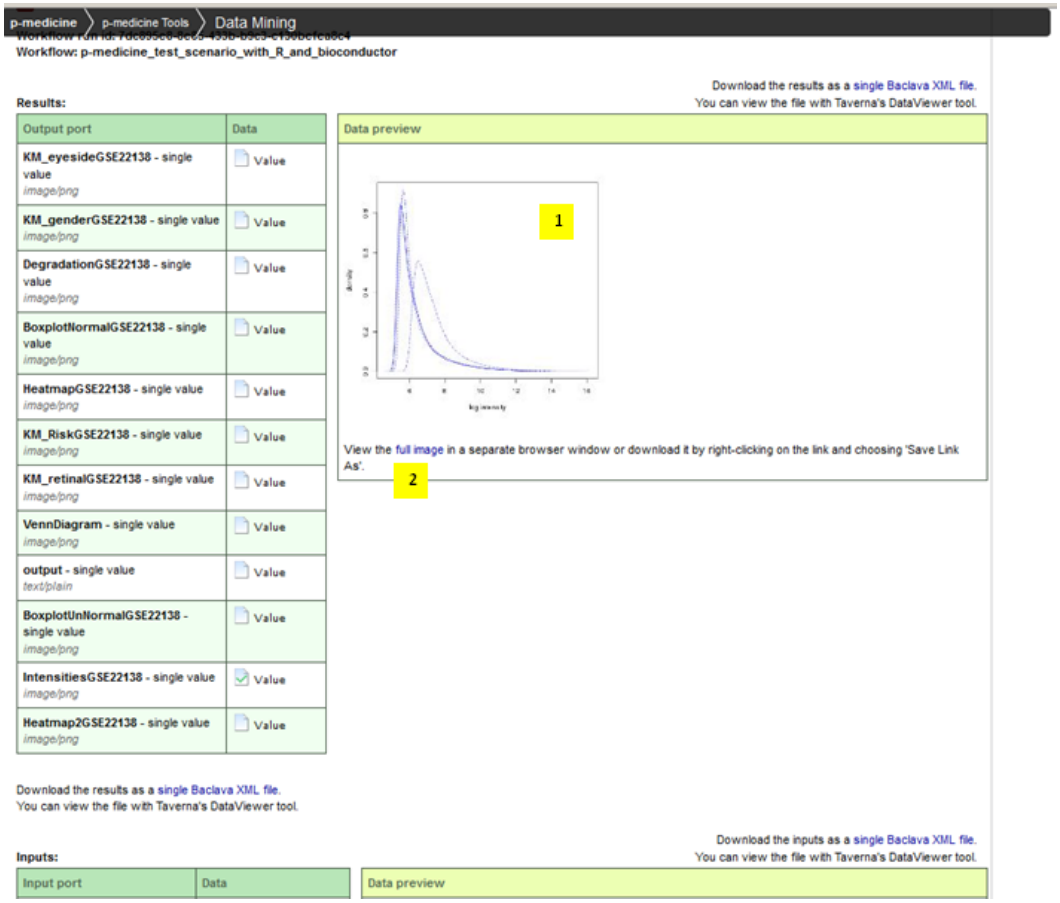

Figure 26: Preview of the value of the executed workflow result

For displaying the result in the full view the user has to use the "full image" link (2, Figure 26). The workflow result value will be displayed in the full view (Figure 27):

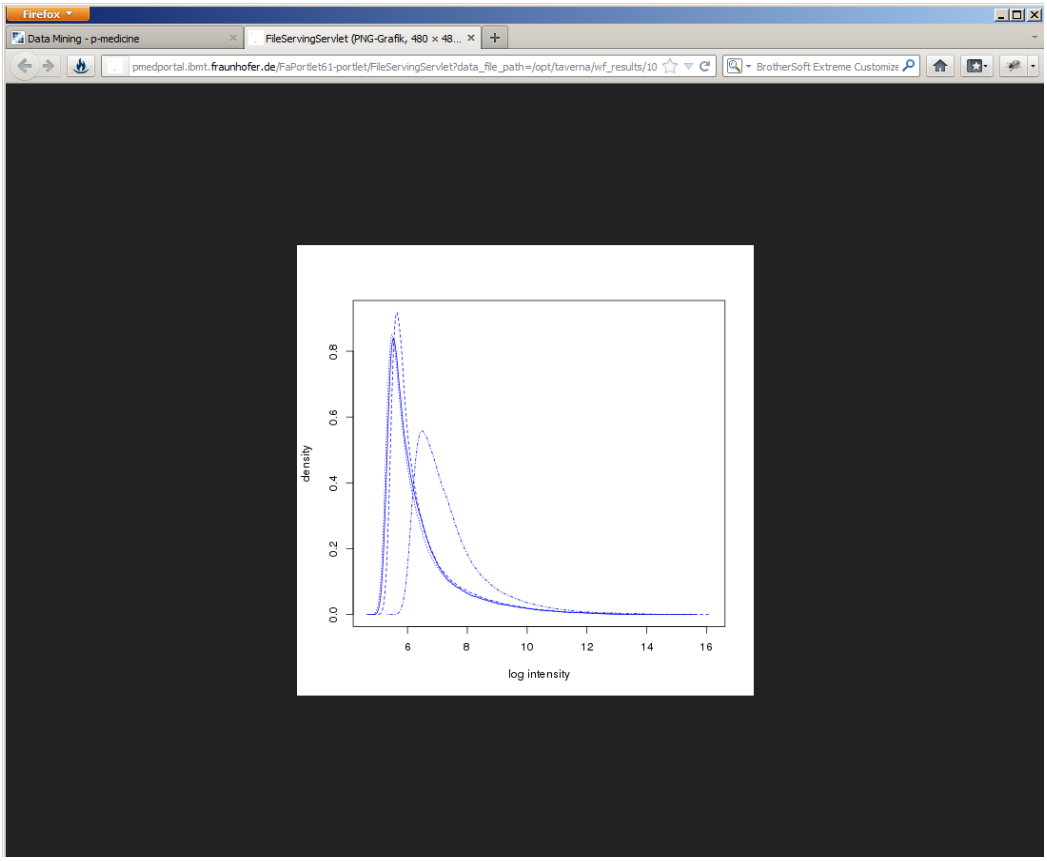

Figure 27: Full size view of the value of the executed workflow result

6.2.4 Integration of ObTiMA

6.2.4.1 Introduction

ObTiMA is a Java Server Faces (JSF) based web application. We provide two ways for accessing ObTiMA in the portal:

- integrated as iframe: ObTiMA will be shown in an iframe portlet inside of the portal
- integrated as a simple web link included in a web content shown in the portal.

We have chosen this way to integrate ObTiMA over the possibility to integrate ObTiMA as a plugin for the following reasons.

For the integration of JSF based web applications into the portal the JSF pages of the application need to be migrated into portlets. The development toolkit provided by the Liferay Company allows performing such a migration using the portlet bridges technology. A portlet bridge allows JSF (Java Server Faces) to work under a portal environment. Due to the complexity of the large ObTiMA application and to the fact that ObTiMA will be further developed and extended during the project lifetime we have decided to use this approach for its integration into the *p-medicine* portal.

6.2.4.2 ObTiMA integrated as iframe

For the integration of ObTiMA in the *iframe* portlet provided by Liferay, the iframe portlet has been selected by using the menu *Add --> More --> Sample* in the portal's Dockbar and added to the page *ObTiMA* created in the main *p-medicine* navigation menu *p-medicine Tools*. Below is shown the iframe portlet added to the *ObTiMA* page:

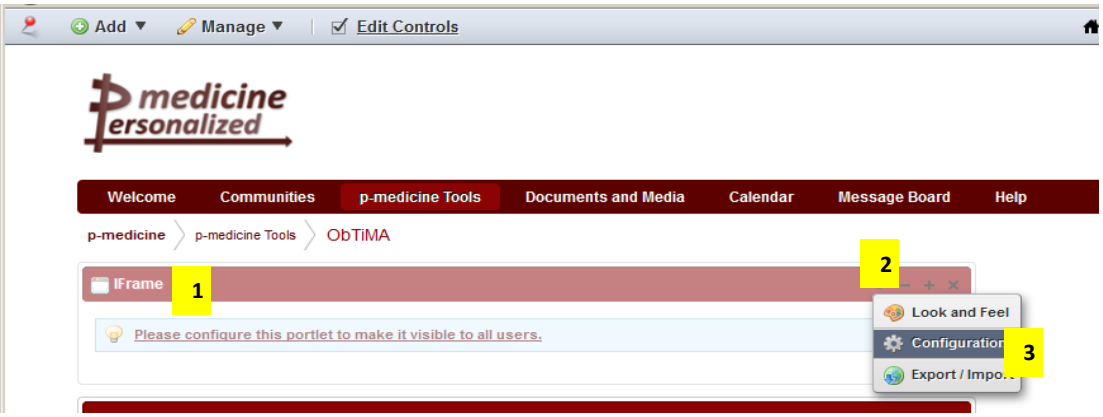

Figure 28: iframe portlet added to a portal page

The name of the portlet can be changed from *iframe* to *ObTiMA* by double clicking on the portlet name *iframe* (1, Figure 28). For configuring the portlet in order to integrate the ObTiMA the icon for configuration should be used (2, Figure 28). In the opened drop-down menu the option Configuration should be selected (3, Figure 28). The window for configuring the *iframe* portlet will be shown:

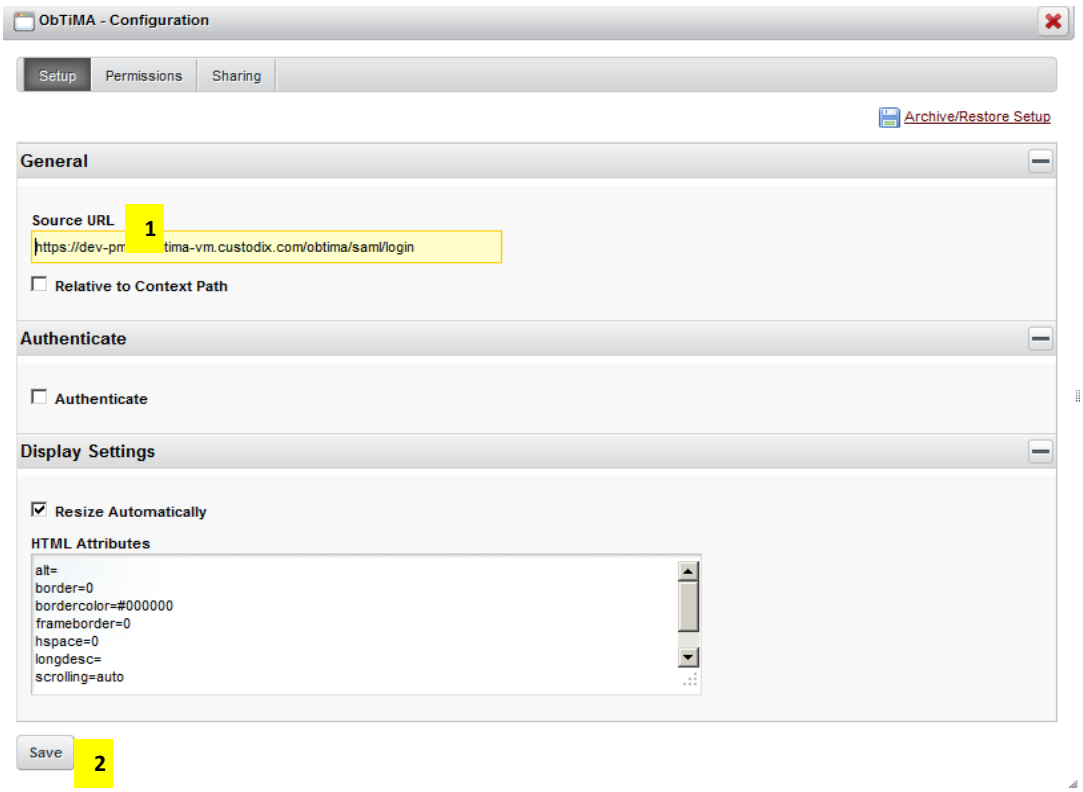

Figure 29: Configuration of the iframe portlet for ObTiMA application

The URL for the ObTiMA application should be entered into the *Source URL* test field (1, Figure 29). After saving the configuration (2, Figure 29) ObTiMA is integrated into the iframe portlet in the portal and is ready for using (1, Figure 30):

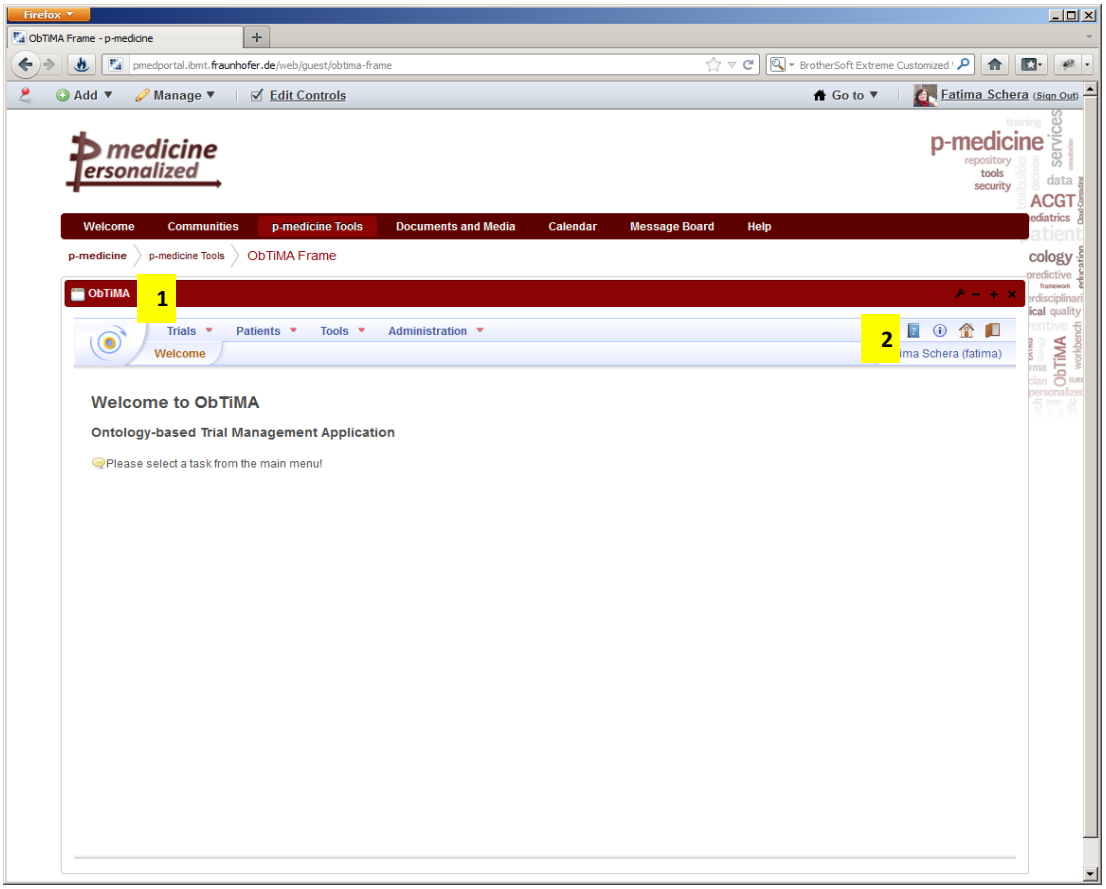

Figure 30: ObTiMA application included to iframe portlet

Due to the fact that the Single Sign-On functionality is implemented in the portal the user doesn't need to login into ObTiMA because he is already logged in into the portal (2, Figure 30).

6.2.4.3 ObTiMA integrated as a simple web link included into a web content

For the integration of ObTiMA as a simple web link the link has been included into the *Web Content Display* portlet provided by Liferay. The portlet has been selected by using the menu *Add --> More --> Content Management* in the portal's Dockbar and then added to the page *ObTiMA* created in the main *p-medicine* navigation menu *p-medicine Tools*. Below is shown the *Web Content Display* portlet added to the *ObTiMA* page:

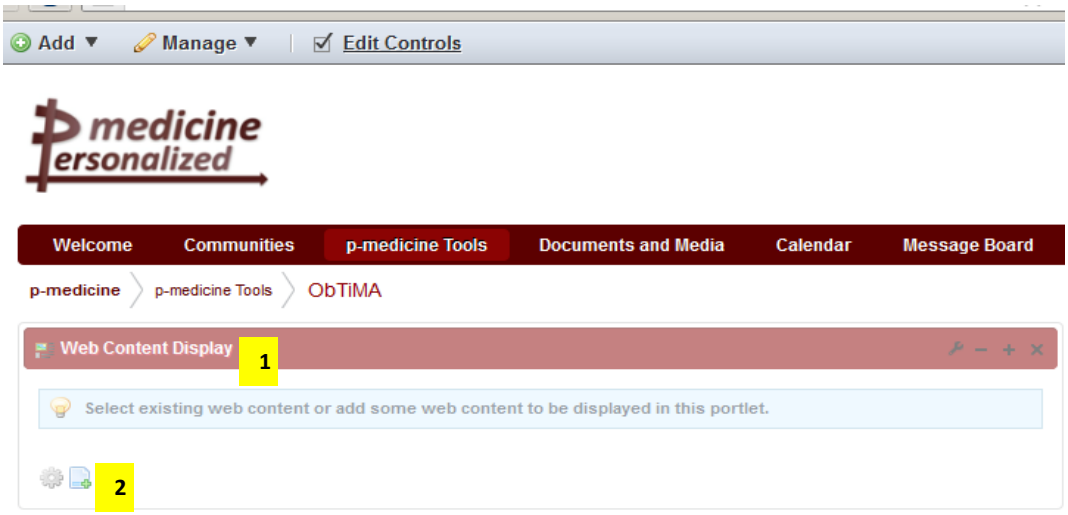

Figure 31: Web Content Display portlet added to a portal page

The name of the portlet has been changed from *Web Content Display* to *ObTiMA* by double clicking on the portlet name *Web Content Display* (1, Figure 31). For editing the content the icon for adding web content should be used (2, Figure 31).

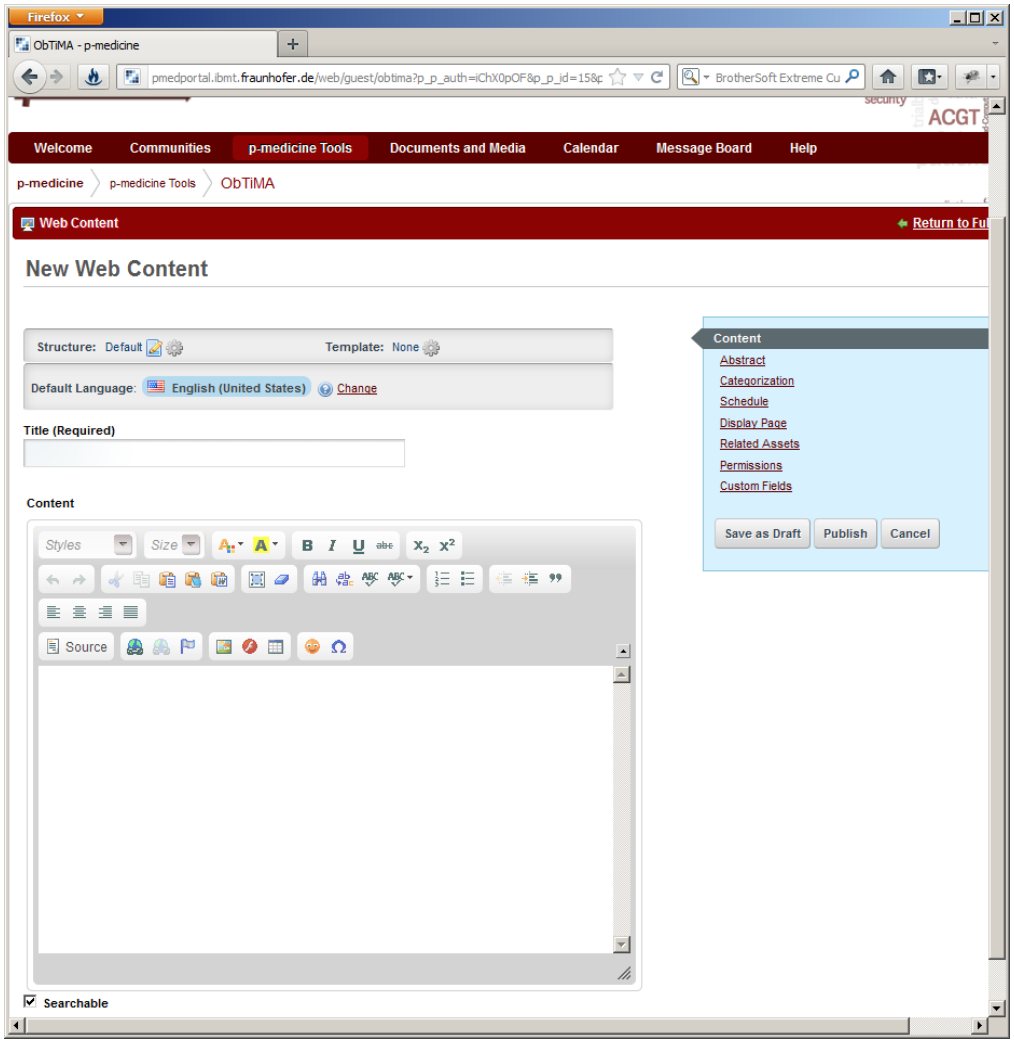

Figure 32: Creation of a content for a Web Content Display

Using the opened rich text editor (Figure 32) the link to the ObTiMA application has been integrated into the portal (1, Figure 33):

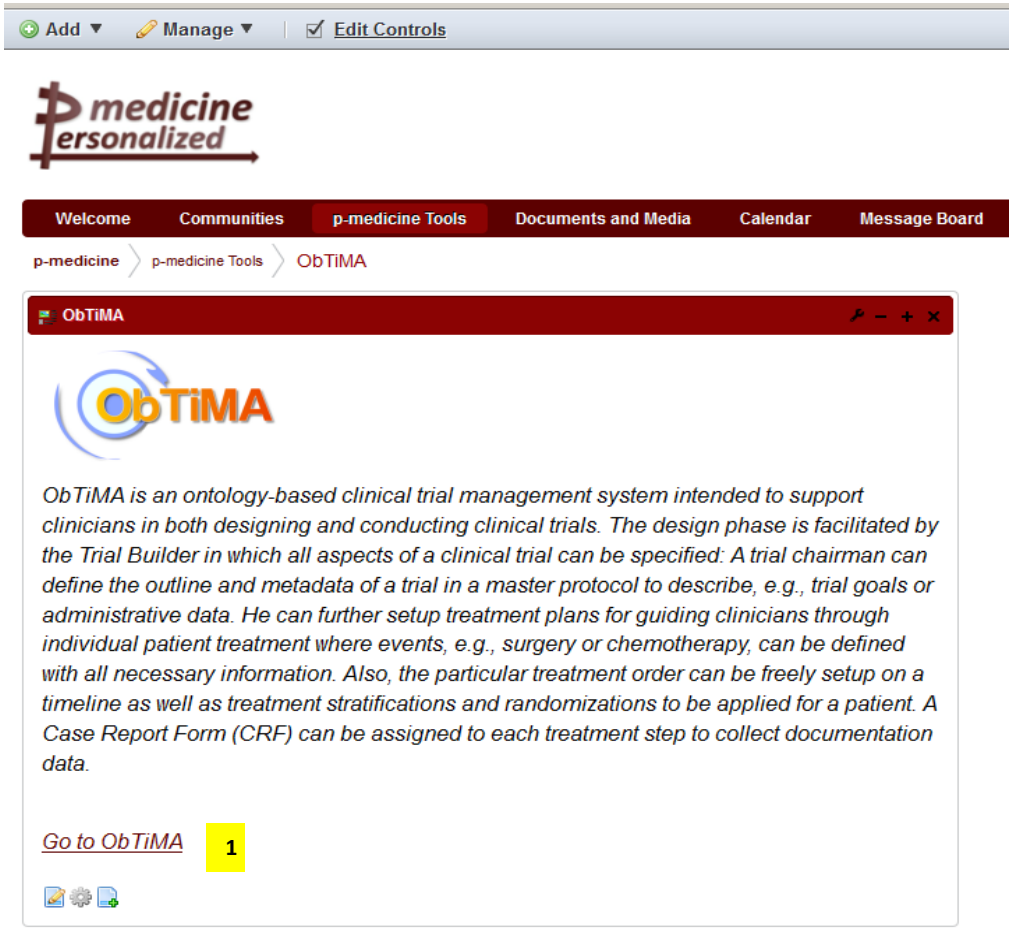

Figure 33: Web link to ObTiMA application

6.2.5 Template site for creation of a default *p-medicine* community

In order to make easier for the *p-medicine* portal administrator to build a new community in the *p-medicine* portal we have decided to create a template for the *p-medicine* community which will include a set of pages containing *p-medicine* related content. The template can be used as basis for a newly created community and it will take less time to edit the already included content or to add additional pages as to create the new community site from scratch.

The default *p-medicine* community can contain the following pages included into the main *p-medicine* user navigation menu described in the section 5.6.1:

- "*Welcome*"
- "*Communities*"
- "*p-medicine Tools*" (a subset of pages)
- "*Documents and Media"*
- "*Calendar*"
- "*Message Board*"
- "*Help*"

These pages have been already included into the main part of the portal.

For the template pages we have used the layout developed for the *p-medicine* look and feel (Figure 17). The template view is shown below (Figure 34):

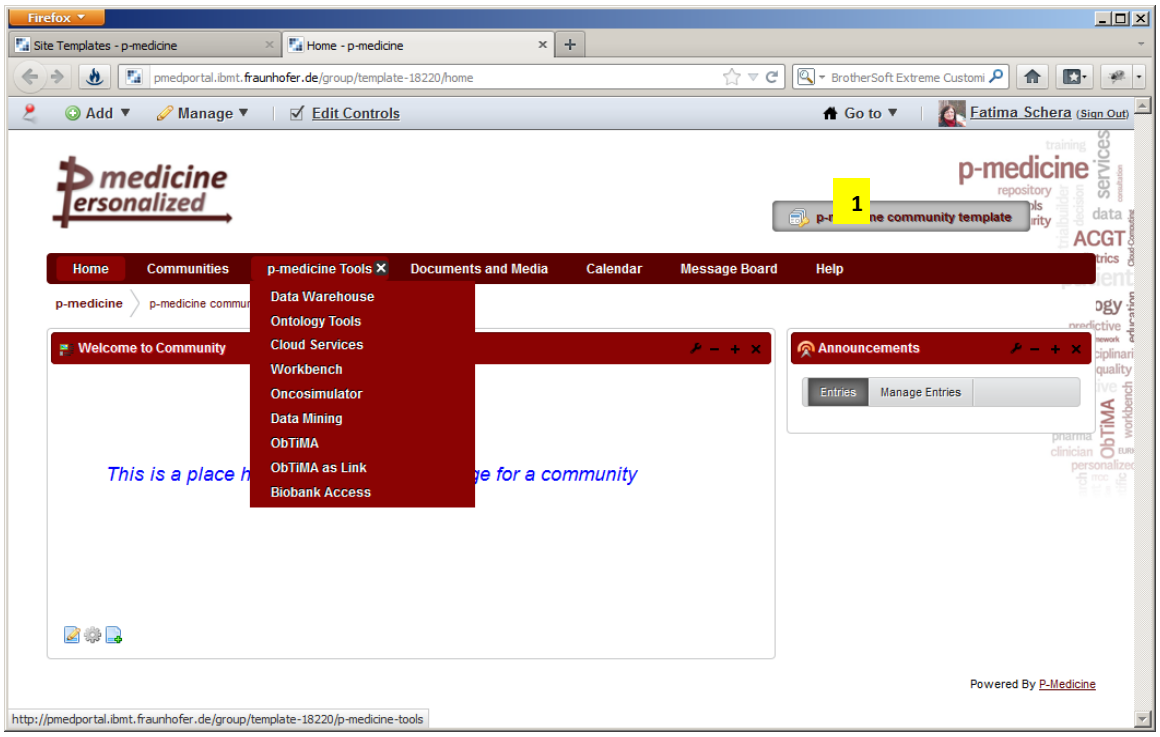

Figure 34: *p-medicine* community template view

The text *p-medicine community template* on the place of the community name (1, Figure 34) indicates that this is a template, not a community site.

The full description how to create a template and how to fill it with content as well as how to use it for creation of a new *p-medicine* community is provided in the Liferay documentation[59].

6.3 Current communities

Based on multiple discussions with *p-medicine* end users we have already included several communities into the portal shown on the Figure 35:

[59] http://www.liferay.com/documentation/liferay-portal/6.1/user-guide

Figure 35: *p-medicine* communities

We have used the terms defining the community groups (1 and 2, Figure 35) for tagging the p-medicine communities (1, Figure 36):

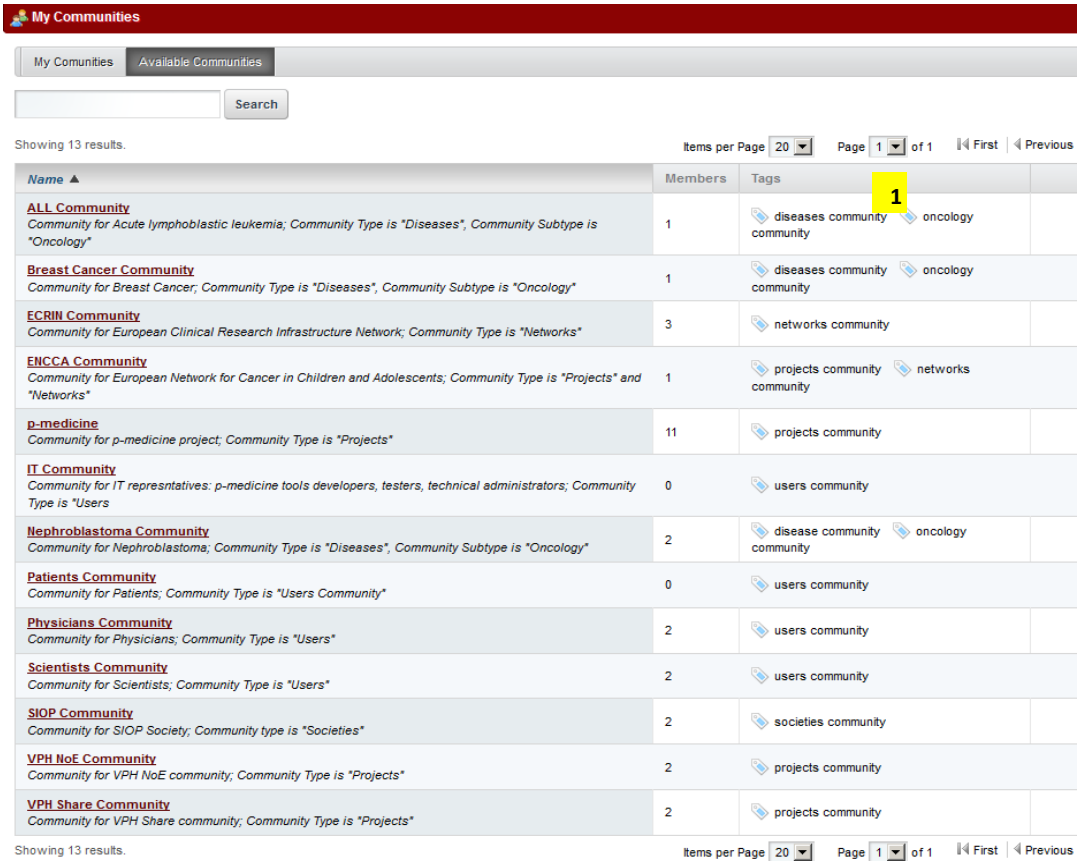

Figure 36: *p-medicine* communities in the portal

7 Conclusion

The *p-medicine* portal is a web application providing comfortable access for the *p-medicine* users to the tools and services integrated into the *p-medicine* environment. The portal has been developed following the requirements of the user groups and their roles for the *p-medicine* portal. After evaluation of several existing portal solutions we have decided to build the *p-medicine* portal based on Liferay framework, a free and open source enterprise portal framework, which fulfils the complex user requirements for the *p-medicine* portal.

We have introduced approaches for the developing and integration of the different components and we defined how the particular *p-medicine* tools and services should be integrated into the portal.

Furthermore, we have developed a layout component representing the *p-medicine* look and feel. We have created a template for a standard *p-medicine* community containing predefined set of pages with the *p-medicine* relevant content which makes it easier to add a new community to the *p-medicine* portal.

The demonstration version of the *p-medicine* portal can be accessed using the URL *http://pmedportal.ibmt.fraunhofer.de.* The actual version contains the initial functionality of the *p-medicine* security framework, tools for data mining as well as ObTiMA. Other tools and services described in section 3.1.2 will be integrated during the course of the project as they become available. Developers of these applications could use the instructions for development and integration given in the chapter 5 and use the examples in the Appendix 2.

Appendix 1 - Abbreviations and acronyms

SOA	Service Oriented Architecture
VPH	Virtual Physiological Human
NoE	Network of Excellence
JSR	Java Specification Request
SSO	Single Sign-On
CATS	Custodix Anonymisation Tool Services
PIMS	Patient identity management system
HDOT	Health Data Ontology Trunk
DICOM	Digital Imaging and Communications in Medicine
ObTiMA	Ontology based Trial Management Application
CRF	Clinical Record Forms
p-BioSPRE	Biomaterial Search and Project Request Engine
CDMI	Cloud Data Management Interface
HIS	Hospital Information System
SPARQL	SPARQL Protocol And RDF Query Language (recursive acronym)
IEmS	Interactive Empowerment Service
ACGT	Advancing Clinico-Genomic Trials
WSO	Web Service Oriented
API	Application Programming Interface
AJAX	Asynchronous JavaScript and XML
UI	User Interface
SOAP	Simple Object Access Protocol
REST	REpresentational State Transfer
RSS	RDF Site Summary, often dubbed Really Simple Syndication
CMS	Content Management System

JCR	Java Content Repository
LDAP	Lightweight Directory Access Protocol
CAS	Central Authentication Service
JOSSO	Java Open Single Sign On
HTML	Hyper Text Markup Language
JSF	Java Server Faces
XML	Extensible Markup Language
ACL	Access Control List
JAAS	Java Authentication and Authorization Service
EAI	Enterprise Application Integration
JSP	Java Server Page
PSML	Portlet Structure Markup Language
RBAC	Role Based Access Control
JDO	Java Data Objects
OQL	Object Query Language
E4X	ECMAScript for XML
WS	Web Service
ROI	Return on Investment
WSDL	Web Services Description Language
XSD	XML Schema Document
APP	Atom Publishing Protocol
IM	Instant messaging
RFI/RFP	Requests For Information/Proposal
MVC	model–view–controller
SQL	Structured Query Language
JEE	Java Enterprise Edition
YUI	Yahoo User Interface

PHP	PHP: Hypertext Preprocessor (recursive acronym)
SDK	Software Development Kit
SVN	SubVersioN (from Apache)
IDE	Integrated Development Environment
ALL	Acute Lymphoblastic Leukemia
SIOP	The International Society for Pediatric Oncology
SP	Service Provider
AE	Architectural Elements
HTTP	Hypertext Transfer Protocol
OS-BRCA	Oncosimulator for Breast Cancer
OS-WT	Oncosimulator for Wilms Tumor
OS-ALL	Oncosimulator for Acute Lymphoblastic Leukemia
RDF	Resource Description Framework
JMX	Java Management Extensions
PACS	picture archiving and communication system
WSGI	Web Server Gateway Interface
SAML	Security Assertion Markup Language
XACML	eXtensible Access Control Markup Language
IdP	Identity Provider
MPPS	Modality Performed Procedure Step
GPWL	General Purpose Worklist
MWL	Modality Worklist
SP	Service Provider

Appendix 2 - Development of portlets for the *p-medicine* portal

General information

- Liferay provides an Ant-based Plugin SDK for developing portal plugins.
- Portlets can be written using any of the java web frameworks that support portlets development, including Liferay's specific frameworks: MVCPortlet or AlloyPortlet
- There are several types of plugins: Themes, Layout Templates and Portlets (incl. Hook and Ext Plugins):
  - Themes and Layout Templates can be developed using CSS and Velocity[60] templates. The existing themes and layout templates can be modified (a) by editing css (for minor changes) or (b) by replacing some templates in the existing Themes and Layout Templates
  - Hook plugins can be used to modify portal properties or to perform custom actions on start up, shutdown, login, logout, session creation and session destruction.
  - It is not recommended to write Ext plugins for users without advanced skills. Even though Ext plugins are deployed as plugins, the server must be restarted for changes to take effect. For this reason, Ext plugins should not be combined with other types of plugins.
- Portlets are represented as .war files, OpenSocial gadgets - as xml files.
- Ways to reuse an **existing** web application:
  - re-write the application as a portlet
  - create a "middleware" portlet for interacting with the application (better using web service)
  - wrap the application as an OpenSocial gadget for displaying it in an iframe
  - Create a portlet that integrates the application either using an iframe or an HTTP proxy (e.g. using Liferay's WebProxy portlet). In this case we need a solution to transfer the authentication between the portal and the application.

If the existing application is a JavaEE application, Liferay provides a technology called **Web Application Integrator** that allows prototyping the integration by deploying the WAR file of the web application (e.g, by using the control panel or by copying it to the deploy directory). As a result Liferay will automatically create a portlet that integrates your application using an iframe.

- Developer can write JavaScript code in the portal by using the JavaScript libraries jQuery, Dojo, YUI, Sencha (former ExtJs), Sproutcore, AlloyUI (from Yahoo) etc.
- For PHP and Ruby developers: they can use the plugins SDK to deploy the PHP and Ruby applications as portlets into the portal (examples are in the public Liferay SVN in the folder plugins/trunk.

Preparation for the development of plugins for the *p-medicine* portal

Preparation steps

Below are the preparation steps for the plugin development:

- Create a folder *[LiferayDeveloperHome]* - this folder will contain two sub-folders: the first folder is for the Liferay framework and the second folder - for the portal's Plugin SDK.

[60] http://velocity.apache.org/

- Copy an unzipped *Liferay6.xx* file downloaded from the page *http://www.liferay.com/downloads*. **Attention**: the folder name should be without space signs.
- Remove for the better performance all folders in *the [LiferayDeveloperHome]\liferay-portal-6.xx\tomcat-xx\webapps* apart from *ROOT* folder.
- Start Liferay *[LiferayDeveloperHome]\liferay-portal-6.xx\tomcat-xx\bin\startup.bat* (For *stopping: CTRL+C* in the terminal window). (For LINUX use the command *startup.sh).*
- Log in as a user with the e-mail *test@liferay.com* (password *test)*.
- Download from the page *http://www.liferay.com/de/downloads/liferay-portal/additional-files* the Liferay Plugins SDK as a file *liferay-plugins-sdk-6.0.6-20110225.zip* **Attention**: on the Liferay page in the *Download* area for *Additional Files* is renamed to *Arquivos Extras*. Unzip the file into *[LiferayDeveloperHome]/pluginsSdk*
- Install ANT:
  - **Download** ANT from *http://ant.apache.org/*
  - **Unzip** the file in any folder: a sub-folder *apache-ant-1.8.2* will be created
  - Set the environment variable ***ANT_HOME*** pointed to the *apache-ant-1.8.2* folder
  - Extend the environment variable **PATH** by extending the text with %ANT_HOME%\bin;
  - For **LINUX** add to the *.bash_profile* the text (if **ANT** is installed in */java*):
    - *export ANT_HOME=/java/apache-ant-1.8.1*
    - *export PATH=$PATH:$ANT_HOME/bin*
  - Check if ANT works: write in command prompt *ant -version*

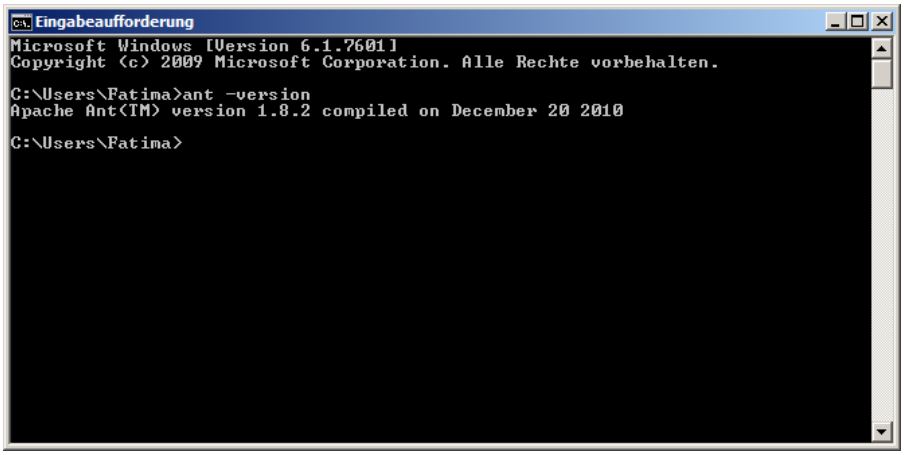

Figure 37: Checking if the ANT tool is running

- configure Plugins SDK
  - The file *build.properties* in *[LiferayDeveloperHome]\pluginsSdk* contains default settings about the Liferay installation and deployment folder. DO NOT EDIT THIS FILE!
  - Create a new file *build.[user.name].properties: build.**Fatima**.properties. user.name* means the user on the Windows system. Add the text into the file: ***app.server.dir*** *= [LiferayDeveloperHome}\\liferay-portal-6.xx\\tomcat-6.xx*. Plugins SDK is now ready for use.

Structure of the SDK

After the installation the Plugins SDK folder has the following structure

- Most folders in the *[LiferayDeveloperHome]\pluginsSdk* folder contains scripts for creating new plugins for that type (*build.xml* files)
- Place for new plugins are:
  - for portlets: *[LiferayDeveloperHome]\pluginsSdk\portlets\[portlet name]*
  - for exts: *[LiferayDeveloperHome]\pluginsSdk\exts\[ext name]*
  - for hooks: *[LiferayDeveloperHome]\pluginsSdk\hooks\[hook name]*
  - for themes: *[LiferayDeveloperHome]\pluginsSdk\themes\[theme name]*

Development of plugins for the portal

In the following, we will show two examples for the plugin development for the *p-medicine* portal:

- development of a portlet: in this section we provide a step by step instructions for the development of a simple portlet and then we show how we have prepared the plugin for Data Mining developed by Taverna for the older portal version (6.0) for using in the *p-medicine* portal based on the newest Liferay version.
- development of a themes-plugin: we have developed a theme-plugin for representing the p-medicine look and feel.

Example 1: Portlet development

In the first part we show the step by step instructions for the development of a simple portlet. The second part will contain an example for preparing the plugin for Data Mining developed by Taverna for the older portal version (6.0) for using in the *p-medicine* portal based on the newest Liferay version.

Simple portlet development

The following steps should be performed for the development of the simple portlet with the name *my-greeting* (without spaces) and with a display name *My Greetings*.

Step 1: Navigate to the portlets directory in the terminal and enter the following command *create.bat my-greeting "My Greeting"*.

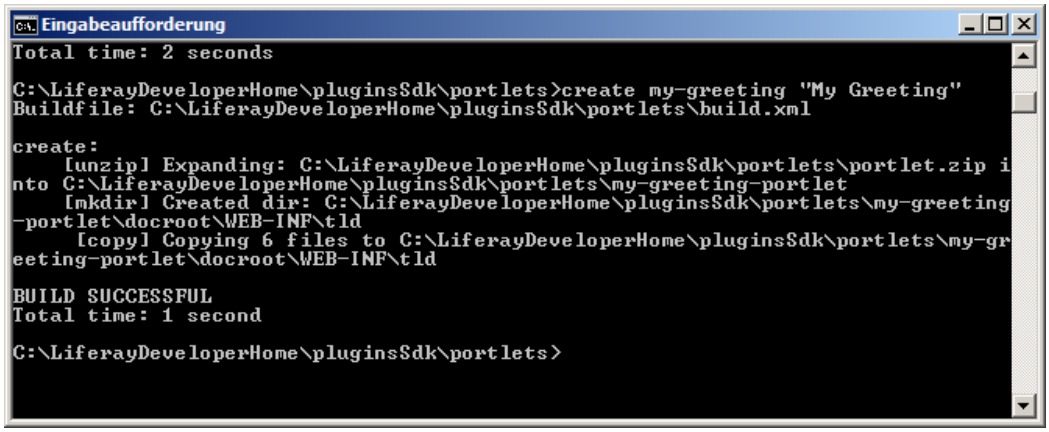

Figure 38: Creation of a portlet

The folder *my-greeting-portlet* is created in the *[LiferayDeveloperHome]\pluginsSdk\portlets* folder.

For implementing of the portlet functionality we have two ways:

- Using Plugins SDK: further develop the portlet where it was created ([LiferayDeveloperHome]\pluginsSdk\portlets\my-greeting-portlet). And then deploy the

portlet by changing in the terminal into the folder *[LiferayDeveloperHome]\pluginsSdk\portlets\my-greeting-portlet* and by executing the command *ant deploy*

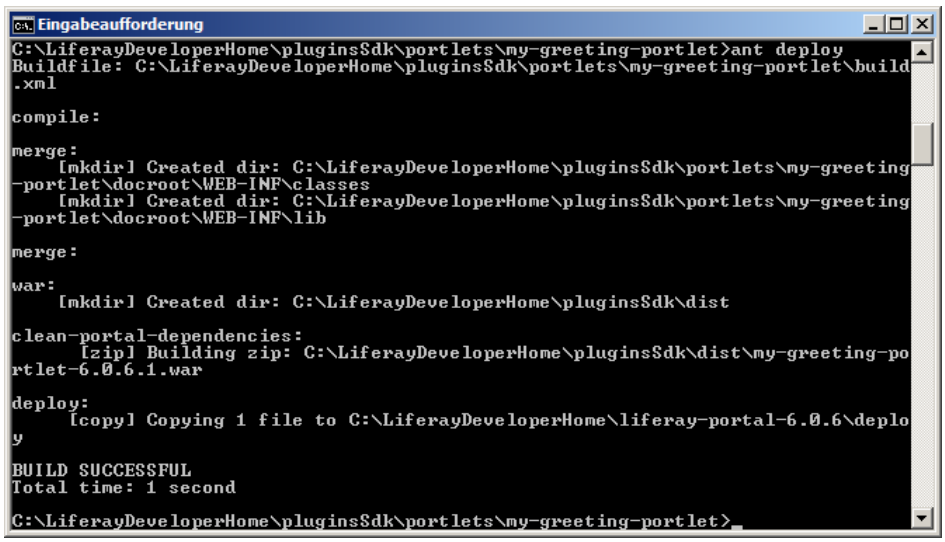

Figure 39: Deploying a portlet

Success case: the message *BUILD SUCCESSFUL* (Figure 39) will be displayed in the terminal and the message: *portlet for my-greeting-portlet is available for use* will be shown in the Liferay console (Figure 40). It means that the portlet is available for using in the Portal.

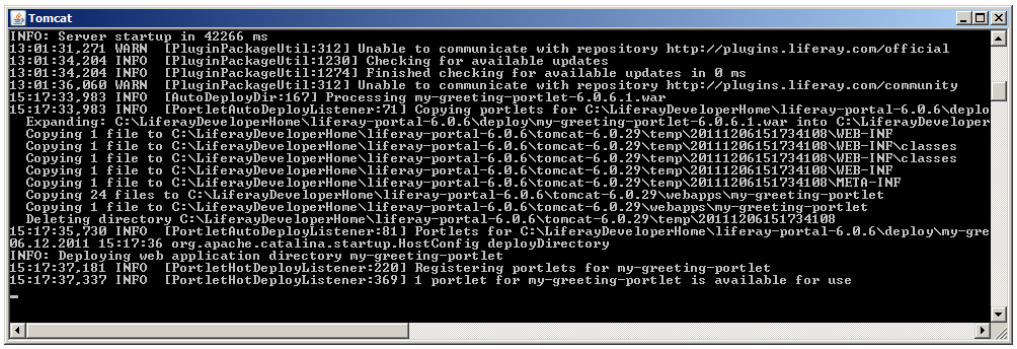

Figure 40: Message about a successfully deploying of a portlet

The portlet is now available in the portal and the user can include the portlet to any page by using the portal's menu *Add--> more --> sample*.

- Using IDE: the newly created portlet should be copied into the IDE and further developed in the IDE. Here the developer can get some compiler errors because some .jar files are missing. The .jars should be added to the project in the IDE. A list of the dependency .jars can be checked in the *[LiferayDeveloperHome]\pluginsSdk\build-common.xml* (see the entries *plugin.classpath* and *portal.classpath*).

Anatomy of a portlet

The portlet's project created in the *[LiferayDeveloperHome]\pluginsSdk\portlets\my-greeting-portlet* folder contains two subfolders: *docroot* (for the development) and *build* (for storing the compiled source code).

The *docroot* folder is the main container for the portlet and includes sub-folders for storing client side files (the *css* folder with files for storing the portlet's layout, the *js* folder with files

for storing the JavaScript files and *jsp* files) as well as the *WEB-INF* folder for the source code and configuration files.

In the Tomcat the *docroot* folder doesn't exist and the client side files are placed direct in the *[LiferayHome]\tomcat-6.0.29\webapps\my-greeting-portlet* folder (or its subfolders).

The structure of the portlet project is displayed on the Figure 41 and Figure 42.

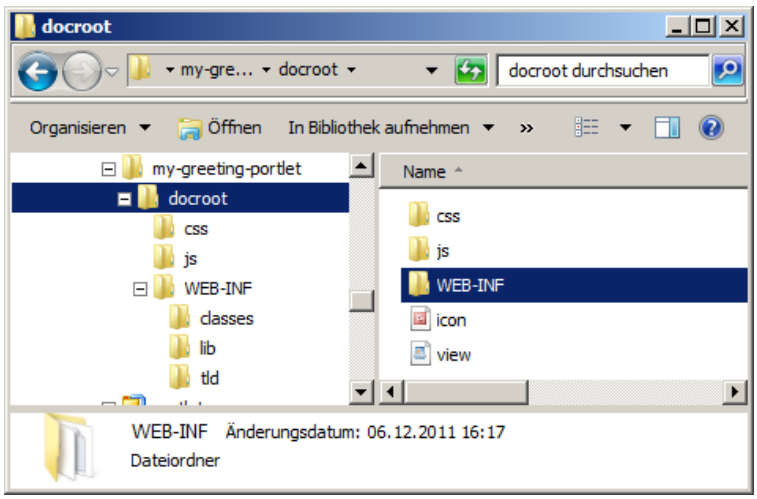

Figure 41: Structure of a portlet project (1)

Portlets created in the SDK are configured by default to use the **MVCPortlet** framework. **MVCPortlet** uses separate JSPs for each page in the portlet. **MVCPortlet** uses by default a JSP with the mode name for each of the registered portlet modes. For example *edit.jsp* for the *edit* mode, *help.jsp* for the *help* mode, *view.jsp* for the *view* mode (default implemented mode), etc. The MVCPortlet is configured in the main standard JSR-286 portlet configuration file: in the *portlet.xml* (Figure 42):

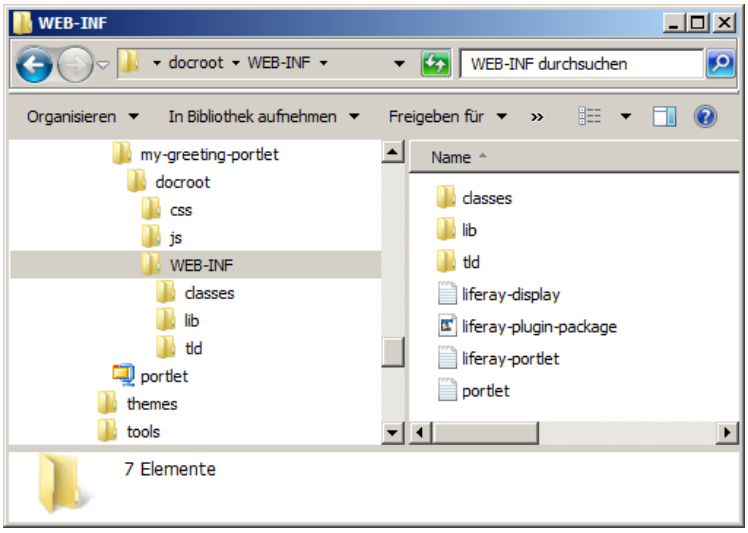

Figure 42: Structure of a portlet project (2)

Configuration files are in the folder *my-greeting-portlet\WEB-INF*:

- The standard JSR-286 portlet configuration is in the *portlet.xml* file
- Optional Liferay specific configuration files are
  - *liferay-display.xml* - describes in which category the portlet should appear on GUI
  - *liferay-portlet.xml* - sets some specific properties, e.g. is the portlet instanceable or not. Instanceable means that the portlet can be added to pages of a community

multiple times. The corresponding DTD is in the definitions folder of the Liferay source code.

- *liferay-plugin-package.properties* - gives the information about the plug-in to Liferay's hot deployer, e.g. which dependency .jars exist for the plug-in. If the .jars already come with Liferay, the hot deployer will extend the plugin's *.war* on deployment by copying the *.jars* inside the plug-in.

For developing more complex portlets (editable portlet, portlet with an action, portlet with multiple actions) as well as other plugins for the portal (ext, hook,etc) see the Development[61] documentation of the Liferay.

Upgrading of the Data Mining portlets developed by Taverna for *p-medicine*

Portlets developed by Taverna for Data Mining are able to run in the older Liferay version (6.0). Because the portlets developed for the version 6.0 cannot run within the portal in version 6.1 which is used for *p-medicine*, we have to upgrade the existing Data Mining portlets. we have performed the following steps for preparing the Data Mining portlets for using them in the *p-medicine* portal:

- Navigate to the *portlets* directory in the terminal and enter the following command *create.bat DataMining "Data Mining".* The folder *DataMining-portlet* was created in *[LiferayDeveloperHome]\pluginsSdk\portlets\.*
- Download the Taverna plugin for Data Mining for the portal version 6.0 from the site *http://dev.mygrid.org.uk/wiki/display/portals/WAR+file+and+source+code* and import them into the Eclipse IDE with the Plugin SDK.
- Edit some files for removing all displayed errors, add missing libraries and compile the project. Here the developer can get some compiler errors because some .jar files are missing. The .jars should be added to the project in the IDE. A list of the dependency .jars can be checked in the *[LiferayDeveloperHome]\pluginsSdk\build-common.xml* (see the entries *plugin.classpath* and *portal.classpath*). The Taverna plugin which includes two portlets (for submission of workflows and for displaying of results of the executed workflows), will be compiled for the newest portal version (6.1).
- Deploy the plugin to the *p-medicine* portal: change in the terminal into the folder *[LiferayDeveloperHome]\pluginsSdk\portlets\DataMining-portlet* and execute the command *ant deploy.*

  Success case: if in the Liferay console the message: portlet for *DataMining-portlet* is available for use is displayed, the portlet is available for using in the portal.

A portal user with corresponding rights can find the Data Mining portlets by using the portal's menu *Add--> more --> Data Mining* (Figure 41). For adding the portlets to any *p-medicine* page a portal has to use the links *Add* next to the portlet name (Figure 43) or he can pick the portlets and drop them to a specific column of the shown portal page.

[61] http://www.liferay.com/documentation/liferay-portal/6.1/development

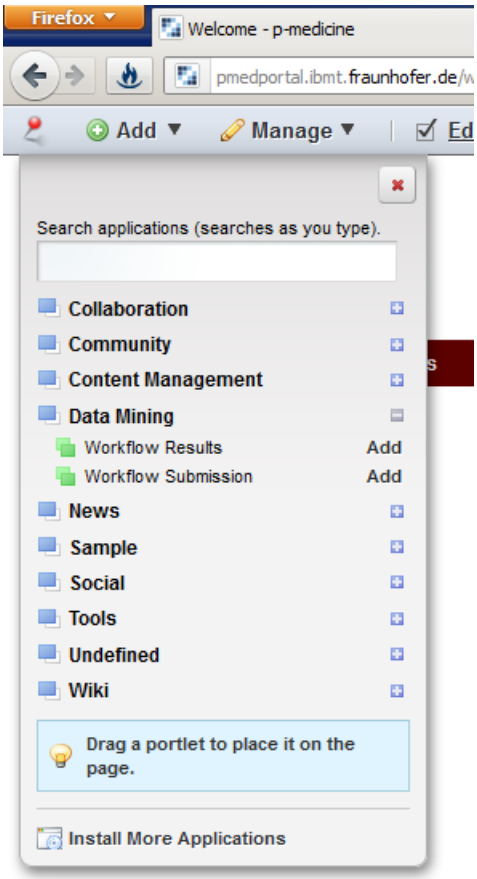

Figure 43: Data mining portlets integrated into the portal

Example 2: development of the *p-medicine* themes-plugin

The following steps should be performed for the development of the themes-plugin for the *p-medicine* layout:

- Creating the new theme *pmed* (without spaces) with the display name P-Med
  - Navigate to the *themes* directory in the terminal and enter the following command *create.bat pmed "P-Med".*

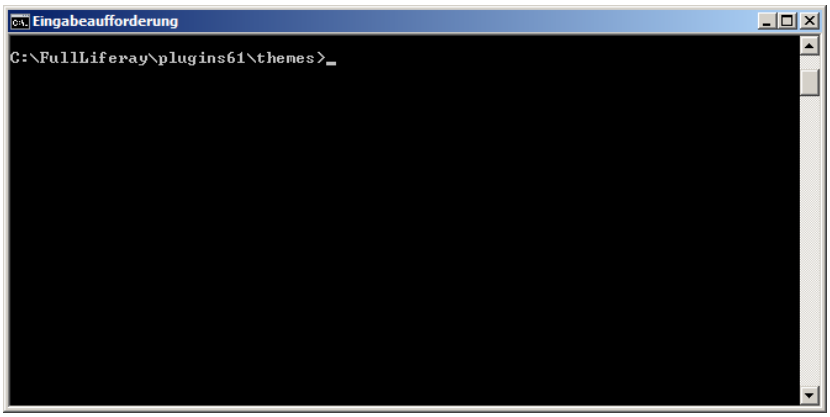

Figure 44: Switching to the "themes" folder for developing a theme-plugin

The folder *pmed-theme* is created in *[LiferayDeveloperHome]\pluginsSdk\themes\.*

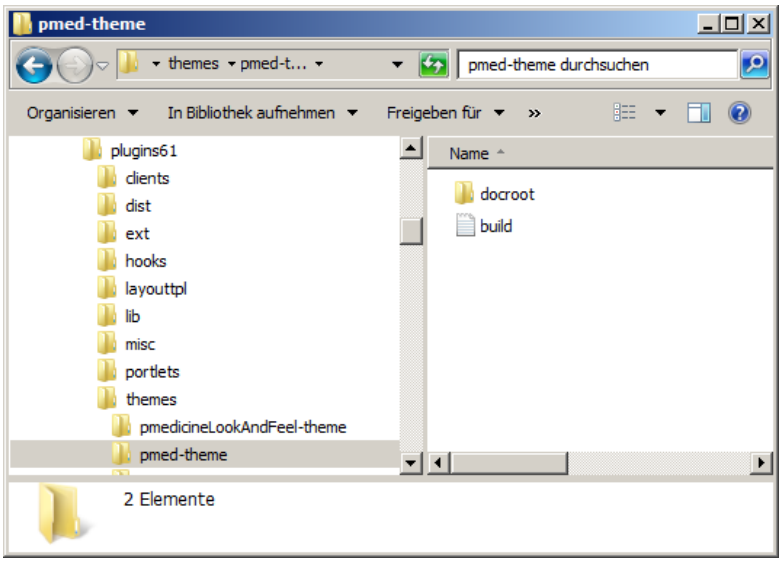

Figure 45: Automatically created project structure for a theme-plugin

- Development of the *p-medicine* theme

  For implementing the plugin functionality we have two ways:

  - Using Plugins SDK: further develop the portlet where it was created *([LiferayDeveloperHome]\pluginsSdk\themes\pmed-theme)*
  - Using IDE (e.g. Eclipse): copy the newly created portlet into the IDE and further develop the portlet in the IDE. Here the developer can get some compiler errors because some .jar files are missing. The .jars should be added to the project in the IDE. A list of the dependency .jars can be checked in the *[LiferayDeveloperHome]\pluginsSdk\build-common.xml* (see the entries *plugin.classpath* and *portal.classpath*).

  The project folder *themes/pmed-theme* contains five sub-folders: *_diffs, css, images, js,* and *templates*.

  The own themes code should be placed in the *_diffs* folder. For customizing of the view the directory structure of the *themes/pmed-theme* should be mirrored in the *_diffs* folder: the subfolders *css, images, js, templates* should be created in the *_diffs* folder where the changed files will be placed. E.g. for customizing the navigation the file *pmed-theme/docroot/templates/navigation.vm* should be copied into *pmed-theme/docroot/**_diff**/templates/navigation.vm.*

Custom styles

For custom styles only one file (*custom.css*) should be copied from the *p-medicine-theme/docroot/css/* to *p-medicine-theme/ docroot/**_diffs**/css/* where all own styles have been defined. The background picture representing the *p-medicine* project on the right side of the portal layout is also defined here as *#pmedlogo*. For adding the picture to the portal layout we have edited the *plugins61\themes\pmed-theme\docroot_diffs\templates\portal_normal.vm* file by adding the text:

```
<div id="pmedlogo">
#pmedlogo()
</div>
```

Thumbnails

For displaying a small preview picture of a new theme for selecting a theme in the look and feel view, a screenshot with the size 150px x 120px of a new view has been placed in *_diffs/imagesthumbnail.png*. The larger version of the screenshot should be saved as *screenshot.png* (1080p x 864p).

- Deploying the new *p-medicine* theme on the portal

  For deploying the plugin functionality we have two ways:

  - Using Plugins SDK: change in the terminal to the folder *[LiferayDeveloperHome]\pluginsSdk\ themes\pmed-theme* and then execute the command *ant deploy*
  - Execute the command *ant deploy* in the IDE (e.g. Eclipse)

    Success case: if in the portal's console the message: *theme for pmed-theme is available for use* the plugin is correctly deployed and is available for using. This new theme can be selected from the list of available themes for using for the layout of the *p-medicine* communities. For doing this the user has to select the menu Manage -->Site Pages (1, Figure 46) on the Dockbar which is the main user navigation menu bar in the portal:

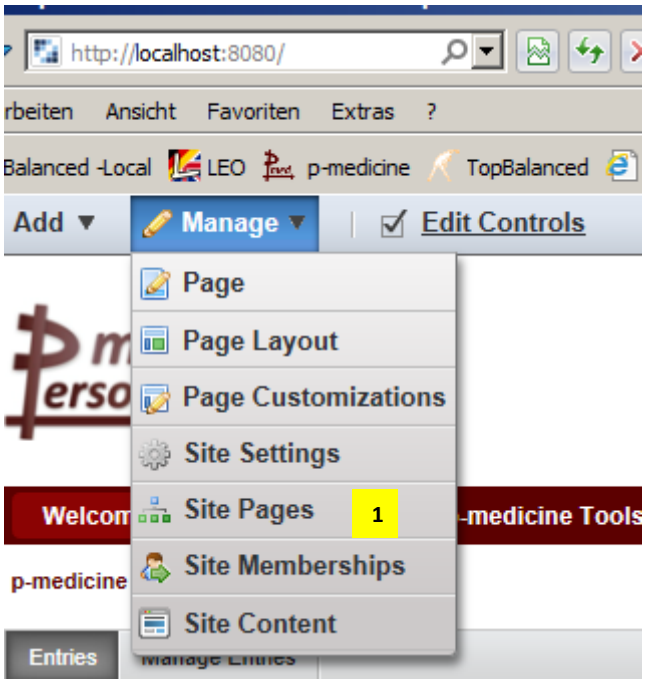

Figure 46: Dockbar menu "Manage" --> "Site Pages"

On the opened view (Figure 47) the user can select in the window opened by using the *Look and Feel* menu either the theme developed for the *p-medicine* project (1) or the default Liferay theme (2).

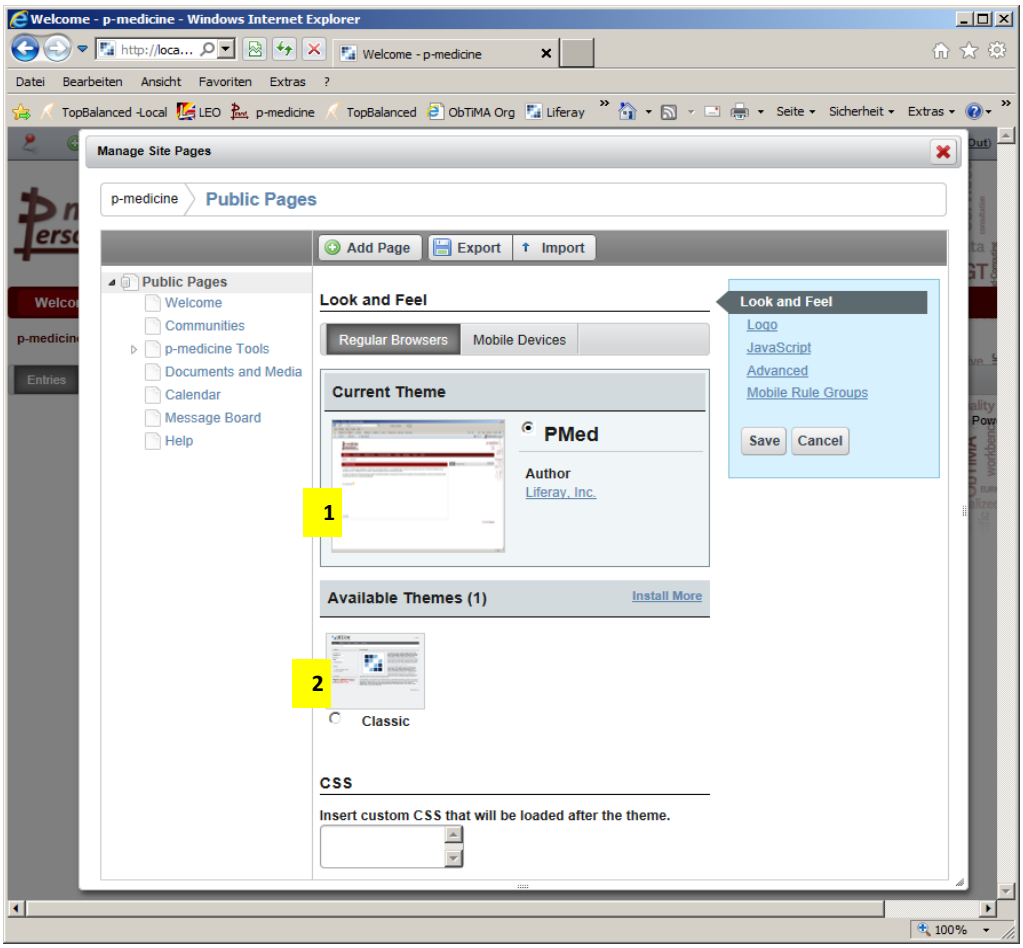

Figure 47: Selecting of a theme-plugin