Intelligent Information Management
Targeted Competition Framework
ICT-2011.4.4(d)

Project **FP7-318652 / BioASQ**

Deliverable **D4.3**

Distribution **Public**

http://www.bioasq.org

Evaluation Infrastructure

Georgios Balikas, Ioannis Partalas, Nicolas Baskiotis,
Thierry Artieres, Eric Gaussier, Patrick Gallinari

Status: Final (Version 1.0)

June 2013

Project

Project ref.no.	FP7-318652
Project acronym	BioASQ
Project full title	A challenge on large-scale biomedical semantic indexing and question answering
Porject site	http://www.bioasq.org
Project start	October 2012
Project duration	2 years
EC Project Officer	Martina Eydner

Deliverable

Deliverabe type	Report
Distribution level	Public
Deliverable Number	D4.3
Deliverable title	Evaluation Infrastructure
Contractual date of delivery	M9 (June 2013)
Actual date of delivery	June 2013
Relevant Task(s)	WP4/Task 4.2
Partner Responsible	UPMC
Other contributors	UJF, AUEB-RC
Number of pages	28
Author(s)	Georgios Balikas, Ioannis Partalas, Nicolas Baskiotis, Thierry Artieres, Eric Gaussier, Patrick Gallinari
Internal Reviewers	Sergios Petridis
Status & version	Final
Keywords	BioASQ, platform, database, web services

Executive Summary

The document describes the overall architecture of the evaluation infrastructure of the on-line BIOASQ Participants Area. The BIOASQ Participants Area is a web application developed for enabling participants to participate in the tasks of the challenge. Its style follows the style of the official website of the challenge (www.bioasq.org). It is publicly available and deployed under bioasq.lip6.fr. Note that this document accompanies deliverable D4.3 which is not report deliverable.

The complete deliverable contains detailed information with respect to the technology that is used for developing the web application. During the description, the flexibility and the scalability of the web application are highlighted as well as the ways it can be adapted and used from other challenges.
In general, participants using the BIOASQ Participants Area can:

- download data and tools for the tasks of the BIOASQ challenge,
- submit their results for each task,
- find guidelines and technical support for the challenge,
- view the evaluation measures for each task of the challenge.

In the document, the BIOASQ Participants Area is considered as the overall project. The project is split into small sub-applications, each one offering a piece of functionality when integrated in the platform. The sub-applications can be seen as reusable modules or components of the main application. There are five sub-applications integrated that offer the total functionality:

- Task1a,
- Task1b-PhaseA,
- Task1b-PhaseB,

} core functionality

- Registration,
- Forum,
- Django-Admin,

The following list summarizes the main points and the logic that is presented in the document:

- The total functionality depends on the functionality that each sub-application adds.
- The sub-applications follow the Model-Template-View pattern.
- The files of each sub-application are in separate folders in the project folder.
- There is a common database where each sub-application defines the necessary tables for its function. Interaction between tables of different applications is supported.
- Users interact with the platform through HTTP requests. The responses of the platform may be HTML pages, JSON strings or whatever type of data is needed.
- Making changes to the application, adding functionality and customizing its behaviour is as simple as adding and editing the files of the Model-Template-View layers. The rest of the platform's functionality remains operational and the minimum trouble is caused.

Contents

List of Figures

1

Introduction

Organizing the series of the BIOASQ challenges requires frequent interaction on behalf of the challenge participants with the BIOASQ team. The BIOASQ challenge consists of two tasks. A short description of each task follows:

- *Task 1A: Large-scale on-line biomedical semantic indexing.* Large-scale semantic indexing is evaluated on the whole of MEDLINE [1]. In particular, participants must classify incoming documents before the human curators do. BioASQ distributes new unclassified MEDLINE documents every week and participants have a limited response time to estimate the MeSH [2] terms of each document.

- *Task 1B: Introductory biomedical semantic QA.* Benchmarks containing development and evaluation questions, as well as golden standard (reference) answers, have been developed. The gold answers have been produced by a team of biomedical experts from research teams around Europe. Established methodologies from QA, summarisation, and classification are followed to produce the benchmarks and evaluate the participating systems The task runs in two phases:

  - *Phase A:* BIOASQ transmits questions from the benchmark while participants have to respond with concepts, snippets and triples in limited time.
  - *Phase B:* BIOASQ transmits questions and concepts, snippets and triples. Participants respond with facts, summaries, etc. The evaluation is based on gold answers while a small percentage will be evaluated manually from the biomedical experts.

For more information about the BIOASQ challenge and the organisation of the tasks consult bioasq.org and Balikas et al. (2013). More information about the preparation of the questions for Task 1B can be found in Malakasiotis et al. (2013).
The evaluation framework and the required functionality for the participants is provided through an online participants area that was developed to support the BIOASQ challenge. In general, the BIOASQ Participants Area (hereafter platform) must:

[1] MEDLINE contains journal citations and abstracts for biomedical literature from around the world
[2] MeSH (Medical Subject Headings) is the vocabulary thesaurus used for indexing articles for MEDLINE.

- support message based communication when interacting and exchanging data,
- support automated communication using web services,
- be modular, as functionality and add-ons are being developed,
- offer the organisers the ability to supervise the challenge in a robust way,
- be easy to be used by the participants.

1.1 A quick tour

The platform is deployed in bioasq.lip6.fr. Note that the platform is different from the main website of the challenge bioasq.org and its role is to provide the evaluation framework needed for the challenge. Figure 1.1 shows the platform's homepage and Figure 1.2 shows the homepage of the administration interface. Participants of the challenge, after logging in to the platform, can:

- find information and guidelines about the BIOASQ challenge,
- exchange data, either using forms or APIs,
- participate in discussions regarding the challenge,
- contact the organisers.

In the following chapters the overall architecture of the platform will be described. In particular,:

- *Chapter 2* describes the overall architecture of the platform,
- *Chapter 3* describes the Model-Template-View layers of each sub-application that is integrated in the platform,
- *Chapter 4* describes the web services that were developed for the platform.

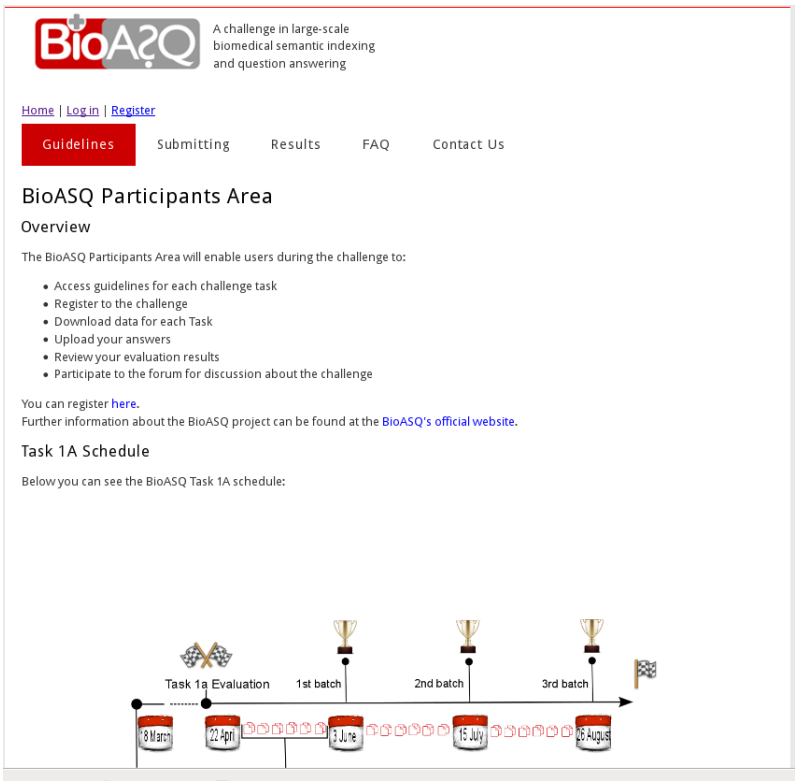

Figure 1.1: The web interface homepage at `bioasq.lip6.fr`

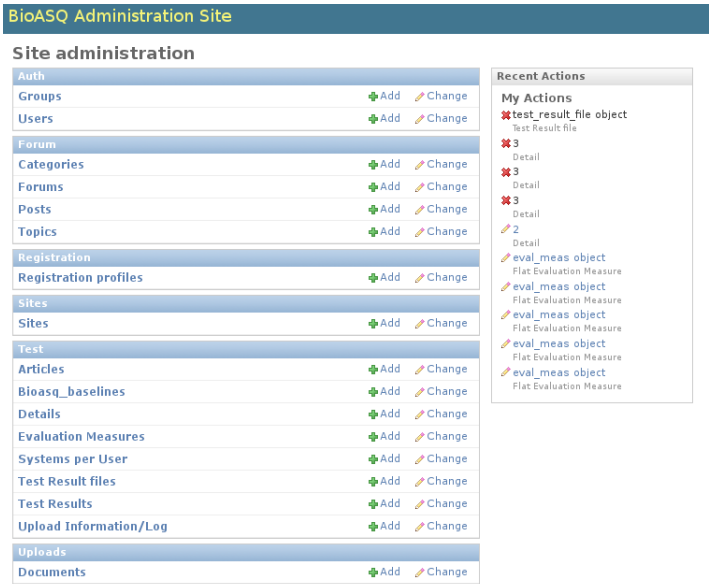

Figure 1.2: The administration interface homepage.

2

Overall architecture

This chapter describes the overall architecture of the platform developed for facilitating the exchange of data between the participants and the BioASQ team. The functionality is presented from:

- the participants and organisers point of view,
- an application point of view following a specific design pattern.

2.1 The Model-View-Controller design pattern

One of the most popular paradigms for user interfaces in software engineering is the Model-View-Controller (MVC)[1] pattern. During the development of the platform a slightly different pattern was followed called Model-Template-View (MTV)[2] pattern. Figure 2.1 presents the basic architecture of the MTV pattern. MTV separates the process of the development of an application into three layers:

1. *the model layer,* which provides an abstraction layer (the "models") for structuring and manipulating the data of an application,
2. *the template layer,* which through a designer friendly syntax provides the rendering of the information presented to the user,
3. *the view layer,* which is responsible for encapsulating the logic behind the application and for processing a user's request and returning the appropriate response.

Each one of the layers is independent from the others. For example, the template layer, which describes the way that the data are presented to the user of the software, is separated from the model layer, which describes the way that the data are stored in the database. This loose coupling of the layers offers the developer the flexibility to adapt the application to the current needs and easily modify it according to future needs. For example, in case the developing team is not satisfied with the user interface of an application or is scheduling major changes in the way the content is presented, changes would occur only in the template layer, leaving the others unaffected.

[1] http://en.wikipedia.org/wiki/Model-view-controller

[2] More information about the MTV pattern and the difference with the MVC pattern in Django Documentation

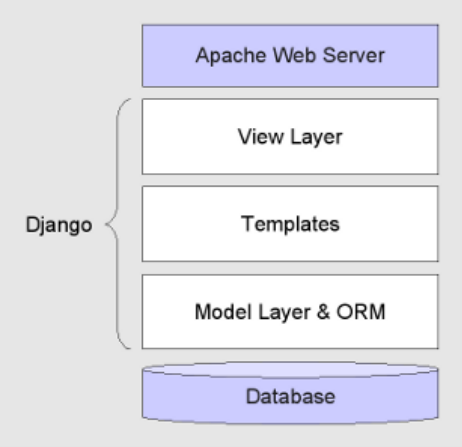

Figure 2.1: The Model-View-Template pattern that was followed during the development of the platform

2.2 The required functionality

There are two groups of users of the platform: the organisers and the participants of the challenge. With respect to this segregation the functionality requirements of the platform can be summarized as following:

- The organisers should be able to:
  - create and provide the data (e.g. test sets) and tools (e.g. a tool for trasnforming XML objects in JSON objects of a particular format) for each task of the BIOASQ challenge,
  - use an administrator interface for supervising the challenge and infering about the participation or about problems that may occur.
- The participants should be able to:
  - download data and tools developed from the BIOASQ Team for the challenge,
  - submit results for the challenge,
  - browse the evaluation measures that are calculated for Tasks of the challenge,
  - find guidelines and technical support for the challenge.
- In a more detailed level, base functionalities include:
  - provide registration to the participants, which can be further split and described as:
    - authentication process,
    - profile editing,
  - provide web services and a web interface for automated ways of interaction and data exchange between the participants and the platform,
  - provide feedback to the participants depending on the result of their actions (e.g. informing messages about errors occurred when submitting results),
  - provide communication between the participants and the BIOASQ team.

The above list of functionalities can also be organised from an application point of view. Consider the BIOASQ Participants Area as the overall application that can be split into small sub-applications, each one offering a piece of functionality when integrated in the platform. The sub-applications can be seen as reusable modules or components of the main application. Each one of these components follows the MTV pattern. The sub-applications integrated in the platform are:

- Task1a,
- Task1b-PhaseA,
- Task1b-PhaseB,

} core functionality

- Registration,
- Forum,
- Django-Admin,

where:

- "Task1a", "Task1b-PhaseA" and "Task1b-Phase B" offer the 'core' functionality required for each one of the tasks of the first year and can be modified according to the requirements of the second year of the BIOASQ project or even adapted to the requirements of other projects.
- "Registration" offers the functionality for registering in the platform.
- "Forum" creates a BIOASQ Discussion Area.
- "Django-Admin" creates the interface where administrators can supervise the BIOASQ challenge.

Figure 2.2 shows the sub-applications of the platform and the logic behind the interaction of the uesrs with the platform. The "Django-Admin" sub-application is omitted from the figure since it is created automatically as it will be described in the following chapters.

An example pointing the benefits of this design follows:

Let the platform have a simple forum that allows participants to discuss only by making posts. If the BIOASQ team decides to change the way that the posts are rendered (e.g. colors, widgets etc.), the developers have to make changes only to the template layer of the "Forum" sub-application and leave the other layers of the application as well as the other applications untouched. In addition, imagine the scenario where the BIOASQ team decides a forum that enables voting and poll is needed. Then, the possible scenarios for the developers are two:

- removing the old forum and developing a new forum or adding the necessary functionality to the layers of the existing one,
- integrating an open source forum and modifying it according to the new requirements.

Either way, only the "Forum" sub-application of Figure 2.2 will change. The other sub-applications remain unaffected and functional while the minimum trouble is caused.

Another common task in the BIOASQ challenges is downloading a test set and submitting results. In order to provide an automated way for doing this, web services have been developed. Using web

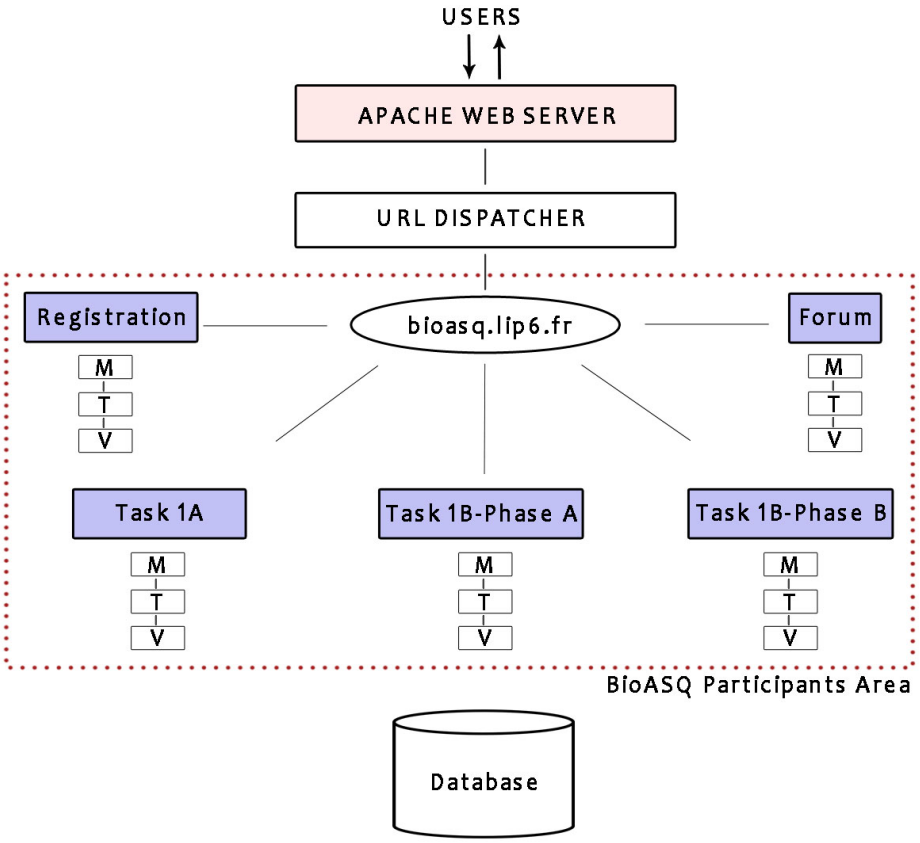

Figure 2.2: The overall architecture of the platform.

services participants can download and/or submit data programmatically, in an automated way. In effect, participants of the BIOASQ challenge, can choose to download the test sets and submit the results of both phases in two ways:

- using the web interface in bioasq.lip6.fr,
- using the web services, which have also been developed under the URL bioasq.lip6.fr.

More information about the ways of submitting results for the BIOASQ challenge can be found in the following chapters of this document and in the online guidelines of Task 1A[3] and Task 1B.[4].

2.3 The technology and the logic used

Currently, there are many frameworks (Ruby on Rails, Django, etc.) aiding the design and development of web applications. In addition, there are many platform and programming language independent (e.g. JSON, XML, YAML) formats that are used for exchanging data. In this section, the technology and the logic that is used in our platform is presented.

2.3.1 The technology

Taking into account the functional requirements and the specifications of the platform an on-line web application has been developed. After comparing the available technologies and frameworks the following design decisions for the tools that would be used were made:

- The Django framework (for more details consult Django). Django is a high-level Python web framework that supports rapid development and clean, pragmatic design. Because Django was developed to support a fast-paced newsroom environment, it was designed to make common web-development tasks fast and easy. It offers many features and there is a vibrant on-line community that supports it. It is based on the DRY principle (Don't Repeat Yourself) and tries to achieve loose coupling between the application's components as this makes them reusable and gives the developer the ability to make changes and improvements in the code without affecting the entire project. The main reasons for selecting Django are:
  - it is a light platform with easy translation of syntax that uses Python, which is one of the simplest programming languages,
  - it helps building an application and adding functionality and add-ons in phases (Task 1A functionality was developed first, Task 1B functionality was added later),
  - there is a very active community supporting it.
- MySQL[5]: Django supports as many features as possible from many databases. The database that was chosen to be used for storing the data is MySQL. Django offers a functional database backend for MySQL. The main reasons for choosing MySQL include the backend that Django offers for this database, the scalability and the flexibility that MySQL is known for as well as the previous experience of the developers using this database.

[3] bioasq.lip6.fr/general_information/Task1a/
[4] bioasq.lip6.fr/general_information/Task1b/
[5] http://www.mysql.com/

- RESTful JSON based web services: the Representational State Transfer[6] (hereafter REST) design style for developing web services was used. REST uses HTTP requests for exchanging data. The format for exchanging data that was selected is Javascript Object Notation (JSON)[7]. Some of the main reasons for choosing JSON are:
  - it is simple and has a structured format,
  - it is human readable and text based,
  - it is programming language independant,
  - the previous experience from BioASQ partners.

2.3.2 The logic of the application

Figures 2.2 and 2.3 present the logic that the application uses. In detail, the platform uses the following logic each time a participant visits the web page:

1. the participant opens a URL in his browser,
2. a URL dispatcher decodes the URL and triggers the appropriate view[8],
3. if the URL asks for a simple static HTML page (e.g., the F.A.Q.), then a generic view is triggered and the page is returned to the participant,
4. if the URL asks for a more complex page, where data from the local database have to be selected depending on the participant's identity, then a custom view is triggered that selects the appropriate data,
5. custom views usually have their own template that inherits from a base template. Thus, we don't repeat the same snippets of HTML code and it is easier for the developers to read and debug the code,
6. the response is sent to the participant's browser. Apart from HTML code, it can be test sets, files, tools, or whatever form of data is necessary for the BioASQ challenge.

In case that the URL does not exist or an error occurs in the views, standard 404.html and 500.html pages are displayed.

[6] http://en.wikipedia.org/wiki/Representational_state_transfer
[7] www.json.org
[8] Views in Django encapsulate the logic, they are functions that control the output based on the input.

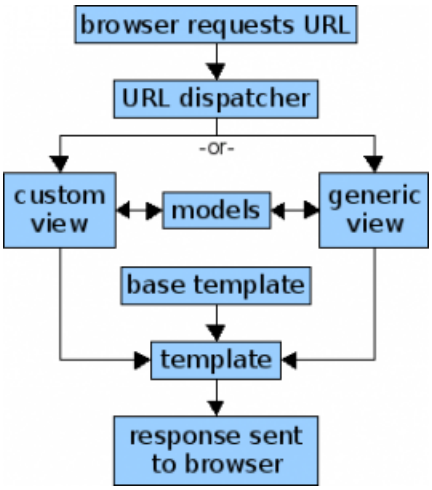

Figure 2.3: The logic the platform uses. URLs are decoded, the appropriate views are triggered and a response is sent to the browser.

3

The Model-Template-View layers

In this chapter the sub-applications along with the functionality they provide when integrated to the platform will be discussed. When necessary, details about the Django web framework and the design decisions that were made will be given. The description of each application begins from the the "model" layer where the database tables are introduced. The "template" layer of the sub-applications is described using screenshots of the platform. The logic of the application (the "view" layer) is also analysed as this helps to understand the actions that the platform performs.

3.1 The file system structure

The filesystem is organised following the requirements of Django. In the project directory each sub-application lives under a folder with the name of the sub-application. For example, all files concerning the "Forum" sub-application live under a folder named "Forum" in the project directory. The folder of each application usually contains some (there are applications that contain all of them) of the following files and folders:

- the "models.py" file. In this file the database tables are defined. When the developer wants to add a table in the database, he adds the description of the table in "models.py". The file is associated with the model layer of the application.
- the "views.py" file. This file contains the view functions, or views for short. They are Python functions that take a web request and return a web response. This response can be the HTML contents of a web page, a redirect, a 404 error, a JSON document, a zip file, or any kind of data needed for the application. The view itself contains whatever arbitrary logic is necessary to return that response. The file contains the biggest part of the logic behind the application.
- the "urls.py" file. In this file the URLs that are registered for the application are defined. For example, to create a Frequently Asked Questions (FAQ) under `bioasq.lip6.fr/faq/` the developer has to register the relative URL `/faq/` under the application that will serve the FAQ.html. This is done by simply adding the `/faq/` in the "urls.py" file. Figure 3.5 shows the FAQ of the platform. The "urls.py" provides a link between the template and the view layer of the application.

- the “forms.py” file. When an application uses forms, the forms are registered in the “forms.py” file of the application. The file is mainly associated with the view template.
- the “Templates” folder that contains the HTML pages that are served for each one of the application’s URLs. The folder is associated with the template layer.
- the “static” folder. Web applications generally need to serve additional files such as images, JavaScript, Jquery or CSS. The static folder of each sub-application contains the necessary files of this kind that the sub-application needs. The folder is also associated with the template layer.

Each time a participant asks for a URL in his browser, the appropriate view is triggered. The view interacts with the database if necessary, and sends a response to the participant. The response may be an HTML page or some data. The way that the application serves the participant’s request is defined in the above-mentioned files.
Making changes or adding functionality is pretty straightforward; adding or removing functionality means adding/removing or editing the specific folders and the files that exist in the project directory. The described file systems structure helps towards this direction as the files have been sorted following a logical basis. From the above it is concluded that developing the application while following the MTV pattern makes it flexible, customizable and modular, which responds to the initial requirements of the platform.

3.2 The MTV pattern for each one of the sub-applications.

3.2.1 The “Registration” sub-application

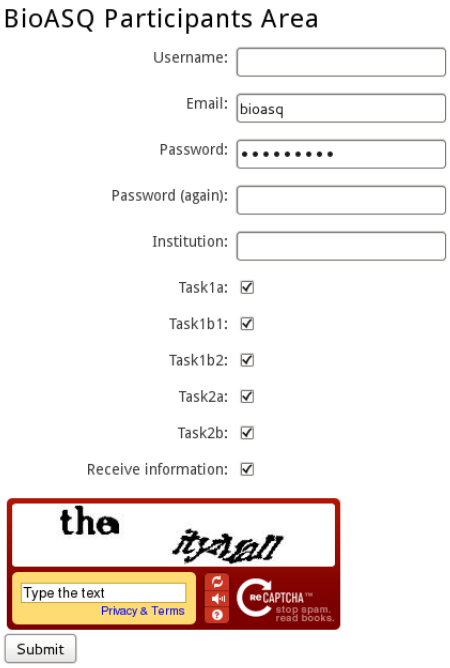

Figure 3.1: Template view: the form the user fills when registering to the platform.

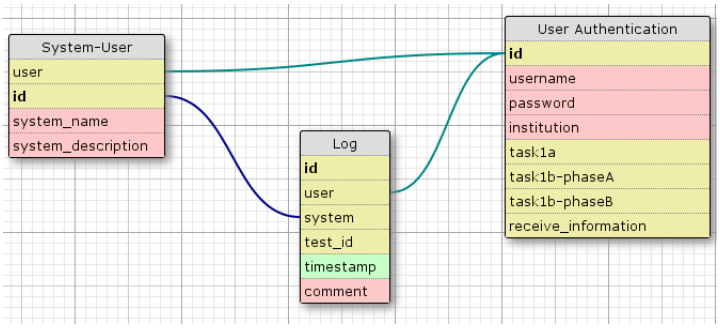

Figure 3.2: Model view: the database schema used from the "Registration" sub-applicaiton.

The "Registration" sub-application addresses the user registration to the platform. Registered users can download and submit data to the platform, can download tools that the BIOASQ team has developed and can access and post to the forum.
The database schema of the sub-application is shown in Figure 3.2. The "User Authentication" table holds the registration information of participants who register to BIOASQ challenge. After filling a registration form, a registration profile is created with a username, a password, an active e-mail account, affiliation details and the tasks where the participant intends to participate. Participants can edit their profile information after logging in the platform.
For each one of the challenge tasks, registered participants are allowed to submit answers for more than one systems. Those systems can be registered in the profile of a logged-in participant by simply filling a form with a "System name" and a "System description". This information is stored in the "System-User" table of Figure 3.2.
The views that offer the "Registration" functionality live under the "Registration" sub-application. When a user submits the registration form of Figure 3.1 the platform checks that all the necessary fields are filled and the username is unique. Then it stores the information to the database. An activation e-mail containing the activation link is sent to the participants. After clicking on the activation link of the e-mail, the participant is considered active and can log in to the platform. When the participant's user information is stored in the database, the password is stored after applying a hash function on it. When he tries to log in and fills in the form his password, the same hash function is applied and the outputs are compared. That way the password of the participants is never stored in his original format in the database. In addition, while registering, participants have to fill the CAPTCHA form shown in figure 3.1. The CAPTCHA form is used to prevent machines from registering.

3.2.2 The "Forum" sub-application

A part of the platform's required functionality concerns the communication between the BIOASQ participants and the BIOASQ team. The platform provides two ways of communication:

- the BIOASQ Discussion area, which is a group of forums available to the logged-in participants,
- a contact form, which is publically available i.e., there is no need to register in the platform to use the contact form.

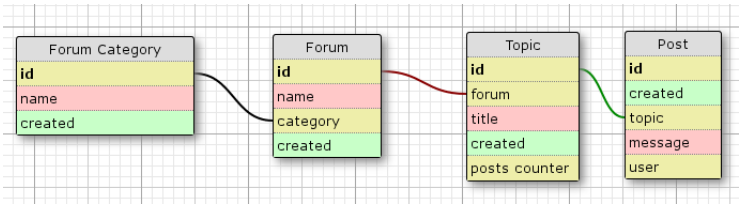

Figure 3.3: The database schema of the "Forum" sub-application.

Figure 3.4: The BIOASQ Discussion Area as can be found in the online platform in bioasq.lip6.fr/forum/.

The BIOASQ Discussions Area

The general schema of the BIOASQ Discussion Area is designed by the administrators of the platform. The administrators can create categories of forums and for each category they can create forums. On the other hand, the participants of the challenge, after logging-in, can create topics and make posts in each one of the forums.
Figure 3.3 shows the database schema that the "Forum" application uses. The "Forum Category", "Forum", "Topic" and "Post" tables are used to store the information that is required. Currently, in BIOASQ Discussion Area, that can be found in http://bioasq.lip6.fr/forum/, there are four forums entitled "BioASQ-General", "BioASQ-Task 1A", "BioASQ-Task 1B/Phase A" and "BioASQ-Task 1B/Phase B". They all belong to the "BioASQ Discussions" category. Figure 3.4 shows the BIOASQ Discussion Area and the available forums.
The views that offer this functionality live under the "Forum" sub-application. Due to the fact that the platform gives access to the forum only to logged in participants, when a participant that is not logged in asks for a forum URL he is redirected to the log-in page of the platform. After logging-in he is redirected back to the BIOASQ Discussion Area. A post and a topic counter is maintained for each one of the forums providing comprehensive information about the participation in the discussions.

The contact form

Apart from the BIOASQ Discussion Area, a contact form (http://bioasq.lip6.fr/contact/) is maintained in the platform. The contact form is available and accessible from every visitor of the plat-

form; it is not required to be logged-in to use it. Its main function is to provide an easy and fast way for people that are interested in the challenge to contact the BIOASQ team, mainly about the issues the platform addresses e.g. the tools, the data formats etc. Notice that for more general or administrative questions the contact information of the BIOASQ challenge organisers is available in bioaq.org.

3.2.3 The tasks sub-applications

An abstract approach

There are three sub-applications serving the "core-functionality" of the BIOASQ challenge. Each sub-application provides the functionality for each one of the tasks. Apart from the details and the differences of each application due to the different data and structure of the tasks, there are some common elements in their functionality. The following list enumerates these elements with a focus on the "model" layer and the "template" layer since the "view" layer is adapted to the special needs of each task.

- In the "model" layer:
  - the test sets of each task are stored in tables with several common fields containing the information about the releases. For example, every test set has:
    - an "id",
    - "started", "finished" date/time fields,
    - a "path" field, indicating where the test file is saved,
    - a "number_of_downloads" field, that is incremented every time a participant downloads a test set,
  - the participants results are saved in "Test Results" tables with each record containing information about the test results (test set, system etc.),
  - a log table is updated with the participant's actions,
  - the evaluation measures of each task are stored in "evaluation measures" tables.
- In the "template" layer:
  - a list with information about the released test sets. In addition, the test sets have to be available for downloading before and after their expiration,
  - when there is an active test, a form for submitting results,
  - a list with the result files a participant has submitted. In addition, the files have to be available for downloading only from the participant that submitted them.

Those common fields have been used to create an abstract layer for each one of the applications that offer the core functionality. The applications inherit from this abstract layer. That way, modifying the platform or adapting it to the needs of another challenge is easy, as the core requirements of each challenge are the same.

The base template

Another feature of the abstract approach that is used is the inheritance of every template from a base template. The template layer provides a designer-friendly syntax for rendering the information to be presented to the user. Templates inherit from a base template following the DRY (Don't repeat yourself) principle. There are certain tags and blocks used in the HTML code where the necessary information

that is passed from the "view" layer to the "template" layer is rendered. One of the most powerful Django's part regarding templates is template inheritance. It allows the developers to build a base skeleton template that contains the common elements of the website and define blocks that child templates can override. Figure 3.5 shows the elements of the BIOASQ web pages. In each page there are four elements:

- the header that contains the BIOASQ logo and some options depending on the situation of the visitor (e.g. logged in users can log out or edit their profile),
- the site navigation menu,
- the content, which depends on the URL,
- the footer, where some information about the project is available.

Three out of four elements of the page are standard and don't change throughout the platform. These elements constitute the base template of the platform that every other template of the platform inherits from.
In the following subsections the sub-applications serving each task will be presented.

The "Task 1A" sub-application

The required functionality for Task 1A includes:

- downloading test sets for Task 1A,
- submitting results for Task 1A,
- downloading participant result files in the format they are stored in the platform's database.

As a result, the function allows to:

- select from the database the test sets of Task 1A,
- check if there is an active test set for submitting results. In case there is one, the "Submit Results" form is initialised with the logged in participant's systems. In figure 3.7 this form is visible. Since participants can submit results for five systems at maximum, the dropdown menu of the form is initialized with the systems that the logged in user has registered,
- search in the database the files the logged in participant has previously submitted and provide them with download links.

Figure 3.6 shows the database schema used for "Task 1A" sub-application. The "Test-Detail" and the "Article" tables hold the data for the test sets of Task 1A. Each test release creates a record on "Test-Detail" with the essential information e.g., starting and finishing timestamps of the test release, paths to the "raw" and vectorized versions in the server and the size of the test release. The PMID, the title and the abstract of each article of the test set are saved in the "Article" table. Each record of this table is associated to the appropriate test release with a foreign key relation. The "Test Result" table holds the files with the results that the participants submit. During the submission of results the platform checks:

- the authentication credentials of the participant, since they can upload results only for systems that belong to them,

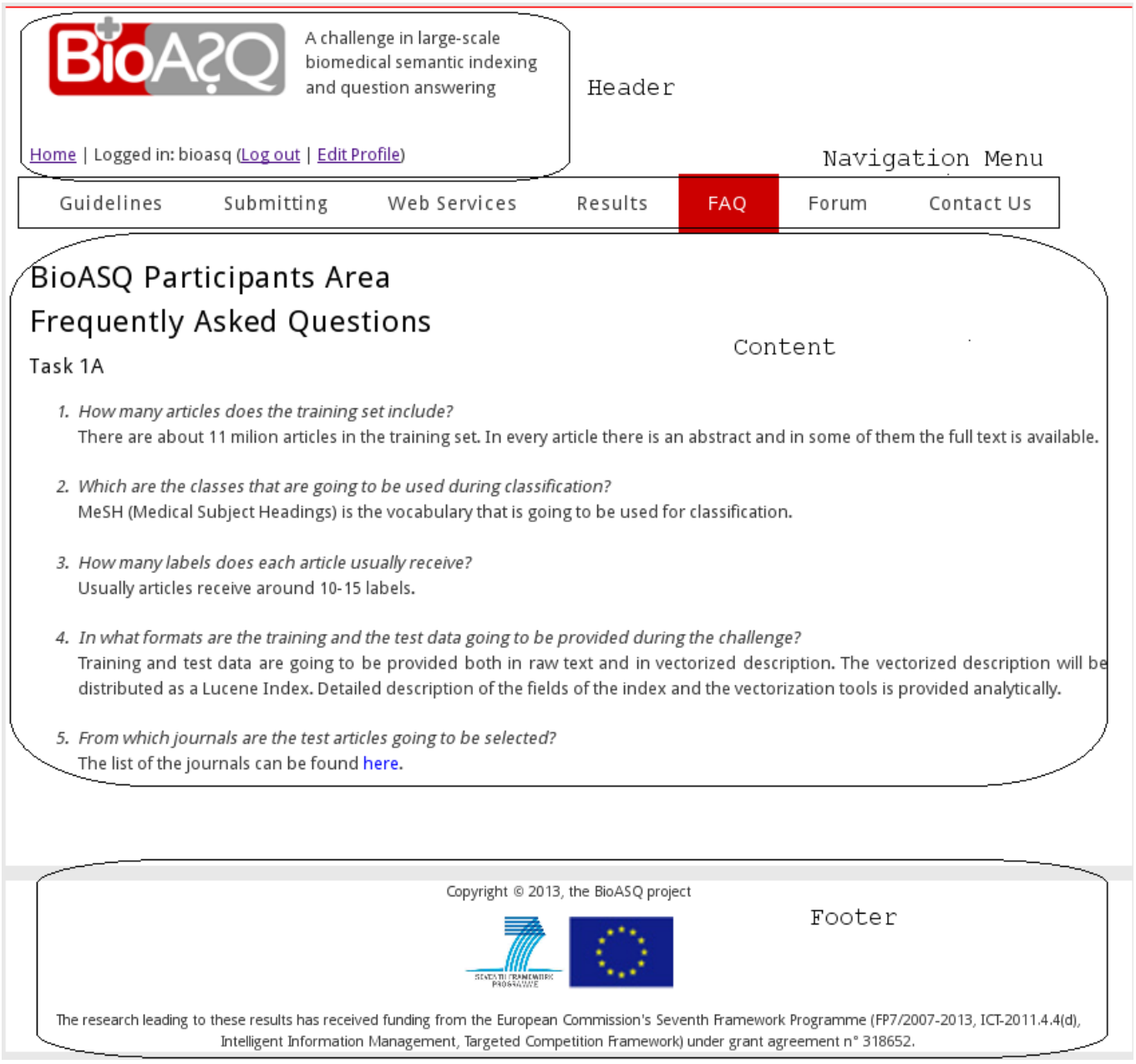

Figure 3.5: The elements of the BIOASQ Participants Area web pages.

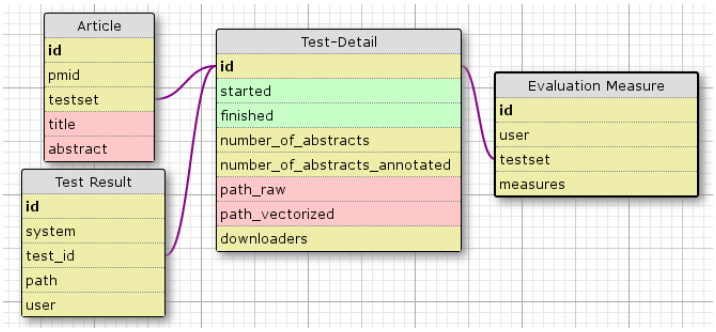

Figure 3.6: The database schema of the "Task 1A" sub-application.

- that the participant submits results for every article of the given test set,
- that the results of the participant are valid, namely that the labels the participant submits for every article of the test set belong to the MeSH hierarchies.

During evaluation, for each one of the "Test Result" table records, a record in the "Evaluation Measures" table is added containing the evaluation measures that were calculated. Figure 3.7 shows the web interface in `bioasq.lip6.fr/Tasks/1a/` where a user can download data, submit results and download the files he has submitted in the format they are stored in the platform's database. In this figure there are three test sets available: the "dry run" test set, and the test sets of week 1 and 2 of the first test batch. There is also an active form for submitting results and a table containing the files that the participant has submitted for this task.

The "Task 1B-Phase A" and "Task 1B-Phase B" sub-applications

The required functionality of "Task 1B-Phase A" and "Task 1B-Phase B" includes:

- downloading test sets for Phase A and Phase B,
- submitting results for the active test sets,
- downloading user result files in the format they are saved in the platform's database,

As a result, the platform allows to:

- select from the database the test sets of Phase A and Phase B,
- check if there is an active test set for submitting results. In case there is, the "Submit Results" form should be displayed and the drop down menu with the "System name" must be initialized with the systems the participant has registered,
- search in the database the files the participant has previously submitted.

The requirements of both phases of the BIOASQ challenge are similar. The difference lays in the answer data types and in the types of errors that the platform checks before storing the data. As a result, the "model" layer and the "template" layer of the sub-applications are the same while the "view" layer is different. Figure 3.8 shows the database schema that "Task 1B-Phase A" and "Task 1B-Phase B" sub-applications use. The tables "User Authentication" and "System-User" are the same with the corresponding ones described for the "Registration" sub-application. With respect to the other tables:

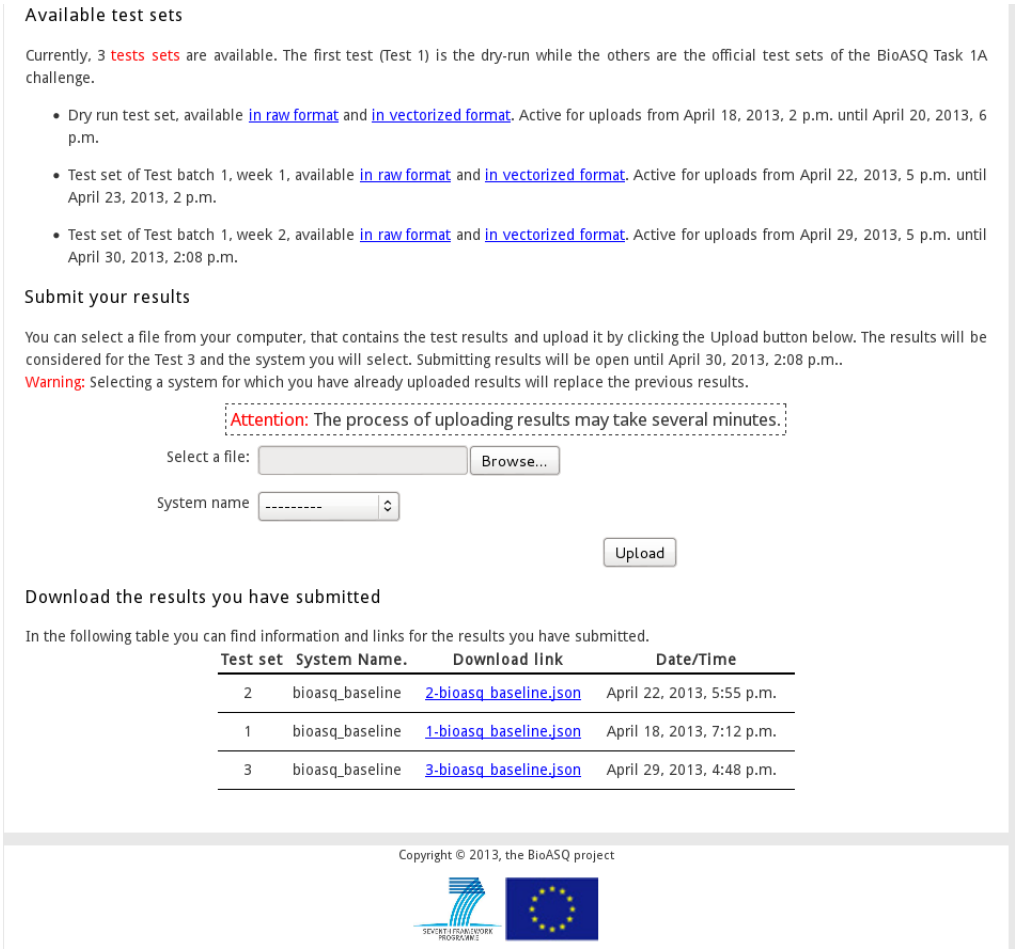

Figure 3.7: The web page where a user can download data, submit results and download the files as they are stered in the platform's database.

- The table "Testset" holds the information about the test releases of both phases e.g. start and finish timestamps, the phase that the test belongs etc.
- Each test set is linked with a foreign key relation to many golden questions ("Golden_question" table). The golden questions are fully annotated questions created by the biomedical experts of the BIOASQ Team.
- The "Test_result_file_1b" table holds the participants submitted result files.
- The "Evaluation_measures_phaseA" and "Evaluation_measures_phaseB" tables hold the evaluation results for the corresponding evalution measures. Since the evaluation measures used in the two phases of Task 1B are totally different between them, two separate tables are used to store the information. Notice that these tables don't contain each one of the evaluation measures that will be calculated in order to save space. For more information on the evaluation measures consult (Balikas et al., 2013).
- The "Log" table holds information about the participant's activity and the errors that may occur.

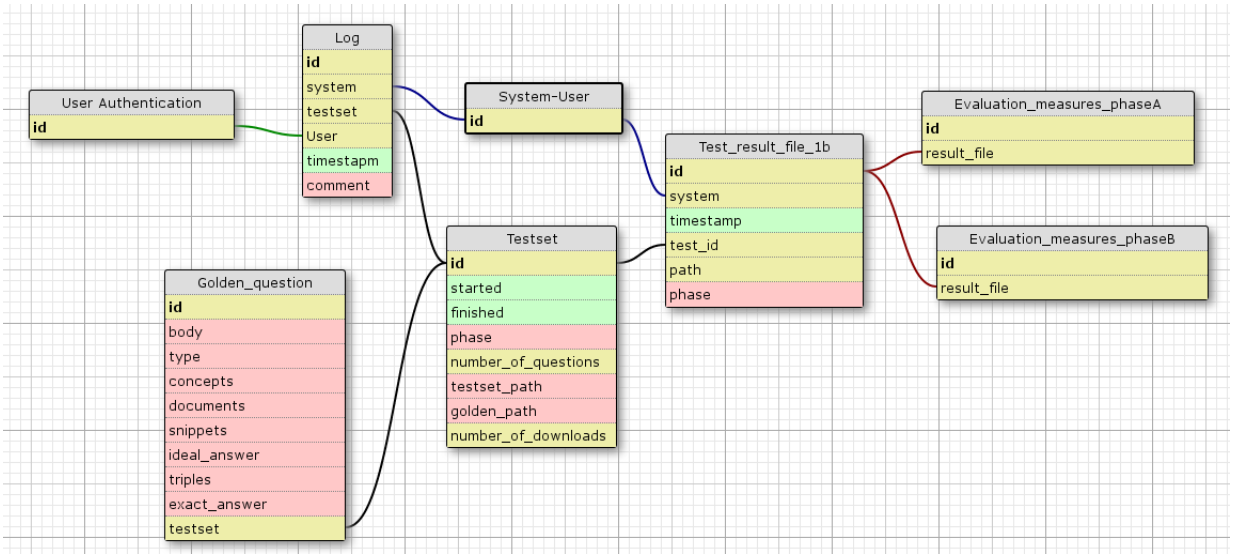

Figure 3.8: The database schema that "Task 1B-Phase A" and "Task 1B-Phase B" sub-applications use.

Figure 3.9 shows the web interface in bioasq.lip6.fr/Tasks/1b/phaseB/ where logged in participants can:

- download the test sets of Phase B,
- submit their results,
- download the files they submitted as they were saved in the platform.

In the Figure 3.9 two test sets are available: a "dry run" test set and the first official test set of the challenge. In addition, there is no active test set and as a result, the form for submitting results is not displayed to the user.
A similar web interface is available for Phase A in bioasq.lip6.fr/Tasks/1b/phaseA/. The logic behind the "Task 1B-Phase A" and "Task 1B-Phase B" sub-applications is similar to the logic of "Task 1A" sub-application. In the example of Figure 3.9, the platform selects from the "Testset" table the available test sets of Phase B. If there is an active test set, the form for submitting results is displayed. If the participant has previously submitted results for the Phase B of the challenge, a list with links to the corresponding files is also displayed. During the submission of results for each phase of the task, the platform checks:

- for Phase A that:
  - the participant submits results for one of his systems,
  - the participant submits at most 100 concepts, 100 articles, 100 snippets, and 100 RDF triples per question,
- for the exact answers of Phase B that:
  - for each yes/no question, the exact answer of each participating system is either "yes" or "no",

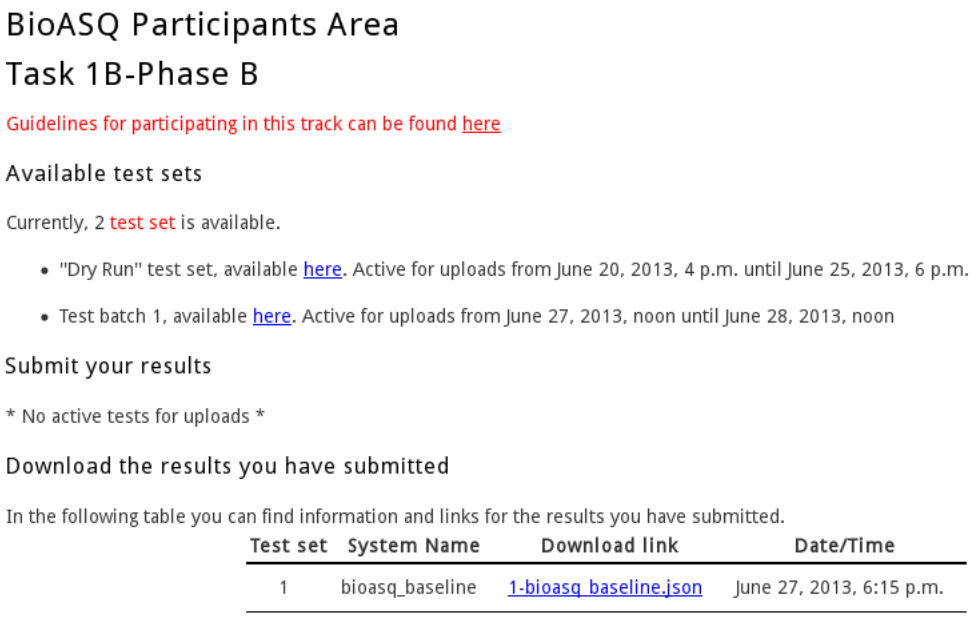

Figure 3.9: The web interface for Task 1B, phase B. A participant can download data, submit results and download the files as they are stered in the platform's database.

  - for each factoid question, each participating system returns a list of up to 5 entity names (e.g., up to 5 names of drugs), numbers, or similar short expressions, ordered by decreasing confidence,
  - for each list question, each participating system returns a single list of entity names, numbers, or similar short expressions, jointly taken to constitute a single answer (e.g., the most common symptoms of a disease). The returned list contains no more than 100 entries of no more than 100 characters each,
  - no exact answers are returned for summary questions.

- for the ideal answers of Phase B that:
  - the ideal answer of each questions is sorter than 200 words.

3.2.4 The administrator interface

One of the main advantages of the Django framework is that an administration web interface is automatically generated. The interface contains a list of tables that correspond to the database tables. The administrators of the BIOASQ challenge can edit, add or remove the content of the tables. The interface is highly customizable. Figure 3.10 shows the table that contains the test articles of Task 1A of BIOASQ challenge. On of the actions supported in the interface is selecting articles (notice the "Select boxes" in the left of each article) and deleting them. More information on the Django Administration site is available in https://docs.djangoproject.com/en/dev/ref/contrib/admin/

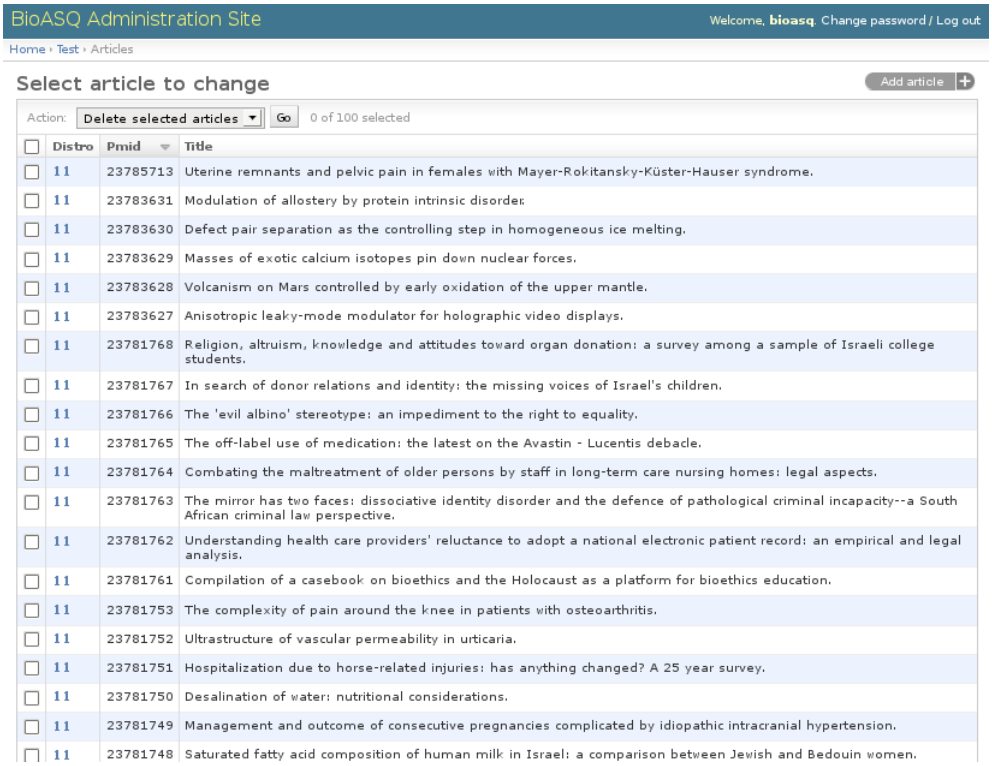

Figure 3.10: The table that contains the test articles of Task 1A in the Django administration interface.

4

The web services

Participants of the BIOASQ challenge exchange data with the platform in a regular basis. In order to automate this process web services have been developed. The services use HTTP-GET and HTTP-POST requests to exchange data. Participants can integrate the web services in their code and automate the process of downloading results, processing them and submitting the results in the platform.
On the other hand, the creation of the test sets and the evaluation process for Task 1A is also based on web services, since the MEDLINE annotations for the articles that are given as tests to the participants become available after the test releases. The design patterns and the details of those services are analysed in the following sections.

4.1 The REST style of developing web services

REST defines a set of architectural principles by which you can design web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. REST is simple to use. The principles it follows are simple and in total agreement with the requirements of the BIOASQ platform. Namely, REST:

- uses HTTP methods explicitly,
- is stateless,
- exposes directory structure-like URIs, and
- can use JavaScript Object Notation (JSON) for exchanging data.

The web services that were developed for downloading and submitting results for both tasks of BIOASQ challenge follow the REST design and support the exchange of JSON strings. In the following sections these services are described in detail.

4.2 General web services for Task 1A

Web services have been developed for exchanging data with the platform for each of the tasks of the BIOASQ challenges. Participants are encouraged to use the web services during the challenge in order to automate the evaluation procedure[1].
For Task 1A of the challenge, participants can:

- download the test sets by sending an HTTP-GET request to:
  - `http://bioasq.lip6.fr/tests/`*`test_number`*`/`

  along with authentication parameters. The *test_number* should be replaced by the test set the participant wishes to download. The web service, internally, after checking that the user is registered, returns a JSON string containing the test set. The test sets remain available for downloading after the expiration date for submitting answers for the test set. The format of the JSON string that is returned is the following:

```
{[{"documentTitle":"Title","documentAbstract":
"Abstract", "PMID":"22511223"},
{"documentTitle":"Title","documentAbstract":"Abstract",
 "PMID":"22511224"},
                    .
                    .
{"documentTitle":"Title","documentAbstract":"Absrtact",
"PMID":"22511225"}
]}
```

- submit results by sending an HTTP-POST request to:
  - `http://bioasq.lip6.fr/tests/uploadResults/`*`test_number`*`/`

  Participants are able to submit their answers for the active test set as many times as they wish before a specified expiration date. Each time a participant submits new answers, the old ones are deleted. The following JSON string illustrates the format of the data in the JSON string in for the HTTP-POST request:

```
{"username":"your_username", "password":"your_password,
"system":"name_of_your_system",
"documents":[{"labels":[index1,..,indexN1],"PMID":"22511223"},
{"labels":[index1,..,indexN2],"PMID":"22511224"},
                    .
                    .
{"labels":[index1,..,indexN3],"PMID":"22511225"}
]}
```

[1] Scripts that perform the POST and GET requests to the web services and can be integrated to the user's code are available in the website http://bioasq.lip6.fr/general_information/Task1a/.

4.3 Internal web services for Task 1A

Creation of test sets

The test sets of Task 1A consist of non-annotated articles of MEDLINE. The creation of the test sets depends on the following web service between the partners of the challenge and MEDLINE:

- `getCitationsAfterDateWithoutMesh`: Each article in MEDLINE is uniquely identified by an integer, namely PMID. The platform makes an HTTP-POST request to the web service's URI with a date parameter and a list of PMIDs to be excluded from the test set. The service returns a JSON string that consists of the non-annotated articles from MEDLINE from the date parameter on, after excluding the PMIDs of the list. The list of the PMIDs contains the PMIDs that have been released in previous test sets. The date parameter is fixed, '2013/04/01'. Measuring the time interval from the upload of an article in MEDLINE until it becomes "In-process"[2] we concluded that several weeks are required. Since the prefered size the test sets is some thousands articles, freezing the date parameter in a past date (e.g. '2013/04/01') resulted in test sets of the required size. The criteria that are used for the articles of the test set are:
  - The articles that are returned from the service are "In-Process".
  - Each article has a PMID, a title and an abstract.
  - The selected articles come from pre-selected journals from a wide variety of biomedical fields. At the same time, the preselected journals have a short average annotation time [3].

  An example of the format of a JSON string that is returned follows:

```
{"result":{
"articlesPerPage": 5,
"documents": [
{
"documentTitle": "A novel method for detecting antigen-specific
human regulatory T cells",
"documentAbstract": "Antigenic epitopes recognized by FoxP3(+)
regulatory T cells (Treg) are poorly defined, largely due
to a lack of assays for determining Treg specificity...",
"PMID":"22265970"},
    .
    .
]
}}
```

Evaluation of the results

The test sets of Task 1A consist of "In-Process" non-annotated articles from MEDLINE. After some time, these articles become annotated with MeSH terms. The following web service is used to retrieve the MeSH terms and evaluate the systems:

[2] PubMed "In-Process" records provide basic citation information and abstracts while these records are reviewed for accuracy of the bibliographic data and assigned subject headings if the subject of the article is within the scope of MEDLINE.

[3] It is possible that some articles will not be annotated until the end of the challenge.

- `checkForMeSH`: The platform performs an HTTP-POST request to the web service's URI with a JSON that contains an array of PMIDs. Server-side, the service checks if the articles with the provided PMIDs have been annotated in MEDLINE. A JSON string with the PMIDs along with their MeSH annotations is returned, as in the following example:

```
{"result":{
"articlesPerPage": 1,
"documents": [
{"PMID":"22265970", "meshHeadings":["D081256",.., "D045211"]},
     .
     .
 "PMID":"22265978", "meshHeadings":["D081487",.., "D045278"]},
]
}}
```

If the returned results outnumber the 5% of the size of the test set, the evaluation function is triggered to update the evaluation measures of the participants' systems.

4.4 Web services for Task 1B

Participants can download the test batches of both phases of Task 1B by sending an HTTP-POST request to:

- `http://bioasq.lip6.fr/Tasks/1b/phaseA/api/`*number*`/` for Phase A and
- `http://bioasq.lip6.fr/Tasks/1b/phaseB/api/`*number*`/` for Phase B

replacing *number* by the number of the test set they wish to download. To obtain the test batch, they should post a JSON string with their username and password, i.e.,

```
{"username":"your_username",
 "password":"your_password"}
```

. Participants can submit their results by sending an HTTP-POST request to:

- `http://bioasq.lip6.fr/Tasks/1b/phaseA/submit/`*number*`/` for Phase A and
- `http://bioasq.lip6.fr/Tasks/1b/phaseB/submit/`*number*`/` for Phase B

replacing *number* by the number of the active test of the batch that the user wants to submit results for. In the HTTP-POST request, users have to post a JSON string with their username, their password, their system name, and their results, i.e.,

```
{"username":"your_username",
 "password":"your_password",
 "system":"your_system",
 "questions":[...]}
```

where the 'questions' field is described below. The JSON format and the names of the fields when exchanging data for Task 1B are as in the following JSON. In each phase only some of the fields of the example are present in the JSON strings that are exchanged.

```
{"questions":[
  {"id":"5118dd1305c10fae750000010",
   "documents": [
   "http://www.ncbi.nlm.nih.gov/pubmed/12723987",
   ...],
   "snippets":[
   {"document": "http://www.ncbi.nlm.nih.gov/pubmed/22853635",
    "text": "The expression and clinical course of RA are influenced
    by gender. In developed countries the prevalence of RA is 0,5 to
    1.0%, with a male:female ratio of 1:3.",
    "offsetInBeginSection": 559,
    "offsetInEndSection": 718,
    "beginSection": "sections.0"
    "endSection": "sections.0"},
    ...],
    "concepts":[
    "http://www.diseaseontology.org/api/metadata/DOID:7148",
    ...],
    "triples": [
   {"s": "http://linkedlifedata.com/resource/umls/id/C2827401",
    "p": "http://www.w3.org/2008/05/skos-xl#prefLabel",
    "o": "http://linkedlifedata.com/resource/umls/label/A17680439"},
    ...]},
   ...]}
```


For more information and code snippets about the web services of Task 1B consult Androutsopoulos et al. (2013).

Bibliography

I. Androutsopoulos, P. Malakasiotis, G. Tsatsaronis, M. Zschunke, and G. Balikas. Resources and Services for Task 1B. 2013.

G. Balikas, I. Partalas, A. Kosmopoulos, S. Petridis, P. Malakasiotis, I. Pavlopoulos, I. Androutsopoulos, N. Baskiotis, E. Gaussier, T. Artieres, and P. Gallinari. Evaluation Framework Specifications. Technical Report D4.1, BioASQ Deliverable, 2013.

Django. *The web framework for perfectionists with deadlines*. URL www.djangoproject.com.

P. Malakasiotis, I. Androutsopoulos, Y. Almirantis, D. Polychronopoulos, and I. Pavlopoulos. Tutorials and Guidelines. Technical Report D3.4, BioASQ Deliverable, 2013.

MEDLINE. *MEDLINE*. URL http://www.ncbi.nlm.nih.gov/pubmed.

MeSH. *MeSH: Medical Subject Headings*. URL http://www.ncbi.nlm.nih.gov/mesh.