SEVENTH FRAMEWORK PROGRAMME
FP7-ICT-2009-6

BlogForever
Grant agreement n°.: 269963

D4.4: Digital Repository Component Design

Editor:	**J. García Llopis, R. Jiménez Encinar**
Revision:	First Version
Dissemination Level:	Public
Author(s):	J. García Llopis, R. Jiménez Encinar, K. Stepanyan, Y. Kim, A. Haberfield, Şenan Postacı, G. Gkotsis, P. Lazaridou, A. Çınar, H. Kalb, V. Banos, S. Arango Docio, N. Kasioumis, T. Šimko, G. Banu Lateci, E. Pinsent
Due date of deliverable:	November 30, 2012
Actual submission date:	November 30, 2012
Start date of the project:	March 01, 2011
Duration:	30 months
Lead beneficiary name:	European Organization for Nuclear Research (CERN)

Abstract:

This report describes in detail the design, architecture and features of the BlogForever's weblog digital repository (web application). First of all, we introduce the Invenio software platform and its merits, explaining the reasons why we decided it should be the basis for the BlogForever digital repository. Then we present a review of the past and current work and highlight how our findings affected and supported some of the key decisions regarding the design of the repository. Furthermore, we analyze and illustrate how the the suggested architecture represents a robust weblog digital repository solution through its various stages: ingestion, management, preservation and dissemination. Finally, we list the various features.

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)

The **BlogForever** Consortium consists of:

Aristotle University of Thessaloniki (AUTH)	Greece
European Organization for Nuclear Research (CERN)	Switzerland
University of Glasgow (UG)	UK
The University of Warwick (UW)	UK
University of London (UL)	UK
Technische Universitat Berlin (TUB)	Germany
Cyberwatcher	Norway
SRDC Yazilim Arastrirmave Gelistrirmeve Danismanlik Ticaret Limited Sirketi (SRDC)	Turkey
Tero Ltd (Tero)	Greece
Mokono GMBH	Germany
Phaistos SA (Phaistos)	Greece
Altec Software Development S.A. (Altec)	Greece

Revision History

Version	Description of Change	Author	Date
0.6	First partial draft (drafting chapters 2, 3, 4, 5, incorporated contributions of people listed on the cover page)	CERN MOKONO UW UG SRDC UL TUB	21/11/2012
0.8	Second partial draft (added more content to chapters 2, 3, 4)	CERN	26/11/2012
0.9	Third partial draft (added more content to chapters 2, 4 and 5, review, corrections and feedback)	CERN UG UL SRDC AUTH	27/11/2012
0.99	Final draft	CERN	28/11/2012
1.0	Final version of the deliverable	CERN	30/11/2012

Table of Contents

List of Figures

List of Tables

Executive Summary

The BlogForever platform consists of the two main software components: the spider component and the repository component. They work independently even if they share interfaces to communicate. This report identifies and describes design decisions for the repository component.

A total of 89 features have been identified and described in this report. The features are based on user requirements identified before in WP4 (*The BlogForever Software Infrastructure*, *D4.1: User Requirements and Platform Specifications Report*[8]) and they follow the metadata structure and preservation recommendations previously given by WP2 (*Weblog Structure and Semantics*, *D2.2: Weblog Data Model*[19]) and WP3 (*The BlogForever Policies*, *D3.1: Preservation Strategy Report*[11]). The features specify functionalities that enable an efficient storage for blog preservation, e.g. how to ingest, check, keep and disseminate blog content, and also functionalities that allow final users to access the information in an structured and intuitive way through a web interface, allowing them to interact and construct a community of users that will further enrich the data. To facilitate implementation and the conducting of Case Studies in WP5 (*Case Studies and Validation*) of the BlogForever project, the same template has been used for the feature specifications of the repository and the spider.

Since the repository design is based in Invenio[1], this report introduces the main functionalities of this software focusing on those that will be more relevant in the BlogForever repository. Then, the most important decisions taken are discussed, including the metadata management, the communication with the spider component, the scalability and the user interface. The outcome of *Task 4.4: Design of the digital repository component* -reported in this document- is the list of feature specifications that is the input for the development of the repository in *Task 4.5: Implementation of the digital repository component.*

[1] https://invenio-software.org/

Chapter 1

Introduction

The BlogForever project aims to develop solutions for aggregating, preserving, managing and disseminating blogs. To achieve these goals, the BlogForever project aims to develop a software platform that enables real-time harvesting of blog entities and preservation of these blogs to facilitate extensive search and exploration functionalities of the archived blogs. The software architecture behind the intended repository system consists of two main components - the weblog spider and the digital repository. The spider is responsible for crawling all the necessary blog data and characteristics designated for preservation while the repository is responsible for long term archiving, preservation and management of the blogs, as well as providing facilities for further analysis and reuse of the content.

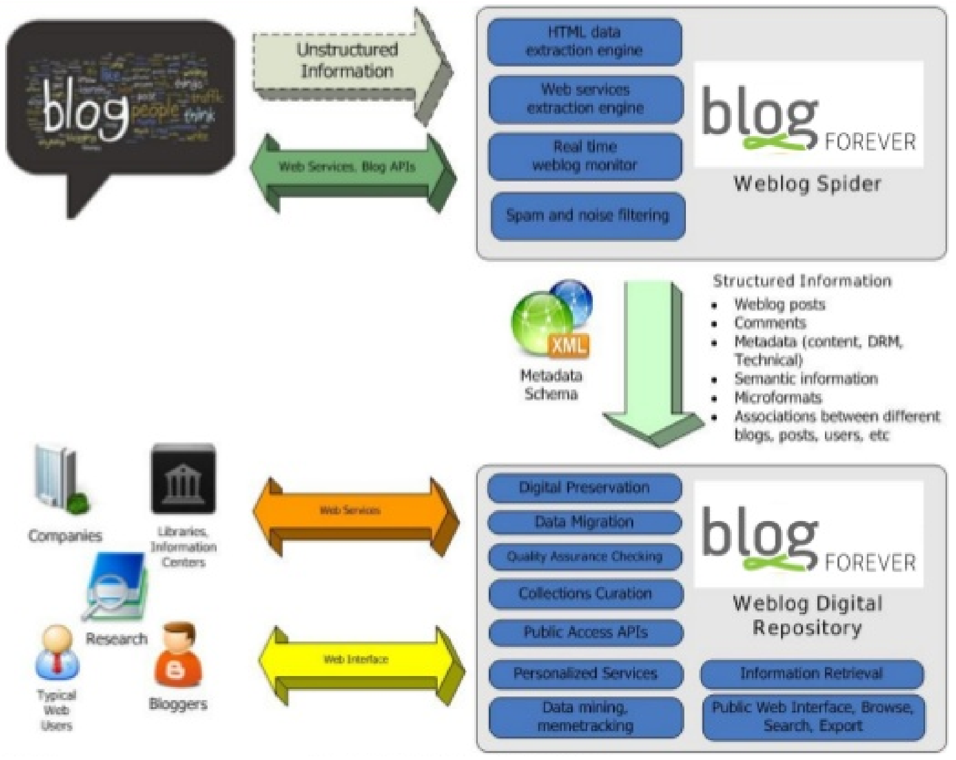


Figure 1.1: BlogForever Platform overview

In the following text the words “weblog” and “blog” will be interchangeably used to describe the same concept, while the weblog spider component and the digital repository component will be referred to as “spider” and “repository” respectively.

1.1 Purpose and scope

In this document the outcome of Task 4.4 of the BlogForever project will be reported. In the Task 4.4 the created deliverables have been studied in order to elaborate on the scope of the modifications required. Reports created in WP2 as well as progress on the WP3 have been of great value in order to determine the expected repository functionality and facilitate the design process. In particular, the weblogs semantics reports (D2.2[19] and *D2.3: WebBlog Ontologies*[9]) have been of high importance in order to identify the data types that will be stored in the digital repository and their special properties and associations, which have great added value. Also the preservation policies deliverable D3.1[11] contributed notably helping the designers to create a robust system capable to preserve blog data. The outcome of the current Task 4.4 will be digital repository component design and will report the design of the digital repository system based on Invenio providing the development stage a list of features to be implemented.

The design -and implementation- of the repository will not be done from scratch, but taking profit of the Invenio software suite developed at CERN [1] (European Organization for Nuclear Research). The design will be expressed in terms of what modifications and extensions Invenio needs in order to meet the requirements previously obtained from D4.1[8].

1.2 Overall approach

Initially, the Invenio software will be studied in order to understand how its structure and functionalities can best map the project requirements in Chapter 2. In Chapter 3 the existing deliverables will be analyzed in order to extract the most relevant part of their findings in the design of the BlogForever platform. This design will be discussed in Chapter 4 introducing the main decisions adopted in order to transform Invenio from a digital library hosting papers, articles and preprints into a digital repository hosting weblog content. Finally, Chapter 5 will present the list of feature specifications.

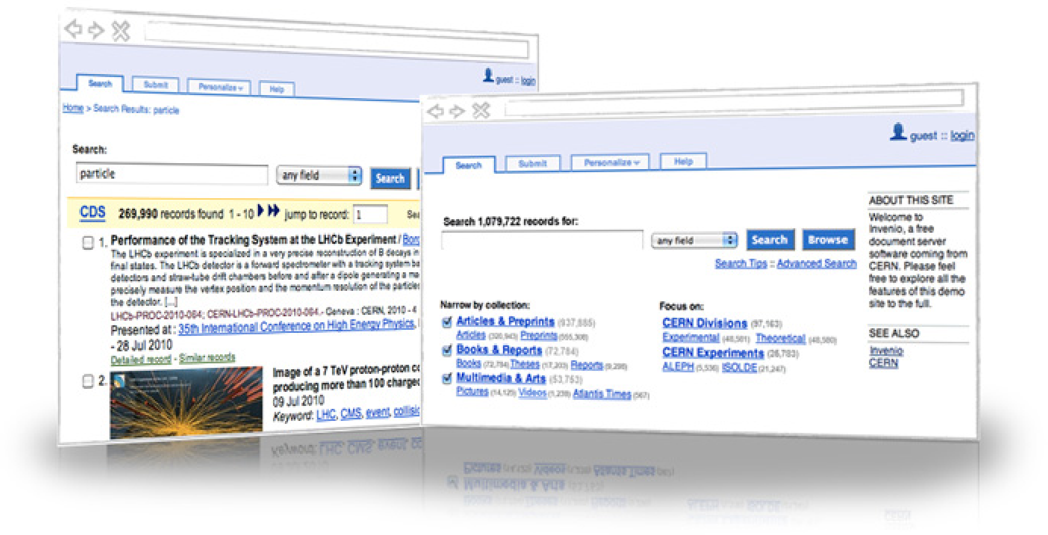
[1]http://www.cern.ch/

Chapter 2

The Invenio Software Platform

This chapter aims to present an overview of the Invenio software platform. In the first parts, bibliographic metadata support and collections organization are outlined. Furthermore, platform architecture is described in more detail and the key Invenio modules are presented. In the last part, Invenio scalability is also addressed from multiple points of view. The Invenio software platform is one of the cornerstones of the BlogForever platform, providing the basis of the blog repository component.

Invenio is an open source software package conceived and developed at CERN, which covers all aspects of digital library management from document ingestion through classification, indexing, and curation to dissemination[3].


Search
Submit
Personalize
Help
guest :: login
Home > Search Results: particle
Search:
particle
any field
Search
CDS
269,990 records found 1 - 10
jump to record: 1
Search 1,079,722 records for:
any field
Search
Browse
Search Tips :: Advanced Search
Narrow by collection:
Articles & Preprints (937,885)
Articles (320,943) Preprints (555,308)
Books & Reports (72,784)
Books (72,784) Theses (17,203) Reports (9,298)
Multimedia & Arts (53,763)
Pictures (14,125) Videos (1,239) Atlantis Times (567)
Focus on:
CERN Divisions (97,163)
Experimental (48,581) Theoretical (48,580)
CERN Experiments (28,783)
ALEPH (5,536) ISOLDE (21,247)
ABOUT THIS SITE
SEE ALSO
Invenio
CERN

1. Performance of the Tracking System at the LHCb Experiment / Borg
The LHCb experiment is specialized in a very precise reconstruction of B decays in
final states. The LHCb detector is a forward spectrometer with a tracking system ba
detectors and straw-tube drift chambers before and after a dipole generating a ma
precisely measure the vertex position and the momentum resolution of the particles
the detector. [...]
LHCb-PROC-2010-064; CERN-LHCb-PROC-2010-064.- Geneva : CERN, 2010 - 4
Presented at : 35th International Conference on High Energy Physics,
- 28 Jul 2010
Detailed record - Similar records

2. Image of a 7 TeV proton-proton c
producing more than 100 charged
09 Jul 2010
Keyword: LHC, CMS, event, collisi

Welcome to Invenio, a free document server software coming from CERN. Please feel free to explore all the features of this demo site to the full.

Figure 2.1: Invenio main search page

Since 1993 CERN has developed this free software (licensed under the GNU General Public Licence (GPL)[1]) primarily for internal needs as an institutional repository, managing over 1,000,000 bibliographic records in high-energy physics since 2002,

[1] http://www.gnu.org/licenses/gpl.html

organized in more than 500 collections, covering articles, books, journals, photos, videos, and more.

Nowadays it represents a suite of applications used by about thirty scientific institutions and universities outside CERN (such as Labordoc in Switzerland, HBZ Digitalisierte Drucke Portal in Germany and National Repository of Grey Literature in Czech Republic) and is being co-developed by an international collaboration including institutes such as DESY[2] (Deutsches Elektronen-Synchrotron), EPFL[3] (École Polytechnique Fédérale de Lausanne), FNAL[4] (Fermi National Accelerator Laboratory) and SLAC[5] (SLAC National Accelerator Laboratory).

From a technical point of view, Invenio runs under GNU/Linux[6] systems, a MySQL[7] database server and an Apache/Python[8] web application server. Its source code is mainly written in Python[9], with some *ad hoc* modules developed in C[10] and Common Lisp[11]. The development strategy used to implement Invenio ensures that it is flexible in every layer. Being based on open standards such as MARC 21[13] and Open Archives Initiative[12] metadata harvesting protocol (OAI-PMH), its interoperability with other digital libraries is guaranteed.

2.1 Metadata

All the bibliographic data in the Invenio system are internally represented in the MARC 21 format. Each record has their associated metadata stored in the database, while the fulltext files and other attached files are stored in the Invenio filesystem. A reference to these files is included in the MARC metadata of the record.

The main reasons why Invenio uses MARC as standard digital format [1]:

- MARC is the standard format in the library world. It is well established and has been used since 1960s.
- MARC is flexible enough to represent any metadata structure. Therefore, Invenio can adapt to your needs without altering its internal data structure.
- MARC technology can be well combined with recent technologies like XML. In fact, whenever bibliographic metadata are to be worked with externally in a file format, Invenio uses recently standardized MARC XML format provided by the Library of Congress[13].

[2] http://www.desy.de/
[3] http://epfl.ch/
[4] http://fnal.gov/
[5] http://www.slac.stanford.edu/
[6] http://www.gnu.org/gnu/linux-and-gnu.en.html
[7] http://www.mysql.com/
[8] http://www.modpython.org/
[9] http://www.python.org/
[10] http://www.open-std.org/jtc1/sc22/wg14/
[11] http://www.common-lisp.net/
[12] http://www.openarchives.org/
[13] http://www.loc.gov/index

The most commonly used metadata fields in Invenio are shown in Table 2.1 (following the MARC typographical conventions[14])

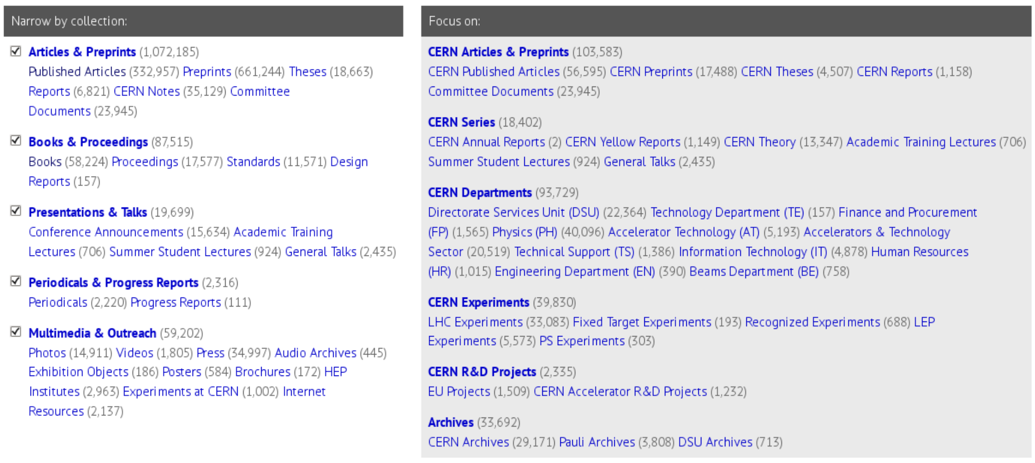
Metadata concept	**Proposed MARC 21 representation**
Abstract	520 $a
Author, first	100 $a
Author(s), additional	700 $a
Collection identifier	980 $a
Email	8560 $f
Imprint	260 $a,b,c; 300 $a
Keywords	6531 $a
Language	041 $a
OAI identifier	909CO $o
Publication info	909C4 $* [many subfields]
References	999C5 $* [many subfields]
Primary report number	037 $a [unique throughout the system]
Additional report number(s)	088 $a
Series	490 $a,v
Subject	65017 $a
Title	245 $a
URL (e.g. to fulltext)	8564 $u, $z

Table 2.1: Main metadata in Invenio

2.2 Record organization

Records in Invenio are organized into collections. The collections are organized in a tree, being this collection tree what the end-users see when they start navigating through the digital library. The collection tree is similar to what other sites call Web Directories that organize Web into topical categories, such as Yahoo! Directory and Open Directory Project (ODP).

[14]http://www.loc.gov/marc/bibliographic/concise/bdintro.html



Narrow by collection:

✓ **Articles & Preprints** (1,072,185)
Published Articles (332,957) Preprints (661,244) Theses (18,663) Reports (6,821) CERN Notes (35,129) Committee Documents (23,945)

✓ **Books & Proceedings** (87,515)
Books (58,224) Proceedings (17,577) Standards (11,571) Design Reports (157)

✓ **Presentations & Talks** (19,699)
Conference Announcements (15,634) Academic Training Lectures (706) Summer Student Lectures (924) General Talks (2,435)

✓ **Periodicals & Progress Reports** (2,316)
Periodicals (2,220) Progress Reports (111)

✓ **Multimedia & Outreach** (59,202)
Photos (14,911) Videos (1,805) Press (34,997) Audio Archives (445) Exhibition Objects (186) Posters (584) Brochures (172) HEP Institutes (2,963) Experiments at CERN (1,002) Internet Resources (2,137)

Focus on:

CERN Articles & Preprints (103,583)
CERN Published Articles (56,595) CERN Preprints (17,488) CERN Theses (4,507) CERN Reports (1,158) Committee Documents (23,945)

CERN Series (18,402)
CERN Annual Reports (2) CERN Yellow Reports (1,149) CERN Theory (13,347) Academic Training Lectures (706) Summer Student Lectures (924) General Talks (2,435)

CERN Departments (93,729)
Directorate Services Unit (DSU) (22,364) Technology Department (TE) (157) Finance and Procurement (FP) (1,565) Physics (PH) (40,096) Accelerator Technology (AT) (5,193) Accelerators & Technology Sector (20,519) Technical Support (TS) (1,386) Information Technology (IT) (4,878) Human Resources (HR) (1,015) Engineering Department (EN) (390) Beams Department (BE) (758)

CERN Experiments (39,830)
LHC Experiments (33,083) Fixed Target Experiments (193) Recognized Experiments (688) LEP Experiments (5,573) PS Experiments (303)

CERN R&D Projects (2,335)
EU Projects (1,509) CERN Accelerator R&D Projects (1,232)

Archives (33,692)
CERN Archives (29,171) Pauli Archives (3,808) DSU Archives (713)

Figure 2.2: CERN Document Server collection tree

2.3 Modules

The architecture of Invenio consists of several modules with precisely defined functionality that works collaboratively but independently inside the system framework. This supports and facilitates extensions to and development of the system. In the following sub-sections, a brief description of each of these modules will be presented by category [2].

A general overview of how some selected Invenio modules interact and what their relationships are can be found in the diagram of the Figure 2.3.

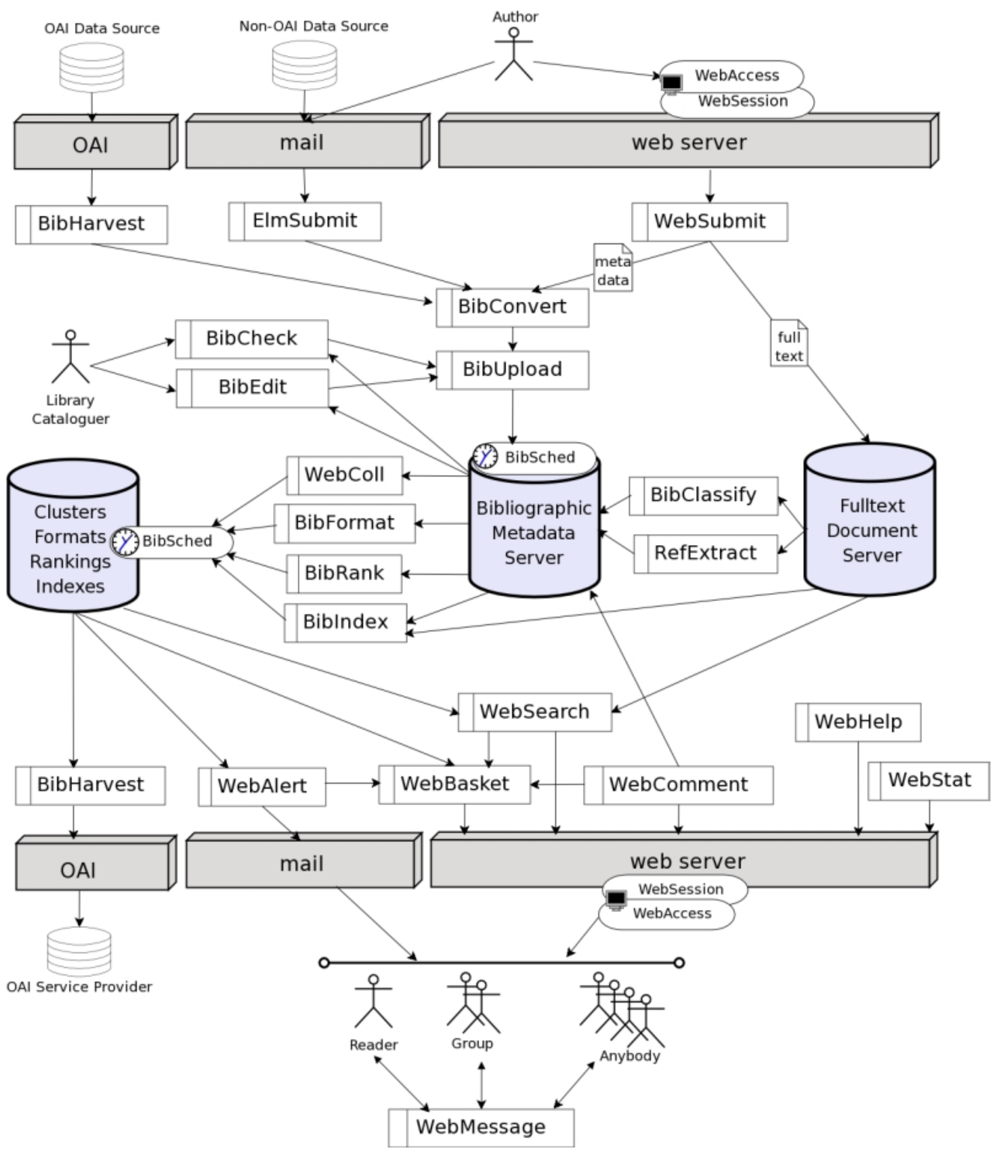


Figure 2.3: Selected Invenio modules overview diagram

The modules in Invenio are named using the following convention: the prefix "Bib" is used for modules related with bibliographic data and the prefix "Web" is related with modules that work with the web interface.

2.3.1 Metadata acquisition

The metadata input into an Invenio system can be done in two different ways:

1. Admin-oriented batch mode: OAI Harvest to get data from OAI repositories, BibConvert to convert any input data into MARC XML, and BibUpload to upload MARC XML files into Invenio

2. Author-oriented interactive mode: WebSubmit or ElmSubmit to submit documents via Web or via e-mail, respectively. Once the data are uploaded in Invenio, the metadata can be modified via BibEdit.

A brief description of each of the modules mentioned above is presented:

- **OAI Harvest** represents the OAI-PMH compatible harvester. It allows the repository to gather metadata from other OAI-compliant repositories and is also in charge of OAI-PMH repository management.
- **BibConvert** allows metadata conversion from any structured or semi-structured proprietary format into any other format. In both WebSubmit and ElmSubmit cases, metadata is gathered in raw form and BibConvert is used to convert it typically to MARC XML, which is used natively in Invenio. Finally, the record is inserted into the system using BibUpload. BibConvert also allows conversions between various sequential and semi-structured formats, such as MODS(Metadata Object Schema) and Dublin Core.
- **WebSubmit** is a submission system, that permits authorized individuals (authors, secretaries and repository maintenance staff) to submit individual documents for ingest into the system.
- **ElmSubmit** is an email submission gateway that allows automatic document uploads from trusted sources via email.
- **BibUpload** allows the uploading of new bibliographic data into the database. If the record has files attached, such as the fulltext file, a reference to them is inserted in the MARC metadata and the file is also uploaded into the system, using BibDocFile to hold them.
- **BibCheck** allows administrators and catalogers to define a variety of automated tests on the metadata to see whether the metadata comply with quality standards. This offers also the possibility of fixing errors.
- **BibMatch** matches bibliographic data in a MARC XML file against the database content. With a MARC XML input file, the produced output shows a selection of records in the input that matches the database content. This way, it is possible to identify potential duplicate entries before they are uploaded in a database.
- **BibEdit** enables manipulate bibliographic data, edit a single record, do global replacements, and other cataloguing tasks via a web interface. The modified metadata is re-submitted to the system using BibUpload.

2.3.2 Indexing and ranking

The metadata output from an Invenio system to the end-user is covered by several modules: BibIndex to index the metadata, BibRank to eventually rank them, BibFormat to format them for the output, WebSearch to provide search interfaces and search engine.

A brief description of each of the modules mentioned above is presented:

- **BibIndex** is in charge of indexing metadata, references and fulltext files.
- **BibRank** enables users to set up a variety of ranking criteria that will be used later by the search engine.
- **WebSearch** handles user requests in searching for a certain word or phrases in the database.
- **BibFormat** is responsible for the formatting of the bibliographic metadata, having several types of outputs. It is fully customizable and with great extendability.

The indexing and ranking modules are at the core of the Invenio system. Invenio offers three types of search: filtered search on selected metadata elements, fulltext keyword search, and reference search. Moreover, Invenio provides the option for users to customize their search further as a combination of all three types of search. For example, in the context of weblogs, it would be possible to search for blog posts containing selected keywords that include a link to entries written by a selected author. Likewise, the results retrieved by the search engine can be ranked according to several criteria. The default Invenio installation includes the classical word-frequency based vector model that permits one to retrieve similar records. Furthermore, a ranking method machinery based on specific metadata values is included. Finally, the new experimental ranking features in Invenio include the capability to rank records by the number of citations and the number of downloads.

2.3.3 Personalization

Invenio interface can be personalized to suit different needs of different end-users. This functionality is covered by several modules: WebStyle to define the general look of Invenio pages, WebSession to identify users and their personal configurations, WebBasket to provide personal baskets or document carts, and WebAlert to set up personal email notification alerts.

A brief description of each of the modules mentioned above is presented:

- **WebStyle** is a library of design-related modules that defines look and feel of Invenio pages.
- **WebSession** is a session and user management module that supports differentiation of users.
- **WebBasket** enables the end users of the system to store the documents they are interested in in a personal basket or a personal shelf. The concept is similar to popular shopping carts. A user may own several baskets. A basket can be either private or public, allowing a simple document sharing mechanism within a group.
- **WebAlert** allows the end user to be alerted whenever a new document matching her personal criteria is inserted into the database. The criteria correspond to a typical user query as if it would be done via the search interface.

- **WebComment** provides a community-oriented tool to rank documents by the readers or to share comments on the documents by the readers.
- **WebMessage** permits the communication between (possibly anonymous) end users via web message boards, to invite readers to join the groups, etc.

Another important aspect worth mentioning is that since Invenio is being developed in an international environment, it supports internationalization and comes translated already in 28 different languages (Afrikaans, Arabic, Bulgarian, Catalan, Czech, German, Georgian, Greek, English, Spanish, French, Croatian, Hungarian, Galician, Italian, Japanese, Kinyarwanda, Lithuanian, Norwegian, Polish, Portuguese, Romanian, Russian, Slovak, Swedish, Ukrainian, Chinese (China), Chinese (Taiwan))), enabling end-users to dynamically select the language of their choice.

2.3.4 Other relevant modules

Other important modules in Invenio that complement the modules presented above are:

- **BibSched** is the central unit of Invenio. It manages and controls module access to the bibliographic database, preventing sharing violation threats and assuring the coherent execution of the database update tasks. This module manages a task queue with different priorities and is able to manage the parallel execution of tasks so that there is no conflict among them.
- **WebAccess** is the module in charge of granting access to users, depending on their roles (content manager, system administrator, registered user, etc.), managing user permissions associated with various actions within the system.
- **MiscUtil** is a collection of miscellaneous facilities that can be used by developers on the other modules to make their work easier.

2.4 Scalability in Invenio

Scalability is a crucial issue as our aim is to cope with the continuously expanding social media data streams. Having long time preservation as a project goal, being able to store large amounts of data and make it accessible in a reasonable time is absolutely necessary. In this section, various aspects of Invenio scalability as well as some concrete examples of current practices will be discussed. In order to illustrate the scalability potential of the repository, the Invenio instance used at CERN will be taken as an example. This instance is called CDS[15] (CERN Document Server) and has been running since 2002 with great success.

[15]http://cdsweb.cern.ch/

2.4.1 Increasing number of visitors

The high scalability is reached by multiplying the number of workers and database slaves that serve the users. For a schematic diagram, see Figure 2.4:

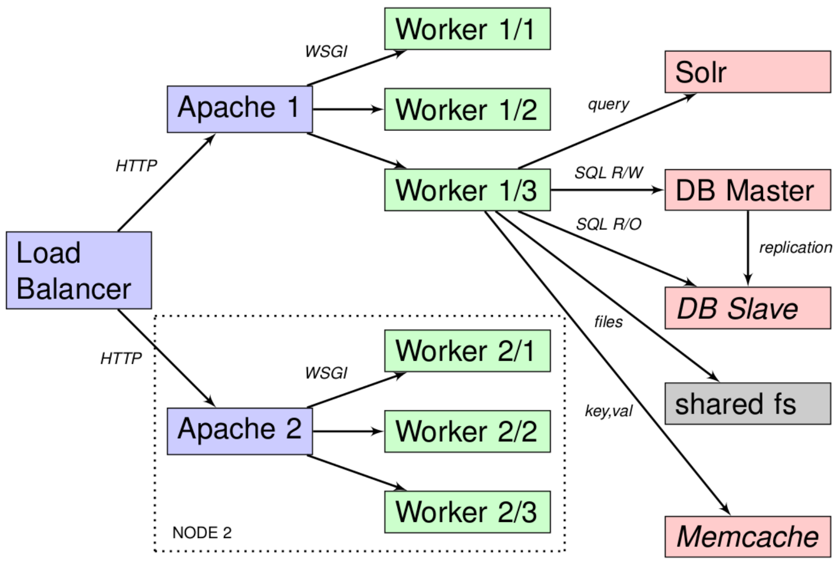


Figure 2.4: Invenio scalability

This technique is used in CDS with the following setup:

a) Incoming traffic goes into a box running HAProxy[16] (High Availability Proxy). The box dispatches incoming requests to backend nodes, using typically "round robin" method for static files and "by-business" method for application web pages. We also have several dedicated jail rules for certain URLs and certain user agents so that e.g. /rss hits or bots would not perturb the regular site operation for regular users.

b) The traffic now reaches boxes running typical Invenio instances. Here, there is a front-end Apache that serves as another proxy for back-end Invenio WSGI[17] (Web Server Gateway Interface) processes. We run 8-30 WSGI processes per box; the number depends on the available memory and on the nature of the data.

c) All the WSGI processes connect to the same database master box on the back end side. It is important to set up MySQL temporary space on a dedicated tmpfs partition due to many queries acting upon TEXT columns.

d) The database master is being replicated live onto a database slave. Certain read-only SQL queries are being dispatched by WSGI processes to run on database slave rather than on database master. This option is still underused, it could be made more effective.

[16]http://haproxy.1wt.eu/
[17]http://wsgi.readthedocs.org/en/latest/

e) In addition, we have an independent Solr[18] instance for full-text indexing, as well as Redis[19] instance for various caches.

This configuration was used by the CERN Document Server on the July 4th 2012 Seminar on Higgs Boson[20] search, hosting images and materials. The setup consisted of 1 HAProxy node, 3 worker nodes serving Invenio application, 2 virtual nodes serving static files, 1 database master and 1 database slave. (The machines are not particularly powerful). There were 7 million user hits that day, with sustained peaks of about 200 requests served per second and 1000 requests processed by the load balancer per second. The load on machines was moderate, there was no need for emergency actions. The Invenio instances were set up as usual.

Note that this multi-node-app-servers, single-node-db-master-server setup can be further tuned by having several databases for several worker node groups. In the CDS instance, it was not really necessary to do that so far.

In the BlogForever use case, the current capabilities of Invenio should be enough load wise even without further measures.

2.4.2 Increasing number of objects

After discussing reader-load-wise scalability, another question is the increasing number of objects being handled by a single database instance. To take once more the CERN Document Server example, the total number of records is 1.5 million. Such a number is conveniently handled by a single-database system. The total database size including indexes, etc. is about 40 GB, not counting object sizes (PDFs, JPEGs) obviously (that would be over TB but it does not matter much for scalability here). Solr fulltext indexes are about 16 GB.

Depending on the nature of the data and on the hardware specifications, it should be possible to hold comfortably databases of 5-10 million on a single box without any big architectural changes. The biggest example of an Invenio instance with regard to data size is ADS[21] (Astrophysics Data System) test server that has about 8 million records. The box requires quite some memory, mostly because of a huge citation map; but this is not the expected scenario in case of BlogForever.

2.4.3 Ingestion speed

Another aspect to consider is ingestion speed. The usual assumption behind Invenio architecture was that records are uploaded, then modified a few times, and then they would not change that much. This would bring us to a situation where there is low number of INSERT and UPDATE statements coming from producers when compared to the high number of SELECT statements coming from consumers. If

[18]http://lucene.apache.org/solr/
[19]http://redis.io/
[20]http://www.atlas.ch/news/2012/latest-results-from-higgs-search.html
[21]http://adswww.harvard.edu/

we take the CERN Document Server as an example, we observe typically up to 20 thousand records updated per day. The system can do comfortably few times more than that, but it is rarely needed.

There is an important bottleneck we would like to mention: there can be only one single BibUpload running at any given point in time. This is due to the need to scan and eliminate duplicates, etc. This limits the general upload speed. In the CDS internal task manager there are several tickets close to being finished that will improve the single-database ingestion speed situation, e.g. (i) by separating uploading incoming information from MARC processing; or (ii) by optimizing append/correct speed by means of record versions. This shows how the ingestion performance is being improved in even single database setup - optimize before scaling out.

2.4.4 Multi-database setup

If a single-database system is not enough to hold all the data, then the problem can be notably taken via a system of collaborating external Invenio instances. Namely, we partition data by collections, say hosting "Preprints" on one instance, hosting "Photos" on another instance, kind of manual partitioning, and one Invenio instance can query other instances during run time. Most of the processing is running on given instances in this setup. There is no need for overall inter-node processing except for searching, sorting, etc.

Chapter 3

Weblogs Structure and Preservation Principles

The primary aims of this chapter are: a) to outline weblog structure and to align it with the concept of a record in Invenio (Section 3.1), and b) to summarize the preservation policies associated with these records (Section 3.2). This chapter also describes the results of the investigation conducted as part of WP2 and WP3. Firstly, it focuses on the inquiry into the semantics of blogs as part of the *Task 2.2: Weblog semantics* within the WP2. The detailed account of the work carried out on this task is available in the BlogForever report D2.2[19]. The brief summary here is intended to provide context and justification to the proposed design of the repository. Secondly, it discusses the preservation policies for aggregating blog content.

3.1 Weblog structure and semantics

The repository software developed as part of BlogForever is expected to capture the structure of weblogs. The data model developed as a result of studying blogs provides a useful foundation for designing a system that aggregates weblogs and keeps their structure intact.

The proposed blog data model was informed by inquiries that aimed to explore the components, the structure and the semantics of weblogs. It took into consideration user views from the earlier conducted online survey, and recommendations from the theoretical inquiry into network analysis, supplemented by the inquiries such as, the existing conceptual models of blogs, the data models of Open Source blogging systems, and data types identified from an empirical study of web feeds. The remainder of this section explains the chosen direction for developing the data model and provides context to explain the outcomes.

Most frequently, data modeling is conducted by defining the requirements. The rationale behind drawing a set of requirements is to ensure that the data model addresses these requirements for the solutions that are being developed [15]. Some of the primary requirements of the project have already been defined and agreed as part of the project agreement[1]. However, the study of the structure and semantics of the blogs extended this list and provided the necessary foundations for developing the data model. This study enabled to extend the list of requirements and identify weblog properties that may be necessary to preserve.

The generic requirements of the task of data modeling was to explore the structure of blogs to be able to accommodate a range of weblogs and their properties. Hence, the proposed blog data model was developed in a number of consecutive phases. Each of the phases contributed to the process of informing the development of the proposed model. At a later stage and in addition to the study, a set of user requirements was studied, which leads to some minor changes in the data model. This document includes the changes introduced after the proposal of the model.

For the purposes of the BlogForever project, conceptual and more detailed logical information levels have been chosen for representing the proposed data model [19]. The decision was based on the necessity to provide both a high level view as well as the more detailed one. The following section provides an outline of the data model and the included properties.

3.1.1 Outline of the data model

It is evident that blogs are multi-faceted entities that may require a range of different data structures to be put in place. However, it is also apparent that most of the blogs share common features and a general outline. Therefore, the development of a generic and simple data model to suffice the preservation of the basic components of the blogs was possible. This basic model - referred here as the core model - can then be extended to ensure the integrity of captured weblogs and to meet the requirements of a successful preservation action.

The components of the core model were identified by looking into user views on blogs, existing models, the structure of their web feeds and types of data distributed by them. By looking into both technical specification as well as a summary of user perceptions, it was possible to identify most prominent conceptual components of weblogs referred to as entities in the proposed model.

These components were further studied in order to identify and to describe their properties. The properties of these components constitute the data that they carry and the metadata that are used to describe them. They have been collected and collated before integrating them into the data model. Once the data type and association with the entities were identified, the properties have been integrated into the data model and associated with one of the suggested entities of the model.

The detailed report about the inquiries used for developing the data model is available in the D2.2 BlogForever report [19]. As a result of developing various BlogForever prototypes [9] and proposing preservation strategies [11], some minor changes have been introduced. The following section outlines the model and highlights the recent changes.

[1]Grant Agreement Annex I - Description of Work (DoW)

3.1.2 Blog core

The inquiry into the structure of blogs suggested that there is an established vocabulary associated with weblogs. While the vocabulary at times seems to contain more than one term for referring to the same concept, the use of many terms has been widely accepted. This observation is confirmed at various stages of the conducted inquiries to identify the semantic components of blogs. For example, the review of the existing models of blogs confirms to the established vocabulary and the use of certain terms. Similar outcomes are revealed after an inquiry into the existing database structures of Open Source blogs. It seems that the concepts such as Post, Comment, Page, and Author, appear frequently to describe various sections of websites referred to as blogs. These conceptual entities have been put together to form the core of the blog as described in the data model.

This data model describes the weblog data and metadata grouped into entities. The graphical representation of the data model is shown in the Figure 3.1[19]. The primary identified entities of a weblog and the interrelation between them is shown and described by the connected lines. The small triangles indicate the directions of the relationships.

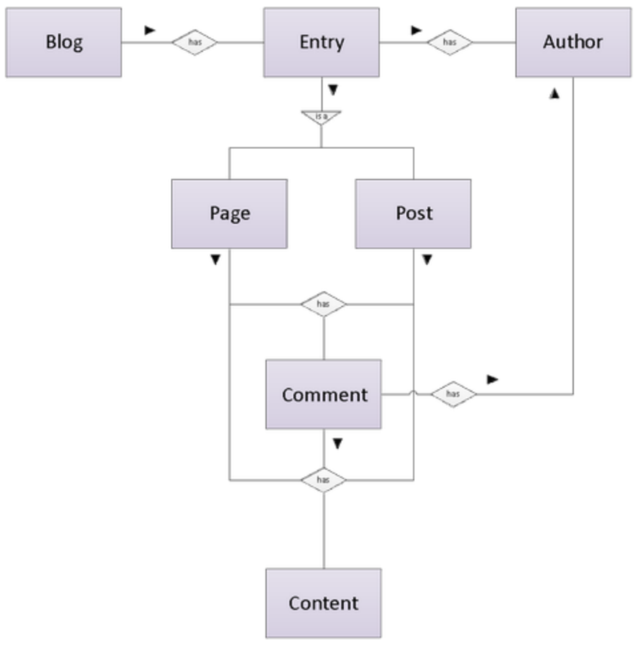


Figure 3.1: Blog core

The model described above demonstrates a high level view of the blog core. However, sets of inquiries mentioned above, allowed identifying the properties that can be associated with each of the entities. These properties were collected and integrated into a more detailed view, while the vocabulary to describe the properties was further collated. The selected naming was discussed and adjusted when further clarity was needed.

After the completion of the WP4 task, that aimed to identify user requirements for the BlogForever system (the detailed account of this work is available in the BlogForever report D4.1[8]), the data model was revisited. These requirements were identified as a result of interviews conducted with a range of stakeholders. The considerations of the requirements led to updating the model with additional blog properties that would be necessary for providing the services according to the identified requirements. Feedback that included the tacit knowledge of partners was also taken into consideration for refining the model.

3.1.3 Records within the repository

This section advances from the previous discussion by presenting and explaining the identified properties of the entities presented as part of the blog core model described above. However, it also introduces the notion of the record and discusses the data model along with the selected records.

While the data model represents the structure of the data contained in blogs, it is also necessary to identify if blog data can be injected into the repository and, subsequently, presented to the repository users as records. Records are information units collected and stored in a repository. Repositories usually contain specific types of records, for instance book, journal or article records. However, apart from representing physical objects such as printed books, the records can also represent digital material. The collection of records can be then indexed and searched by users. By looking into the core data model we can see that there are a number of prominent entities associated with a blog. It is likely that users of the repository will be interested in searching through certain units of information. Taking into account the above, the following four types of records have been identified: Blog, Post, Comment and Page.

Each of the record types can be used for implementing a faceted search functionality, as well as general search by keywords. The keywords entered by the users for searching through the repository can then be compared with the metadata/data stored in association with the records. While keyword search can be based on some complex concepts such as author, the result of the search will be presented as a list of records of the chosen type.

The following sections describe the attributes of the records as presented within the blog data model.

3.1.3.1 Blog as a record

A Blog record contains the primary description of the object. With respect to the data model, it can be described by the attributes presented in Table 3.1 [19].

Record	Attribute	Description
Blog	title	Title of the entry
	html_title	Contains the title of the HTML head element
	alt_title	Alternate title may include subtitles of the blog or other titles
	alt_title_type	Alternative title type specified the type of the alt_title
	URI	URI of the blog
	aliases	*Alternative URLs/aliases associated with the blog*
	status_code	Status code (may reflect whether the blog ceased to exist)
	language	Retrieved language field, as defined by the blog
	encoding	Retrieved encoding (character set) field, as defined by the blog
	sitemap_uri	URI of the blog sitemap if exists
	platform	Platform of the blog powering service, retrieved where available
	platform_version	Versioning information about the platform
	webmaster	Information about the webmaster where available
	hosting_ip	IP address of the blog
	location_city	Location city based on the hosting details
	location_country	Location country based on the hosting details
	last_activity_date	Date as retrieved from the blog, including time zone
	post_frequency	As retrieved from the blog
	update_frequency	As retrieved from the blog
	copyright	Notes of copyright as retrieved from the blog
	ownership_rights	Notes of ownership rights as retrieved from the blog
	distribution_rights	Notes of distribution rights as retrieved from the blog
	access_rights	Notes of access rights as retrieved from the blog
	license	License of the content

Table 3.1: Blog record attributes

3.1.3.2 Post and Page as a record

Post and Page records share most of their properties, presenting as well a very similar structure. For instance, both Post and Page can have a name, a unique URL, creation date, etc. Hence, the shared attributes have been combined here as Entry. They are presented in Table 3.2 [19].

Record	Attribute	Description
Entry	title	Title of the entry
	subtitle	Subtitle of the entry if available
	URI	Entry URI
	aliases	*Alternative URLs/aliases associated with the entry*
	alt_identifier (UR)	A common alternative identifier similar to DOI
	date_created	Retrieved from the blog or obtained from the date/time crawling, including time zone
	date_modified	Retrieved from the blog or obtained from the date/time crawling, including time zone
	version	Auto-increment: derived version number (versioning support)
	status_code	Information about the state of the post: active, deleted, updated (versioning support)
	response_code	HTTP response code
	geo_longitude	Geographic positioning information
	geo_latitude	Geographic positioning information
	access_restriction	Information about accessibility of the post
	has_reply	Derived property (as in SIOC[2])
	last_reply_date	Derived property (as in SIOC), including time zone
	num_of_replies	Derived property (as in SIOC)
	child_of	ID of entry parent if available

Table 3.2: Post and page records shared attributes

While Page and Post are similar in their properties, they are conceptually different. Posts are Entries published by the blog Author, appear in a chronological order or in categories, and are distributed by web feeds. On the other hand, Pages are Entries which content is not distributed via web feeds and they are not displayed in a chronological order either. However, Pages usually contain relevant information that may describe the Author, and/or provide basic information about the Blog. Hence capturing Pages in addition to Posts is considered important. A different template is used for Pages and this is the only property exclusively associated with them.

The attributes of the Entry are then extended to include the attributes relevant for the Post and the Page record, which are presented in the following Tables 3.3 and 3.4 [19].

[2]Semantically-Interlinked Online Communities, http://sioc-project.org/

Record	Attribute	Description
Post	type	Custom type of the post if specified (e.g. WordPress): attachment, page/post or other custom type
	posted_via	Information about the service used for posting if specified
	previous_URI	URI to the previous post is available
	next_URI	URI to the next post if available
	author	See Section 3.1.3.4
	content	See Section 3.1.3.4

Table 3.3: Post record extended attributes

Record	Attribute	Description
Page	template	Information about the design template if available and if different from the general blog
	author	See Section 3.1.3.4
	content	See Section 3.1.3.4

Table 3.4: Page record extended attributes

3.1.3.3 Comment as a record

Comments are published by others or the Author him/herself as a response to the original Page/Post/Comment. Comments appear along with the published Entry/Comment and provide an opportunity for readers to voice their views. The control over the publication of the Comments is held by the authors/administrators of the Blog. The properties of the Comment are presented in Table 3.5 [19].

Record	Attribute	Description
Comment	subject	Subject of the comment as retrieved
	URI	URI of the comment if available
	status	Information about the state of the comment: active, deleted, updated (versioning support)
	date_added	Date comment was added or retrieved, including time zone
	date_modified	Date comment was modified or retrieved as modified, including time zone
	addressed_to_URI	Implicit reference to a resource
	geo_longitude	Geographic positioning information
	geo_latitude	Geographic positioning information
	has_reply	Derived property (also SIOC)
	num_replies	Derived property (also SIOC)
	is_child_of_post	Indicates information about the parent post
	is_child_of_comment	Indicates information about the parent comment
	author	See Section 3.1.3.4
	content	See Section 3.1.3.4

Table 3.5: Comment record attributes

3.1.3.4 Other data associated with a record: Content and Author

In addition to the properties discussed along with records, it is necessary to highlight the existence of other data associated with records. Most prominent types of data associated with records are: Author data and published Content. These data, unlike other entities, cannot be described using a single property. For example, Authors can have a first/second name, a username or a URL to a user profile. Furthermore, these data can be associated to more than one type of record. For example, Pages, Posts and Comments can all be associated with a specific Author. Yet, Author is not being considered as a separate record, as the search results are more likely to require the Content published by the Authors. Hence, it makes sense to separate the description of the Author from the tables describing records. The same argument can be held for the published Content. To make sure that searching through various types of information integrated into the published Content can be organized, the Content is being categorized, yet associated with all the relevant records (which are Posts, Pages and Comments). The properties of the Author and the Content are presented in Tables 3.6 and 3.7 [19].

Entity	Attribute	Description
Author	username	Username of the profile
	name	Author name credentials where available
	profile_uri	URI to the profile
	avatar_uri	URI to the avatar file
	name_displayed	Name of the poster as displayed
	role	*Role of the user*
	email_displayed	Email address of the poster as displayed
	is_anonymous	Boolean property to indicate anonymity

Table 3.6: Author entity attributes

Entity	Attribute	Description
Content	full_content	Full content as extracted
	full_content_format	*Content format associated with the conceptual entity (e.g. Comment, Post)*
	note	Additional notes if available
	encoding	Information on encoding of the content
	copyright	Notes of copyright as retrieved from the blog
	ownership_rights	Notes of ownership rights as retrieved from the blog
	distribution_rights	Notes of distribution rights as retrieved from the blog
	access_rights	Notes of access rights as retrieved from the blog
	license	License of the content

Table 3.7: Content entity attributes

3.1.4 Changes to the data model

Since the development of the data model some refinement changes were proposed. The following has been accommodated and highlighted in italic font in the tables presented above:

- Entity Content: An additional property called *full_content_format* is added to capture sections of the HTML page that correspond to conceptual elements such as Comment or Entry. This is needed to eliminate the transfer of the full content of HTML from the Spider side when it is not necessary and to simplify the task of further processing of the section.
- Entities Blog and Entry: A property called *aliases* (if more than one, a set of properties or a new entity with one-to-many relationship) is necessary to capture the URL aliases that a Blog Entry or a Blog may have.
- Entity Author: A property called *role* to indicate the role of the user would be needed (e.g. contributor, editor, administrator, moderator).

3.1.5 Extended data model components

Capturing the data associated with the core of the blog may not be sufficient for some preservation initiatives. Hence, the repository should be able to capture additional properties. Towards this aim, the core blog model was extended (i.e. extended data model) to be able to accommodate additional data exhibited in blogs.

This section outlines the components of the extended data model. It introduces additional entities that are grouped according to their nature. These groups are referred here as categories. The categories capture various aspects of blogs and provide a descriptive foundation to enable preservation of additional blog data. While the changes within the defined components are possible, they represent a necessary foundation that can be used for capturing additional information if necessary. An example for possible extension can be the integration of additional technical metadata fields into the Categorized Content for addressing the requirements of the project.

The categories enable storing the following types of blog data:

- Blog Context: descriptive data provided by the bloggers themselves
- Network and Linked Data: a range of network data
- Community: information about the user base
- Categorized Content: descriptive data about the captured content
- Standard and Ontology Mapping: additional structures enabling mapping into other standards
- Semantics: information generated based on the analysis of the captured content
- Spam Detection: spam mark-up and associated descriptive data

- Crawling Info: specifics about the crawling
- Ranking, Category and Similarity: various measures based on the analysis of existing data
- Feed: information about the web feeds used

  The graphical representation of the categories in relation to the blog core is presented in Figure 3.2.

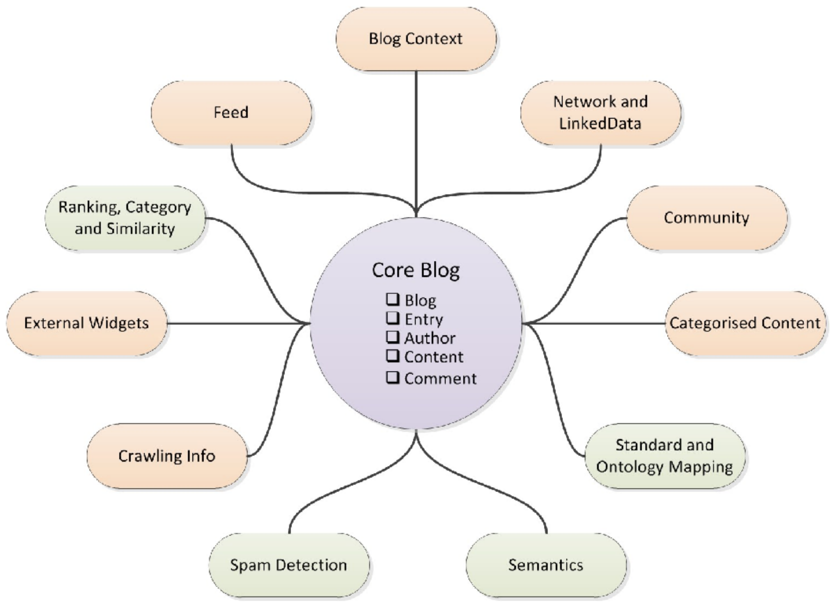


Figure 3.2: Blog core and components

There are primarily two types of categories described in the data model. The first type of category requires the data to be collected and extracted from the blog. The second type includes primarily derived properties and relies on the data already collected and stored in the repository. These two types of categories are represented in the diagram in different colours. The details about each of the components are accessible from the original D2.2 BlogForever report [19].

It is a common practice to anticipate some changes within the data model at the later stages of the project development. After initiating the design of the spider, some elements have already been discussed and modified. Due to agile methods adopted for the design and development of the repository component, some additional changes to the data model can be expected. We have put in place a mechanism for tracking and documenting these changes using an internal wiki. New suggestions are being added to the wiki, and the history of accepted changes is recorded.

3.1.6 Data model and Invenio

Achieving the aims of this project that revolve around solutions for preserving, managing and disseminating blogs requires a carefully designed software application. Using Invenio (already described in Section 2) for developing a repository system, provides a range of advantages. The system is designed and is shown to be capable of providing the functionality that is necessary for digital repository management and use. Reusing these functional facilities reduces the time required for designing and developing these from scratch. However, Invenio is not designed to work with blog records, so it requires some customization and adaptation of the system. This customization includes the changes necessary to be introduced to Invenio's database system. Otherwise the system will not be able to store, version, update or provide faceted search functionality using the above discussed records. Hence, the proposed data model can be used to inform the changes required to be added to the current structure of Invenio. However, due to possible changes to the structure of blogs in the future, it would be beneficial to design the database in a way that enables easier maintenance according to future possible changes.

3.2 Weblog preservation policies

This section is intended to summarize some of the key results from the deliverable D3.1 [11] that is deemed relevant to the repository software design. In particular, the current section describes the information, associated properties, and relationships targeted for capture to support preservation, in view of preservation objectives within BlogForever. The section ends with a summary of how these will be described and transmitted using controlled vocabularies, selected metadata schemas, and transmission standards.

3.2.1 Target information

In BlogForever we aim to capture information on three levels: micro-level, macro-level, and community-level. On the **macro-level**, there are four repository record types supported within BlogForever: Blog, Post, Comment, Page, which were already described in Section 3.1.3.

On the **micro-level**, the blog records come with associated embedded content. The six most common types of content associated to blogs are presented in Table 3.8 [11].

Content type	Example file formats	Common extensions
Structured text	Hyper Text Markup Language	HTML, HTM
	Extensible HyperText Markup Language	XHTML, XHT
	Extensible Markup Language	XML
	PHP Script Page	PHP
	HTML File Containing Server Side Directives	SHTML
	Cascading Style Sheet	CSS
Image	Portable Network Graphics	PNG
	Graphics Interchange Format	GIF
	Bitmap	BMP
	JPEG	JPG
	Scalable Vector Graphics	SVG
Documents	MS Word for Windows Document	DOC
	MS Office Open XML	DOCX
	OpenDocument Text	ODT
	Portable Document Format	PDF
	Plain Text File	TXT
	MS Excel Workbook	XLS
	MS Excel for Windows	XLSX
	OpenDocument Spreadsheet	ODS
	MS PowerPoint	PPT
	MS PowerPoint for Windows	PPTX
	OpenDocument Presentation	ODP
Audio	MPEG 1/2 Audio Layer 3	MP3
	Waveform Audio	WAV
Moving image	MPEG-1 Video Format, MPEG-2 Video Format	MPEG, MPG
	Audio/Video Interleaved Format	AVI
	QuickTime	MOV
	3GPP Audio/Video File	3GPP
	Macromedia FLV	FLV
Executables	Postscript	AI, EPS, EPSF, PS
	Base64-encoded bytes	MM, MME
	UNIX tar file, Gzipped	GZ, TGZ, Z, ZIP
	Compressed archive fle	ZIP
	Gzip compressed archive file	GZ
	Tape Archive Format	TAR
	Zip Format	ZIP
	Executable file	EXE, DLL, MSI
	XPInstall	XPI
	Atom Syndication Format feed	ATOM
	Really Simple Syndication feed	RSS
	Resource Description Framework	RDF
	Really Simple Discovery	RSD
	JavaScript	JS

Table 3.8: Digital object types commonly embedded within blogs

A blog is a form of social network media. As such, each blog record type does not stand on its own but must be interpreted in relation to other blogs, posts, comments and how they share technology, form networks and share information. The profile of the community from which the blog arises supports the end-user by allowing them to interpret the blog's authenticity, reliability and integrity. Three types of profiles, as a proxy of community-level information, have been recommended in D3.1[11]: a profile on the basis of technical characteristics, network structure, and user generated categorizations.

The properties associated to the three levels of information types (i.e. four record types, embedded content and community level profiles) that support preservation will be summarized in the next section.

3.2.2 Target properties and associations for preservation

The properties deemed significant for digital preservation in association to the four record types of Section 3.1.3 were described, in deliverable D3.1[11], as macro-level properties. These properties comprise those properties identified in Section 3.1.3 which , in particular, support stakeholder requirements examined in the work submitted as part of the D4.1 BlogForever report [8].

The deliverable D4.1[8], presents an integrated analysis of results from stakeholder interviews and results from the questionnaire survey reported in *D2.1: Survey implementation report* [6]. That is, it represents user requirements on several levels (e.g. blogger community, content managers and system administrators) examined using two independent methods for scoping user needs. This deliverable forms the foundation of the features being implemented in WP4. This ensures that the properties on the macro-level significant for preservation are being captured. An example of how properties of blog components on the micro-level are matched to user requirements to capture significant properties is illustrated in Figure 3.3.

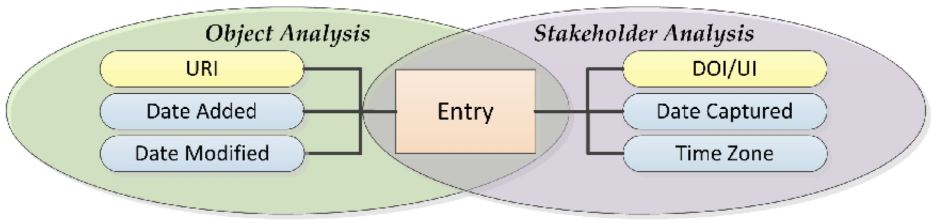


Figure 3.3: Cross matching the data model properties to user requirements

The properties that support the preservation of micro-level content embedded within each record type relies on their render-ability over time and has been studied in many digital preservation initiatives already. An investigation of previous work in the identification of core technical metadata was presented in D3.1[11] and adapted to yield the properties presented in the tables below with respect to five of the six most common media types (structured text, image, document, audio, and moving image) defined in the previous Table 3.8. These properties represent the technical metadata of content embedded within blog records to support renderability.

Object type	**Semantic unit**
Structured text	Title
	Creator
	Date
	Keywords
	Rights
	Div
	Span
	Language
	Paragraph
	Line break
	Headings
	Emphasis
	Bold
	Italics
	Underline
	Strong emphasis
	Strikethrough
	Horizontal rule
	Inserted text
	Deleted text
	Samp
	Cite
	Defined terms (DFN)
	Code
	Abbreviation
	Acronym
	Quotations
	Subscript / Superscript
	Address
	Button
	List elements
	Table elements
	Image
	Link
	Applet
	Frame
	Frameset

Table 3.9: Technical metadata of structured text

Object type	Semantic unit
Image	Image width Image height X sampling frequency Y sampling frequency Bits per sample Samples per pixle Extra samples

Table 3.10: Technical metadata of image

Object type	Semantic unit
Document	PageCount WordCount CharacterCount ParagraphCount LineCount GraphicsCount Language Fonts FontName IsEmbedded Features

Table 3.11: Technical metadata of document

Object type	Semantic unit
Audio	Duration Bit depth Sample rate Number of channels Sound field Sound map location for each channel Description Originator OriginatorReference OriginationDate OriginationTime Coding history Quality report Cue Sheet

Table 3.12: Technical metadata of audio

Object type	Semantic unit
Moving image	imageStreams
	audioStreams
	Lenght
	Width
	Height
	bibDepth
	colourModel
	colourSpace
	pixelAspectRatio
	frameRate
	Lossess
	CompressionRatio
	Codec
	Interlace
	Metadata

Table 3.13: Technical metadata of moving image

A weblog, as a form of social network media, carries the potential as an information source for the community. A community can be defined by three aspects:

- The styles, conventions, and types of information used by the community
- The level and patterns of interaction taking place between the community
- The scope and stability of concepts shared within the community

While all these are high-level concepts difficult to define objectively as specific properties of the blog, there are proxies of each dimension that is represented by formal characteristics exhibited by the webpage. These proxies and properties of blogs presented by community-level are:

- **Shared technical characteristics**
  1. Properties:
     - Type and number of media
     - Type and number of platforms in use
     - Types and number of formats in use
     - Layout characteristics
  2. Proxies:
     - Variety and number of HTML tags such as: <img>, <a>, <embed>, <video>, <object>, <audio>, <iframe>
     - Attribute values of HTML tags such as: <meta>, <link>, <script>
     - Variety and numbers of file extension patterns in the HTML tag attributes
     - Variety and number of layout HTML tags such as: <span>, <div>, <frame>, <style>, <em>, <strong>

- **Community network structure**
  1. Properties:
     - How many resources the target blog cites (internal, external)
     - How many resources cite the target blog
  2. Proxies:
     - Resources included as the attribute value of HTML tags: <img>, <script>, <a>, <link> (internal or external)
     - Link back information
     - Collection based link statistics
- **Shared user generated categorization**
  1. Properties:
     - Types and numbers of user generated categories
     - Number of shared categories across blog collections
  2. Proxies:
     - Categories and tags associated to blog posts

The properties related to network structure and user generated categories are already being extracted as part of the properties arising from the data model and user requirements. The technical characteristics of the original HTML code of the page is available as part of the original HTML code as received from the HTTP request for the webpage stored using a preservation friendly format (e.g. the digital forensics format aff[3]) as provenance information and a digital finger print of the blogging community. The community-level properties, as described before, are subject for consideration to be recorded as partial evidence of authenticity, integrity, and reliability, and, should ideally be exposed to the end-user community through the user interface.

3.2.3 Workflow for capturing the information and associated properties

A workflow for capturing the information (Section 3.2.1) and associated properties (Section 3.2.2) was proposed to map repository features (Section 5.1) to the recommendations of the Reference Model for an Open Archival Information System[4] in a way that meets the needs of weblog preservation.

Going through all the preservation recommendations defined in D3.1[11], we saw that most of them were already fulfilled explicitly or implicitly in the existing repository features. Just four new repository features were created in order to meet all the preservation recommendations.

The 17 preservation service recommendations were split into sub-recommendations to make easier the mapping between them and the repository features. How

[3]http://afflib.org/
[4]http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24683

the preservation recommendations were matched with the repository features is described in detail below:

1. **Receive submission**
   a) An interface to the Crawler (spider), usable by the repository managers
      - RF31 - The archive offers a complete blog submission interface to submit, modify and delete blogs/posts, includes implicitly this functionality

   b) A submission interface, allowing Producers to submit blogs in the forms of Submission Information Package (SIP)
      - RF31 - The archive offers a complete blog submission interface to submit, modify and delete blogs/posts, includes implicitly this functionality

   c) The SIP will probably be a METS wrapper which contains the original XML data as crawled by the spider, along with MARC and MIX metadata, and links to locally-attached files
      - RF12 - The archive can import METS

2. **Quality assurance**
   a) The repository should perform validation of the transmitted content to ensure that the transmission was successful and that the content is eligible for admission to the repository
      - RF40 - The archive validates the content received from the spider

3. **Generate descriptive information**
   a) Functions to create and edit the descriptive metadata are already implemented in Invenio. If the producer supplies metadata with their SIP, it's acceptable for the repository staff to enhance this metadata and create an "Updated SIP" in OAIS terms (already implemented in Invenio)
      - New feature defined: RF86 - The archive offers functions to edit metadata

4. **Generate Archival Information Package (AIP)**
   a) The repository should transform a submitted SIP into an archival AIP
      - New feature defined: RF87 - The archive transforms the SIPs received from the spider to AIPs

   b) The creation of an AIP will involve storing the content in two different databases
      - New feature defined: RF88 - The archive stores the content of the AIPs in two different databases for preservation purposes

   c) There would be a submission ID stored in the METS header, so all the stored data in both databases would be characterized by that UID and would be the OAIS AIP

- This is going to be in the platform, and in fact it has already been implemented, but we don't see how this is a feature on its own. It is more an implementation decision in the spider-repository communication and the metadata architecture

d) There may also be a recommendation for normalization or format-shifting (i.e. migration) of the media attachments found in blogs, such as text and images

- New feature defined: RF89 - The archive carries out the normalization and/or migration of the media attachments

5. **Co-ordinate updates**

   a) The repository must should move AIPs into archival storage, and store descriptive information in the database

   - RF9 - The archive stores and displays accordingly all the record metadata received from the spider

6. **Receive data**

   a) The repository should move an AIP into permanent storage

   - This is covered by features matched with recommendation 4.

7. **Manage storage hierarchy**

   a) The repository should implement a backup strategy

   - RF80 - The archive provides mechanisms to control data redundancy

8. **Replace media and migration strategy**

   a) The repository should be capable of reproducing the AIPs over time. This includes error checking for media failure in storage, but also the migration of file formats when necessary

   - RF60 - The archive can export all its content, database entries and file system for a migration

9. **Error-checking**

   a) The repository should provide assurance that the storage and data transfer process has not corrupted the AIP

   - This fuctionality is included in the design of RF9 (The archive stores and displays accordingly all the record metadata received from the spider), and also Invenio has the capability of periodically checking the data correctness.

10. **Disaster recovery**

    a) The repository should duplicate the contents of the archive and store the copies in a remote facility

    - RF80 - The archive provides mechanisms to control data redundancy

11. **Administer database**

a) The repository should have a database which contains descriptive information and system information
   - This is covered by features matched with recommendation 4.

12. **Perform queries**

a) The repository database should perform queries that can locate and retrieve blogs in response to requests
   - RF84 - The archive offers a complete range of search options to the user

13. **Generate report**

a) The repository system should create reports (e.g. on size of holdings in the archive, or usage statistics)
   - RF20 - The archive's statistics are exported as CSV

14. **Receive database updates**

a) The repository system should add, modify or delete database information in response to updates, such as ingest or access requests.
   - RF18 - The archive detects duplicated content and keeps only one copy
   - RF55 - The archive provides advanced APIs for developers to interact with the archive's content

15. **Co-ordinate access activities**

a) The repository should provide a user interface to the archive holdings
   - RF5 - The web interface provides harmonized access and ensures compatibility with major browsers

16. **Generate DIP**

a) The repository should allow an AIP to be converted into a DIP automatically
   - RF7 - Export data using the OAI-PMH protocol
   - RF8 - Export data in Dublin Core schema
   - RF59 - Export data using XML
   - RF62 - Export records as PDF and JPEG

17. **Deliver response**

a) The repository should deliver responses to consumers
   - RF3 - “Share” option in ”Your History” box
   - RF30 - Users are able to bookmark records, also using external bookmarking engines
   - RF38 - Users can communicate within the archive sharing and exchanging resources
   - RF50 - The archive offers the option to disseminate newly archived content in external social platforms

Metadata schemas for generating the AIP comprising the properties outlined in Section3.2.2 have also been recommended in the deliverable D3.1[11]. We refer to D3.1 for the criteria and selection process that led to the final recommendation. Here, we summarize the schemas selected for five of six object types described in Section 3.2.1. The type "executable" was not included in the current framework as it was agreed in the project earlier on that, while we will retrieve and store executables where we can, the focus of the repository, in establishing the archival information package of the repository, would be placed on the embedded content of the blog. The selected metadata schemas are summarized in Table 3.14.

Metadata type	**Description**	**Metadata schema**
Descriptive metadata	To locate and retrieve the blog and perform search queries in the repository	MARCXML[5]
Administrative metadata	Information related to establishing the authenticity, integrity, reliability and usability of the object	
Provenance metadata	Metadata related to Provenance, Preservation, and Rights	
Provenance and preservation	Provenance relates to source information and acquisition method (e.g. URI, crawler version, dates). Preservation relates to scheduled and performed changes to the data and metadata	PREMIS[6]
Rights metadata	For example: ownership, copyright, modification rights, access rights, and licenses	PREMISRights[7]
Technical metadata	Metadata related to renderability	
Structured Text	For example, number and positions of divisions	TextMD[8]
Image	For example still image resolution	MIX[9]
Audio	For example sound sample rate	AES57-2011[10]
Moving image	For example video frame rate	MPEG/7, Version 10, 2004[11]
Document	For example, length of the document	Florida Digital Archive/Harvard University Archive Document Metadata[12]

Table 3.14: Metadata schema

[5]http://www.loc.gov/marc/marcxml.html
[6]http://www.loc.gov/standards/premis/
[7]http://www.loc.gov/standards/premis/Rights-in-the-PREMIS-Data-Model.pdf
[8]http://www.loc.gov/standards/textMD/
[9]http://www.loc.gov/standards/mix/
[10]http://www.loc.gov/standards/amdvmd/audiovideoMDschemas.html
[11]http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm
[12]http://fclaweb.fcla.edu/uploads/

Note that the metadata types presented in the left hand side column of Table 3.14 reflects the structure of the Metadata Encoding and Transmission Standard (METS)[14] which categorize metadata into two high level sections: sections for descriptive metadata (dmdSec) and administrative metadata (amdSec). The administrative metadata, in turn, are divided into subcategories: a section related to provenance (digiProv) and related to technical specifications of embedded content (techMD). Provenance relates to any information that provides evidence of the authenticity and reliability of the data and metadata, which includes information on previous custodians of the information and any existing rights associated to the information.

The association with METS is reflective of the fact that the metadata for each component will be wrapped, encoded, and exposed using METS. The policies for how this will be wrapped within METS was described thoroughly in D3.1[11].

In addition to the METS object implemented and associated to each blog record, it is essential that a METS profile describing the purpose of the repository and the types of objects and formats supported within the repository is made available at a fixed URI within the repository.

The principles governing the management of the information ingested into the BlogForever repository was proposed in the METS profile and example included in the deliverable D3.1[11]. The profile stipulates:

- The controlled vocabularies and syntax to be used to fill metadata values (e.g. ISO standards for describing time, date, geographic locations)
- The set of external metadata schemas that are being incorporated into the METS standard (e.g. MIX for image objects), and how they will be included within the native METS syntax, that is which section will contain the information
- Any tools that might be employed for object characterization
- Description and structural rules for employment in writing METS objects

The profile should be considered to be a work in progress to be adapted, if necessary as the implementation of the repository continues to unfold. The final profile should be made available with a fixed public URI within the repository and also submitted for inclusion in a public registry such a the Library of Congress[13]. A draft examples of the METS profile and METS object are included in Appendix of the deliverable D3.1[11].

3.2.4 End-user user interface features to support preservation

In addition to core information capture, storage, description and encoding strategies described in above sections, to support continued repository improvement and

[13] http://www.loc.gov/standards/premis/

preservation activities, in D3.1[11], user interface features for end-users were suggested. These features are:

- To add recommendation features: for example, a request to have a blog harvested triggers the recommendation that other related blogs be harvested
- To allow end-users to contribute missing metadata values
- To allow users to request items missing or items that would add value to a target blog to be harvested
- To allow end-users to broadcast access problems to the community and request solutions (for example, migration request)
- To allow end-users to contribute solutions to access problems publicly
- To allow end-users outwith the repository to cite the material in the repository
- To document all of the above activities and make it transparent and public within the community

These features are designed to fill the gap for missing content and metadata, add value to existing content (by adding associated content and providing citation mechanisms), to stimulate community-based solutions to meet stakeholder needs and make the processes transparent.

Most of these features are already being supported in Invenio (e.g. methods for the citation of content). In addition, the community-level information and properties discussed in Sections 3.2.1 and 3.2.2 could be made transparent to the community through the interface to provide evidential support for end-users to gauge and contribute to maintaining authenticity, integrity and reliability of repository information.

Chapter 4

The BlogForever Repository Component

The BlogForever platform consists of two main components, the BlogForever spider [10] and the BlogForever repository.

The BlogForever project will extend and adapt the globally acknowledged and widely used Invenio package already described in detail in Chapter 2 to implement the repository. Its flexibility and performance make it a comprehensive solution for the management of document repositories of large size and render it as an ideal basis for the BlogForever repository design and construction.

In this chapter the process followed to carry out the repository component design will be introduced. As mentioned before, the repository design takes the Invenio suite as starting point and builds a Blog repository on top of it, merging the novel features with the Invenio existing ones. As mentioned and described in previous sections, the main inputs of this deliverable are the data model from D2.2[19], the preservation recommendations from D3.1[11] and the list of requirements from D4.1[8]. We will outline how these three inputs were considered in order to elaborate the list of repository features that will be incorporated on the top of Invenio in order to fulfill the BlogForever project requirements.

The 153 requirements coming from D4.1[8] were grouped by the designers when they considered that one functionality would make the repository meet all the requirements of the group. At the same time, other requirements needed more than one functionality to be fulfilled, so they were split in sub-requirements. Therefore, this allowed us to create a new list of features that needed to be in the repository and the many-to-many mapping to the requirements. Since the requirements of D4.1[8] were classified in three different categories (essential, recommended and optional) this information was used to determine the priority of the features as high, medium or low. The features were labeled with unique identifiers depending on the component that needed to implement the described functionality: the repository features are those starting by "RF" and the spider features are those starting by "SF". The spider features were discussed in *D4.2: Weblog spider component design*[10], while in this document (specifically in Section 5) we will focus on the repository features.

Later on, D3.1[11] proposed a list of 17 recommendations for the repository design. According to this WP3 deliverable, following these recommendations will make the repository a robust preservation archive. In the design process we considered each one of these recommendations, comparing them with the list of features coming from the requirements. We found that some recommendations were impossible to be included in the repository design since the recommendations were related to policies or hardware. Among the rest of the recommendations, most of them were already fulfilled explicitly or implicitly in the existing features. Finally, only four new features were needed to be added in order to meet all the D3.1[11] recommendations. How the recommendations were matched with the features is described in detail in Section 3.2.3. The complete list of features can be found in Chapter 5.

In the following sections the most relevant high level design decisions will be discussed.

4.1 Blog management with Invenio

A metadata structure has been developed in cooperation with WP3. As already explained in D3.1[11] and D4.2[10], the metadata standard chosen is METS. The structure of the METS profile that we defined is presented in D3.1[11].

4.1.1 Overall architecture

As mentioned in the introduction of this chapter, Invenio is the base of the BlogForever repository. In order to fulfill the requirements of the project, new modules and features have been designed to be incorporated on the top of it. These novel modules and features are:

- **BibIngest module**: This module is very important from the preservation point of view since this module is the responsible of storing the submitted material (ingestion packages) in its original format covering, at the same time, the requirements of the Open Archival Information System (OAIS) specification for the acquisition and storage of the Submission Information Package (SIP).
- **WebTag module**: The WebTag module will enable users to add free-form tags to blog records. Tags could easily be used for self-organization, it will be possible for a user to show all blog records tagged with a concrete tag and other related navigations.
- **Spam filtering**: The integration of this system will improve the archive's fidelity and integrity. Whenever BlogForever's spider captures new content, it will be evaluated using the spam detection web services before proceeding to archiving and digital preservation facilities. The filters work by combining information about spam captured on a large set of weblogs and then using those spam rules to block future spam.

- **Billing system**: This system has been designed to allow the system's administrators to exploit added value services. For instance, disk space in the digital repository could be free until users reach a specific quota. Additional disk space could be available for a monthly fee.
- **Spider communication**: An spider-repository API has been design in order to develop the communication between these independent components. There are two directions defined:

  1. repository - spider: The repository informs the spider that new blogs were submitted. The urls of these blogs are sent to the spider.
  2. spider - repository: The spider finishes crawling blogs and the repository retrieves all the crawled content.

- **Export options**: BlogForever repository comes with new options to export records such as “Export to METS”, “Export to PDF” and “Export to JPEG”.
- **Social features**: New social features such as display blogs that were read by people who also viewed a specific blog, disseminate newly archived content in external social platforms, subscribe and navigate activities of other users, etc. will be integrated into the repository.
- **Blog rendering**: New blog templates will be defined in order to represent the four types of records: Blog, Blog Post, Page and Comment.
- **Blog metadata**: In order to define the blog related properties, and therefore, define the internal BlogForever MARC schema to represent all the BlogForever related metadata, the Invenio MARC schema has been extended.

The Figure 4.1 represents an abstraction of the BlogForever repository design including the modules and features described above.

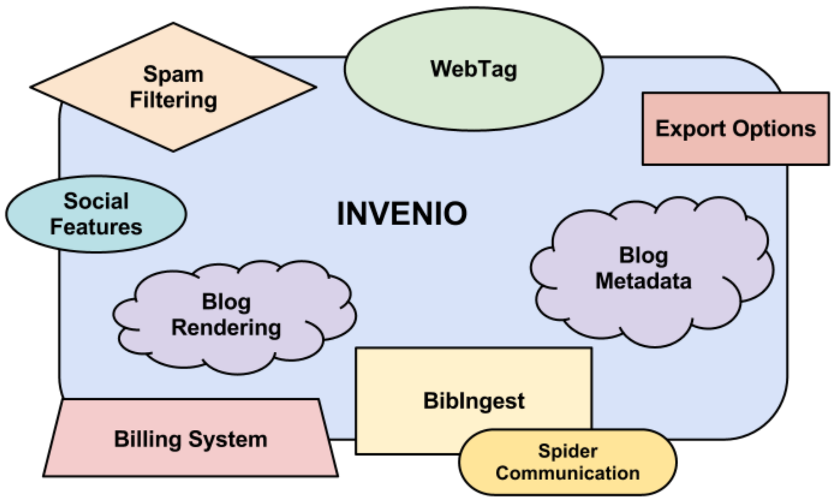


Figure 4.1: Overall architecture

4.1.2 Ingestion work-flow

In this sub-section the blog-related metadata management will be described. How the data arrives to the repository will be presented in first place, how it is stored and used will be next and finally the dissemination options will be introduced.

A communication protocol has been defined between the spider and the repository components. In the spider side, as mentioned in D4.2[10], an API interface has been developed, which is accessible via a web service that the repository is able to use. On the other hand, in the repository side, a plug-in system has been designed. The repository needs a series of commands to send to the spider, but this library should not be specific to one spider. Instead, the repository code has been designed in a way that allows future developers to extend it in an easy way by writing a plug-in that translates the set of commands used in the repository code to the specific syntax of the corresponding spider. The plug-in of the BlogForever spider will be developed within the project.

This design enhances the interoperability of both components, allowing the spider to be used by other repositories and also allowing the repository to potentially use other spiders. This interoperability is further enhanced if we consider that the metadata container chosen for the communication is the same METS profile defined in D3.1[11].

The entry point for data is the repository. Blog URLs can be inserted to the system using a submission form in the web interface. There is also a command-line tool that allows the submission of a batch of new URLs, but this option is only available for administrators. The access to this form is in principle authorized only to content managers, but the administrators can open it to any registered user. Should this happen, their submission would go though a refereeing process where designated content managers would be responsible of the approval or rejection of the suggested new blog. Once the new blog is approved to be in the repository, an Invenio record is created. This record will have only the information that the submitter provides (the blog URL and the title) and it will be stored temporarily in a hidden collection (only administrators and content managers have access to it). As soon as this happens, the repository will use the spider API to send the information about this new blog that needs to be crawled.

In the Section 2 the module BibSched was introduced. In parallel with the submission procedure and the push action to the spider, the repository will have a daemon running as a BibSched task. It will run periodically querying the spider to check if there are any new content available to be downloaded. The frequency of the check can be easily customized thanks to BibSched. The spider response will be a list of the object IDs for the new content available. These objects correspond to future repository records, and as such they can be a Blog, a Page, a Post, or a Comment. The repository will then go through the list of objects and will download the METS file as well as all the files attached to it. This includes images, videos and other files that the blog author might include in the content of the blog, and also a copy of the HTML and CSS files in the original server response. The Figure 4.2 shows a diagram of the repository-spider communication.

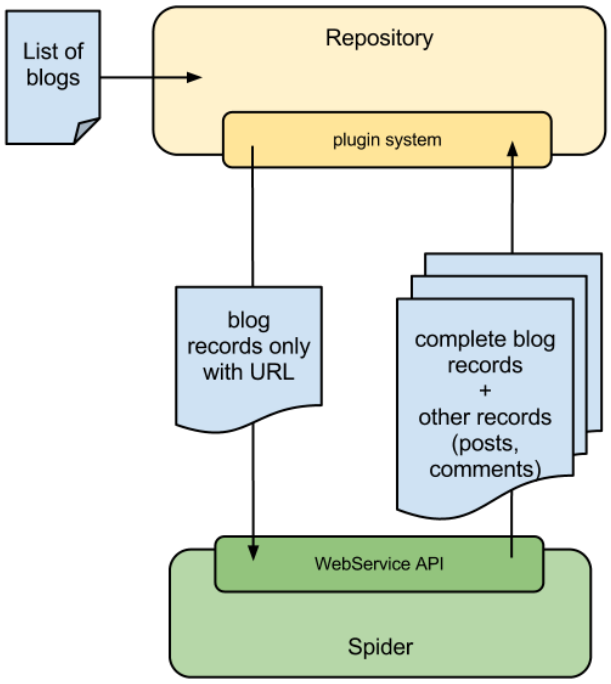

List of blogs
Repository
plugin system
blog records only with URL
complete blog records + other records (posts, comments)
WebService API
Spider

Figure 4.2: Repository-spider communication

Once all the content is downloaded in the repository side, there will be a call to BibUpload. Two plug-in systems have been designed allowing developers to pre and post process the records before and after the upload. The upload will then have 3 phases:

1. **Pre-process:** The MARC metadata embedded in the METS file is extracted. Then, it is enriched with FFT[1] ("Fulltext File Transfer") tags corresponding to each file downloaded from the spider and also the connections with other existing records are established. Since the metadata includes the URL of the parent Blog in the case of Posts and Pages and the parent Post or Comment in case of Comments, the presence of these related records will be checked and included in the MARC metadata. This does not alter the SIP since the MARC information in the METS file remains unaltered. It is only the internal management copy of Invenio that is enriched.

[1] http://cdsweb.cern.ch/help/admin/bibupload-admin-guide

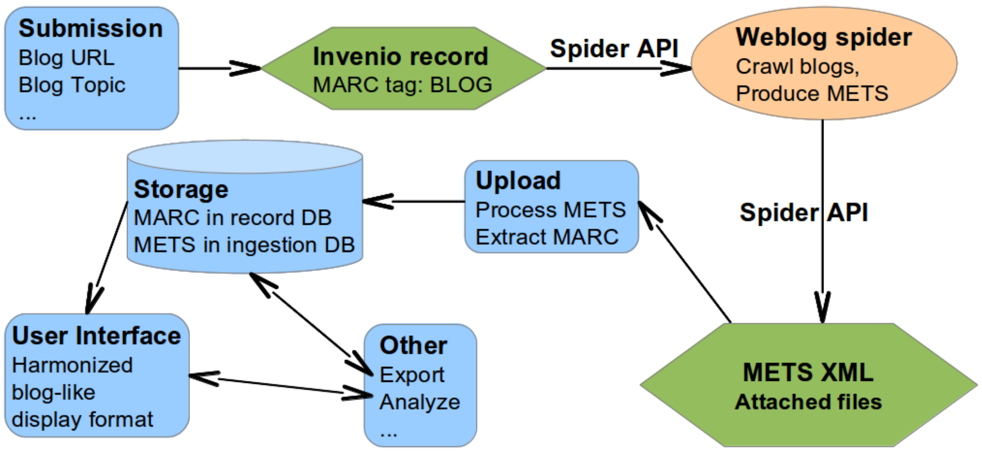


Figure 4.3: Metadata workflow in the repository

2. **Upload:** The BibUpload task will start, inserting the record in the Invenio databases in the usual way. Also, the attached files will be copied to the corresponding directories where Invenio (the BibDocFile module, more precisely) can manage them.
3. **Post-process:** Finally, and only if the upload process has been successful, the original METS document retrieved from the spider will be stored into a document-oriented database. The characteristics of this database will be better explained in the next subsection.

All this communication between the repository and the spider (querying and downloading files) will be done using the spider's API. The Figure 4.3 represents the metadata workflow in the repository. Starting in the submission, then the communication with the spider and finally the ingestion and storage.

4.1.3 BlogForever metadata structure

In the data model defined in deliverable D2.2[19] there are three kinds of entities: Blog, Entry and Comment. An Entry can be both a Post or a Page. All these entities became different kind of records in the repository design, as explained in Section 3.1.3. The decision adopted is to have all the records as first-class citizens in the system and to include in their metadata links to other records in order to keep the original structure, instead of having a hierarchical structure of 3 layers (Blog, Post/Page, Comment). Following the input from WP2 detailed in 3.1.3, four different kinds of records have been designed. But the difference between them is not the structure, but which tags will be present in the MARC metadata. The only difference in the behaviour is how they are rendered when the records are displayed in the web interface. There are two reasons behind this decision: this approach is much more flexible in the parent-child structure, allowing nested comments (Comments may address Posts and Pages, but also other Comments [19]) and it also fits much better in the Invenio architecture. An example of the logical structure of the four kinds of records can be found in the Figure 4.4.

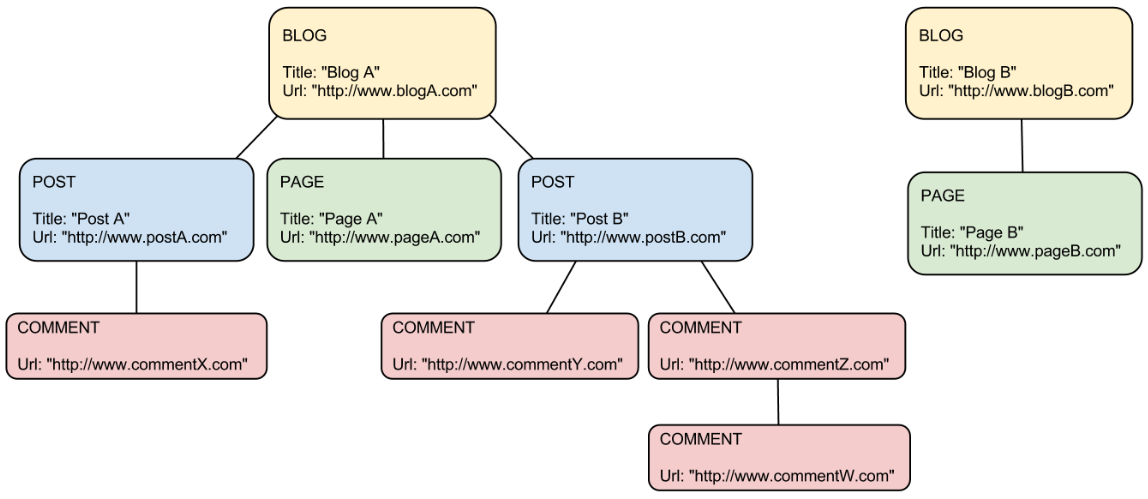

BLOG
Title: "Blog A"
Url: "http://www.blogA.com"
BLOG
Title: "Blog B"
Url: "http://www.blogB.com"
POST
Title: "Post A"
Url: "http://www.postA.com"
PAGE
Title: "Page A"
Url: "http://www.pageA.com"
POST
Title: "Post B"
Url: "http://www.postB.com"
PAGE
Title: "Page B"
Url: "http://www.pageB.com"
COMMENT
Url: "http://www.commentX.com"
COMMENT
Url: "http://www.commentY.com"
COMMENT
Url: "http://www.commentZ.com"
COMMENT
Url: "http://www.commentW.com"

Figure 4.4: Logical view of records structure

This view does not correspond with the way the records are stored in the repository. In the database, all the records share a common space and are not split in different holders for each record type. The information of the kind of record is stored in the MARC tag 980, as shown in Table 2.1. The relationships between the records are preserved including them in the metadata - mainly the parent record. The bubbles in Figure 4.5 do not reflect the reality but a conceptual splitting of the records in collections. It represents the same example than Figure 4.4 but the lines representing parent-child relationships have been replaced by tags that behave as links pointing to other records. In order to make these relationships more visible, the boxes representing records have been colored depending on the type of record they represent, and the tags inside a record that point to another record have been colored depending on the type of the record they point to.

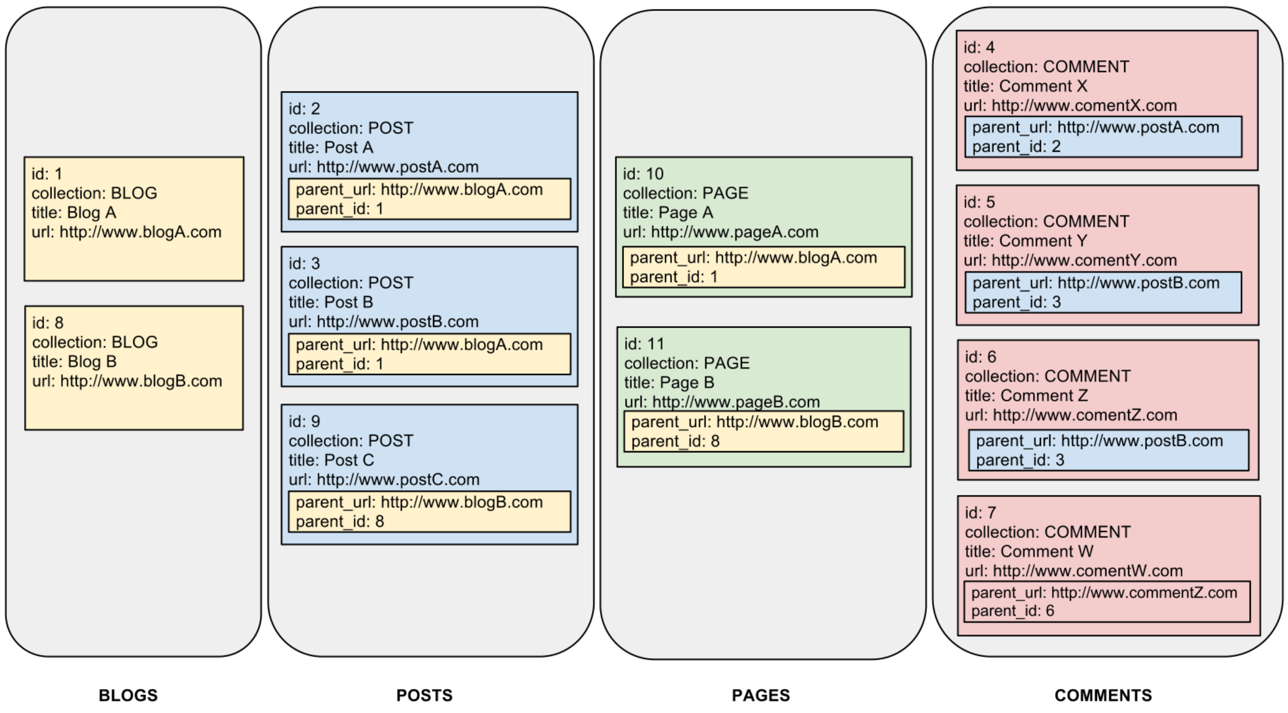

id: 1
collection: BLOG
title: Blog A
url: http://www.blogA.com
id: 8
collection: BLOG
title: Blog B
url: http://www.blogB.com
id: 2
collection: POST
title: Post A
url: http://www.postA.com
parent_url: http://www.blogA.com
parent_id: 1
id: 3
collection: POST
title: Post B
url: http://www.postB.com
parent_url: http://www.blogA.com
parent_id: 1
id: 9
collection: POST
title: Post C
url: http://www.postC.com
parent_url: http://www.blogB.com
parent_id: 8
id: 10
collection: PAGE
title: Page A
url: http://www.pageA.com
parent_url: http://www.blogA.com
parent_id: 1
id: 11
collection: PAGE
title: Page B
url: http://www.pageB.com
parent_url: http://www.blogB.com
parent_id: 8
id: 4
collection: COMMENT
title: Comment X
url: http://www.comentX.com
parent_url: http://www.postA.com
parent_id: 2
id: 5
collection: COMMENT
title: Comment Y
url: http://www.comentY.com
parent_url: http://www.postB.com
parent_id: 3
id: 6
collection: COMMENT
title: Comment Z
url: http://www.comentZ.com
parent_url: http://www.postB.com
parent_id: 3
id: 7
collection: COMMENT
title: Comment W
url: http://www.comentW.com
parent_url: http://www.commentZ.com
parent_id: 6
BLOGS
POSTS
PAGES
COMMENTS

Figure 4.5: A lower level view of records structure

The Figure 4.6 shows the internal structure of an Invenio record, that can be considered as an OAIS AIP. An Invenio record mainly consists of three components. The MARC metadata, stored in the database, holds not only the information describing the record but also links to other related records. The second component is the BibDocFile module, where the attached files are stored. These are the files coming from the spider (original HTML and CSS files, pictures, etc.) and are used to display the record in the web interface. The third component is the database already introduced in the post-process phase of the ingestion work-flow. In the repository design MongoDB[4] was chosen for this purpose, which is a scalable, high-performance, open-source and document-oriented database. A new Invenio module called BibIngest[3], has been designed to hold the submission METS files and to cover the needs of persistent storage of submitted material by using MongoDB. Since BibDocFile, BibIngest, and Invenio's MARC support versioning, this characteristic will be exploited by the repository in its preservation features.

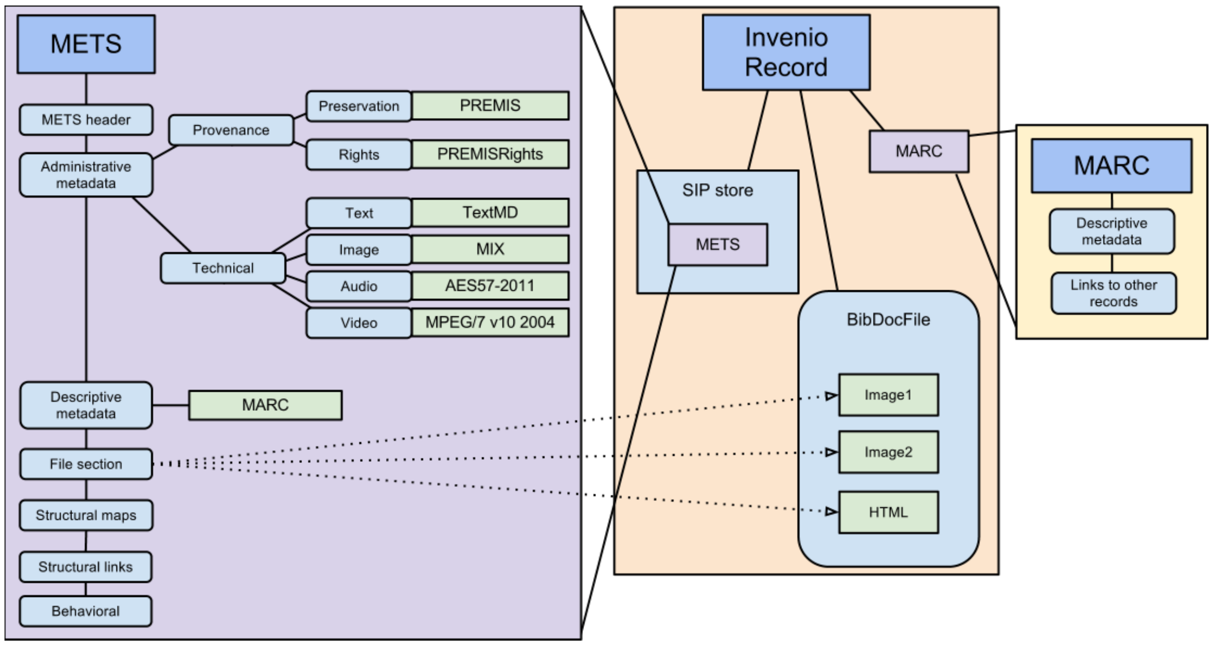


Figure 4.6: Metadata architecture

4.1.3.1 Mapping of blog attributes to MARC

As explained in Section 2, Invenio uses MARC 21 internally to represent the bibliographic metadata. This MARC schema has been chosen as a starting point in order to define the internal BlogForever MARC schema and to represent all the BlogForever related metadata.

In the previous Section 3.1.3, four type of records were identified to be stored in the BlogForever repository: Blog, Post, Comment and Page, as well as the attributes that define each of them. What is intended to present in the tables below (following the MARC typographical conventions[2]) is the set of MARC tags chosen to represent all these attributes and to compose, therefore, the BlogForever MARC schema.

[2]http://www.loc.gov/marc/bibliographic/concise/bdintro.html

Record	Attribute/Metadata concept	MARC 21 representation
Blog	title	245 $a
	subtitle	245 $b
	URI	520 $u
	aliases	100 $g
	status_code	952 $a
	language	041 $a
	encoding	532
	sitemap_uri	520
	platform	781 $a
	platform_version	781 $b
	webmaster	955 $a
	hosting_ip	956 $a
	location_city	270 $d
	location_country	270 $b
	last_activity_date	954 $a
	post_frequency	954 $b
	update_frequency	954 $c
	copyright	542
	ownership_rights	542
	distribution_rights	542
	access_rights	542
	license	542 $f

Table 4.1: Blog record attributes - MARC tags mapping

Record	Attribute/Metadata concept	MARC 21 representation
Entry (Post and Page)	title	245 $a
	subtitle	245 $b
	full_content	520 $a
	full_content_format	520 $b
	author	100 $a
	URI	520 $u
	aliases	100 $g
	alt_identifier (UR)	0247 $a
	date_created	269 $c
	date_modified	260 $m
	version	950 $a
	status_code	952 $a
	response_code	952 $b
	geo_longitude	342 $g
	geo_latitude	342 $h
	access_restriction	506
	has_reply	788 $a
	last_reply_date	788 $c
	num_of_replies	788 $b
	child_of	760 $o $4 $w

Table 4.2: Post and Page record shared attributes - MARC tags mapping

Record	Attribute/Metadata concept	MARC 21 representation
Post	type	336
	posted_via	781 $a
	previous_URI	780
	next_URI	785

Table 4.3: Post record extended attributes - MARC tags mapping

Record	Attribute/Metadata concept	MARC 21 representation
Page	template	962

Table 4.4: Page record extended attributes - MARC tags mapping

Record	Attribute/Metadata concept	MARC 21 representation
Comment	subject	245 $a
	author	100 $a
	full_content	520 $a
	full_content_format	520 $b
	URI	520 $u
	status	952 $a
	date_added	269 $c
	date_modified	269 $m
	addressed_to_URI	789 $u
	geo_longitude	342 $g
	geo_latitude	342 $h
	has_reply	788 $a
	num_replies	788 $b
	is_child_of_post	773 $o $4 $w
	is_child_of_comment	773 $o $4 $w

Table 4.5: Comment record attributes - MARC tags mapping

4.1.4 Data export

The Invenio data export capabilities have been extended in the design. The blog-related content and its metadata can be exported in a variety of ways, enhancing interoperability. The available options are the following:

- **OAI-PMH:** The Open Archives Initiative metadata harvesting protocol (OAI-PMH) can be used in Invenio to import and also export data.
- **XML:** The metadata is automatically converted and exported in a variety of XML standards: BibTeX[3], MARC, MARCXML, DC[4], EndNote[5], RefWorks[6] and METS.
- **PDF and JPEG:** Blog content can be downloaded in a human-friendly printable format. Including the child posts are included when a Blog record is exported in this format will be optional.
- **SRU:** External machines are able to query the repository using the standarized querying syntax of SRU[7] and retrieve metadata in MARC or DC formats.

[3]http://www.bibtex.org/
[4]http://dublincore.org/
[5]http://endnote.com/
[6]http://www.refworks-cos.com/refworks/
[7]http://www.loc.gov/standards/sru/

4.2 Scalability

After describing the scalability techniques used in Invenio instances and why they are important, in Section 2.4, the recommended for potential administrators of the BlogForever repository will be described in this section.

Concerning the BlogForever case, it is possible to have several repository instances (∼10), each holding different kinds of blogs, each having its own API connection to spider, each having its own BibUpload queue, etc. In other words, these repository instances would be fully dissociated. And then there would be one repository above them that would not hold any data, but would simply dispatch user queries in “external hosted collection” style. Depending on the nature of data, this setup could easily hold several tens of millions of records and more.

The next step on the scalability ladder is the situation where it is not practical to setup “manual” horizontal partitioning based on the document corpus specifics, in the way mentioned above. Here, automatic sharding would be suitable. This is currently not possible out of the box. The solution would be to take the previous technique and bring it to non-search scenarios too. The master instance would basically keep consecutive sequence of record IDs from 1 to 100 million, say, with notes that record 17 lives on shard 1 as record 10, record 18 lives on shard 2 as record 7, etc. An automated layer would then be called by master BibUpload to dispatch create/replace/update/delete tasks to shards. *Ditto* for search queries, similar to hosted search. This setup could be nested. This will allow for full automatic sharing, without having to think too much of document corpus based partitioning. This solution is already being implemented by the Invenio team at CERN and it is expected to be usable by March 2013.

4.3 BlogForever user interface

The repository requirements address the need for attention to a qualitative user interface experience for users of the repository. The design of the existing interface, adopted from Invenio, did not fully meet these requirements, and a decision was made to seek out a modern, standards-compliant, mobile-adaptive framework from which to develop the design for the front-end of the repository.

4.3.1 User interface related requirements and features

The major user interface requirements related to the front-end design of the repository, which have been extracted from the BlogForever D4.1 report [8], are:

- UI1 - Web interface
- UI16 - Easy to learn/Intuitive
- UI14 - User interface for mobiles

- UI15 - Search interface
- FR37 - Web portal
- FR43 - Access to content in a harmonized way

These requirements are covered by the following defined features (see the description of these features in Section 5.2):

- RF5 - The web interface provides harmonized access and ensures compatibility with major browsers
- RF77 - The archive provides a mobile version

So far, these features are not met by the existing Invenio archive design. With no templating function, the interface is built directly into the Python code, and provides an old-fashioned visual design, a mark-up structure that reduces the ability to customize the interface and to add new features. Small font sizes provide poor readability, and the page mark-up is not optimized for accessibility.

However, Invenio has reached the point where it is useful to rewrite part of the code base using a new software stack in order to keep flexibility, manageability of growing number of modules, and speed-up prototyping and development of new ones[12]. This includes an upgrade to the existing Invenio interface providing a more modern appearance, a responsive and accessible design, and an uncluttered interface with clear, simple terminology to assist users to easily and logically navigate through the repository.

Taking advantage of all this work, the development of the BlogForever user interface will be carried out using the same technologies that have been adopted to improve the existing Invenio interface. These technologies will be described in the next sub-section.

4.3.2 User interface technologies

In this sub-section are described the adopted technologies by Invenio in order to achieve the goal described above, and therefore, the technologies that will be used by the front-end BlogForever programmers.

4.3.2.1 Twitter Bootstrap framework

The Twitter Bootstrap[5] framework solves many of the problems mentioned above with the existing interface. It is a fully modern HTML5[8]/CSS3[9] responsive framework, so full mobile functionality is already built in, and its existing CSS styles include high quality typography that is ideal for a text-heavy resource such as the BlogForever repository.

[8]http://www.w3schools.com/html/html5_intro.asp

[9]http://www.css3.com/

As a fully-fleshed out framework with existing classes and HTML structures for most interface elements, it allows for very fast development of attractive and functional front-end features. It is widely supported by a community of developers and designers, which allows BlogForever to take advantage of existing resources from the community such as “FontAwesome”[10] icons and “Bootswatch”[11] color templates, and because of its well-structured HTML and CSS mark-up, it will be very easy for individual repository owners to customise the appearance of their front-ends to match institutional style guidelines.

Bootstrap

Sleek, intuitive, and powerful front-end framework for faster and easier web development.

Figure 4.7: Twitter-Bootstrap logo

4.3.2.2 Jinja templating

The use of Jinja templating[17] introduces a templating function into the repository, so that the code for the visual display of the repository can be maintained and edited separately to the back-end components. This allows faster and easier customization of the user interface and provides a more modular code framework that is easier to maintain.

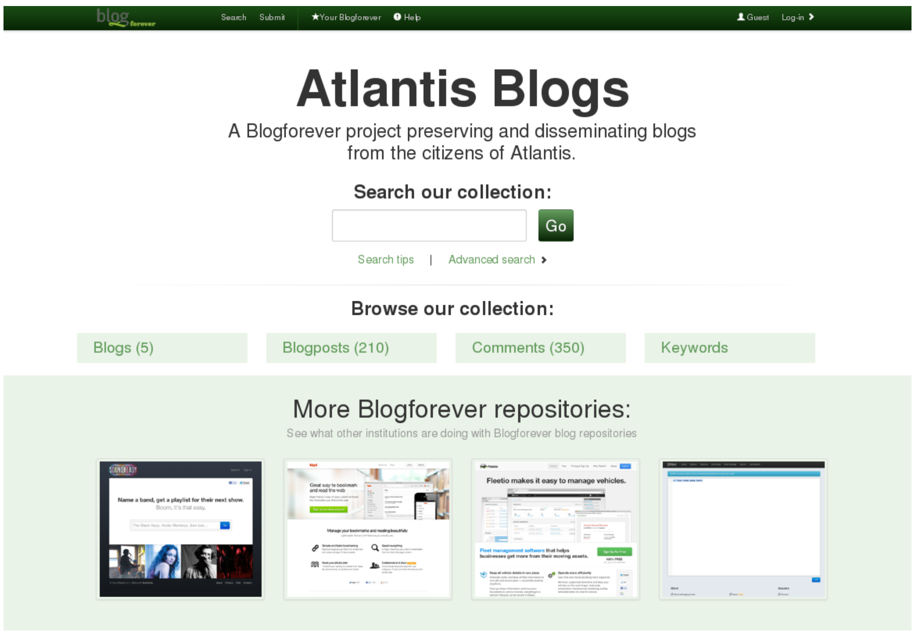
Figure 4.8: Jinja logo

4.3.3 BlogForever user interface prototype

The design of the BlogForever user interface started with the design and development of a user interface prototype. What has been done so far is the development of some

[10]http://fortawesome.github.com/Font-Awesome/
[11]http://bootswatch.com/

mock-ups taking advantage of all the work already done in Bootstrap. These mock-ups will be improved as long as the development takes place. The first mock-up (Figure 4.9) corresponds to the main search page of the BlogForever repository.



Atlantis Blogs

A Blogforever project preserving and disseminating blogs from the citizens of Atlantis.

Search our collection:

Browse our collection:

More Blogforever repositories:

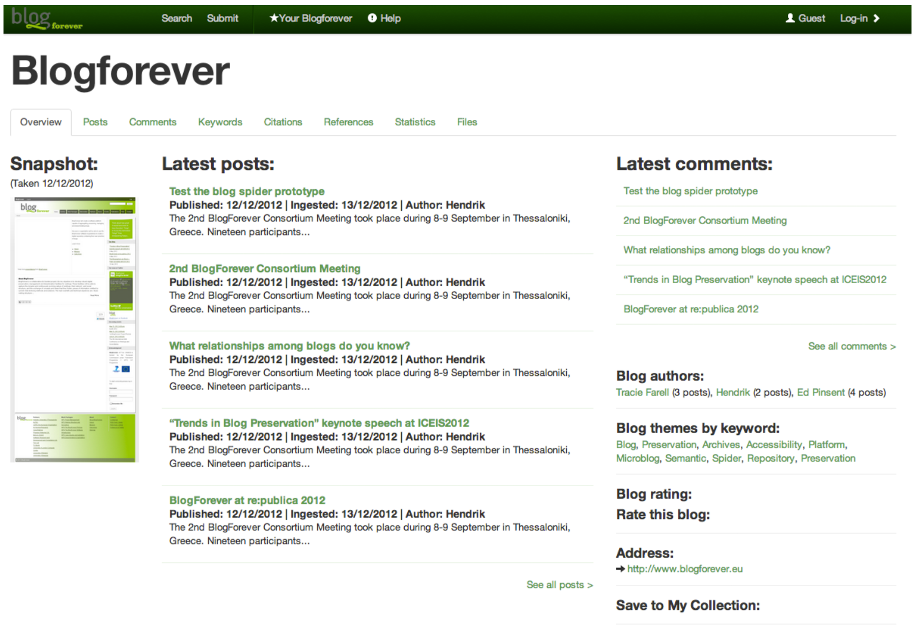
See what other institutions are doing with Blogforever blog repositories

Figure 4.9: Main search page prototype

The focus is on clear as well as accessible text-based navigation and features, with the search box front and center, and the “browse” functions immediately below. The built-in JQuery[12] functions allowed for fast development of front-end features such as the log-in drop-down, advanced search functions and tabbed browsing of records.

The next images (Figures 4.10, 4.11, 4.12) correspond to the developed mock-ups for each of the type of records that will be stored in the BlogForever repository: Blog, Entry(Post and Page, due to the fact that both types of record have a similar template just the mock-up for Post is showed) and Comment.

[12]http://jquery.com/



Blogforever

Snapshot:
(Taken 12/12/2012)

Latest posts:

Test the blog spider prototype
Published: 12/12/2012 | Ingested: 13/12/2012 | Author: Hendrik
The 2nd BlogForever Consortium Meeting took place during 8-9 September in Thessaloniki, Greece. Nineteen participants...

2nd BlogForever Consortium Meeting
Published: 12/12/2012 | Ingested: 13/12/2012 | Author: Hendrik
The 2nd BlogForever Consortium Meeting took place during 8-9 September in Thessaloniki, Greece. Nineteen participants...

What relationships among blogs do you know?
Published: 12/12/2012 | Ingested: 13/12/2012 | Author: Hendrik
The 2nd BlogForever Consortium Meeting took place during 8-9 September in Thessaloniki, Greece. Nineteen participants...

"Trends in Blog Preservation" keynote speech at ICEIS2012
Published: 12/12/2012 | Ingested: 13/12/2012 | Author: Hendrik
The 2nd BlogForever Consortium Meeting took place during 8-9 September in Thessaloniki, Greece. Nineteen participants...

BlogForever at re:publica 2012
Published: 12/12/2012 | Ingested: 13/12/2012 | Author: Hendrik
The 2nd BlogForever Consortium Meeting took place during 8-9 September in Thessaloniki, Greece. Nineteen participants...

See all posts >

Latest comments:

Test the blog spider prototype
2nd BlogForever Consortium Meeting
What relationships among blogs do you know?
"Trends in Blog Preservation" keynote speech at ICEIS2012
BlogForever at re:publica 2012

See all comments >

Blog authors:
Tracie Farell (3 posts), Hendrik (2 posts), Ed Pinsent (4 posts)

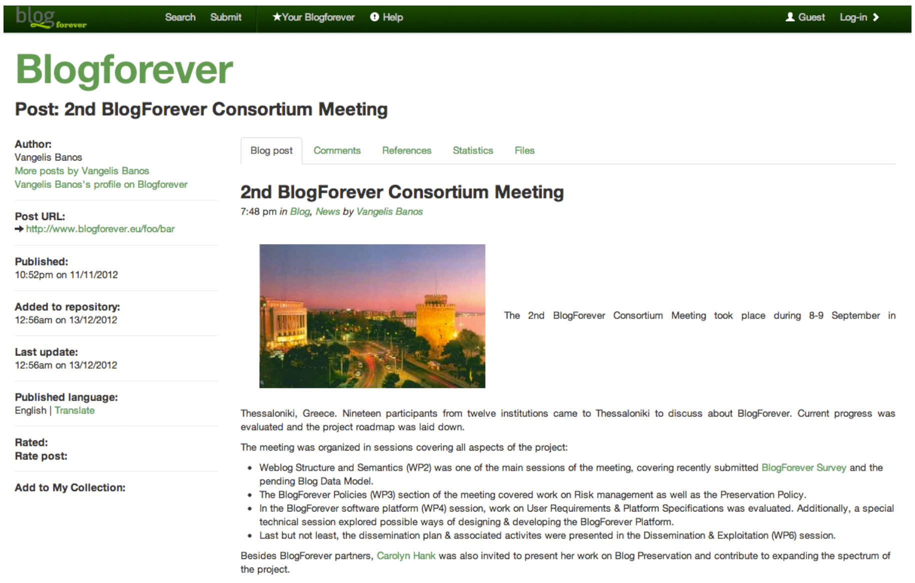
Blog themes by keyword:
Blog, Preservation, Archives, Accessibility, Platform, Microblog, Semantic, Spider, Repository, Preservation

Blog rating:
Rate this blog:

Address:
➜ http://www.blogforever.eu

Save to My Collection:

Figure 4.10: Blog record prototype



Blogforever

Post: 2nd BlogForever Consortium Meeting

Author:
Vangelis Banos
More posts by Vangelis Banos
Vangelis Banos's profile on Blogforever

Post URL:
➜ http://www.blogforever.eu/foo/bar

Published:
10:52pm on 11/11/2012

Added to repository:
12:56am on 13/12/2012

Last update:
12:56am on 13/12/2012

Published language:
English | Translate

Rated:
Rate post:

Add to My Collection:

2nd BlogForever Consortium Meeting
7:48 pm *in Blog, News by Vangelis Banos*

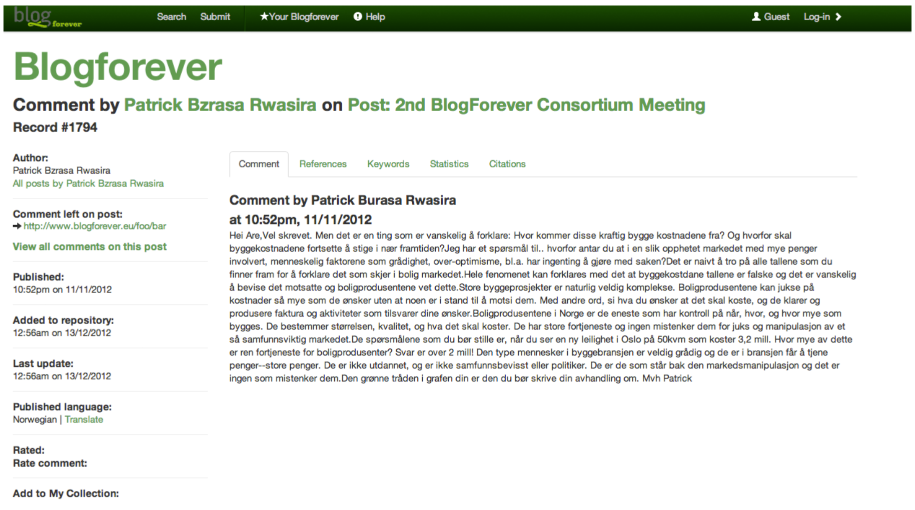
The 2nd BlogForever Consortium Meeting took place during 8-9 September in Thessaloniki, Greece. Nineteen participants from twelve institutions came to Thessaloniki to discuss about BlogForever. Current progress was evaluated and the project roadmap was laid down.

The meeting was organized in sessions covering all aspects of the project:

- Weblog Structure and Semantics (WP2) was one of the main sessions of the meeting, covering recently submitted BlogForever Survey and the pending Blog Data Model.
- The BlogForever Policies (WP3) section of the meeting covered work on Risk management as well as the Preservation Policy.
- In the BlogForever software platform (WP4) session, work on User Requirements & Platform Specifications was evaluated. Additionally, a special technical session explored possible ways of designing & developing the BlogForever Platform.
- Last but not least, the dissemination plan & associated activites were presented in the Dissemination & Exploitation (WP6) session.

Besides BlogForever partners, Carolyn Hank was also invited to present her work on Blog Preservation and contribute to expanding the spectrum of the project.

Figure 4.11: Post record prototype



Blogforever

Comment by Patrick Bzrasa Rwasira on Post: 2nd BlogForever Consortium Meeting

Record #1794

Author:
Patrick Bzrasa Rwasira
All posts by Patrick Bzrasa Rwasira

Comment left on post:
➜ http://www.blogforever.eu/foo/bar
View all comments on this post

Published:
10:52pm on 11/11/2012

Added to repository:
12:56am on 13/12/2012

Last update:
12:56am on 13/12/2012

Published language:
Norwegian | Translate

Rated:
Rate comment:

Add to My Collection:

Comment by Patrick Burasa Rwasira
at 10:52pm, 11/11/2012

Hei Are,Vel skrevet. Men det er en ting som er vanskelig å forklare: Hvor kommer disse kraftig bygge kostnadene fra? Og hvorfor skal byggekostnadene fortsette å stige i nær framtiden?Jeg har et spørsmål til.. hvorfor antar du at i en slik opphetet markedet med mye penger involvert, menneskelig faktorene som grådighet, over-optimisme, bl.a. har ingenting å gjøre med saken?Det er naivt å tro på alle tallene som du finner fram for å forklare det som skjer i bolig markedet.Hele fenomenet kan forklares med det at byggekostdane tallene er falske og det er vanskelig å bevise det motsatte og boligprodusentene vet dette.Store byggeprosjekter er naturlig veldig komplekse. Boligprodusentene kan jukse på kostnader så mye som de ønsker uten at noen er i stand til å motsi dem. Med andre ord, si hva du ønsker at det skal koste, og de klarer og produsere faktura og aktiviteter som tilsvarer dine ønsker.Boligprodusentene i Norge er de eneste som har kontroll på når, hvor, og hvor mye som bygges. De bestemmer størrelsen, kvalitet, og hva det skal koster. De har store fortjeneste og ingen mistenker dem for juks og manipulasjon av et så samfunnsviktig markedet.De spørsmålene som du bør stille er, når du ser en ny leilighet i Oslo på 50kvm som koster 3,2 mill. Hvor mye av dette er ren fortjeneste for boligprodusenter? Svar er over 2 mill! Den type mennesker i byggebransjen er veldig grådig og de er i bransjen får å tjene penger--store penger. De er ikke utdannet, og er ikke samfunnsbevisst eller politiker. De er de som står bak den markedsmanipulasjon og det er ingen som mistenker dem.Den grønne tråden i grafen din er den du bør skrive din avhandling om. Mvh Patrick

Figure 4.12: Comment record prototype

The design is content-driven, with blog post and comment content being highlighted first to invite reading, discovery and browsing actions by users, followed by comments and then associated metadata and post-related functions. The user is kept informed about their location in the repository with breadcrumbing of blog title, blog post and comments, so that information is always presented in context.

Bootstrap's built-in responsive styles mean that the mock-ups are already fully mobile-compatible, and the clear, readable typography ensures a coherent visual experience throughout the repository. Further development of an HTML cleaning and parsing tool strengthens the unified display of content in the repository.

The speed of development in a Bootstrap interface means that as BlogForever features and requirements are refined through the Case Study process [18], the front-end will be able to be rapidly modified to meet these needs without long development periods. The work done to introduce the Jinja templating layer and the Twitter Bootstrap framework lay a strong base from which to continue developing a high quality, responsive and user-focused interface for the BlogForever repository.

Chapter 5

Feature Specifications

In this chapter, the final list of repository features and associated specifications is presented. As we already mentioned in Chapter 4, during the design phase of the BlogForever repository, the requirements defined in D4.1[8] were used as input to define repository features. The definition of requirement and feature can be summarized as follows [16] [20]:

- A **requirement** is a capability that a product must possess or something a product must do in order to ultimately satisfy a customer need. A requirement tends to be more granular, and is usually written with the implementation in mind.
- A **feature** is a set of related requirements that allows the user to satisfy a business objective or need. A feature tends to be a "higher-level" objective than a requirement (and is usually more focused on business needs rather than implementation).

The rest of this section is structured as follows: in Section 5.1 the methodology of mapping requirements to features is explained. In Section 5.2 the feature specifications are presented in detail.

5.1 Mapping of requirements to features

Using the knowledge gathered during the requirements analysis we were able to identify a list of features. This was done by functionally decomposing the domain (the BlogForever platform) into subject areas. The Figure 5.1 shows the followed steps in the process of mapping requirements to features.

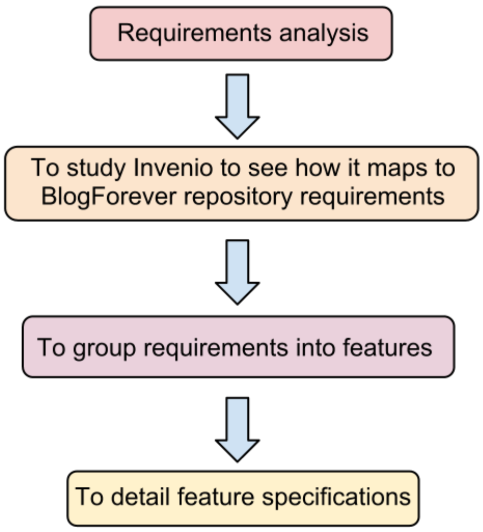

Requirements analysis
To study Invenio to see how it maps to BlogForever repository requirements
To group requirements into features
To detail feature specifications

Figure 5.1: Requirements to features process

Therefore, the set of requirements was thematically grouped together and mapped to a single feature. On the other hand, some requirements were broken down and mapped to more than one features since they were described by distinct sets of software attributes.

The list of the 89 features that were created as result of the process described above is presented in Table5.1.

Feature	**Requirements**
RF1 - Customizable user dashboard	**UI13** - Customizable user dashboard
RF2 - "Your History" box as part of the user dashboard	**UI3** - History of own activities in the archive
RF3 - "Share" option in "Your History" box	**UI14** - Subscribe and navigate activities of other users
RF4 - Bibformat output templates to display blogs and blog posts differently	**DR11** - Differentiate between blog and blog post
RF5 - The web interface provides harmonized access and ensures compatibility with major browsers	**UI1** - Web Interface **FR37** - Web portal **FR43** - Access to content in a harmonized way
RF6 - Latest posts are displayed sorted by addition date	**UI6** - Latest posts
RF7 - Export data using the OAI-PMH protocol	**IR4** - Expose parts of the archive via OAI-PMH based on specified criteria **IR6** - Facilities to enable interoperability
RF8 - Export data in Dublin Core schema	**IR3** - Export data using OAI-PMH protocol and Dublin Core schema **IR6** - Facilities to enable interoperability
RF9 - The archive stores and displays accordingly all record metadata received from the spider	**DR17** - Metadata for blogs **DR21** - Long term digital preservation
RF10 - Archive user passwords are stored encrypted in the database	**SR1** - Passwords are stored encrypted
RF11 - The web interface is available in many different languages	**FR46** - Internationalization **UI29** - Multiple language support
RF12 - The archive can import METS	**DR22** - METS **IR6** - Facilities to enable interoperability **DR21** - Long term digital preservation **OP2** - OAIS
RF13 - UTF-8 is used as the default character encoding in the archive	**FR51** - UTF8 the default character encoding **DR21** - Long term digital preservation
RF14 - Descriptive statistics are offered by record	**FR3** - Descriptive statistics for the archive **DR15** - Visits of blogs and blog posts **FR5** - Descriptive statistics for a single blog or blog post **DR12** - Demographics
RF15 - Option to disseminate archive content in major social web platforms	**FR7** - User dissemination channels for blog post **UI16** - Easy to learn/Intuitive **UI28** - Integration/Combination with other systems

RF16 - The archive offers an RSS channel of its latest updates and/or users can receive notification when new content of their interest is added to the archive	**FR12** - Notification about changes in the archive **UI16** - Easy to learn/Intuitive **UI28** - Integration/Combination with other systems
RF17 - The archive displays a disclaimer about the originality of the content	**UI21** - Archived content is clearly stated as such **DR3** - Disclaimer **UI16** - Easy to learn/Intuitive
RF18 - The archive detects duplicated content and keeps only one copy	**FR23** - Detection of duplicates **DR21** - Long term digital preservation
RF19 - The archive can be indexed by external search engines	**EI4** - Accessible via search machines
RF20 - The archive's statistics are exported as CSV	**EI5** - Export as CSV **UI28** - Integration/Combination with other systems
RF21 - The archive offers the option to login using SSO/LDAP	**IR1** - Single Sign On/Interoperates with Authentication System especially LDAP
RF22 - "Your Preferences" box as part of the user dashboard	**UI33** - User profiles **UI16** - Easy to learn/Intuitive
RF23 - The archive stores the comments of blog posts and displays them as part of the blog posts	**DR13** - Comments **DR21** - Long term digital preservation **FR54** - What to archive: text and comments
RF24 - Links to other sources within blog posts and comments are displayed separately	**UI17** - Display references (links) to other sources inside or outside the archive **UI16** - Easy to learn/Intuitive
RF25 - The archive displays the tags of blogs and blog posts	**UI7** - Tags for blogs and blog posts
RF26 - BlogUploader command line to upload, update and delete a list of blogs	**FR15** - Selection of blogs to archive **UI16** - Easy to learn/Intuitive
RF27 - The archive displays a unique URL (DOI) for each record	**DR2** - URI and metadata for referencing/citing **IR8** - Digital Object Identifier **IR6** - Facilities to enable interoperability
RF28 - The archive displays the author of blog posts and comments	**DR4** - Display the author of the blog, blog post, comment
RF29 - The archive alerts users when there are software updates	**SM2** - Software updates
RF30 - Users are able to bookmark records, also using external bookmarking engines	**FR10** - Bookmarking of blog posts **UI16** - Easy to learn/Intuitive

RF31 - The archive offers a complete blog submission interface to submit, modify and delete blogs/posts	**FR32** - Add user suggested blogs to the archive **UI34** - Simple submission by authors **UI35** - Workflow to manage blog submissions **FR1** - Deletion by the blog author **OP2** - OAIS **DR21** - Long term digital preservation
RF32 - Users are able to remove their personal data	**DR18** - Remove private data of archive users
RF33 - The archive can display only the very core information for each record	**UI8** - Overview with metadata and summary **UI24** - Display with only core information **UI22** - Density of displayed information
RF34 - The archive displays and suggests similar records to the user	**UI19** - Display similar blogs and posts
RF35 - The archive displays other blogs that were viewed by people who also viewed the current blog	**UI20** - Display blogs that were read by people who have read a specific blog
RF36 - The archive identify and stores the topic of blogs and blog posts to let users navigate through the archive by topic	**FR34** - Topic/Subject detection **UI23** - Categories/Topics are shown in different tabs **FR8** - Topics (Categories) for blogs and blog posts
RF37 - The archive restricts the access to its content to specific IP ranges	**SR2** - Access restricted to IP range
RF38 - Users can communicate within the archive sharing and exchanging resources	**UI30** - Creation of a community of providers and recipients within the archive platform **FR19** - Sharing and collaboration
RF39 - Free open-source archive software	**LR5** - Open source software license is preferable
RF40 - The archive validates the content received from the spider	**RA2** - Correct information in the archive **FR47** - Data integrity **DR21** - Long term digital preservation **OP2** - OAIS
RF41 - The archive detects and eliminates spam content	**FR42** - Weblog content validation and spam filtering
RF42 - The archive extracts bibliographic metadata from content embedded in blogs	**FR30** - Extract bibliographic metadata from blog contents
RF43 - For each record the archive stores the search keywords used to find them	**DR16** - Search key words **UI15** - Search interface
RF44 - The archive enables pingback/-trackback services	**EI3** - Pingback, Trackback
RF45 - The archive is able to interoperate with federated search engine dbwiz (SRU Server)	**IR5** - Connection with federated search engine dbwiz

RF46 - Users can create personal collections of their favorite blogs	**FR20** - Favorite list of blogs and topics **UI16** - Easy to learn/Intuitive
RF47 - Description of how to cite archived records is presented prominently with each record.	**UI5** - Citation is presented prominently **UI16** - Easy to learn/Intuitive
RF48 - The archive provides the option to translate its content on demand	**FR9** - Content translation
RF49 - The archive distinguishes institutional/corporate blogs from personal blogs	**DR19** - Distinguish institutional/corporate blogs from personal blogs
RF50 - The archive offers the option to disseminate newly archived content in external social platforms	**FR33** - Dissemination of newly archived items in external social platforms (ex. Twitter) in connection with author profiles
RF51 - The archive is able to search within external sources	**UI18** - Search in external sources
RF52 - Users can tag archived records with personal tags	**UI32** - Tagging system
RF53 - The archive respects content licenses and displays useful information about them	**DR1** - Rights and Licenses **LR1** - Copyright laws **LR2** - Privacy laws **LR3** - Additional national laws **LR4** - License of the content **DR23** - Mashup activities **FR39** - Digital rights management
RF54 - The archive keeps all the different versions of a record	**OP1** - Versioning **DR21** - Long term digital preservation
RF55 - The archive provides advanced APIs for developers to interact with the archive's content	**OP3** - APIs for developers
RF56 - The archive provides a journal view of the new blog posts	**FR22** - Summaries/Journals about new archive content **UI2** - Magazine/Journal view **UI16** - Easy to learn/Intuitive
RF57 - The archive provides a ranking method based on the user rating of content	**FR11** - Recommendation system **UI31** - Ranking of archived posts **FR27** - Ranking of blogs and blog posts
RF58 - A user can rank archived content based on specific users' content rating	**FR11** - Recommendation system **UI31** - Ranking of archived posts **FR27** - Ranking of blogs and blog posts
RF59 - Export data using XML	**EI1** - API for external clients to query data **EI2** - Data access/export as XML **FR4** - Blog export **IR6** - Facilities to enable interoperability **UI28** - Integration/Combination with other systems

RF60 - The archive can export all its content, database entries and file system for migration	**SM3** - Data export for migration
RF61 - The archive ranks blogs based on their views and downloads	**FR31** - Define important blogs and filter junk **FR27** - Ranking of blogs and blog posts
RF62 - Export records as PDF and JPEG	**FR17** - Print/Export as PDF, JPEG **IR6** - Facilities to enable interoperability **UI28** - Integration/Combination with other systems
RF63 - The archive keeps snapshots of all the different designs of a blog	**FR53** - Snapshot versions of blog designs in the archive
RF64 - The archive offers the option to login using external (universal) credentials	**FR55** - Universal Login and central login
RF65 - The archive analyzes blog links and stores the connections between them separately	**DR9** - Connections/Links
RF66 - The archive provides a historical/chronological blogs navigation	**UI11** - Historical/Chronological view on a blog **UI26** - Historical/Chronological view on blogs combined with corresponding statistics
RF67 - The archive fetches and stores embedded content	**FR54** - What to archive: text and comments **DR14** - Embedded objects
RF68 - The archive provides information diffusion analysis mechanisms	**FR36** - Memetraking and trend detection
RF69 - The archive facilitates searching by providing fuzzy indexing and stemming	**FR38** - Multidimensional indexing **UI15** - Search interface
RF70 - The archive can provide services under some cost using a billing system	**FR40** - Billing system **FR25** - Paid access/Billing system
RF71 - The archive provides a personalized annotating and highlighting tool for users	**UI12** - Annotations and Highlighting
RF72 - The archive provides a visualization of the blogs network structure	**FR18** - Analyze the network structure of blogs **UI9** - Network view on topics, blogs, posts, authors, etc. **UI27** - Dynamic network view on topics, blogs, posts, etc. **EI6** - Export links between blog content
RF73 - The archive recommends blogs to users based on the ratings and preferences	**FR28** - Recommend a cluster of blogs according to user preferences
RF74 - The archive enables/disables certain functionalities based on the content rights	**UI10** - Available services depend on the content rights

RF75 - The archive can do sentiment analysis on the content	**FR21** - Sentiments analysis on blog post level
RF76 - The archive detects content's originality and ranks it accordingly	**FR35** - Detection and ranking of the originality **UI31** - Ranking of archived posts **FR27** - Ranking of blogs and blog posts
RF77 - The archive provides a mobile version	**UI14** - User interface for mobiles **UI16** - Easy to learn/Intuitive
RF78 - The archive displays content after filtering it with user preferences	**UI25** - Filtered, personalized aggregation of content for end-users **FR45** - Personalized filtering services
RF79 - The archive can handle a very large number of content and users	**CS1** - Amount of archived blogs **CS2** - Amount of blog posts per day **CS3** - Amount of users **PR2** - Storage data concurrently **PR1** - Amount of blog posts to capture **CS4** - Clustering and high availability architectures
RF80 - The archive provides mechanisms to control data redundancy	**SP2** - Mechanisms to avoid data loss **RA1** - Recovery of the system **SM1** - Migration/Updating without down time
RF81 - The archive is built based on a modular service-oriented architecture	**CS5** - Modularity
RF82 - The archive can be deployed using a range of different database server technologies	**SP1** - Support for different SQL-data bases
RF83 - The archive provides multiple different views of the archive for each user	**FR24** - User specific collections/projects
RF84 - The archive offers a complete range of search options to the user	**FR16** - Search by author **FR13** - Keyword/Metadata search **FR14** - Full-text search **FR44** - Advanced searching **FR26** - Context-sensitive search by keyword **UI16** - Easy to learn/Intuitive **UI15** - Search interface
RF85 - The archive provides support for OpenURL	**IR7** - Open URL support **IR6** - Facilities to enable interoperability
RF86 - The archive offers functions to edit metadata	**DR17** - Metadata for blogs **DR21** - Long term digital preservation **OP2** - OAIS
RF87 - The archive transforms the SIPS received from the spider to AIPS	**DR17** - Metadata for blogs **DR21** - Long term digital preservation **OP2** - OAIS

RF88 - The archive stores the content of the AIPS in two different databases for preservation purposes	**RA2** - Correct information in the archive **PR2** - Storage data concurrently **DR21** - Long term digital preservation **OP2** - OAIS
RF89 - The archive carries out the normalization and/or migration of the media attachments	**DR21** - Long term digital preservation **OP2** - OAIS

Table 5.1: Features/Requirements

5.2 Feature specifications list

A functional specification does not define the inner workings of the proposed system; it does not include the specification of how the system function will be implemented. Instead, it focuses on what various agents outsides (people using the program, computer peripherals, or other computers, for example) might "observe" when interacting with the system. A typical functional specification might state the following: "When the user clicks the OK button, the dialogue is closed and the focus is returned to the main window in the state it was in before this dialogue was displayed." One of the primary purposes for functional specifications is to decide on what the program is to achieve before making the more time-consuming effort of writing source code and test cases, followed by a period of debugging. In prototypical systems development, functional specifications are typically written after or as part of requirements analysis. After this, typically the software development and testing team write source code and test cases using the functional specification as the reference. While testing is performed, the behaviour of the program is compared against the expected behaviour, as defined in the functional specification.

Therefore, following the methodology described above, once we had defined the whole list of features showed in the previous Section 5.1, we wrote the corresponding functional specification and the expected software behaviour for each of them. In order to simplify the work of writing these specifications and to make their presentation more clear to the reader, a template was designed. This template is presented in Table 5.2.

Field	**Description**
Feature ID	Short feature identifier: RFXX (Repository Feature XX)
Name	One sentence clear enough to make someone who has already read the specification remember the description
Priority	The priority is set taking into consideration the priority of the requirements associated and the use case it is included in. (Possible values: High/Medium/Low)
Effort	Expected implementation time (Possible values: Days/Weeks/-Months)
Components	Invenio modules affected
Requirements	Requirements related to the feature according to table 5.1
Description	High level description of the feature
Logical constraints	Basic high level technical details and logical constraints
Notes and questions	Other comments/suggestions
Assigned to	Person or partner in charge of the development

Table 5.2: Feature specification template

Having said this, in Sections 5.2.1, 5.2.2 and 5.2.3 will be presented each of the features specifications that have been written, listing them by priority:

1. **High priority features**: Implies that the software will not be acceptable unless these features are provided in an agreed manner
2. **Medium priority features**: Implies that these are features that would enhance the software product, but would not make it unacceptable if they are absent
3. **Low priority features**: Implies a class of functions that may or may not be worthwhile. This gives the supplier the opportunity to propose something that exceeds the Software Requirements Specification.

Given that some features were already available in Invenio out-of-the-box or needed only a small customization from the administrator side (but no development), it was decided to fill their templates as follows:

- Since there is no development needed, the "Effort" field will read "Already in Invenio".
- For the same reason, the "Assigned to" field will be empty.
- The "Components" field will reflect roughly which Invenio modules offer this functionality.

5.2.1 High priority features

Feature ID	RF4
Name	Bibformat output templates to display blogs and blog posts differently
Priority	High
Effort	Days
Components	BibFormat
Requirements	DR11 - Differentiate between blog and blog post
Description	A different Bibformat template will be designed and developed to display blogs and blog posts. Therefore, new Bibformat elements should be created to build these templates.
Logical Constraints	The blog template should contain: • Title and url of the corresponding blog • All the blog posts of the blog (with option "Show all posts/Show less posts") The post blog template should contain: • Title of the corresponding blog post • All the comments of the blog post (with option "Show all comments/Show less comments") • Url of the parent blog • Navigation menu containing all the other blog posts that belong to the same blog
Notes and Questions	
Assigned to	CERN

Table 5.3: Feature RF4

Feature ID	RF5
Name	The web interface provides harmonized access and ensures compatibility with major browsers
Priority	High
Effort	Days
Components	BibFormat
Requirements	UI1 - Web Interface FR37 - Web portal FR43 - Access to content in a harmonized way

Description	Blogs and blog posts should have a common way of being displayed since different blogs may have different layouts and different menu structure and this could confuse users.
Logical Constraints	Blogs and blog posts Bibformat templates will be quite similar.
Notes and Questions	
Assigned to	CERN

Table 5.4: Feature RF5

Feature ID	RF6
Name	Latest posts are displayed sorted by addition date
Priority	High
Effort	Weeks
Components	WebSearch, WebColl
Requirements	UI6 - Latest posts
Description	The goal of this feature is to create general infrastructure where latest addition lists can be made easily customizable by administrators.
Logical Constraints	• We can have various latest addition listing algorithm living as small Python files in a plugin directory such as “Sort by addition date”, “Sort by title”, etc. • New database table “collection_latestadditions” will permit admins to decide which lister plugin will be used for which collection • WebSearch Admin UI should be enhanced to allow such configuration
Notes and Questions	
Assigned to	CERN

Table 5.5: Feature RF6

Feature ID	RF7
Name	Export data using the OAI-PMH protocol
Priority	High
Effort	Already in Invenio
Components	OAIHarvest
Requirements	IR4 - Expose parts of the archive via OAI-PMH based on specified criteria IR6 - Facilities to enable interoperability

Description	The repository metadata will be exported using the OAI-PMH protocol. Other machines will be able to download metadata by giving the appropriate commands using an HTTP request.
Logical Constraints	An XSLT (Extensible Stylesheet Language Transformations) template to make the XML output more human-readable will be used when displaying the metadata in a browser.
Notes and Questions	The metadata will be able to be exported in both MARC and DC formats. Example of how to use it: http://cdsweb.cern.ch/oai2d?verb=ListRecords&metadataPrefix=oai_dc
Assigned to	

Table 5.6: Feature RF7

Feature ID	RF8
Name	Export data in Dublin Core schema
Priority	High
Effort	Already in Invenio
Components	BibExport, BibConvert
Requirements	IR3 - Export data using OAI-PMH protocol and Dublin Core schema IR6 - Facilities to enable interoperability
Description	The repository will be able to transform the metadata to Dublin Core schema and include it as an export option in the web interface as well as in OAI-PMH (see RF7)
Logical Constraints	There has to be a mapping between the MARC tags and Dublin Core tags.
Notes and Questions	
Assigned to	

Table 5.7: Feature RF8

Feature ID	RF9
Name	The archive stores and displays accordingly all record metadata received from the spider
Priority	High
Effort	Weeks
Components	BibUpload, BibFormat
Requirements	DR17 - Metadata for blogs DR21 - Long term digital preservation

Description	A pre-ingestion plugin should be implemented in order to extract the MARC metadata embedded in the METS files produced by the spider, and to enrich this MARC metadata with the schema used by Invenio. This pre-ingestion process will be executed before calling BibUpload. Once BibUpload finishes, all the blog metadata are stored into the repository and ready to be displayed to the user.
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.8: Feature RF9

Feature ID	RF10
Name	Archive user passwords are stored encrypted in the database
Priority	High
Effort	Already in Invenio
Components	WebSession, WebAccess
Requirements	SR1 - Passwords are stored encrypted
Description	User passwords should be stored into the database encrypted using some industry-standard cipher, such as the Message-Digest Algorithm 5 (MD5).
Logical Constraints	
Notes and Questions	
Assigned to	

Table 5.9: Feature RF10

Feature ID	RF11
Name	The web interface is available in many different languages
Priority	High
Effort	Already in Invenio
Components	MiscUtil
Requirements	FR46 - Internationalization UI29 - Multiple language support
Description	The BlogForever repository should be support all the European Languages, therefore, users can easily switch to the language of their choice in the interface.
Logical Constraints	
Notes and Questions	
Assigned to	

Table 5.10: Feature RF11

Feature ID	RF12
Name	The archive can import METS
Priority	High
Effort	Days
Components	BibUpload
Requirements	DR22 - METS IR6 - Facilities to enable interoperability DR21 - Long term digital preservation OP2 - OAIS
Description	The archive should be able to import the METS files produced by the spider (see RF9)
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.11: Feature RF12

Feature ID	RF13
Name	UTF-8 is used as the default character encoding in the archive
Priority	High
Effort	Already in Invenio
Components	All modules
Requirements	FR51 - UTF8 the default character encoding DR21 - Long term digital preservation
Description	The default character encoding used in the archive should be UTF-8 (UCS Transformation Format—8-bit) in order to be able to represent every character in the Unicode character set.

Logical Constraints	
Notes and Questions	
Assigned to	

Table 5.12: Feature RF13

Feature ID	RF15
Name	Option to disseminate archive content in major social web platforms
Priority	High
Effort	Already in Invenio
Components	WebStyle
Requirements	FR7 - User dissemination channels for blog post UI16 - Easy to learn/Intuitive UI28 - Integration/Combination with other systems
Description	In the detailed view of a records there will be a series of buttons to allow users to share a link to the record they are viewing.
Logical Constraints	At least the following sharing options will be available: email, LinkedIn, Twitter, Facebook, Google and del.icio.us The respective APIs of the different platforms will be used in the implementation.
Notes and Questions	
Assigned to	

Table 5.13: Feature RF15

Feature ID	RF16
Name	The archive offers an RSS channel of its latest updates and/or users can receive notification when new content of their interest is added to the archive
Priority	High
Effort	Already in Invenio
Components	
Requirements	FR12 - Notification about changes in the archive UI16 - Easy to learn/Intuitive UI28 - Integration/Combination with other systems

Description	A user of the archive should be able to subscribe to blogs or topics s/he is interested in, and gets notified when new content is added to those blogs and/or topics. Invenio already offers an RSS feed per collection, so users are notified when new content comes to any of the collections they subscribed to. Therefore, for topics it should be exactly the same, there will be an RSS feed per topic (which actually are collections), in a way that users could subscribe to them and then receive notifications when new content is added to any of those topics. On the other hand, in the case of blogs, there will not be an RSS feed per blog, but users will be able to set alerts specifying the blog/s from they would like to receive notification of latest updates.
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.14: Feature RF16

Feature ID	RF17
Name	The archive displays a disclaimer about the originality of the content
Priority	High
Effort	Days
Components	WebStyle
Requirements	UI21 - Archived content is clearly stated as such DR3 - Disclaimer UI16 - Easy to learn/Intuitive
Description	A disclaimer will be displayed in every detailed record page saying that the corresponding blog, post, page or comment, is just an archived copy of the original one.
Logical Constraints	A link to the the original element (Blog, Blog Post, Comment, Page) will be displayed.
Notes and Questions	
Assigned to	CERN

Table 5.15: Feature RF17

Feature ID	RF18
Name	The repository detects duplicated content and keeps only one copy
Priority	High
Effort	Days
Components	BibMatch

Requirements	FR23 - Detection of duplicates DR21 - Long term digital preservation
Description	Duplicates of any content will be automatically detected and only one copy of this content will be kept in the repository. This will be done by re-using the existing command-line tool in Invenio called BibMatch. BibMatch matches bibliographic data in a MARCXML file against the database content. With a MARCXML input file, the produced output shows a selection of records in the input that matches the database content. Therefore, it is possible to identify potential duplicate entries before they are uploaded in a database. This tool offers flexible matching options such as: 1. print-new, (the default behaviour) which will print out unmatched records 2. print-match, which will print out matched records 3. print-ambiguous, which will print out records that matches more than one existing record 4. print-fuzzy, which will print out records that match the longest words in existing records In our case we will have a BibMatch daemon that will be run in the background periodically in a way that duplicates are detected and only one copy of these records is kept into the repository.
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.16: Feature RF18

Feature ID	RF19
Name	The archive can be indexed by external search engines
Priority	High
Effort	Already in Invenio
Components	WebSearch, Bibindex
Requirements	EI4 - Accessible via search machines
Description	In order to facilitate external search engines indexing, several techniques are recommended.

Logical Constraints	• Expose the metadata, using direct links in the record detailed view. • Use site maps that let the external search engine better understand the structure of the web site. • Include tags like GoogleScholar in the header.
Notes and Questions	
Assigned to	

Table 5.17: Feature RF19

Feature ID	RF20
Name	The archive's statistics are exported as CSV
Priority	High
Effort	Already in Invenio
Components	WebStat
Requirements	EI5 - Export as CSV UI28 - Integration/Combination with other systems
Description	The usage statistics associated to the collections should be computed and exported as CSV (Comma-Separated Values) files.
Logical Constraints	The platform should be able to create usage data associated to the collections and be able to display the datasets in a standardized and reliable way onto CSV files. The users are able to download the corresponding CSV file from the repository interface.
Notes and Questions	Only for admin statistics
Assigned to	

Table 5.18: Feature RF20

Feature ID	RF23
Name	The archive stores the comments of blog posts and displays them as part of the blog posts
Priority	High
Effort	Days
Components	BibFormat
Requirements	DR13 - Comments DR21 - Long term digital preservation FR54 - What to archive: text and comments

Description	The blog post bibformat template should be enriched with a new bibformat element in order to obtain all the comment records of a specific blog post and, therefore, display them to the user together with the specific blog post.
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.19: Feature RF23

Feature ID	RF25
Name	The archive displays the tags of blogs and blog posts
Priority	High
Effort	Days
Components	BibFormat
Requirements	UI7 - Tags for blogs and blog posts
Description	The blog post and blog bibformat templates should be enriched with a new bibformat element in order to obtain all the tags associated to them.
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.20: Feature RF25

Feature ID	RF26
Name	BlogUploader command line to upload, update and delete a list of blogs
Priority	High
Effort	Days
Components	WebBlog
Requirements	FR15 - Selection of blogs to archive UI16 - Easy to learn/Intuitive
Description	A new command line tool should be implemented in order to let admins to edit a list of blogs and to insert, delete or update them into the archive.

Logical Constraints	• The following modes should be offered: – insert list of blogs: -i, –blog_insert – delete list of blogs: -d, –blog_delete – update list of blogs: -U, –blog_update • The input file could be a CSV file where the data to be included depends on the mode: – Insert: url,[title],topic,license – Delete: url – Update: url,[title],topic,license
Notes and Questions	
Assigned to	CERN

Table 5.21: Feature RF26

Feature ID	RF27
Name	The archive displays a unique URL (DOI) for each record
Priority	High
Effort	Already in Invenio
Components	BibUpload, MiscUtil
Requirements	DR2 - URI and metadata for referencing/citing IR8 - Digital Object Identifier IR6 - Facilities to enable interoperability
Description	Digital Object Identifier (DOI) is a permanent identifier given to an electronic resource that, in contrast to a URL, does not depend on the electronic document's location.
Logical Constraints	Records will be stamped with DOIs in their metadata, and this information will be ready to be indexed allowing users and external machines to search digital objects by DOI.
Notes and Questions	The MARC tag to be used may be the 0247a.
Assigned to	

Table 5.22: Feature RF27

Feature ID	RF28
Name	The archive displays the author of blog posts and comments
Priority	High
Effort	Days

Components	BibFormat
Requirements	DR4 - Display the author of the blog, blog post, comment
Description	The blog post and comment bibformat templates should be enriched with a new bibformat element in order to display the author.
Logical Constraints	The author will be displayed as a link in such way that a search by author is triggered by clicking on it.
Notes and Questions	
Assigned to	CERN

Table 5.23: Feature RF28

Feature ID	RF29
Name	The archive alerts the user when there are software updates
Priority	High
Effort	Days
Components	WebSession
Requirements	SM2 - Software updates
Description	When the system is updated, the users should be informed about the update and they should be able to see change-log.
Logical Constraints	• A warning about system's last update will appear on the users' your account page when the users log in. • The change-log will be displayed in your account page too or there will be a link that directs the users to the page displaying the details of the update. • The changes will be specified in a manner that the user will be able to understand easily what are the changes.
Notes and Questions	
Assigned to	SRDC

Table 5.24: Feature RF29

Feature ID	RF39
Name	Free open-source archive software
Priority	High
Effort	Already in Invenio

Components	All modules
Requirements	LR5 - Open source software license is preferable
Description	The BlogForever repository should be licensed under an open source software license license. This should be stated on the website and in the source code.
Logical Constraints	
Notes and Questions	
Assigned to	

Table 5.25: Feature RF39

Feature ID	RF40
Name	The archive validates the content received from the spider
Priority	High
Effort	Days
Components	BibIngest
Requirements	RA2 - Correct information in the archive FR47 - Data integrity DR21 - Long term digital preservation OP2 - OAIS
Description	The repository stores the information correctly as users can found it in the original blog. In order to be sure that this is done properly, the repository will validate all the information coming from the spider by using the cryptographic hash function MD5 (Message-Digest Algorithm 5). In case the repository finds that any content coming from the spider has not the same MD5 hash, an alert will be sent to the spider communicating that something was wrong and the content should be sent again to the repository.
Logical Constraints	The validation will be done at pre-ingestion time.
Notes and Questions	
Assigned to	CERN

Table 5.26: Feature RF40

Feature ID	RF41
Name	The archive detects and eliminates spam content
Priority	High
Effort	Months
Components	BibUpload
Requirements	FR42 - Weblog content validation and spam filtering

Description	The repository checks new records to specify if they can be classified as spam. The record URI is used for checking against popular online services providing an API for this purpose. If a record is identified as spam, it is flagged by adding a specific element to its metadata. The following 3rd party services will be used for spam identification: • http://www.spamhaus.org/ • http://www.phishtank.com/ • http://www.uribl.com/ • http://www.surbl.org/
Logical Constraints	The classification will take place only once, right after ingestion
Notes and Questions	
Assigned to	AUTH

Table 5.27: Feature RF41

Feature ID	RF47
Name	Description of how to cite archived records is presented prominently with each record.
Priority	High
Effort	Days
Components	BibFormat
Requirements	UI5 - Citation is presented prominently UI16 - Easy to learn/Intuitive
Description	A user needs to link and cite the content of the archive. Therefore, the way how to link and how to cite a record should be presented prominently with the content.

Logical Constraints	• A new bibformat element should be created which will display the description of how users should cite any content of the archive. • Citation description for blogs will read: "title"(record_creation_date). record_url Retrieved from the original "original_url" • Citation description for blog posts will read: author. "title". Blog: "blog_title". (record_creation_date). record_url Retrieved from the original "original_url" • Citation description for comments will read: author. Blog post: "blog_title". (record_creation_date). record_url Retrieved from the original "original_url"
Notes and Questions	
Assigned to	CERN

Table 5.28: Feature RF47

Feature ID	RF53
Name	The archive respects content licenses and displays useful information about them
Priority	High
Effort	Weeks
Components	WebSearch, BibFormat, WebBlog
Requirements	DR1 - Rights and Licenses LR1 - Copyright laws LR2 - Privacy laws LR3 - Additional national laws LR4 - License of the content DR23 - Mashup activities FR39 - Digital rights management
Description	The system should respect any content license information. Therefore, the archive should detect, store, preserve and display to the user all digital rights and authority information on the blog content.

Logical Constraints	• License information should be displayed to users every time they views a blog detailed record • The user will be able to view copyright information of the content • The archive will follow copyright laws on both international and national level • The archive will follow privacy and data protection laws • The archive will respect any additional laws of organizations or authorities in the country in which the archive is running • The archive will respect the license under which the content is published • Content from other sources should be identified and associated with a disclaimer around copyright infringement to protect the archive and hosting institution, which can be easily found by the user • The platform will support a fully functional and robust Digital Rights Management (DRM) system • The system will support Open Digital Rights Language (ODRL), which is supported by the W3C
Notes and Questions	List of licenses to be discussed. Suggestion: Creative Common Licenses, http://creativecommons.org/licenses/
Assigned to	CERN

Table 5.29: Feature RF53

Feature ID	RF54
Name	The archive keeps all the different versions of a record
Priority	High
Effort	Days
Components	BibIngest
Requirements	DR21 - Long term digital preservation OP1 - Versioning
Description	Versioning will be enabled for every data object stored in the digital repository. When any data object is modified, a new version of it will be kept in the ingestion database storage. Therefore, the repository will store all the different versions of all the data objects.

Logical Constraints	• This feature should be covered within the BibIngest module by adding a new parameter called version to the usage settings. • As ingestion database storage we could use mongoDB.
Notes and Questions	
Assigned to	CERN

Table 5.30: Feature RF54

Feature ID	RF55
Name	The archive provides advanced APIs for developers to interact with the archive's content
Priority	High
Effort	Already in Invenio
Components	All modules
Requirements	OP3 - APIs for developers
Description	The repository should provide APIs for developers to interact with weblog data in order to read and write data, configure the repository and perform maintenance actions.
Logical Constraints	
Notes and Questions	
Assigned to	

Table 5.31: Feature RF55

Feature ID	RF66
Name	The archive provides a historical/chronological navigation
Priority	High
Effort	Weeks
Components	WebSearch
Requirements	UI11 - Historical/Chronological view on a blog UI26 - Historical/Chronological view on blogs combined with corresponding statistics
Description	The user will be able to navigate a blog's content and layout through its timeline and display it as it was in a specific date. The navigation will be possible through a slider.

Logical Constraints	• The user will be able to navigate a blog's timeline using a slider that is provided in the detailed record page • When the user adjust the slider in a particular date, the system will display: – blog posts – comments that were published at that date and the corresponding snapshot of the layout of blog
Notes and Questions	
Assigned to	CERN

Table 5.32: Feature RF66

Feature ID	RF69
Name	The archive facilitates searching by providing fuzzy indexing and stemming
Priority	High
Effort	Months
Components	BibIndex, WebSearch
Requirements	FR38 - Multidimensional indexing UI15 - Search interface
Description	The user should be able to retrieve any record that is likely to be relevant to his search arguments. The results should be records that contain either the exact arguments or similar spelling variations.
Logical Constraints	• The user should have the option to rank the displayed returned results according to the similarity between the search arguments and the words contained in results • The user should have the option to search intentionally a truncated term in order to retrieve records that contain variations of this term
Notes and Questions	
Assigned to	CERN

Table 5.33: Feature RF69

Feature ID	RF70
Name	The archive can provide services under some cost using a billing system
Priority	High
Effort	Weeks
Components	WebAccess, WebSession
Requirements	FR25 - Paid access/Billing system FR40 - Billing system
Description	The admin should be able to restrict the collections with an infinite/finite time interval for some cost. The users should be able to buy the right to access restricted collections with their credit cards or PayPal accounts.
Logical Constraints	• The admin will be able to create premium packages which contain right to access to one or more collections with some time interval for some cost. • The users will not access the restricted collections without purchasing corresponding premium package.
Notes and Questions	
Assigned to	SRDC

Table 5.34: Feature RF70

Feature ID	RF79
Name	The archive can handle a very large number of content and users
Priority	High
Effort	Already in Invenio
Components	All modules
Requirements	CS1 - Amount of archived blogs CS2 - Amount of blog posts per day CS3 - Amount of users PR2 - Storage data concurrently PR1 - Amount of blog posts to capture CS4 - Clustering and high availability architectures

Description	The number of blogs can vary from some thousands up to one million or more. Therefore, the archive should be able to handle large amount of archived blogs. Moreover, the archive should be able to handle large amount of daily added blog posts. The number of the users increases with the increase of the number of blogs/blog posts, as well. The archive should also be able to handle large amount of users.
Logical Constraints	The performance of the archive will not be effected by the number of users and the blogs/blog posts.
Notes and Questions	
Assigned to	

Table 5.35: Feature RF78

Feature ID	RF80
Name	The archive provides mechanisms to control data redundancy
Priority	High
Effort	Already in Invenio
Components	MiscUtil
Requirements	SP2 - Mechanisms to avoid data loss RA1 - Recovery of the system SM1 - Migration/Updating without down time
Description	The data should be recoverable. To avoid data loss there should be periodic or manual backups. A mechanism that minimizes data redundancy due to backups should be implemented.
Logical Constraints	The administrator will be able to specify the period of the backups. The administrator will be able to manually backup the system. The administrator will be able to recover the archive by selecting a backup. There may be an option to backup/recover specific content rather than the whole.
Notes and Questions	To avoid data redundancy, an incremental backup strategy may be applied.
Assigned to	

Table 5.36: Feature RF80

Feature ID	RF81
Name	The archive is built based on a modular service-oriented architecture
Priority	High
Effort	Already in Invenio
Components	All modules

Requirements	CS5 - Modularity
Description	The system architecture should be divided in modules with clear boundaries and well documented interfaces between them. A module should be easily extracted from the system or inserted to the system.
Logical Constraints	How to insert a new module or create an API on the system will be defined clearly.
Notes and Questions	
Assigned to	

Table 5.37: Feature RF81

Feature ID	RF84
Name	The archive offers a complete range of search options to the user
Priority	High
Effort	Already in Invenio
Components	WebSearch
Requirements	FR16 - Search by author FR13 - Keyword/Metadata search FR14 - Full-text search FR44 - Advanced searching FR26 - Context-sensitive search by keyword UI16 - Easy to learn/Intuitive UI15 - Search interface
Description	The repository will include a very fast search engine optimized for large repositories (millions of documents) on simple infrastructures, combining metadata and fulltext search in a simple Google-like query language. Advanced users will be also given the opportunity to perform advanced queries, such as find document written by Ellis from years 2000 to 2010, mentioning “higgs boson” in the fulltext, referring to documents written by Randall, and cited more than 50 times.

Logical Constraints	Search modes: • Simple search: The default search mode is simple search that basically provides users with one input box where users can type their query, followed by a possibility to choose one of the common indexes to search within. • Advanced search: The advanced search interface provides users with explicit tools to play with such as change the matching type from the default word matching to phrase searching or the regular matching, use boolean queries in several indexes, etc. Searching techniques: • Ranges: “->”, e.g.: muon decay year:1983->1992 • Combined searches (metadatafulltext) • Regular expressions
Notes and Questions	
Assigned to	

Table 5.38: Feature RF84

Feature ID	RF85
Name	The archive provides support for OpenURL
Priority	High
Effort	Already in Invenio
Components	WebSearch
Requirements	IR7 - Open URL support IR6 - Facilities to enable interoperability
Description	OpenURL is a standardized format of Uniform Resource Locator (URL) intended to enable Internet users to more easily find a copy of a resource that they are allowed to access. The OpenURL standard is designed to enable linking from information resources such as abstracting and indexing databases (sources) to library services (targets), such as academic journals, whether online or in printed or other formats. The linking is mediated by ”link resolvers”, or ”link-servers”, which parse the elements of an OpenURL and provide links to appropriate targets available through a library by the use of an OpenURL knowledge base. The repository should offer an OpenURL link resolver to resolves OpenURL to content directly hosted in the corresponding repository.

Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.39: Feature RF85

Feature ID	RF86
Name	The archive offers functions to edit metadata
Priority	High
Effort	Already in Invenio
Components	BibEdit
Requirements	DR17 - Metadata for blogs DR21 - Long term digital preservation OP2 - OAIS
Description	Content managers are able to edit and enhance metadata submitted to the repository by the content provider and/or the weblog spider. End-users are able to submit missing metadata and corrections to existing metadata. Metadata submitted by end-users, content managers, and content providers will be distinguished, stored, and managed accordingly to meet the policies of the repository.
Logical Constraints	When metadata is edited the database must be updated. The history of metadata edit to be recorded as provenance metadata in METS. This feature is inter-linked with RF89
Notes and Questions	The repository must determine how to represent, validate and integrate the three different types of metadata edit.
Assigned to	

Table 5.40: Feature RF86

Feature ID	RF87
Name	The archive transforms the SIPS received from the spider to AIPS
Priority	High
Effort	Weeks
Components	BibUpload, BibIngest
Requirements	DR17 - Metadata for blogs DR21 - Long term digital preservation OP2 - OAIS

Description	The content of the submitted blog must be transformed into an Archival Information Package (AIP). The AIP consists of the following: 1. Parsed blog content. 2. Raw HTML from the blog. 3. Categorised content of the blog. 4. Reference information. 5. Descriptive metadata. 6. Rights metadata. 7. Provenance metadata. 8. Preservation metadata. 9. Fixity information. 10. Packaging information.
Logical Constraints	1. The parsed blog is delivered as XML. 2. The reference information provided as a permanent URI. 3. The technical metadata is expected to be delivered using the schemas recommended in Section 3.2. 4. Descriptive metadata is delivered using MAR-CXML. 5. provenance and preservation metadata including the fixity information is delivered using PREMIS and PREMISRights. 6. The package is delivered using METS.
Notes and Questions	The metadata schemas, controlled vocabulary, syntax, and encoding is further discussed in Section 3.2 and the deliverable D3.1 [11](Chapter 5 and METS profile and example drafts included in the Appendix).
Assigned to	CERN

Table 5.41: Feature RF87

Feature ID	RF88
Name	The archive stores the content of the AIPS in two different databases for preservation purposes
Priority	High
Effort	Weeks
Components	BibIngest
Requirements	RA2 - Correct information in the archive PR2 - Storage data concurrently DR21 - Long term digital preservation OP2 - OAIS
Description	A copy of the MARC metadata (i.e. the descriptive information) would be stored in the "Main Storage Database" where it would be processed in order to extract information and retained for further processing and output. The METS file as submitted would be stored in a separate "Ingestion Database" for preservation purposes.
Logical Constraints	The information must be updated when metadata is edited (see RF86).
Notes and Questions	
Assigned to	CERN

Table 5.42: Feature RF88

Feature ID	RF89
Name	The archive carries out the normalization and/or migration of the media attachments
Priority	High
Effort	Weeks
Components	BibUpload
Requirements	DR21 - Long term digital preservation OP2 - OAIS
Description	The content manager must be able to specify selected formats for conversion to target formats (see RF90 and Section 3.2) included in blog records. The repository must offer the ability to carry out migration on access: that is, by HTTP content-negotiation by clients accessing the content. Additionally, ideally, the repository should provide the possibility of carrying out at two additional points: at ingest, as a batch process after ingest.

Logical Constraints	The system must be able to re-direct content specified by the content manger for conversion at ingest. The system must be able to redirect content stored in selected formats specified by the content manager for conversion. The system must be able to redirect stored content for conversion to requested formats upon end-user access. The system administrator must be able to specify the tools for converting these formats to target formats.
Notes and Questions	
Assigned to	CERN

Table 5.43: Feature RF89

5.2.2 Medium priority features

Feature ID	RF1
Name	Customizable user dashboard
Priority	Medium
Effort	Weeks
Components	WebSession
Requirements	UI13 - Customizable user dashboard
Description	The current user informative dashboard of Invenio (called “Your Account”) will be updated to customizable dashboard. The user will be able to permanently choose what information is visible in his/ her dashboard, and if possible in what order. Available information will include Invenio features such as “Your Baskets”, “Your Searches”, “Your Messages” etc. It should be relatively easy for future developers to add new available features.
Logical Constraints	• The user will be able to choose his/her available features from a drop down menu. • For each feature the user should be able to: – add or remove it – (possibly) adjust its position – (possibly) adjust its size • The user’s preferences should be stored permanently
Notes and Questions	
Assigned to	SRDC

Table 5.44: Feature RF1

Feature ID	RF2
Name	"Your History" box as part of the user dashboard
Priority	Medium
Effort	Days
Components	WebSession
Requirements	UI3 - History of own activities in the archive
Description	The user should have the option to add the feature: "Your Activity" to his/her customizable dashboard. This feature will gather and display information about the latest user's activities within the repository. These will include activities such as: user searches, user actions on baskets, user messages etc.
Logical Constraints	• The user will be able to add this feature from a drop down menu in his/her dashboard. • The user should be able to – add this feature – remove this feature – (possibly) adjust the feature's position in the dashboard • The user's preference about this feature should be stored permanently
Notes and Questions	
Assigned to	SRDC

Table 5.45: Feature RF2

Feature ID	RF3
Name	"Share" option in "Your History" box
Priority	Medium
Effort	Days
Components	WebSession
Requirements	UI4 - Subscribe and Navigate activities of other users
Description	The user should be able to share his activities within the archive with other users. Through the "Your Activity" feature in the user's customizable dashboard, the option "Share..." should be available, either as an icon, or a link. Choosing to share an activity the user should be able to send a message to any other user within the archive, mentioning the said activity.

Logical Constraints	• The user will be able to click on the "Share..." option, for each activity • The "Share..." option will open a dialog where the user will be able to choose to whom he/she can share the chosen activity with.
Notes and Questions	
Assigned to	SRDC

Table 5.46: Feature RF3

Feature ID	RF14
Name	Descriptive statistics are offered by record
Priority	Medium
Effort	Weeks
Components	WebSession
Requirements	FR3 - Descriptive statistics for the archive DR15 - Visits of blogs and blog posts FR5 - Descriptive statistics for a single blog or blog post DR12 - Demographics
Description	For each record, there should be descriptive statistics which are: • the number of visitors, downloads and comments • overall rating • who visited record and how often? • list of records viewed by the users who visited this record • which search keywords lead to the record For generic (platform related) statistics, there should be the following: • amount of records and categories • the number of users, total page view • most popular records, groups, baskets • demographic statistics The usage statistics of each record will be extended to contain specified categories above. To gather demographic statistic, the registration form will become more detailed (e.g., nationality, sex, age, profession)

Logical Constraints	• The user will be able to see the statistics from the "Usage Statistics" tab in each record. • The user will be able to reach the generic statistics by visiting "Platform Statistics" page.
Notes and Questions	
Assigned to	SRDC

Table 5.47: Feature RF14

Feature ID	RF21
Name	The archive offers the option to login using SSO/LDAP
Priority	Medium
Effort	Already in Invenio
Components	WebAccess
Requirements	IR1 - Single Sign On/Interoperates with Authentication System especially LDAP
Description	The single sign-on (SSO) property provides access control to multiple related systems. The secure access to a Lightweight Directory Access Protocol (LDAP) repository is required. The system should provide the corresponding LDAP authentication.
Logical Constraints	The user logs in once and gains access to all systems without being prompted to log in again at each of them.
Notes and Questions	Internal interface requirement.
Assigned to	

Table 5.48: Feature RF21

Feature ID	RF22
Name	"Your Preferences" box as part of the user dashboard
Priority	Medium
Effort	Days
Components	WebSession
Requirements	UI33 - User profiles UI16 - Easy to learn/Intuitive
Description	Each user in the system should have a user profile and is able to access this profile via a personal user page. From the user page, the users should be able to access all user related pages and set user preferences.

Logical Constraints	• The user page will be user friendly and consist of boxes each of which will be related with one user action (messages, alerts, searches, history, etc.) • User preferences and user profile will be taken into consideration for some actions like similar blog/record recommendation, user history.
Notes and Questions	Maybe merge with RF1. Another box in the user dashboard that shows recommended content.
Assigned to	SRDC

Table 5.49: Feature RF22

Feature ID	RF24
Name	Links to other sources within blog posts and comments are displayed separately
Priority	Medium
Effort	Days
Components	BibFormat
Requirements	UI17 - Display references (links) to other sources inside or outside the archive UI16 - Easy to learn/Intuitive
Description	The user interface will display in a clear way the links used as references when showing a blog post.
Logical Constraints	• Creates on the detailed record page of every post a menu containing all the reference links within each blog post. • Offers a link to the corresponding record when a reference link points to an archived record.
Notes and Questions	
Assigned to	CERN

Table 5.50: Feature RF24

Feature ID	RF30
Name	Users are able to bookmark records, also using external bookmarking engines
Priority	Medium
Effort	Already in Invenio
Components	WebBasket

Requirements	FR10 - Bookmarking of blog posts UI16 - Easy to learn/Intuitive
Description	There should be a tool that would allow repository users to bookmark content that they are currently viewing. As well this tool should allow the bookmarking of the current document within other external social bookmarking services. The integration of the repository with the world of social bookmarking is provided.
Logical Constraints	The user should click on the provided links to related content, either within the repository or elsewhere on the web, as specified by the provided bookmarking service.
Notes and Questions	
Assigned to	

Table 5.51: Feature RF30

Feature ID	RF31
Name	The archive offers a complete blog submission interface to submit, modify and delete blogs/posts
Priority	Medium
Effort	Weeks
Components	WebSubmit
Requirements	FR32 - Add user suggested blogs to the archive UI34 - Simple submission by authors UI35 - Workflow to manage blog submissions FR1 - Deletion by the blog author OP2 - OAIS DR21 - Long term digital preservation
Description	The archive will offer a complete submission interface to let users and admins to submit nee blogs, to modify certain specific metadata of a blog, and to delete either a blog (as result all its comments and blog posts will be deleted) or a single blog post.

Logical Constraints	• Adds action on detailed record page to let users to delete a blog or post record. ("Ask for Deletion") • Adds action on detailed record page to let users to modify blog metadata. ("Ask for Modification") • Creates blog submission interface for users (refereed) and admins: – Blog Submission Actions (for admins): Submit a Blog, Modify a Blog, Delete a Blog, Delete a Post – Blog Submission (Refereed) Actions (for users): Submit a Blog, Approve Blog Submission, Modify a Blog, Approve Blog Modification, Delete a Blog, Approve Blog Deletion, Delete a Post, Approve Post Deletion • Creates two new collections in order to manage the submission Provisional Blogs (contains the submitted blogs), Rejected Blogs (contains the rejected blogs)
Notes and Questions	
Assigned to	CERN

Table 5.52: Feature RF31

Feature ID	RF32
Name	Users are able to remove their personal data
Priority	Medium
Effort	Days
Components	WebSession, WebAccess
Requirements	DR18 - Remove private data of archive users
Description	The users should be able to deactivate/activate their accounts. The users should be able to completely remove their accounts.

Logical Constraints	• The users will access this feature from their “Your Account” page. • Password confirmation will be required to deactivate or remove the account. • If the user selects to remove her/his personal data too, database entries only related with that user should also be deleted. However, entries related with other users too such as messages, comments, etc. should be kept. • If the user selects to keep her/his personal data, s/he will have the option of activation of her/his account. • When the user tries to log in with an inactive account credentials, the system will simply accepts the user and load her/his personal data. • There may be an option that enables/disables the option of suspension of the accounts.
Notes and Questions	
Assigned to	SRDC

Table 5.53: Feature RF32

Feature ID	RF33
Name	The archive can display only the very core information for each record
Priority	Medium
Effort	Already in Invenio
Components	BibFormat
Requirements	UI8 - Overview with metadata and summary UI22 - Density of displayed information UI24 - Display with only core information
Description	The user can adjust the density of the displayed information. A user can see a blog post free of redundant elements like advertisements, so only the text and images/videos of it will be displayed. This overview should show at least the title, author, amount of comments, and a brief summary of the content (Brief format).
Logical Constraints	
Notes and Questions	
Assigned to	

Table 5.54: Feature RF33

Feature ID	RF34
Name	The archive displays and suggests similar records to the user
Priority	Medium
Effort	Days
Components	BibClassify
Requirements	UI19 - Display similar blogs and posts
Description	When displaying a post, links to other posts on the same subject should be displayed as suggestions. In a blogs page exists a component with similar blogs (blogs with analogous topics). Suggestions of similar blogs and posts are showed when displaying a blog or post “View similar records“ in the detailed records page.
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.55: Feature RF34

Feature ID	RF35
Name	The archive displays other blogs that were viewed by people who also viewed the current blog
Priority	Medium
Effort	Days
Components	BibRank
Requirements	UI20 - Display blogs that were read by people who have read a specific blog
Description	The repository will display to the user the list of blogs that were viewed for people who also viewed the current blog. With each viewed blog will be showed the number of different people who viewed it.
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.56: Feature RF35

Feature ID	RF36
Name	The archive identify and stores the topic of blogs and blog posts to let users navigate through the archive by topic
Priority	Medium
Effort	Weeks
Components	BibUpload, WebSearch

Requirements	FR34 - Topic/Subject detection UI23 - Categories/Topics are shown in different tabs FR8 - Topics (Categories) for blogs and blog posts
Description	Users are presented with the option of viewing content under different categories. Users should be presented with the appropriate tool to select from a list of predetermined categories to output filtered blog content. Displaying all the posts from a category on a page could help visitors of the archive to find older blogs content.
Logical Constraints	The specific categories are created internally via templates or plugins.
Notes and Questions	
Assigned to	CERN

Table 5.57: Feature RF36

Feature ID	RF37
Name	The archive restricts the access to its content to specific IP ranges
Priority	Medium
Effort	Already in Invenio
Components	WebAccess, WebSearch
Requirements	SR2 - Access restricted to IP range
Description	Access to the platform content should be controlled by the use of sets of IP addresses
Logical Constraints	The BlogForever platform should be responsible for issuing and terminating control, verifying the status of authorized users, providing lists of valid sets of IP addresses if applicable, and updating such lists on a regular basis.
Notes and Questions	
Assigned to	

Table 5.58: Feature RF37

Feature ID	RF38
Name	Users can communicate within the archive sharing and exchanging resources
Priority	Medium
Effort	Already in Invenio
Components	WebSession, WebMessage, WebComment
Requirements	UI30 - Creation of a community of providers and recipients within the archive platform FR19 - Sharing and collaboration

Description	The users should able to use share button from different pages of Invenio to share records, comments, etc.. In addition, a user can share his/her basket with specific users, not only share with a group or public, which is currently available.
Logical Constraints	• The users will be able to click a "Share" button, for each content. • The users will be able to invite specific users to access his/her basket with specified access rights.
Notes and Questions	RF3 can be extended
Assigned to	

Table 5.59: Feature RF38

Feature ID	RF42
Name	The archive extracts bibliographic metadata from content embedded in blogs
Priority	Medium
Effort	Days
Components	BibIngest
Requirements	FR30 - Extract bibliographic metadata from blog contents
Description	The BlogForever archive will extract bibliographic metadata (e.g. Title, Author) from PDF documents, LaTeX files and image files that are attached to blogs, or embedded in the post of a blog. The extraction will be done using reference management software, such as EndNote, RefWorks, or BibTeX which are already supported in Invenio. The extracted metadata will be stored in the database and used to populate an index. Users will be able to search and browse the metadata for the attachments, using a special web interface.
Logical Constraints	
Notes and Questions	
Assigned to	CERN

Table 5.60: Feature RF42

Feature ID	RF43
Name	For each record the archive stores the search keywords used to find them
Priority	Medium
Effort	Already in Invenio

Components	MiscUtil
Requirements	DR16 - Search key words UI15 - Search interface
Description	The system will be able to store the keywords used for each search and provide to the user the option to retrieve the used keywords and execute the past searches again.
Logical Constraints	• The user will have to type one or more keywords and press “Search” in order to be stored in the system • The user can display the list of his past searches through his account settings menu • The displayed list of past searches will be in chronological order • For each one of the searches the user will be able to view: – the collections in which the search was executed – time and date of last execution • The user will be able to execute any past search using exactly the same search settings he had used before
Notes and Questions	
Assigned to	

Table 5.61: Feature RF43

Feature ID	RF44
Name	The archive enables pingback/trackback services
Priority	Medium
Effort	Already in Invenio
Components	WebLinkBack
Requirements	EI3 - Pingback, Trackback
Description	There should be a mechanism that keeps track of the links, references to the content in the system.
Logical Constraints	Linkback methods like pingback or trackback will be used. Spam filters will be used to prevent spam links or pingback method may be used which is less susceptible to spamming.
Notes and Questions	
Assigned to	

Table 5.62: Feature RF44

Feature ID	RF45
Name	The archive is able to inter-operate with federated search engine dbwiz (SRU Server)
Priority	Medium
Effort	Months
Components	WebSearch
Requirements	IR5 - Connection with federated search engine dbwiz
Description	Search/Retrieval via URL (SRU) protocol support will be implemented in BlogForever. The admin of a dbwiz installation will be able to add the BlogForever SRU Server URI to its system in order to be able to perform queries using SRU protocol. Results from BlogForever will appear in dbwiz federated search.
Logical Constraints	The CQL implementation required to have full SRU protocol support is quite complex.
Notes and Questions	SRU standard: http://www.loc.gov/standards/sru
Assigned to	AUTH

Table 5.63: Feature RF45

Feature ID	RF46
Name	Users can create personal collections of their favourite blogs
Priority	Medium
Effort	
Components	WebBasket
Requirements	FR20 - Favourite list of blogs and topics UI16 - Easy to learn/Intuitive
Description	Users should be able to define personal collections of blogs.
Logical Constraints	This should be covered by using baskets or by using tags (see RF52)
Notes and Questions	
Assigned to	CERN

Table 5.64: Feature RF46

Feature ID	RF56
Name	The archive provides a journal view of the new blog posts
Priority	Medium
Effort	Weeks
Components	WebBlog
Requirements	FR22 - Summaries/Journals about new archive content UI2 - Magazine/Journal view of the new blog posts UI16 - Easy to learn/Intuitive

Description	The system will archive every blog post and the corresponding comments, if there are any. The user will be able to display for each record the full text of the post and the comments.
Logical Constraints	• The user will have to click the option "Detailed record" to display a record from his search results • The record will be displayed in the main frame • Below the record, the user will have the option to find a set of attached files that will include: – the full text of the post – the comments of the post (if there were any)
Notes and Questions	
Assigned to	CERN

Table 5.65: Feature RF56

Feature ID	RF57
Name	The archive provides a ranking method based on the user rating of content
Priority	Medium
Effort	Days
Components	BibRank
Requirements	FR11 - Recommendation system UI31 - Ranking of archived posts FR27 - Ranking of blogs and blog posts

Description	The BlogForever archive is providing a set of services that allow the recommendation of content among users. Explicit recommendation is taking place by allowing users to share records with selected users of the archive through direct sharing gestures. For example, user X has found an interesting blog post and forwards the record to his friend Y. This forwarding is facilitated through an internal messaging system developed for the archive. Therefore, sending a recommendation to another user is equal to automating the process of sending another message to a user, which includes completing the body of the message with the desired link. Optionally, this type of message may be tagged as a specific type in order to differentiate from ordinary user-to-user messages. Contrary to explicit, implicit recommendation is facilitated by the system and the user has no direct control over the content he receives as a recommendation. Through feedback received by the users on records, the content is processed and recommended to users with similar taste. In order to support the above service, powerful models such as Support Vector Machines are deployed and run periodically in order to estimate affinity between content and users.
Logical Constraints	For the end user, recommendations appear as supplementary information in the archive's pages. The information provided may be dependent on the record shown (i.e. recommended records based on the currently visiting record) or not (i.e. special page for messages with recommendations from other users). For the case of explicit recommendation, a messaging system is required. This messaging system allows the communication among users through simple messages. In order to describe a message, the sender and the receiver, as well as the title (optional) and body of the message must be completed. For the case of implicit recommendation, the system sets up a periodic calculation of relevant information through a configuration page. This page, which is accessed and maintained by the administrator, allows entering information such as the frequency of the calculations as well as the amount or persistence of recommendations shown to users.
Notes and Questions	
Assigned to	SRDC

Table 5.66: Feature RF57

Feature ID	RF58
Name	A user can rank archive content based on specific users' content rating
Priority	Medium
Effort	Days
Components	BibRank
Requirements	UI31 - Ranking of archived posts FR27 - Ranking of blogs and blog posts FR11 - Recommendation system
Description	The users should be able to search the records by the average score calculated with only scores given by some specific users.
Logical Constraints	• The users will be able to create lists of users that they trusts. • Only ratings of the users who are in a trusted list will be used to rank the records. • The users will be able to select user list(s) among their already created lists to rank the records.
Notes and Questions	
Assigned to	SRDC

Table 5.67: Feature RF58

Feature ID	RF59
Name	Export data using XML
Priority	Medium
Effort	Already in Invenio
Components	WebSearch, BibExport
Requirements	EI1 - API for external clients to query data EI2 - Data access/export as XML FR4 - Blog export IR6 - Facilities to enable interoperability UI28 - Integration/Combination with other systems
Description	The repository will offer different formats to export content as XML. This formats are: MARCXML, EndNote, RSS, OAI DC, RefWorks, MODS, SRU
Logical Constraints	
Notes and Questions	
Assigned to	

Table 5.68: Feature RF59

Feature ID	RF60
Name	The archive can export all its content, database entries and file system for migration
Priority	Medium
Effort	Already in Invenio
Components	MiscUtil
Requirements	SM3 - Data export for migration
Description	The system should provide the option to export all archive data so that they can be migrated to another system.
Logical Constraints	• The system will export the data in a widely known and standardized file format • The exported data will consist of: – the blog content and elements that were crawled – metadata with information regarding the preservation process in the system
Notes and Questions	
Assigned to	

Table 5.69: Feature RF60

Feature ID	RF61
Name	The archive ranks blogs based on their views and downloads
Priority	Medium
Effort	Days
Components	BibRank
Requirements	FR27 - Ranking of blogs and blog posts FR31 - Define important blogs and filter junk
Description	The users should be able to search the records by their view count. The users should be able to search the records by their download count.
Logical Constraints	New ranking methods to rank the records will be introduced: to rank the records by their download and to rank the records by their view count .
Notes and Questions	
Assigned to	SRDC

Table 5.70: Feature RF61

Feature ID	RF62
Name	Export records as PDF and JPEG
Priority	Medium
Effort	Weeks
Components	BibFormat, WebSearch, WebStyle
Requirements	FR17 - Print/Export as PDF, JPEG
Description	The users should be able to export the records as PDF or JPEG. To construct the PDF file, first latex code should be created from the record in HTML form. If CSS style of the record is present, it should also be applied. Later, with latex tools (pdflatex, xelatex, etc.), latex to PDF conversion should be completed and the PDF file should be provided to the users.
Logical Constraints	• The PDF file should be constructed based on a latex template.
Notes and Questions	
Assigned to	SRDC

Table 5.71: Feature RF62

Feature ID	RF67
Name	The archive fetches and stores embedded content
Priority	Medium
Effort	Days
Components	BibUpload
Requirements	FR54 - What to archive: text and comments DR14 - Embedded objects
Description	All the embedded objects of blogs are downloaded from the spider. These objects should be stored in the archive at pre-ingestion time.
Logical Constraints	To amend the pre-ingestion plugin (see RF9) in order to enrich the MARC metadata of each record (Blog, Blog Post, Page, Comment) with FFT tags.
Notes and Questions	
Assigned to	CERN

Table 5.72: Feature RF67

Feature ID	RF68
Name	The archive provides information diffusion analysis mechanisms
Priority	Medium
Effort	Weeks

Components	BibClassify
Requirements	FR36 - Memetracking and trend detection
Description	The archive provides the user with a set of tools that allow detecting the provenance of various topics, phrases and memes. In addition to their provenance, the archive provides the full history of a given meme, allowing for the user to adjust the time of content and study the spread and the diffusion of information in the archive. To support the above mechanisms, the archive detects influential memes and processes them according to time. An interface that maps this information diffusion to aggregations and their visualisations is provided by the system. This analysis is conducted by taking into account the network of blogs that reside in the archive and adopts the state-of-the-art technologies.
Logical Constraints	Memetracking and trend detection are expected to be complex and resource-costly services. The mechanisms that are deployed are expected to run periodically, at a low priority and configured by the administrator of the archive. Some initial input for the mechanisms may be provided by external social media analytics providers or the administrator himself. Concerning the end-user, exporting mechanisms are provided, such as the generation of reports, allowing printing of the visualisations and social media sharing.
Notes and Questions	
Assigned to	CERN

Table 5.73: Feature RF68

Feature ID	RF71
Name	The archive provides a personalized annotating and highlighting tool for users
Priority	Medium
Effort	Weeks
Components	WebSearch, WebSession
Requirements	UI12 - Annotations and Highlighting
Description	The users should be able to highlight any part of a record. The users should be able to annotate the part they highlighted.

Logical Constraints	• There will be a tool to highlight the content in the record page. • The users will be able to highlight any part of the record. • The users will be able to choose the highlighting color, clear all of the highlights on a record and undo their move. • The users will be able to erase their highlights. • The users will be able to change the color of their highlights. • The users will be able to annotate the highlighted text. • The users will be able to edit or delete their annotations. • The annotations and notes will be saved into database. The users will be able to load them later.
Notes and Questions	
Assigned to	SRDC

Table 5.74: Feature RF71

Feature ID	RF72
Name	The archive provides a visualization of the blogs network structure
Priority	Medium
Effort	Weeks
Components	WebSearch
Requirements	FR18 - Analyze the network structure of blogs UI9 - Network view on topics, blogs, posts, authors, etc. UI27 - Dynamic network view on topics, blogs, posts, etc. EI6 - Export links between blog content
Description	The connections between two blog entities (e.g. posts) can be aggregated to a network. Various network views are possible for the different entities in the repository, for example, networks among blogs, posts or authors. The repository will provide the possibility of simple network visualization and, thus, the option of a visual exploration to the experienced end-user.

Logical Constraints	• The network visualization will utilize the connections that would be provided by the feature RF65. • The network visualization will be either included as part of the repository or accomplished via the communication with an external visualizer. The implementation will be decided according to performance considerations due to the fact that the rendering for network graphics can be source consuming.
Notes and Questions	
Assigned to	TUB

Table 5.75: Feature RF72

Feature ID	RF73
Name	The archive recommends blogs to users based on the ratings and preferences
Priority	Medium
Effort	Days
Components	WebSearch, WebSession
Requirements	FR28 - Recommend a cluster of blogs according to user preferences
Description	The system should find the records/blogs that the user may interested in and display them.
Logical Constraints	• The records/blogs that the user may interested in will be offered in the account page. • The recommended records/blogs may be listed in the order of ratings of the records/blogs that they are similar to. • The recommended records/blogs will be similar to the ones that the user's most visited, rated or favourite records/blogs. • To find the recommended records/blogs, ranking methods such as word similarity, content tags, times referenced/cited will be used.
Notes and Questions	
Assigned to	SRDC

Table 5.76: Feature RF73

Feature ID	RF74
Name	The archive enables/disables certain functionalities based on the content rights
Priority	Medium
Effort	Already in Invenio
Components	WebAccess
Requirements	UI10 - Available services depend on the content rights
Description	Digital content could be available with different rights. Services and functionalities should only be used by the users/user groups with required rights. The user interface should support these rights by enabling/disabling the allowed services/functionalities.
Logical Constraints	The admin will be able to restrict the services/functionalities. The admin will be able to give rights to specific users or user groups for restricted services/functionalities. The system will check if the user has necessary rights to use corresponding service/functionality. If the user doesn't have the rights, s/he will not be able to use that service/functionality.
Notes and Questions	
Assigned to	

Table 5.77: Feature RF74

Feature ID	RF82
Name	The archive can be deployed using a range of different database server technologies
Priority	Medium
Effort	Already in Invenio
Components	
Requirements	SP1 - Support for different SQL-databases
Description	The archive should support different SQL-databases to ease the administration for different institutions.
Logical Constraints	There will be a list of sql based databases that the system supports. The admin will be able to choose one of the supported databases and the system will use specified database.
Notes and Questions	
Assigned to	

Table 5.78: Feature RF82

5.2.3 Low priority features

Feature ID	RF48
Name	The archive provides the option to translate its content on demand
Priority	Low
Effort	Days
Components	WebSession, WebAccess
Requirements	FR9 - Content translation
Description	Users should be able to translate the content of the records, comments, reviews and messages.
Logical Constraints	• Google's translate gadget will be accessible for each page and the users will be able to translate the content (e.g., record, comment, review and message) by selecting the language from this gadget.
Notes and Questions	
Assigned to	SRDC

Table 5.79: Feature RF48

Feature ID	RF49
Name	The archive distinguishes institutional/corporate blogs from personal blogs
Priority	Low
Effort	
Components	
Requirements	DR19 - Distinguish institutional/corporate blogs from personal blogs
Description	Since the repository size can become very big, it would be easier to navigate if institutional/corporate blogs would be separated from personal blogs, therefore, they could be also searched separately.
Logical Constraints	A MARC tag will be used to classify blogs in these two groups.
Notes and Questions	
Assigned to	CERN

Table 5.80: Feature RF49

Feature ID	RF50
Name	The archive offers the option to disseminate newly archived content in external social platforms
Priority	Low
Effort	Weeks
Components	WebAlert
Requirements	FR33 - Dissemination of newly archived items in external social platforms (ex. Twitter) in connection with author profiles
Description	This feature concerns the blog authors and aims at increasing their awareness on the BF activity through social media. Every user of the repository who has a weblog archived, may select to link his BF account to one or more social media platforms. In order to achieve this, we assume that the blogger has or creates a user account and its corresponding profile in the social media platform. The BF archive will have to verify that the two user accounts belong to the same person. The user accounts' cross-matching and verification takes place through technologies, such as OAuth or OpenID. In any case, the user grants the permission to the BF archive to access and confirm that he is the owner of the social media account that he claimed. Afterwards, the BF repository is adding this information to the user profile and is using it for disseminating newly archived content. More specifically, for every record inserted in the archive, which belongs to a blogger verified by the above process, the BF social media account will publish a short message including the user account and the link to the record. In order to avoid having the official BF social media account overloaded, the messages posted will be personalized and not visible to every friend or follower of the BF archive. For example in Twitter, the BF account will mention the blogger's account or in the case of Facebook a post on the user's wall might be more appropriate. Alternatively, the BF account may post on behalf of the blogger as follows: "John Smith - Hello World archived by @BlogForever-Archive."

Logical Constraints	• The bloggers must be able to connect their social media accounts without the need to provide their credentials (i.e. username and password), since the BF archive supports open authentication methods. • The BF archive allows opting out and removing the linkage between accounts. • The feature may provide the option to select the frequency of updates or similar customizations.
Notes and Questions	
Assigned to	SRDC

Table 5.81: Feature RF50

Feature ID	RF51
Name	The archive is able to search within external sources
Priority	Low
Effort	Already in Invenio
Components	WebSearch
Requirements	UI18 - Search in external sources

Description	This feature allows the administrator of the BlogForever archive to setup additional sources for searching. More specifically, the administrator can choose to make use of web services that support open searching, such as OpenSearch[1] or other RESTFul services. While the above services require a technical expertise to understand them, the administrator of the archive should be able to select from a drop-down menu between a list of popular search engines. Some more options, which are not mandatory and depend on the web service to be consumed, may be provided that describe the settings of the search results provided to the end-user (e.g. following a link is opening a new window, the summary of the resource is provided, the banner or the name of the provider or the destination is visualised etc.). The provision of this feature will allow users of the platform to access resources provided by third parties, such as blog search engines as well as other social media content providers or aggregators. Depending on the technology used, a short summary of the external resource may be provided. In every case, the link to this external resource is clearly visualised so as to illustrate that following this link will lead the user to a website outside the BlogForever website archive.

[1]http://en.wikipedia.org/wiki/OpenSearch

Logical Constraints	• Adding a new source for searching contains a pre-existing list of search engines. This allows the administrator of the archive to learn from the examples provided and add a new source. • Adding a new source is supported with some customization options with respect to the behaviour of the system. For example, the administrator of the platform may select: – if clicking a link opens in new window, – whether a link displays the address of the search engine (for example, http://blogsearch.com) or the address of the record (for example http://example.com/my-new-blog-post) or both. – the metadata that describe the record, – the maximum number of search results provided by external search engines. • Search results linking to external sources must be clearly illustrated, such that the end-user will expect that following them will result in exiting the archive.
Notes and Questions	
Assigned to	

Table 5.82: Feature RF51

Feature ID	RF52
Name	Users can tag archived records with personal tags
Priority	Low
Effort	Weeks
Components	WebTag
Requirements	UI32 - Tagging system
Description	The user will be able to assign tags to the records with his personal metadata. With the tags, the user will be able to organize the records in personal collections according to his preferences. The user should have the option to choose freely the words he wants to use as tags. Tagging feature is served through the user baskets. The user will assign a tag to a record by adding the record to the homonymous basket.

Logical Constraints	• The user will be able to assign tags to the records that: – he adds in his personal basket – are already included in his personal basket • The user will have anytime the option to: – edit or delete tags, when displaying the corresponding basket through the options "Edit basket" and "Delete basket" correspondingly – create new tags, in the "Display baskets" menu through the option "Create basket" – add more tags to a record, in the "Display baskets" menu through the option "Copy item" with which he will be directed to copy the record to another basket – remove a tag from a record, in the "Display baskets" view through the option "Remove item" in the corresponding basket • The user will be able to create a new tag or use one of those he previously created • The user will be asked to assign a tag every time he adds a new record in his personal basket • User's tags will be also visible to all archive users
Notes and Questions	
Assigned to	CERN

Table 5.83: Feature RF52

Feature ID	RF63
Name	The archive keeps snapshots of all the different designs of a blog
Priority	Low
Effort	Days
Components	BibUpload
Requirements	FR53 - Snapshot versions of blog designs in the archive
Description	The system archives a snapshot of the layout design of blogs each time there is any modification. The snapshots will be provided to the user as attached, detailed image files.

Logical Constraints	• The snapshots will be available for blog items • The snapshots will be in JPEG format • The user will be able to view the snapshots of current and past versions of a blog's design as attached files below the detailed record of blog • The snapshot of the latest version can also been viewed by clicking the snapshot icon next to each blog item in the search results
Notes and Questions	
Assigned to	CERN

Table 5.84: Feature RF63

Feature ID	RF64
Name	The archive offers the option to login using external (universal) credentials
Priority	Low
Effort	Weeks
Components	WebSession, WebAccess
Requirements	FR55 - Universal Login & Central Login
Description	Users should be able to login to the platform with their Facebook, Google+ and Twitter accounts.
Logical Constraints	• Users who logged in with external accounts will be able to share contents from platform. • When the user logged in with external account, s/he will be able to access all archives that s/he has permission to access.
Notes and Questions	
Assigned to	SRDC

Table 5.85: Feature RF64

Feature ID	RF65
Name	The archive analyzes blog links and stores the connections between them separately
Priority	High
Effort	Weeks
Components	WebBlog
Requirements	DR9 - Connections/Links

Description	Blogs and inherent entities (posts, comments, authors) can be related to other entities. These relations occur either as explicit hyperlinks (e.g. link, blogroll) or as implicit relations (e.g. co-occurrences). The BlogForever repository will make these connections visible and navigable in the user interface.
Logical Constraints	• Explicit connections will be differentiated in: – external links, when destination is outside the repository – internal links, when destination is also an archived resource in the repository. Furthermore, explicit links will be also differentiated in: – Citations – Blogroll – Pingback/Trackback • Among the implicit connections, the following will be generated: – Blog-Connection: One or more posts of a blog A link to one or more posts of a blog B – Author-Citations: One or more posts or comments written by author A link to one or more posts or comments written by author B – Author-Co-Citations: One or more posts or comments written by author A link to the same resource than one or more posts or comments written by author B
Notes and Questions	
Assigned to	TUB

Table 5.86: Feature RF65

Feature ID	RF75
Name	The archive can do sentiment analysis on the content
Priority	Low
Effort	Months
Components	BibClassify
Requirements	FR21 - Sentiments analysis on blog post level
Description	To give the users an idea about the blog post's attitude sentiments of the post should be analysed.

Logical Constraints	Sentiment analysis will be run by the administrator. The user should be able to see the sentiment scores of the contents. The users will be able to add a search criteria that ranks the results based on their sentiment scores.
Notes and Questions	
Assigned to	CERN

Table 5.87: Feature RF75

Feature ID	RF76
Name	The archive detects content's originality and ranks it accordingly
Priority	Low
Effort	Months
Components	BibRank
Requirements	FR35 - Detection and ranking of the originality UI31 - Ranking of archived posts FR27 - Ranking of blogs and blog posts
Description	The archive is analyzing the content of a blog and processes it in order to assess its novelty. In order to achieve this, computational linguistics technologies are deployed, such as n-grams. The feature is built on top of services provided by RF67. For the end-user, an originality index is displayed for each blog, which will approximate the novelty of the content. Optionally, the text that appears to be most original will be annotated. For the remaining of the text, mechanisms that annotate near-duplicate statements will allow tracing back to the original author or other external sources.
Logical Constraints	The mechanism that analyzes the originality of blog content is expected to be expensive and erratic. A disclaimer about the quality of the results presented is required. Additionally, the mechanism should be invoked periodically in a semi-supervised manner. The administrator of the blog should be able to intervene and bypass the results of the process in order to reduce the number of false assessments.
Notes and Questions	
Assigned to	SRDC

Table 5.88: Feature RF76

Feature ID	RF77
Name	The archive provides a for mobile version
Priority	Low
Effort	Months
Components	All modules
Requirements	UI14 - User interface for mobiles UI16 - Easy to learn/Intuitive
Description	The users should be able to view the mobile phone supported pages from their mobile phones.
Logical Constraints	New web page templates to accommodate the screen of the mobile phones will be introduced. (Optional) Surfing on the Internet via mobile operators costs some for the users. Therefore, there will be some regulations: • The CSS files will be separated into modules not to load unnecessary ones. Moreover, they will be minimized to get rid of unnecessary white spaces. • The minimized versions of the JavaScript files will be loaded to pages.
Notes and Questions	
Assigned to	MOKONO

Table 5.89: Feature RF76

Feature ID	RF78
Name	The archive displays content after filtering it with user preferences
Priority	Low
Effort	Weeks
Components	WebSearch, BibRank, BibFormat
Requirements	UI25 - Filtered, personalized aggregation of content for end-users FR45 - Personalized filtering services
Description	There should be a personalized service that allows registered users to systematically filter their searches based on their individual research interests. Moreover, the users should be able to use personalized list of favorite blogs/blog posts. This list could also be filled automatically with blogs and posts matching certain searching criteria inserted by the user. The users should be able to configure a personalized filter for the content view in the platform, including old and new content.

Logical Constraints	The platform will be able to display different contents for each user according to their configurations. The users will be able to use personalized list of favorite blogs/blog posts. The users will be able to add a blog/blog post to their favorite lists. The favorite lists will be able to be filled by the search keywords automatically. The users will be able to save their search keywords and information. These keywords and information will be able to be used by the user later.
Notes and Questions	
Assigned to	CERN

Table 5.90: Feature RF78

Feature ID	RF83
Name	The archive provides multiple different views of the archive for each user
Priority	Low
Effort	Weeks
Components	WebSearch
Requirements	FR24 - User specific collections/projects
Description	The users should be able to save their searching preferences to search always in one specific collection or topic.
Logical Constraints	The users will be able to save their search preferences. The preferences would be appended automatically at the end of every search query.
Notes and Questions	
Assigned to	CERN

Table 5.91: Feature RF83

Chapter 6

Conclusions

The overall BlogForever system architecture was presented. The BlogForever platform consists of two main components: the spider is responsible of fetching the blog data, extract entities and present them in a structured format, while the repository is responsible of hosting the data, preserve it and make it available in a web interface. The communication between these components is done using APIs and open standards like METS, keeping the dependence between them in a minimum. While the spider component design was presented in D4.2[10], this document focused on the repository component design. The repository is reusing the already existing Invenio digital library software platform, as stated in the Description of Work (DoW).

Some relevant Invenio characteristics were introduced. Invenio already has a modular architecture composed of ∼40 modules that makes it easily expandable. Invenio's flexibility and expandability has been proved during this design process, making the design process intuitive. This makes us pretty confident that the resulting repository design is also flexible for potential future extensions.

The mapping between the existing components and the blog archive use case was discussed. Some components needed adaptation to our BlogForever use case (e.g. ingestion process) while new components needed to be designed (OAIS, spam, tags). The improvements to the user web interface design were discussed presenting also UI mockups.

The mapping of mandatory/optional features to concrete requirements was described as well as their prioritization. The priorities have influence on the development plan. We shall be using agile development method, so that components will be developed via rapid prototyping and then, tested and feedback from tests, will be taken back into development continuously. We shall also work by use case schedule in collaboration with WP5. Finally at later stages we shall introduce nice UI for the end users.

References

[I] CDS Document Server. http://cds.cern.ch. [Retrieved November 22, 2012], 2012.

[II] Invenio documentation. http://invenio-demo.cern.ch/help/admin/. [Retrieved November 25, 2012], 2012.

[III] Invenio Project. http://invenio-software.org/. [Retrieved November 11, 2012], 2012.

[IV] mongoDB. http://www.mongodb.org/. [Retrieved March 22, 2012], 2012.

[V] Twitter Bootstrap. http://twitter.github.com/bootstrap/. [Retrieved November 7, 2012], 2012.

[VI] S. Arango-Docio, V. Banos, K. Stepanyan, and M. Joy. *D2.1: Survey Implementation Report.* Work package, University of London, August 2011. Work Package Two Deliverables.

[VII] J. Caffaro and S. Kaplun. *Invenio: A Modern Digital Library for Grey Literature.* Technical report, Twelfth International Conference on Grey Literature, Prague, Czech Republic, December 2010.

[VIII] H. Kalb, N. Kasioumis, J. García Llopis, S. Postaci, and S. Arango-Docio. *D4.1: User Requirements and Platform Specifications Report.* Work package, B. Consortium (Ed.): Technische Universität Berlin, December 2011. Work Package Four Deliverables.

[IX] H. Kalb, Y. Kim, and P. Lazaridou. *D2.3: Weblog Ontologies.* Work package, A. I. Cristea & M. Joy (Eds.): Technische Universität Berlin (TUB), May 2012. Work Package Two Deliverables.

[X] Hendrik Kalb, Paraskevi Lazaridou, and Matthias Trier. *D4.2: Weblog spider component design.* Work package, B. Consortium (Ed.): Technische Universität Berlin (TUB), June 2012. Work Package Four Deliverables.

[XI] Y. Kim, S. Ross, K. Stepanyan, E. Pinsent, P. Sleeman, S. Arango-Docio, H. Kalb, et al. *D3.1 Preservation Strategy Report.* Work package, Y. Kim & S. Ross (Eds.): University of Glasgow, September 2012. Work Package Three Deliverables.

[XII] Jiří Kunčar and Tibor Šimko. New Features and Technologies in Current and Future Invenio Versions. Technical report, European Organization for Nuclear Research (CERN), October 2012.

[XIII] Library of Congress. *MARC 21 Format for Bibliographic Data*, 1999 edition.

[XIV] Library of Congress. *METS*, October 2012.

[XV] P. Ponniah. *Data modeling fundamentals: a practical guide for IT professionals*. Hoboken, New Jersey, USA: Wiley-Blackwell, 2007.

[XVI] Suzanne Robertson and James C. Robertson. *Mastering the Requirements Process*. Addison-Wesley, 2nd edition, March 2006.

[XVII] A. Ronacher. Jinja2 (The Python Template Engine). http://jinja.pocoo.org. [Retrieved November 7, 2012], 2012.

[XVIII] P. Sleeman E. Pinsent G. Gkotsis T. Farrell S. Kopidaki M. Rynning S. Arango-Docio. *D5.1: Design and Specification of Case Studies*. Work package, University of London, June 2012. Work Package Five Deliverables.

[XIX] K. Stepanyan, M. Joy, A. Cristea, Y. Kim, E. Pinsent, and S. Kopidaki. *D2.2: Weblog Data Model*. Work package, B. Consortium (Ed.): University of Warwick (UW), October 2011. Work Package Two Deliverables.

[XX] Karl E. Wiegers. *Software Requirements*. Microsoft Press, 2nd edition, 2003.