Project Acronym: Awareness

Grant Agreement number: 297261

Project Title: Europeana Awareness (WP2)

D2.2 Report on Infrastructure and Tools for supporting User Contributed Content

Revision V1.4

Date of submission 30 March 2012

Author(s) Natasa Sofou, Anna Christaki, David Haskiya

Dissemination Level Restricted to a group specified by the consortium (including the Commission Services)

Project co-funded by the European Commission within the ICT Policy Support Programme

Contents

1 Introduction and context 3

2 Digital Storytelling Platform 3
2.1 Architecture 3
2.2 Design 5
2.2.1 Frontend design - Storytelling platform UIs 5
2.2.2 Administration Console 14
2.3 Data Model 18

3 Other platforms in use by Europeana for User Generated Content 24
3.1 1914-1918/RunCoCo 24
3.2 1989/Historypin 25

4 Conclusions 25

References 26

Appendix 1. API 27

1 Introduction and context

This deliverable reports on all the work done until now in the context of Task 2.1 *Operationalise tools to enable end user contributions to Europeana content*. It also marks a milestone, namely MS9 Release of the integrated toolset, version1.

Specifically, it aims to:

- define and specify an infrastructure for hosting User Generated Content, such as Flickr, Wikimedia Commons and Open Images

This deliverable focuses on the description of the architecture and design of the Digital Storytelling Platform (DSP), which is the outcome of Awareness project. The data model and API used for the development of the DSP are also provided in this document.

All details about architecture, design, data model and API refer to first release V1 of the DSP as indicated by MS9. Improvements, changes and revisions may occur in next releases.

Along with the DSP, other platforms for user generated content used by Europeana are briefly described, such as 1914-1928/RunCoCo and 1989/HistoryPin.

2 Digital Storytelling Platform

User contributed stories/narratives can be a starting point for exploring the interconnections between items discovered in Europeana based on intersecting time/space/topic components. Although digital stories can manifest in different ways, they share a set of characteristics which makes them ideal for the purpose of communicating cultural heritage to audiences who may find it difficult - or boring - to access content via passive channels such as search engines or catalogues. Such digital stories are rapidly becoming globally acknowledged as powerful tools for learning, integration, and preservation by providing unique platforms for creativity in the cultural sphere. Stories act to transfer knowledge from previous generations and help to uncover ethnic heritage emanating from different regions and locations in the culturally rich and diverse tapestry of Europe.

Europeana Awareness focuses on providing a Digital Storytelling Platform that should encourage users to create and share stories that include content from Europeana and other open resources. In the following section we provide detailed descriptions of how this platform was designed and developed. Particularly we provide description of the architecture, design, and data model used for the implementation of the platform. The API developed for the communication of the front-end and back-end of the platform is provided in Appendix 1.

2.1 Architecture

The Digital Storytelling Platform is being developed using a REST-style [1] client-server architecture. REST was the selection of choice since a REST-based application is a lightweight alternative to Web Services (SOAP, WSDL, etc) and RPC (Remote Procedure Call).

Much like Web Services, a REST service is:

- Platform-independent
- Language-independent
- Standards-based (runs on top of HTTP)

Since the decoupling of frontend and backend was an important requirement for DSP development, a REST-based architecture gives the freedom for application parts to evolve independently.

The frontend of the DSP application includes all the User Interfaces and will be collecting input in various forms from the user and processing it to conform to a specification the back-end can use. Themes and their corresponding skins will be used to change the appearance of the frontend while the user searches for Digital Stories within a theme (Europeana 1914-1918, Europeana 1989, etc).

The frontend is being developed as a thin client using HTML5 and CSS, and interacts with the backend API issuing AJAX (Asynchronous Javascript and XML) calls. The data format of choice for the interaction between frontend and backend is JSON, a light-weight data serialization format based on a subset of JavaScript.

The backend of the DSP is being built as a RESTful web API that will process the incoming data from requests, validate it and save it to a MONGO Database [2], a document-oriented database system that stores structured data as JSON-like documents. The backend API service is being built in Java and will run on the Play! Web application framework, which targets RESTful architectures and supports agile development (less configuration, faster testing etc).

All the user uploaded media files (apart from images that will be saved in Mongo DB) in future versions will be hosted on the OpenImages [3] media platform taking away the complexity of setting up a dedicated media server for DSP.

The frontend and backend of the application will run a one web service, residing on the same server (Play!) [4]. However configurable widgets of the frontend (search stories and story play out) will be made available for installation as embedded components across partner websites. The DSP application will also need to interact with a number of other available web services like Europeana, YouTube, Flickr etc, to make use of their resources as Digital Story components. Therefore a number of connectors to the APIs exposed by these web services must be developed and embedded to the DSP application.

For the indexing and searching of Digital Stories we will use Solr4 [5] a search platform that includes powerful full-text search, hit highlighting, faceted search and quick database integration.

For security and encryption of DSP sensitive data, an HTTPS Proxy will be used. The whole application is being developed as a Europeana hosted service but additionally the digital stories will be published to an OAI/PMH repository where Europeana can harvest them and make them available in the Europeana portal, API and Linked Open Data. All the selected technologies used in the development of DSP are open source technologies.

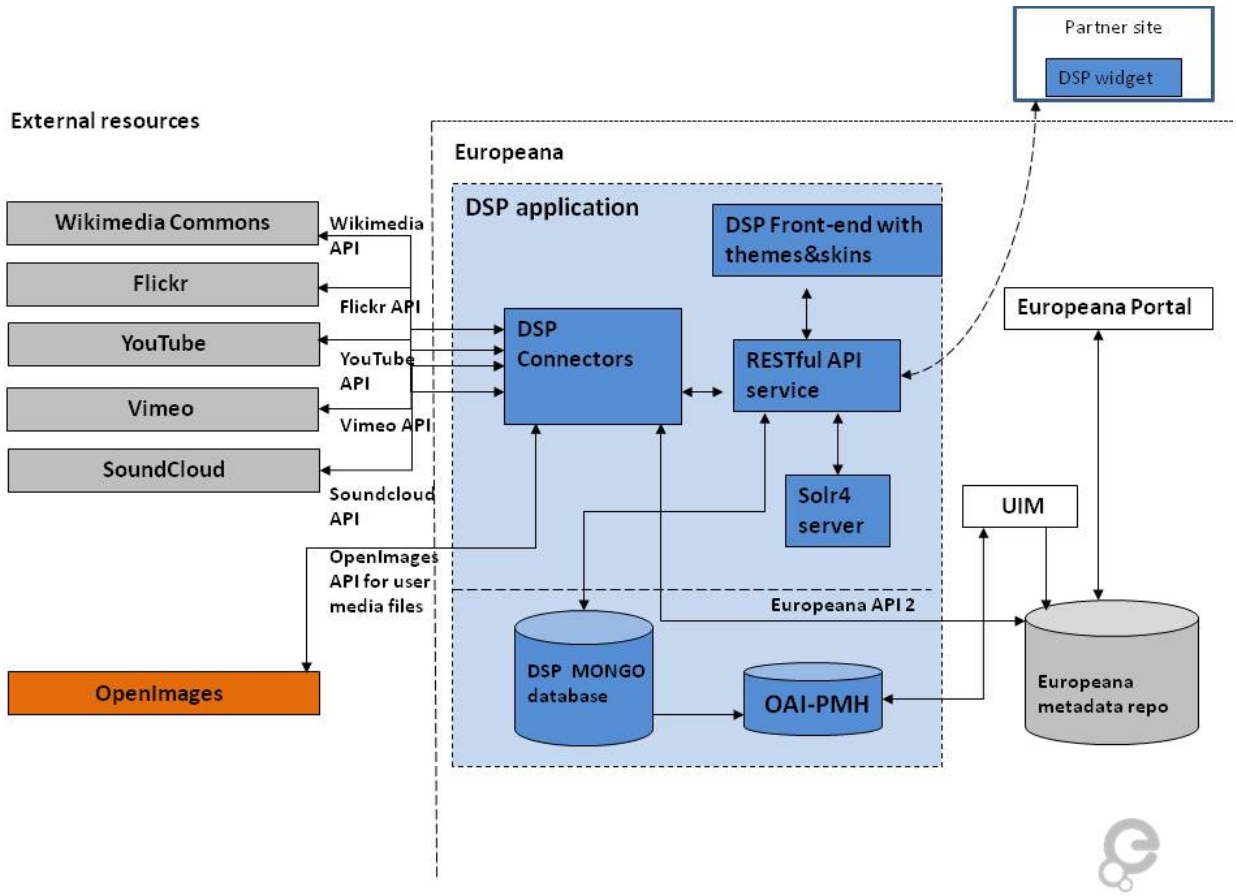


Figure 2-1 DSP Architecture

2.2 Design

As explained in 2.1 there are two main components that constitute the storytelling platform: the frontend which consists of the UIs and handles the interaction with the user and the backend which processes the incoming data from requests, validates and saves data in the database, and provides appropriate JSON responses to the frontend for search and retrieval requests.

The DSP frontend design is responsive taking into account the viewing experience across a wide range of devices. The functionality of the platform has also been tested in all the major browsers (IE 8+, Safari, Firefox, Chrome).

In the following sections we provide information about the design and related workflow of the Frontend UIs and the backend administration console.

2.2.1 Frontend design - Storytelling platform UIs

In this section we provide a description of the main storytelling platform functionalities available to user (either registered or not) screen along with the corresponding UI screens.

2.2.1.1 Splash screen

The first screen a users sees when visiting the storytelling platform is the one illustrated in Figure 2-2. The theme of the splash screen is “Europeana 1914 -1918”. All stories created under this theme are displayed and overlayed is a Help screen with directions on how to use the DSP. With this screen as starting point the user has the option to view the Help, About or Contact section, Join if he is not a registered user or Log In if he is already a registered user.

Even if not registered, the user can select one of the available themes of the platform, set the language from the available languages list or perform a free text search from within a theme.
Figure 2-2 Storytelling platform splash screen

Figure 2-2 Storytelling platform splash screen

2.2.1.2 Themes

Europeana story telling platform has 3 themes (provided by Europeana) currently available:

- Migration
- Europeana 1914-1918
- Europeana 1989

The user is able to select among the available themes and set the desired theme for the storytelling platform. When selecting a theme, all stories under that theme are displayed on screen.

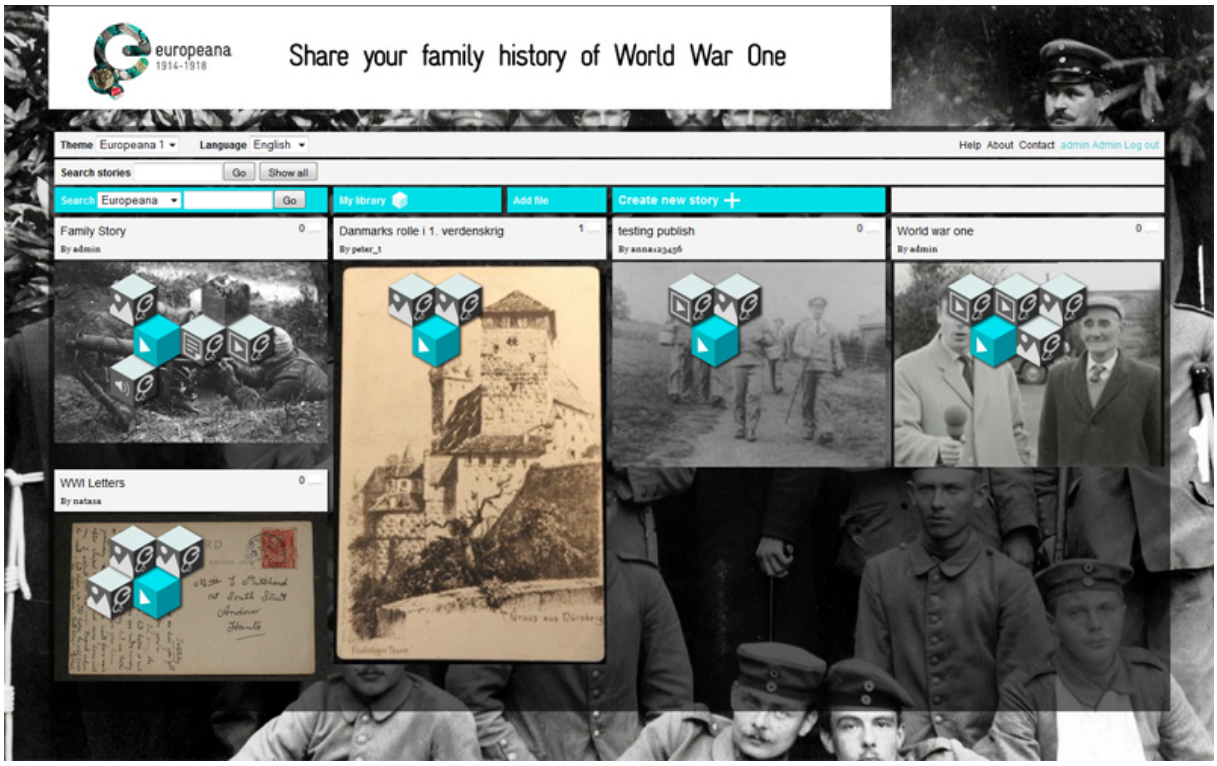


Figure 2-3 Europeana 1914-1918 theme

By selecting a different theme the screen changes as illustrated in Figure 2-3. Themes, apart from providing a filtering option for stories, also give skinning parameters to the DSP frontend. All stories under theme “1914-1918” are displayed in the example above, with a relevant banner and background to that theme.

2.2.1.3 Search Story

After setting the preferred theme, the user can perform a keyword(s) based search among all published stories related to that theme. When more than one keywords are used, the results are ranked based on their relevance to the search terms i.e. a story with the most search “hits” will be displayed first. In the case illustrated in Figure 2-4, the user searches for stories of Europeana 1914-1918 using the keyword “postcard”. The returned results are presented. Each story is presented as a grid of cubes, where each cube is a story object. The blue cube with the “play” icon triggers the story player.

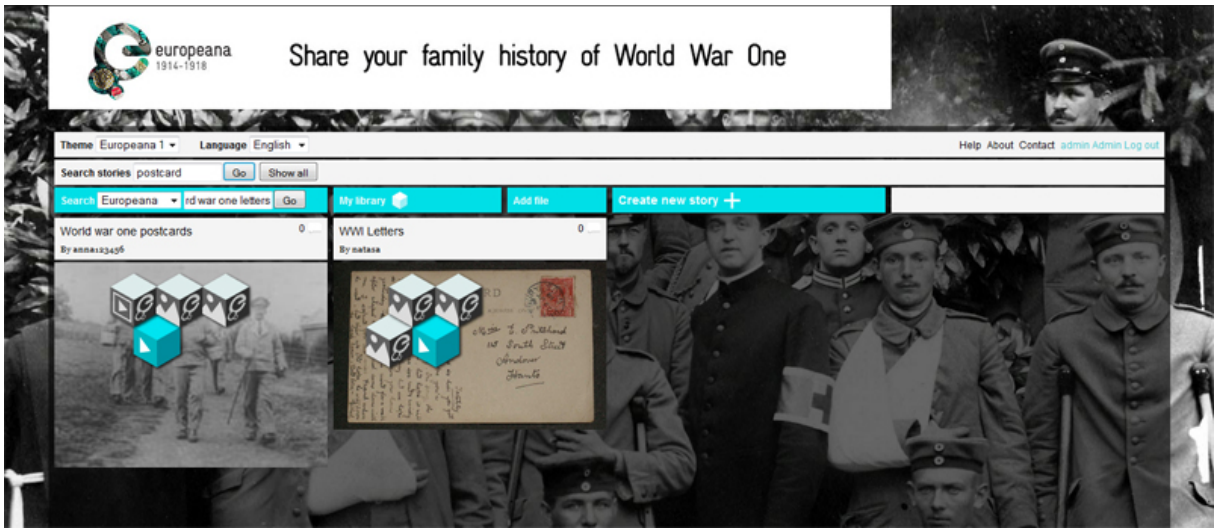


Figure 2-4 Search stories

2.2.1.4 Story play out

By clicking on any of the story cubes the user can have a preview of the corresponding story object, as in Figure 2-5.

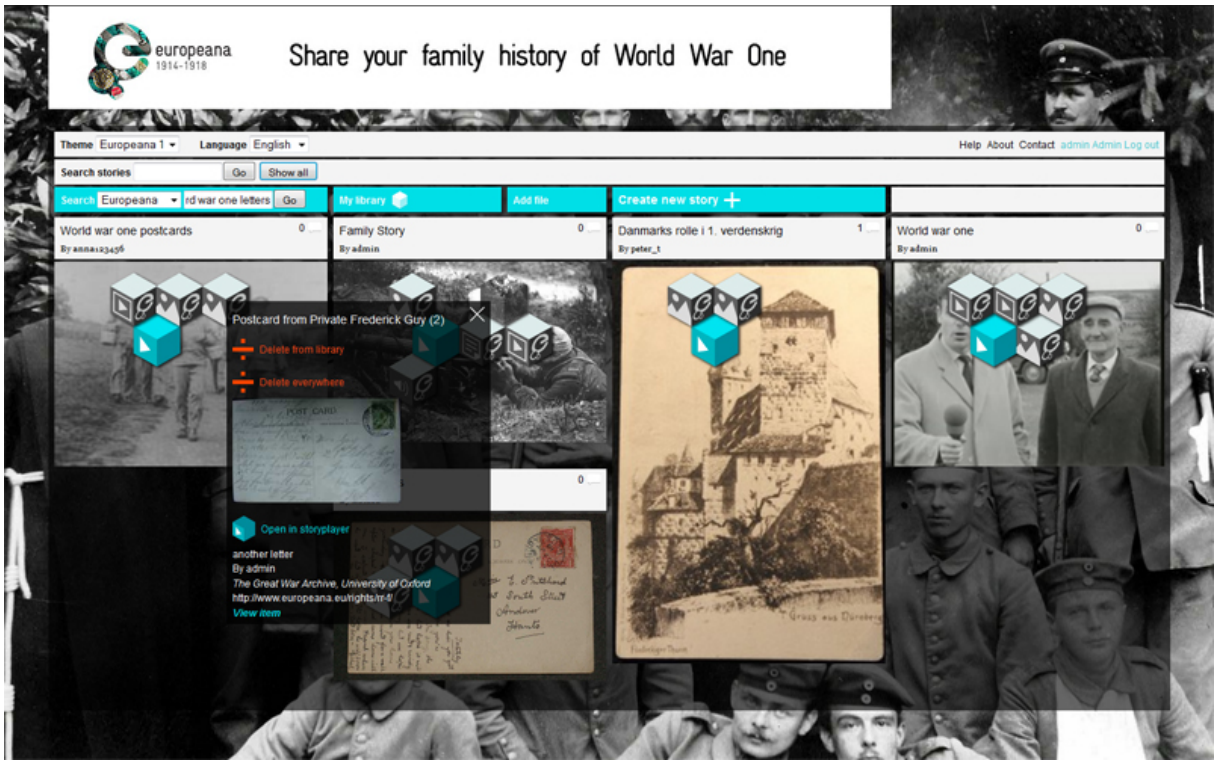


Figure 2-5 Story object preview

By clicking on the blue cube of a story grid the story player window opens on top of the search results, as in Figure 2-6.

The story player window area is divided in three columns (see Figure 2-6). The first column consists of the story grid and the corresponding story objects placed on it and also shows the title and description of the story. Europeana story objects are indicated by a lowercase “e” displayed on the rightmost side of the item cube. Along with this “e” the type of the item is

displayed on the leftmost cube side. A Europeana story object type can be one of the following:

- image
- video
- audio
- text
- 3D

In the current version, story objects can also be generated from user-uploaded photos. In this case the photo is projected on the story object cube that appears on grid. Future versions of the platform will handle other types of user uploaded content such as video, audio and text.

In each of the next two columns of the story player window a preview of consecutive items of the story (in terms of adjacent cubes on grid) appears. The item, its metadata as well as the story script the user inserted for that item is displayed in each column. For Europeana items an additional link is included to "view item" which opens up a browser window displaying the item in the Europeana portal.

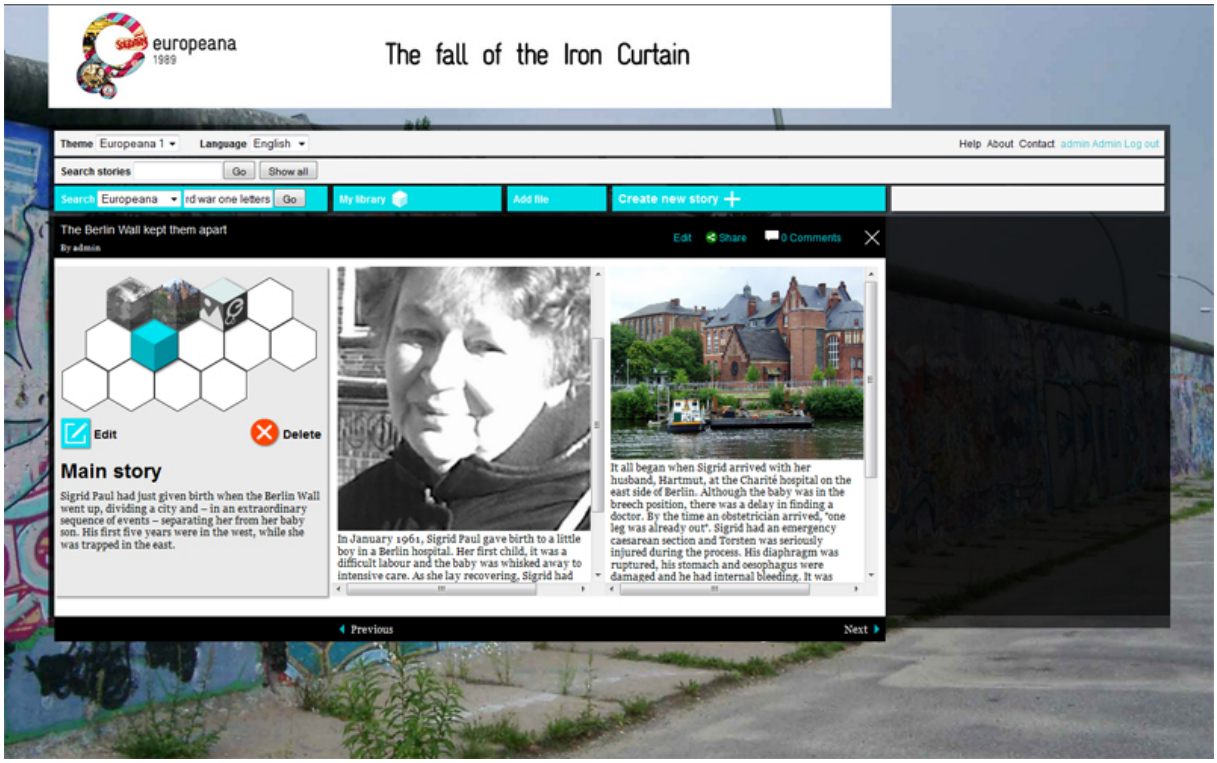


Figure 2-6 DSP story player

2.2.1.5 Login

The user can join or login to the storytelling platform, thus obtaining access to advanced platform functionalities such as creating stories. By joining, the user registers to the DSP and creates an account. Different roles can be assigned to each user (such as editor or contributor) by the administrator's console as explained in 2.2.2.

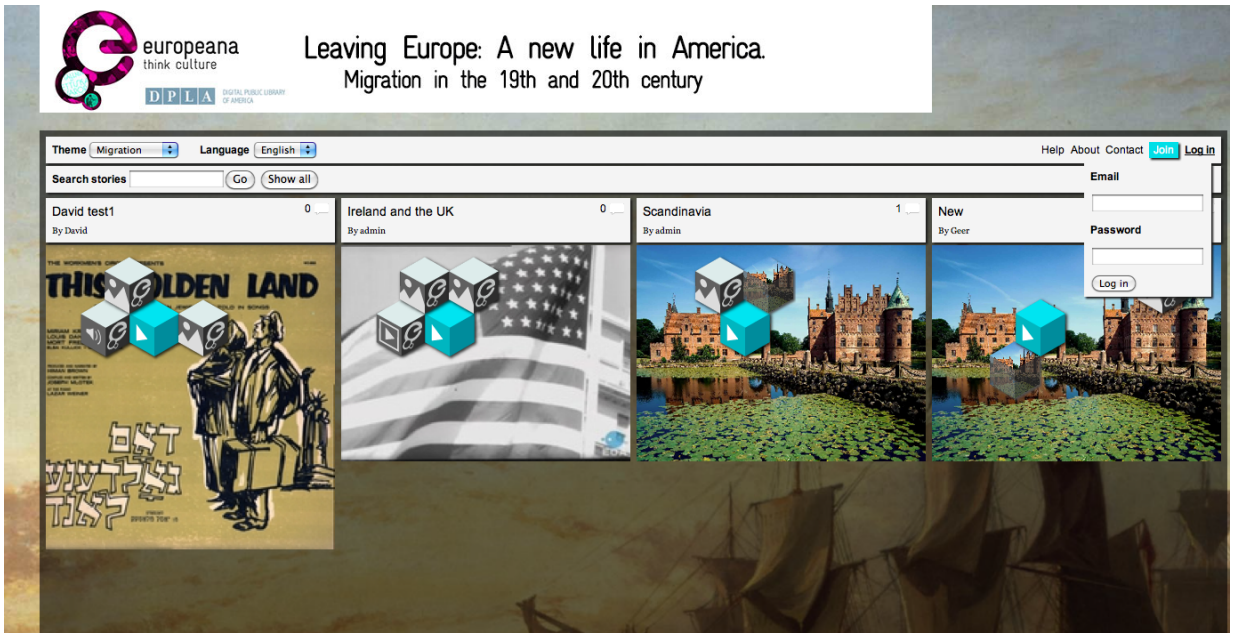


Figure 2-7 Log in

When logged in, in addition to searching among theme stories, a user has the ability to:

- Search among Europeana items
- Search among user uploaded files
- Preview "My Library"
- Upload files to his/her library and
- Create new story

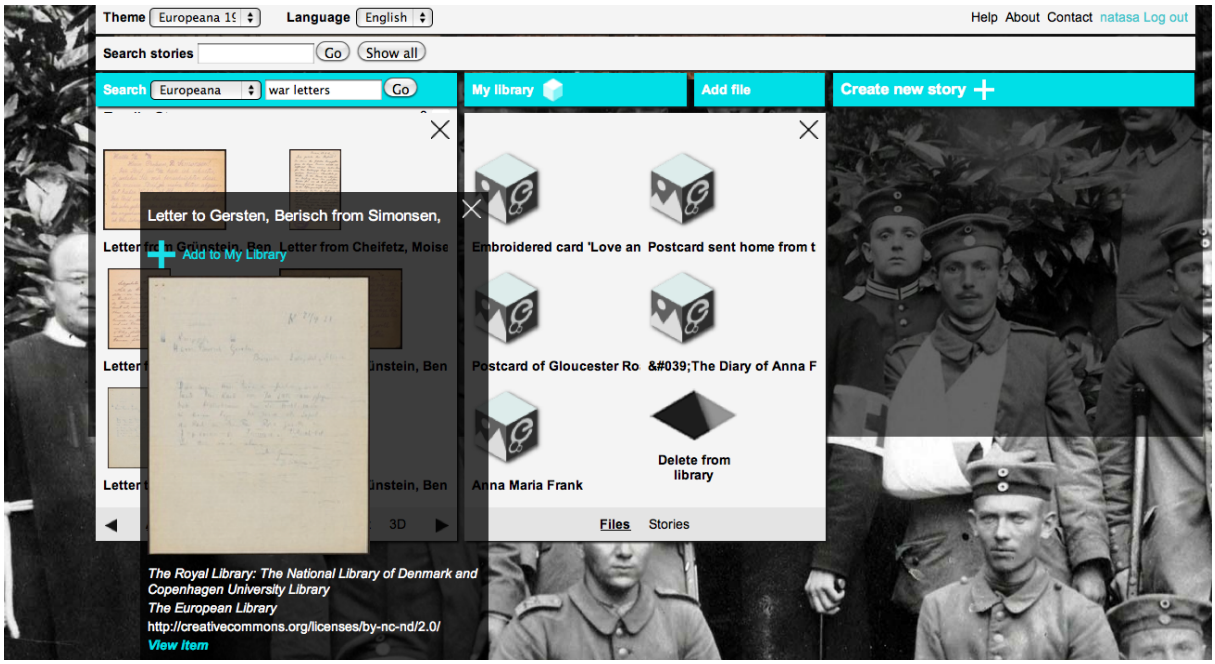


Figure 2-8 Add item from search results to My Library

Future versions of the DSP will include searching of items in additional data sources such as YouTube, Flickr etc.

When searching Europeana with specific keywords, the returned results are displayed in a separate window under the “Search Europeana” field area. When searching DSP uploaded content the results are also returned in a separate window under the “Search DSP” field area. Filtering parameters by item type are included in the bottom of the window. By clicking on each returned item, a preview window appears, containing information about the object, as seen in Figure 2-8. The item preview window provides also a link referred to as ”Add to My Library” that enables the user to add the specific object to his “library”. Alternatively adding an item to “my library” can be done by dragging the item from search results to “my library” window.

Clicking on “my library” opens up a preview window, where all items (files or stories) that belong to user’s library appear, as in 2-8.

“Add file” gives the user the option to choose a file locally stored in his computer and upload it to “My library”, adding a title and a description to it, as illustrated in Figure 2-9.

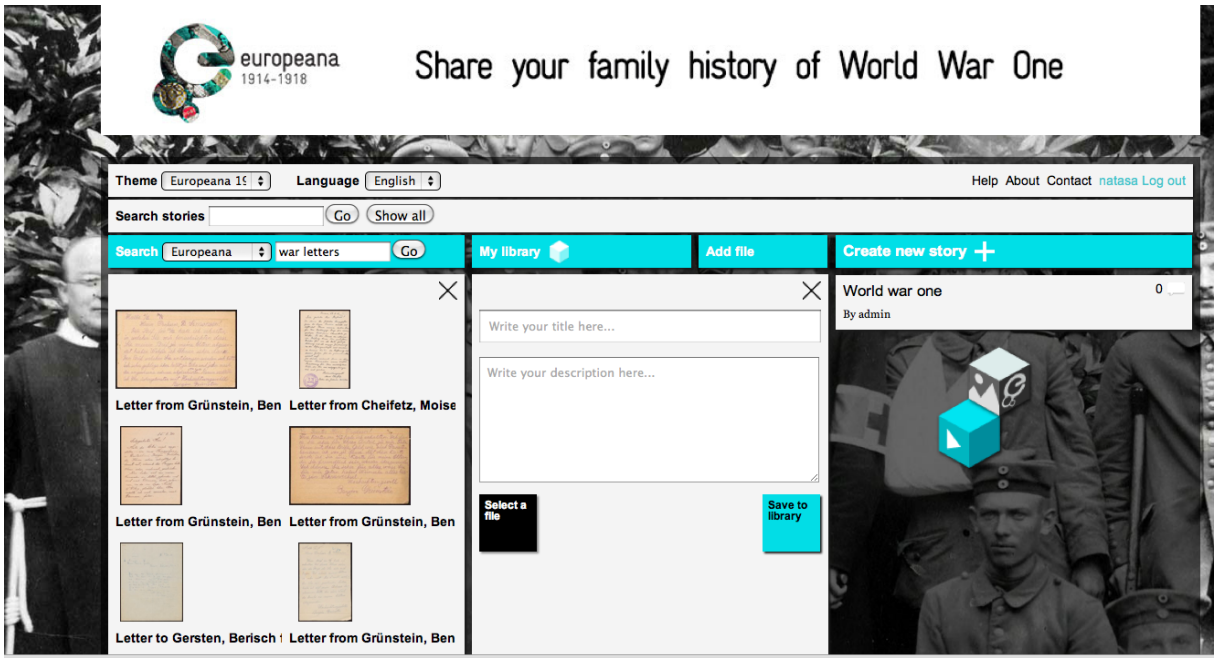


Figure 2-9 Add File

In this first version of the platform only image files can be uploaded, however future versions also will include upload capabilities for other media types (video, audio, text).

2.2.1.6 Create new story

A registered user has the option of creating a new story. In order to do so, the user can search files and explore stories to collect story objects for his new story. He can drag objects of interest to his library or he can upload his own files (as explained above). Then by clicking “Create new story,” a new window opens under the “create new story” field, where as a first step the user can enter the title of the story, some text describing the story, and related tags, as illustrated in Figure 2-10. He can also set the theme of the story and the language.

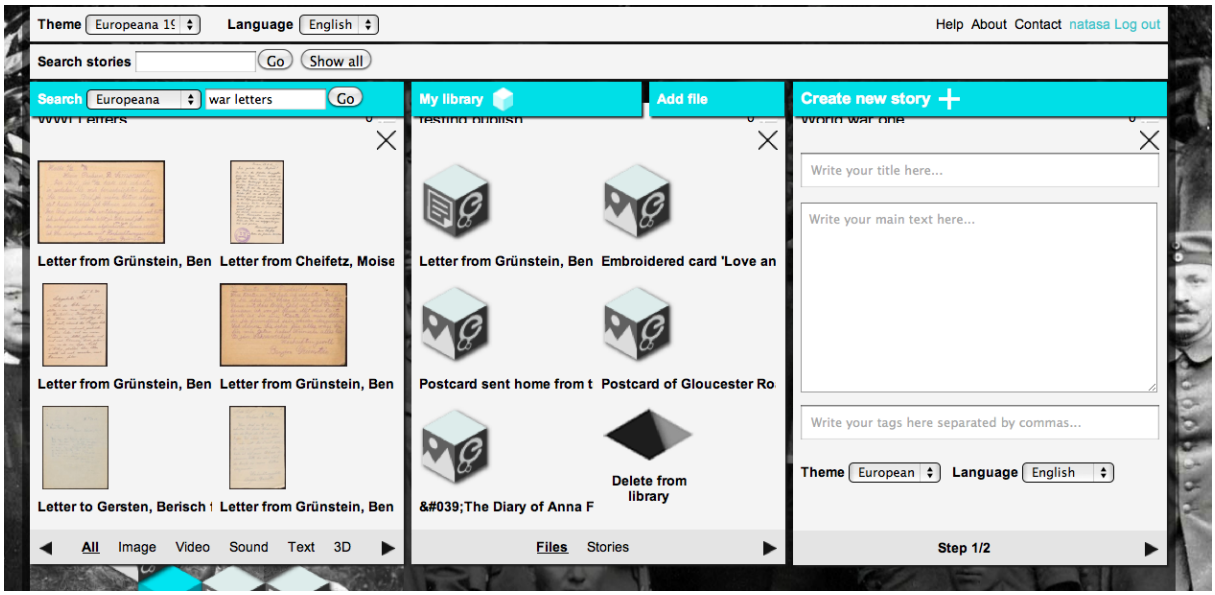


Figure 2-10 Create New Story step1/2

In the next step the user can drag story objects from his library to the cube grid to build his story (see Figure 2-11). It should be noted that a story is only publishable if it contains at least one Europeana object. By pressing the publish button the user can share the newly created story making it available to the story telling platform as seen in Figure 2-12.

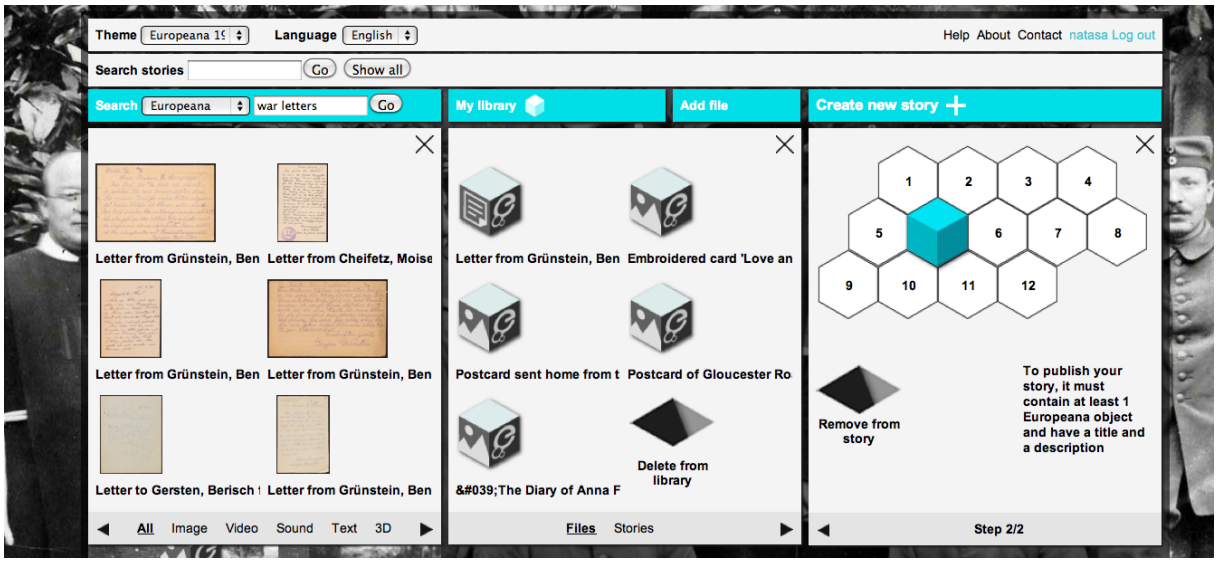


Figure 2-11 Create New Story step2/2

Once the story is created and published the user can see the story under the specific theme's stories, can open it, edit it, and delete it.

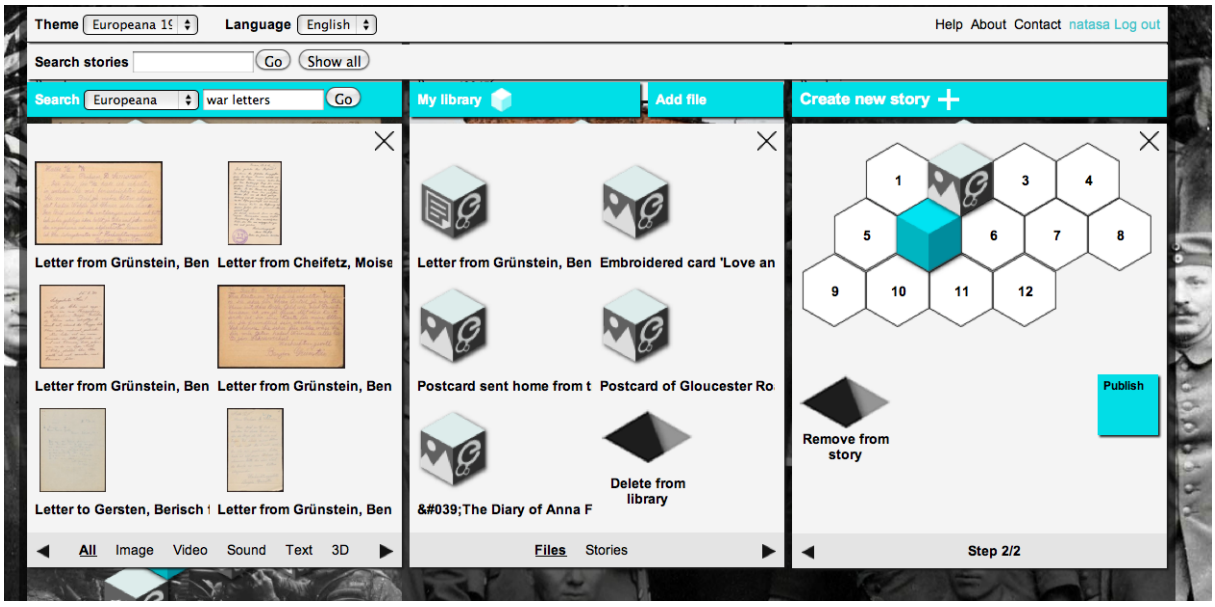


Figure 2-12 Create New Story step2/2: Publish

2.2.1.7 Story Comments

A logged in user has the ability to add comments to existing stories and edit his own comments. Story comments addition functionality is illustrated in Figure 2-13.

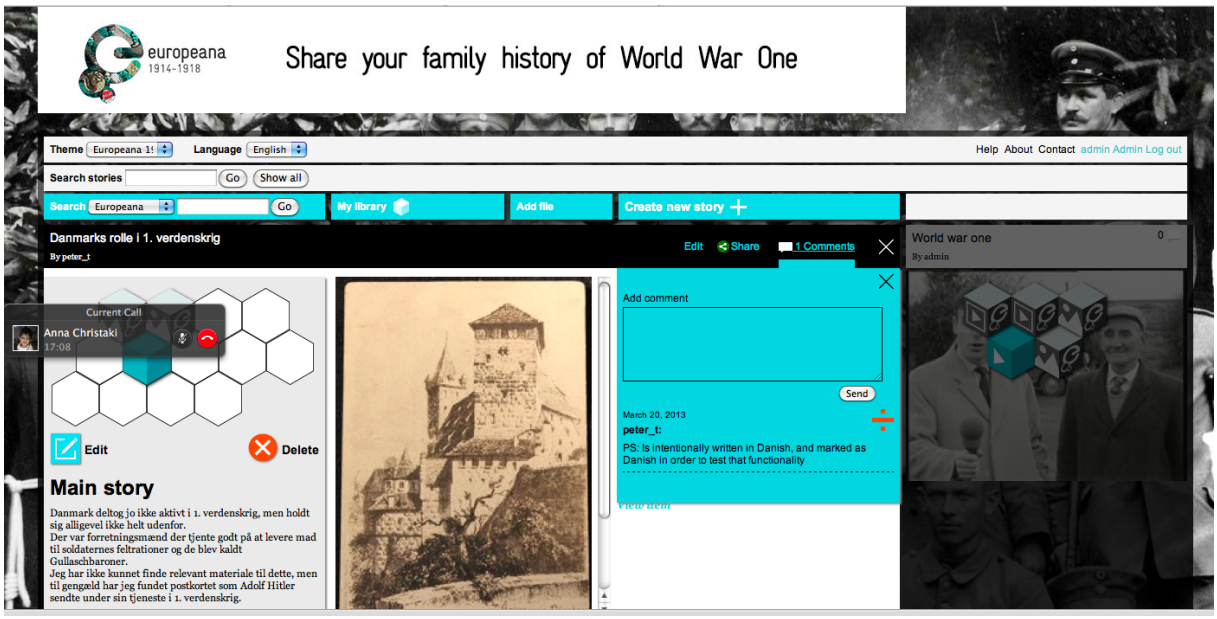


Figure 2-13 Story Comments

2.2.1.8 User roles and Frontend administrative functions

The DSP platform supports three different user roles:

1. Contributors: Users free to create stories and add comments.
2. Editors: Users with editing rights for all stories, story objects and comments
3. Administrators: Editing access to all stories, story objects and comments and additionally access to the administrative console where they can perform user management and theme management actions

When users register to the DSP frontend their role is limited to that of a “contributor”, meaning the user is free to create stories or comment on other stories already published in the DSP. He also has full control over his uploaded content and library contents, story objects and stories which can be deleted and edited at any point.
However when someone logs in to the DSP with an “Admin” or “Editor” role he gets additional functionalities. Administrators and editors can see buttons to edit any story, delete any story, and remove story objects from the DSP completely, thus removing them also from any stories that use them. They also have full access and can edit any comments added to stories.

An administrator’s view of a story object with an additional link to “Delete everywhere” can be seen below. Upon selecting this link, the story object will be removed from any stories that use it and from all the libraries of DSP users.

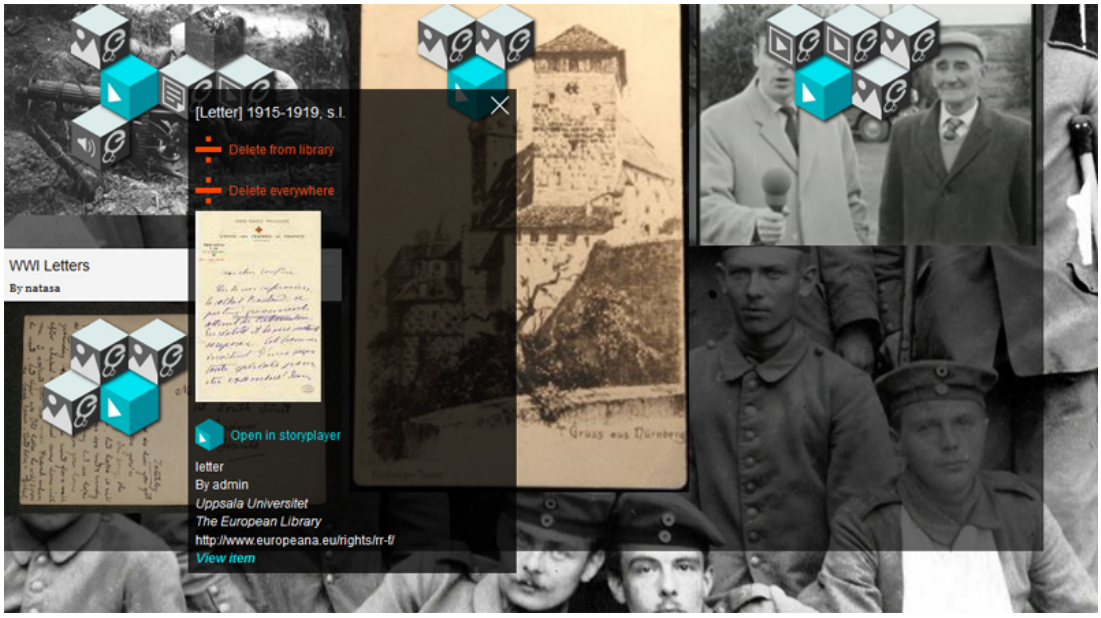


Figure 2-14 Administration view for story object preview: Delete from everywhere link

2.2.2 Administration Console

The administration console interface can be viewed in Figure 2-15. Administration console, which is only accessible to users with the “Administrator” role, provides functionalities like user management, theme management and in future versions of the platform, flagged content management. As already mentioned in 2.2.1.8, story and story object management is not supported by administration console since story management functionalities are operated via frontend UIs. An overview of the administration functionalities can be viewed in the Dashboard as illustrated in Figure 2-15.

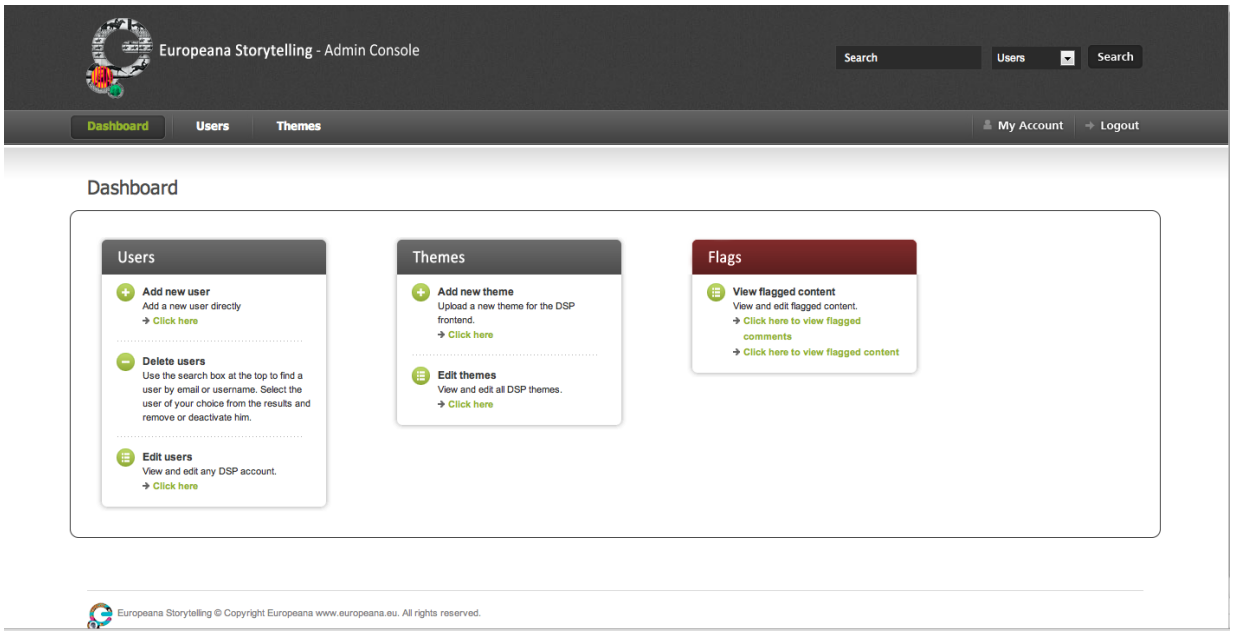


Figure 2-15 Administration Console- Dashboard

By selecting "Users" tab the Users list appears along with the user details, as seen in Figure 2-16. For each user information is provided regarding: username, email, status (active or not), user role (contributor, editor, and administrator), number of stories, date on which the user account was created. The administrator has the authority to edit or permanently delete user accounts (Figure 2-17). New user accounts can be created directly through the administration console.

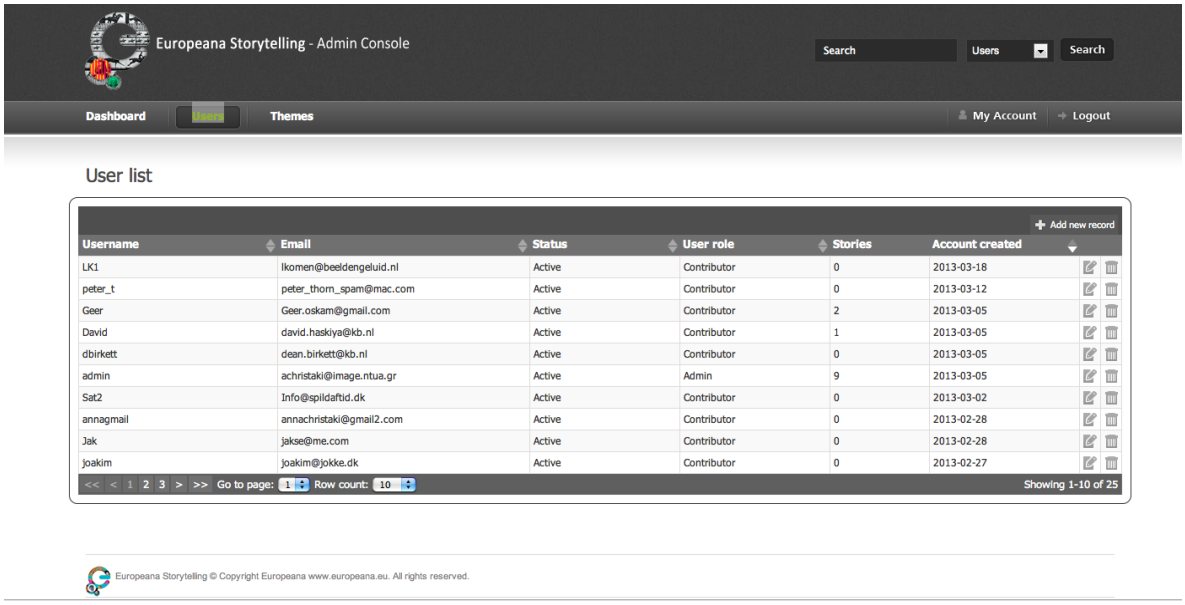


Figure 2-16 Administration Console - User List

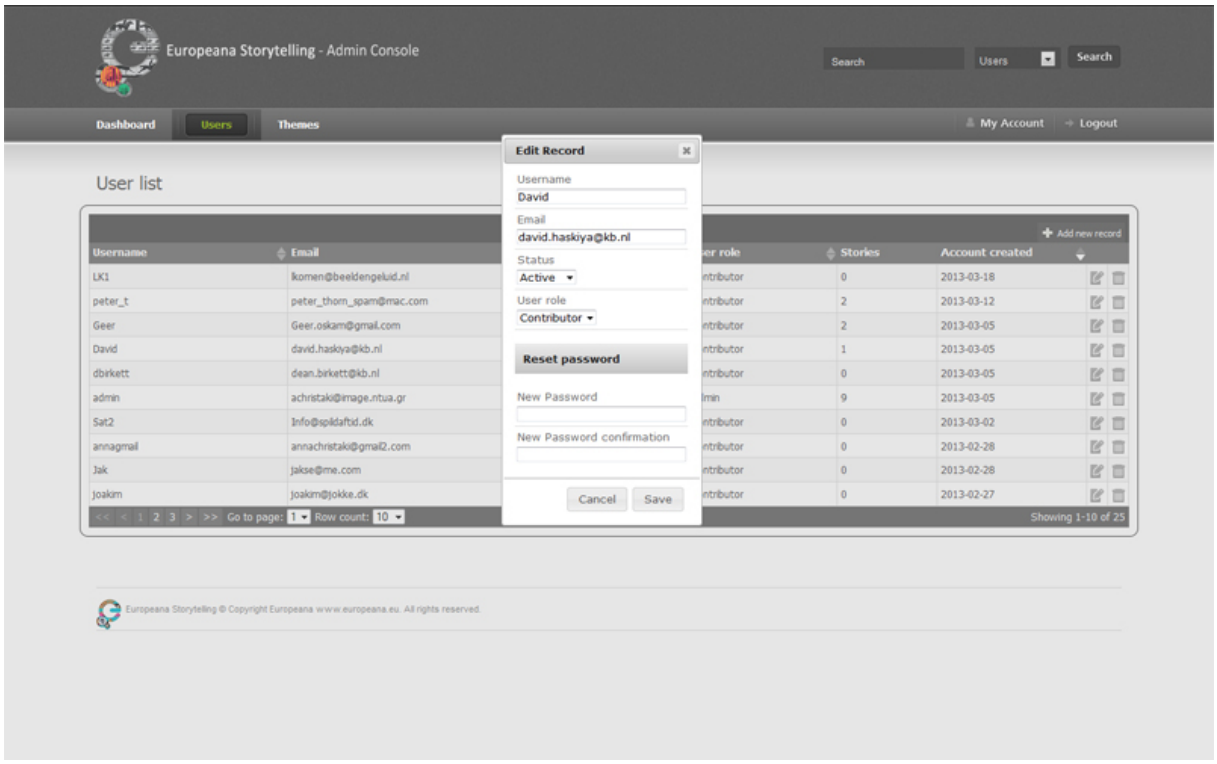


Figure 2-17 Administration Console - Edit User

By selecting the "Themes" tab the theme list appears, as in Figure 2-18. For each theme there is information about: title, description, number of stories under this theme. There is also preview, edit and delete option for each theme. The administrator can add new themes to the DSP (see Figure 2-19).

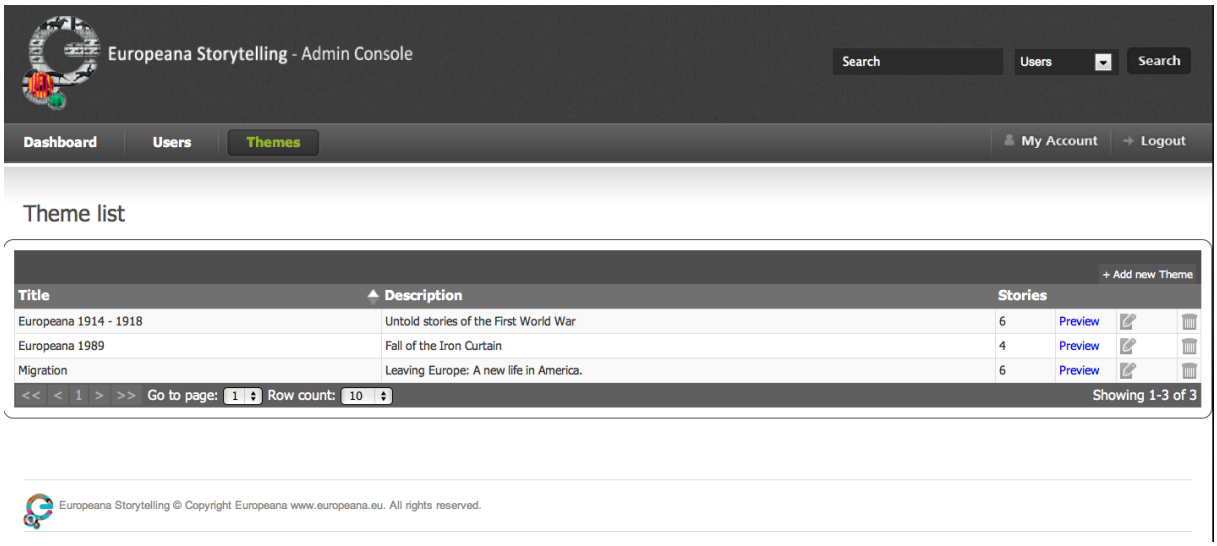


Figure 2-18 Adminidtration Console - Themes

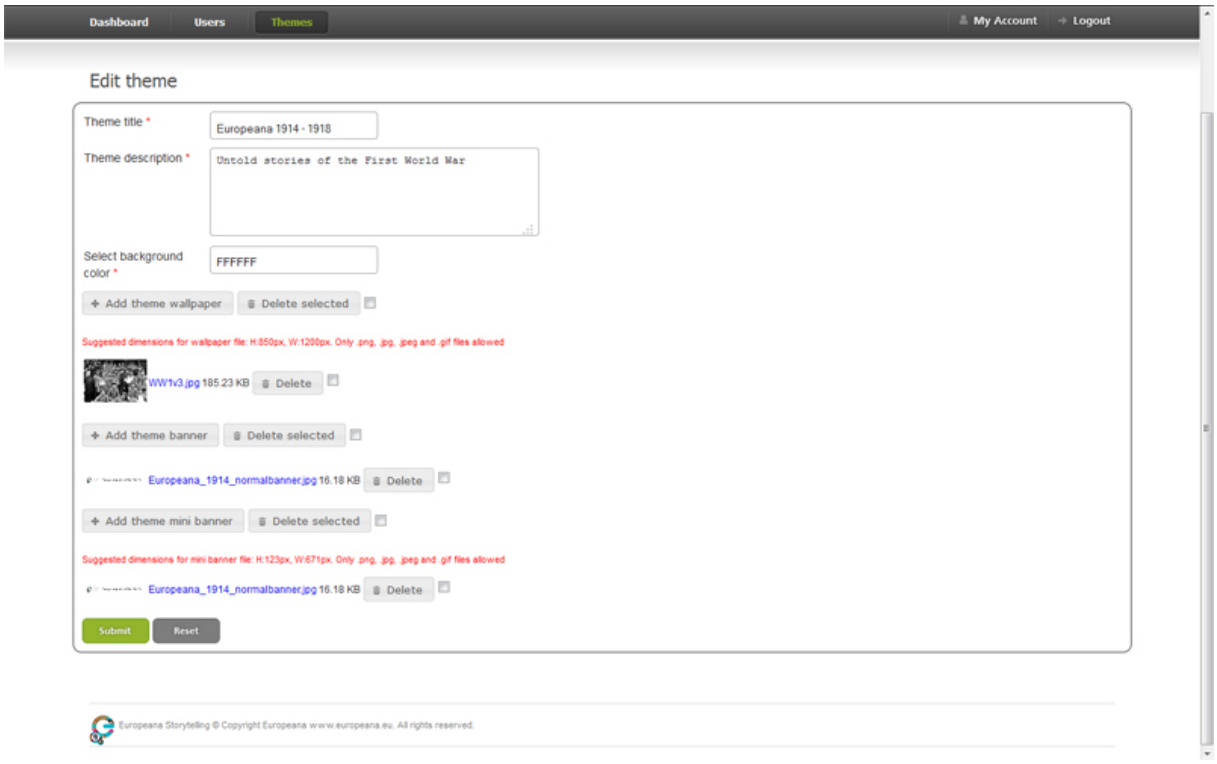


Figure 2-19 Administration Console - Edit Theme

Finally, search for users and themes functionality is also available through the DSP admin console.

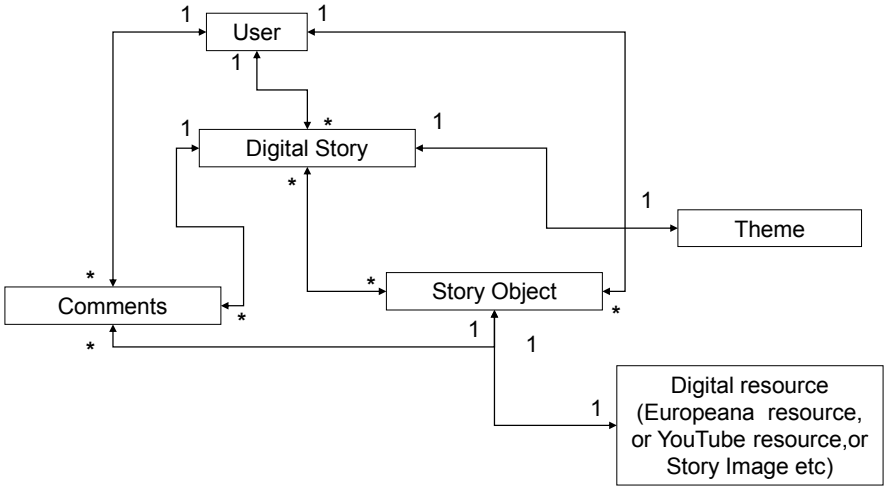
2.3 Data Model

The main entities of the data model used for the digital storytelling platform can be viewed in the following table together with the interactions among them.

A mapping to the EDM corresponding value is also provided for story elements, since all the stories will be eventually converted to the EDM format and made searchable also through the Europeana Platfrom.

Entities:

DigitalStory
StoryObject
User
Story Image
Theme
Story Comments



DigitalStory:

Property Name	Type	EDM Value Type	Mandatory	Cardinality	Description
title	String	Literal	Yes	1	The title of the Digital Story
description	String	Literal	Yes	1	A description provided by the user for the Digital Story
tags	Array of strings	Array of literals	no	1	An array of tags added by the user for indexing/search purposes, display

					of related story objects & stories.
creator	DBRef	reference	Yes	1	Reference to the ID of the user entity who created the story
coverImage	String (URL)	reference	No	1	A URL link for the story's cover image.
isPublished	Boolean	boolean	Yes	1	Indicates if the digital story will be visible or not on browsing/search
isPublishable	Boolean	boolean	Yes	1	If all mandatory fields exist then true.
forReview	Boolean	boolean	No	1	Flags the digital story for review.
id	ID object	reference	Yes	1	A unique ID for the DigitalStory assigned by the system at creation time.
storyObjects	Array with DBRef for story object, story object position, order and scriptPart: [DBRef,int,int ,String]	3D Array [referenc e, int, int,Literal]	Yes	2-12	An array of references to story objects with a position (position in grid) and order (play out order) for each one and a script description. Min size=1 (one Europeana story object . Max 12.)
theme	DBRef	reference	Yes	1	Reference to the id of the theme the digital story belongs to
language	String	Literal	Yes	1	The language used for title & description. ISO 639 will be used
dateCreated	Date	Literal	Yes	1	The date the story was created (auto generated in DB).
license	string	Referenc e	yes	1	the value given here should be the rights statement that applies to the digital representation at the URL. By default set to CC0.

StoryObject:

Property Name	Type	EDM Value type	Mandatory	Cardinality	Description
title	String	Literal	Title or description must be present	1	The title of the Story Object. Inherited from the digital resource or given by the user if digital resource is user generated
description	String	Literal	Title or description must be present	1	A description provided by the user for the Story Object if the digital resource is uploaded by the user else inherited from the digital resource (YouTube, Flickr etc).
provider	String	reference	No	1	Europeana provider
dcCreator	String	reference	No	1	Reference to the profile of the user entity that created the digital resource or the name of the creator of the object. (e.g a link to the youTube user that created the video, a link to the profile of the DSP user that uploaded the object etc). For DSP uploaded content this is same value as creator.
creator	DBRef	reference	Yes	1	Reference to the ID of the DSP user who created this story object
dataProvider	String	reference	No	1	Europeana DataProvider
source	String (url)	reference	Yes	1	The url of the digital resource in its original datasource (e.g. the youTube link where the youTube video resource can be viewed, the Europeana landing page for the item)
tags	Array of	Array of	No	1	An array of tags

	strings	literals			added by the user for indexing/search purposes
type	Enumerated string	Literal	Yes	1	Value will be one of the story object types accepted by the system, e.g. video or audio etc. The type will also be used to display the thumbnail of the story object. Possible values: (“video”,”image”,” sound”,text”, 3D, map)
url	String (url)	reference	Yes	1	A valid URL pointing to a digital resource associated with the Story Object(e.g. youtube video link)
id	ID object	reference	Yes	1	A unique ID for the StoryObject assigned by the system at creation time.
language	String	Literal	No	1	The language used for the story object metadata. ISO 639 will be used
loc	2d array [x,y] Decimals (long,lat)	3d array [x,y,z] of decimals	No	1	Coordinates in decimal degrees of the longitude and latitude associated with the story object
dateCreated	Date (predefined format)	Literal	No	1	A date associated with the story object.
license	String(url)	Reference	No	1	URL of the rights statement that applies to the digital representation at the URL.
sosource	Enumerated string	Literal	yes	1	The story object ‘s data source origin. Values one of:“local”, “Europeana”,”You tube”, “Flickr”.. (all the external datasources and “local” for awareness

					uploaded content). If no value is given on story object creation system will inspect the "source" url field and automatically add a value.
thumbnail	String(url)	reference	no	1	URL to cube thumbnail image for this story object
storyImage	String(ID)	reference	no	1	Gridfs id for story image, only present for story objects generated by user uploaded images.

User:

Property name	Type	Mandatory	Cardinality	Description
id	ID object	Yes	1	Unique id assigned by the system
password	String	Yes	1	Min length=6. Stored in Mongo after MD5 encryption is applied
username	String	Yes	1	Unique
email	String	Yes	1	Unique. Used for log in.
address	String	No	1	
town	String	No	1	
country	String	No	1	
gender	String	No	1	
age	Int	No	1	
role	Enumerated String	Yes	1	Value is one of: admin,editor,contributor. Set by default on registration to "contributor".
accountActive	boolean	yes	1	Indicator for active/inactive accounts. On registration it is set by the system to true.
accountCreated	Date	yes	1	System generated date of user creation

Story Image:

This model is used for the user uploaded images. The story images will be stored on MongoDB GridFS and will be accessed over HTTP just as every other digital resource. The user will only upload the original image and then the rest of the thumbnails etc will be automatically generated, stored and linked (java.awt is used in the backend for image manipulation).

Property name	Type	Mandatory	Cardinality	Description
id	ID object	Yes	1	Unique id assigned by the system
original	String	Yes	1	A unique FileName identifier that points to

				the GridFS location where the Original Image is stored
coverImage	String	Yes	1	A FileName identifier pointing to GridFS for a cover image version of the original one.
thumbnail	String	Yes	1	A thumbnail image to be used in 'search files' results.
objectPreview	String	Yes	1	Pointer to a Preview Image for the Object.
objectThumbnail	String	YES	1	Pointer to a Thumbnail for the Object.(mapped cube)
userId	String	YES	1	The user who upload the image
title	String	Title or description must be present	1	A title for the image
description	String	Title or description must be present	1	Image description

Themes:

Apart from being a parameter of a Digital Story, a theme also provides skinning options for the ui:

Property	Type	Mandatory	Cardinality	Description
id	ID object	Yes	1	Unique id assigned by the system
title	String	Yes	1	The theme title as it will appear on the theme list. Max length=30 chars
description	String	Yes	1	Theme description (to be used for sub banner). Max length=300 chars
wallpaper	DBRef	Yes	1	A reference to the image id to be used for the background of the theme. The image is stored in GridFS (specific dimensions required)
banner	DBRef	Yes	1	A reference to the image id that will be used as theme banner. Image should contain the theme title, Europeana logo and logo of partner domain. (specific dimensions required)
minibanner	DBRef	Yes	1	A reference to the image used as theme banner on smaller screen sizes. Minibanner will be generated by the system when banner is loaded.

				(specific dimensions required)
background	String	yes	1	Hex notation of the background color used for the main screen
defaultTheme	boolean	yes	1	Indicates if this is the default theme (active when user chooses to search in all themes)

Story comments:

Property	Type	Mandatory	Cardinality	Description
id	ID object	Yes	1	Unique id assigned by the system
text	String	Yes	1	The text of the comment
userId	DBRef	Yes	1	The id of the user who created the comment
storyId	DBRef	Yes	1	The id of the story where the comment belongs
dateCreated	Date	yes	1	System generated date of comment creation

3 Other platforms in use by Europeana for User Generated Content

Europeana Awareness is running two campaigns aiming to collect stories and media from the public. The campaigns are centred on the First World War and the great political changes of 1989 in Europe.

In terms of timing these campaigns needed to begin earlier than the scheduled delivery of the fully fledged Digital Storytelling Platform. For this reason Europeana turned to existing partners and their platforms making only the necessary adaptations for those to work in the context of Europeana. Those two platforms, one already in production and the other undergoing customisation, are briefly described below.

3.1 1914-1918/RunCoCo

For the community collection of stories and media related to the First World War the technical solution used is based on the RunCoCo [6] software. RunCoCo, short for Run Community Collections Online, is developed by Europeana Awareness partners Oxford University and was adopted because it's the software used in the Great War Archive project [7].

For the purposes of supporting the Europeana 1914-1918 campaign RunCoCo has gone through two rounds of customisation and adaptation.

The first round of development was completed in Q1 2011 and was a very basic adaptation with no functionality added or extended. However, it was enough to support the community collection days throughout 2011 and the first half of 2012.

The second round of development was initiated in January 2012 and completed by end of June the same year. This round of development was more ambitious aiming at a re-design of the user experience for the creator of stories, the cataloguers supporting users at collection

days and the user accessing the stories. A more detailed description of the consideration that went into the re-design is available in a series of blog posts here. [8].

The development resulted in the Europeana 1914-1918 site [9] available today and what we consider the first truly operational version of the site. After it went into operation it has since been update with the addition of the ability to search for institutional World War I-related content in Europeana (via the Europeana API) and a component allowing harvesting of the entire Europeana 1914-1918 collection via the OAI-PMH standard. These additions taken together means that the site functions as a stand-alone site where data flows to and from the main Europeana database via open technical interfaces and standards.

The source code for Europeana 1914-1918 is available on Europeana Labs [10]. It is available for others to re-use and adapt as it is released as Open Source.

3.2 1989/Historypin

For the Europeana 1989 community collection campaign Europeana has decided to form a partnership with an existing not-for-profit community driven site namely Historypin [11]. Historypin is developed and maintained by We Are What We Do [12], a UK non-profit.

There are multiple reasons for joining an existing site and re-using an existing platform:

- We join an active community of users accustomed to annotate and improve each others content
- We decrease sustainability issues post-project. After Europeana Awareness ends we have no dedicated resources to further develop RunCoCo. As an organisation we have a hard time keeping pace with best practices for user generated content sites as it's not part of our core skill sets.
- We can offer Europeana 1989 users features that would be very costly to develop ourselves. These features include advanced geo-tagging down to the street view level, augmented reality style overlays of old photographs on current photographs [13] annotations allowing users to help each other in providing more detailed information about objects and dedicated mobile applications for accessing and contributing content on the go.

The first version of Europeana 1989 on Historypin will be available in June 2013. Further development will take place within the Europeana Creative project (where We Are What We Do are consortium members).with a focus on automating the data flow between Historypin and Europeana

4 Conclusions

This document provided a detailed description of the Digital Storytelling Platform in terms of architecture, design, data model and APIs. Any information provided for the aforementioned constitutional parts of the platform refer to the current version (version1), which was released end of February 2013 (as marked by MS9) and is subject to usability tests that will take place in the following period. A combination of in-person tests and remote tests on different platforms (desktop, tablets, etc.) as well as technical browser and device compatibility tests under predefined test scenarios are planned for the following period. Evaluation of the platform using the obtained usability test results and insights will contribute to the improvement of the platform in future version releases.

References

[1]. RESTful-Wikipedia : http://en.wikipedia.org/wiki/Representational_state_transfer

[2]. MONGODB homepage: http://www.mongodb.org/

[3]. Play Framework Homepage: http://www.playframework.org/

[4]. Open Images homepage: http://www.openimages.eu/

[5]. Solr homepage: http://lucene.apache.org/solr/

[6]. RunCoCo: http://projects.oucs.ox.ac.uk/runcoco/

[7]. Great War Archive Project : http://www.oucs.ox.ac.uk/ww1lit/gwa

[8]. http://kadmeianletters.wordpress.com/2012/06/10/europeana-1914-1918-re-designed/

[9]. Europeana 1914-1918 : http://www.europeana1914-1918.eu/en

[10]. Europeana Labs: http://europeanalabs.eu/browser/europeana1914-1918

[11]. HistoryPin: http://www.historypin.com/

[12] . We Are What We Do: http://wearewhatwedo.org/

[13]. http://www.historypin.com/map/ - !/geo:50.292724,2.780468/zoom:19/

Appendix 1. API

Call Route	HTTP Method Type	Serialization
http://{hostname}/awareness/digitalStories/save	POST	JSON
http://{hostname}/awareness/digitalStories/delete	GET	JSON
http://{hostname}/awareness/digitalStories/list	GET	JSON
http://{hostname}/awareness/digitalStories/{id}	GET	JSON
http://{hostname}/awareness/digitalObjects/save	POST	JSON
http://{hostname}/awareness/digitalObjects/delete	GET	JSON
http://{hostname}/awareness/digitalObjects/list	GET	JSON
http://{hostname}/awareness/digitalObjects/{id}	GET	JSON
http://{hostname}/awareness/Users/save	POST	JSON
http://{hostname}/awareness/Users/delete	GET	JSON
http://{hostname}/awareness/Users/list	GET	JSON
http://{hostname}/awareness/Users/{id}	GET	JSON
http://{hostname}/awareness/Users/login	POST	JSON
http://{hostname}/awareness/images/{id}	GET	JSON
http://{hostname}/awareness/FileReader/save	POST	octet-stream
http://{hostname}/awareness/EuropeanaSearch/search	POST	JSON
http://{hostname}/awareness/EuropeanaSearch/record	POST	JSON
http://{hostname}/awareness/Themes/save	POST	JSON
http://{hostname}/awareness/Themes/delete	GET	JSON
http://{hostname}/awareness/Themes/default	GET	JSON
http://{hostname}/awareness/Themes/list	GET	JSON
http://{hostname}/awareness/Themes/{id}	GET	JSON
http://{hostname}/awareness/Themes/{id}/stories	GET	JSON
http://{hostname}/awareness/Users/{id}/stories	GET	JSON
http://{hostname}/awareness/Users/{id}/objects	GET	JSON
http://{hostname}/awareness/StoryImages/save	POST	JSON
http://{hostname}/awareness/StoryImages/{id}	GET	JSON
http://{hostname}/awareness/StoryImages/delete	GET	JSON
http://{hostname}/awareness/digitalStoryComment/{id}	GET	JSON
http://{hostname}/awareness/digitalStories/comments/save	POST	JSON
http://{hostname}/awareness/digitalStories/comments/delete	GET	JSON
http://{hostname}/awareness/digitalStories/{id}/comments	GET	JSON
http://{hostname}/awareness/Users/{id}/comments	GET	JSON
http://{hostname}/awareness/session/user	GET	JSON
http://{hostname}/awareness/Users/logout	GET	JSON
http://{hostname}/awareness/Admin/deleteObject	GET	JSON

API Calls Detailed Description:

http://{hostname}/awareness/digitalStories/save

Description: Saves on MONGODB a valid digital story JSON object.

Parameters: none, the actual object is part of the body of the HTTP Post request.

Return: The saved JSON object, together with the unique ID which is assigned by MongoDB.

Comments: The same method is also used for updating a digital story object, the only requirement is that the submitted JSON object should contain the assigned MongoDB ID, if and only if this ID exists then the object is updated, otherwise a new object is created and a new ID is assigned.

http://{hostname}/awareness/digitalStories/delete

Description: Deletes an existing digital story JSON object from MongoDB.

Parameters:

Parameter Name	Mandatory	Description
id	Yes	The id of the story to be deleted

Return: an http code 200 if the operation was successful.

http://{hostname}/awareness/digitalStories/list

Description: Lists all the available digital stories that are stored in MongoDB. It supports pagination.

Parameters:

Parameter Name	Mandatory	Description
from	No	Indicates the starting point from where the data will be fetched.
to	no	Indicates the ending point for the returned objects.

Return: A json document containing an array of object IDs.

Query: http://localhost:9000/awareness/digitalStories/list?to=4&from=1

{ "totalSize" : 7 , "start" : 1 , "to" : 4 , "values" : ["5058490f8aa1d838fc56a829" , "50584c078aa135e16443529c" , "5058a1888aa144d4d986d9e9"]}

http://{hostname}/awareness/digitalStories/{id}

Description: Retrieves a specific user story based on its unique ID.

Parameters: none

Return:

- An http code 200 with the complete JSON object of a story
- an http code 404 (not found) if no digital story was found

http://{hostname}/awareness/digitalObjects/save

Description: Saves on MongoDB a valid digital object JSON object.

Parameters: none, the actual object is part of the body of the HTTP Post request.

Return: The saved JSON object, together with the unique ID which is assigned by MongoDB.

Comments: The same method is also used for updating a "digital object" JSON object, the only requirement is that the submitted JSON object should contain the assigned MongoDB ID, if and only if this ID exists then the object is updated, otherwise a new object is created and a new ID is assigned.

http://{hostname}/awareness/digitalObjects/delete

Description: Deletes an existing digital object.

Parameters:

Parameter Name	Mandatory	Description
id	Yes	The id of the object to be deleted

Return: an http code 200 if the operation was successful.

http://{hostname}/awareness/digitalObjects/list

Description: Lists all the available digital objects that are stored in MongoDB. It supports pagination.

Parameters:

Parameter Name	Mandatory	Description
from	No	Indicates the starting point from where the data will be fetched.
to	no	Indicates the ending point for the returned objects.

Return: A json document containing an array of object IDs.

Query: http://localhost:9000/awareness/digitalStories/list?to=3&from=0

{ "totalSize" : 8 , "start" : 0 , "to" : 3 , "values" : ["5058490f8aa1d838fc56a829" , "50584c078aa135e16443529c" , "5058a1888aa144d4d986d9e9"]}

http://{hostname}/awareness/digitalObjects/{digital object id}
Description: Retrieves a specific digital object based on its unique ID.

Parameters: none

Return:

- An http code 200 with the complete JSON object of a "digital object".
- an http code 404 (not found) if no digital object was found

http://{hostname}/awareness/users/save

Description: Saves on MongoDB a valid user JSON object.
Parameters: none, the actual object is part of the body of the HTTP Post request.
Return: The saved JSON object, together with the unique ID which is assigned by MongoDB.
Comments: The same method is also used for updating a user object, the only requirement is that the submitted JSON object should contain the assigned MongoDB ID, if and only if this ID exists then the object is updated, otherwise a new object is created and a new ID is assigned.

http://{hostname}/awareness/Users/{id}

Description: Retrieves a specific user based on its unique ID.

Parameters: none

Return:

- an http code 200 with the JSON object of a user.
- an http code 404 (not found) if no user was found

http://{hostname}/awareness/Users/delete

Description: Deletes an existing user JSON object from both the MongoDB

Parameters:

Parameter Name	Mandatory	Description
Id	Yes	The id of the user to de deleted

Return: an http code 200 if the operation was successful.

http://{hostname}/awareness/Users/list

Description: Lists all the user ids that are stored in MongoDB. It supports pagination.

Parameters:

Parameter Name	Mandatory	Description
from	No	Indicates the starting point from where the data will be fetched.
to	no	Indicates the ending point for the returned objects.

Return: A json document containing an array of user object IDs.

Query: http://localhost:9000/awareness/Users/list?to=4&from=1

{ "totalSize" : 4 , "start" : 0 , "to" : 4 , "values" : ["5058490f8aa1d838fc56a829" , "50584c078aa135e16443529c" , "5058a1888aa144d4d986d9e9" ,

```
"5058a28e8aa103f3cfde5d52"]}
```

http://{hostname}/awareness/Users/login

Description: Logs user into the DSP.

Parameters:

Parameter Name	Mandatory
email	yes
password	Yes

Return:

- an http code 200 if the operation was successful with a JSON user object
- an http code 404 (not found) if email was not found
- an http code 403 (forbidden) if email/password combination is not valid.

http://{hostname}/awareness/images/{id}

Description: Retrieves an image from GridFS using the unique id (to be used as the img src attribute)

Parameters: none

Return: the image.

http://{hostname}/awareness/FileReader/save

Description: saves a file to the system temp folder.

Parameters: none, the file is the body of the HTTP Post request.

Return:

- On success: an http code 200 and a json document with the success status and the name of the temporary file
- On failure: http error code

Example reply:

```
{success: true, fname: 'Awareness8157498529327360565cpy'}
```

http://{hostname}/awareness/EuropeanaSearch/search

Description: Free text search that returns Europeana items using Europeana Search Api v2.

Parameters:

Parameter Name	Mandatory	Description
term	yes	The search term(s).
pageNumber	Yes	Results are paginated (6 results per page). This parameter indicates the number of page results to retrieve.

Return:

- an http code 200 if the operation was successful with a JSON response and the following parameters:

Parameter Name	Description
itemsCount	How many results on current page
totalResults	Total result number
term	Search term/s
pageNumber	Number of page results retrieved
items	An array of Europeana items. (europeanaid, title, type, rights, provider, url, source)

Example reply:

{ "itemsCount" : "6" , "totalResults" : "425" , "term" : "berlin wall" , "pageNumber" : 1 , "items" : [{ "europeanaid" : "/2022005/79B3CE605EE1B1C4DECE85DF161C45B2BCCAF6E2" , "type" : "IMAGE" , "source" : "http://preview.europeana.eu/portal/record/2022005/79B3CE605EE1B1C4DECE85DF161C45B2BCCAF6E2.html?utm_source=api&utm_medium=api&utm_campaign=api2demo" , "title" : "After exchanging of pieces of land a new wall was built. In the background the new wall. Near Bernauer Strasse." , "provider" : "International Institute of Social History" , "url" : "http://europeanastatic.eu/api/image?uri=http%3A%2F%2Fhdl.handle.net%2F10622%2F30051000468691%3Flocatt%3Dview%3Alevel2&size=LARGE&type=IMAGE" , "license" : "http://www.europeana.eu/rights/rr-f/"} ,…}]}

- an http code 400 on malformed input parameters
- an http code 500 on error getting results.

http://{hostname}/awareness/EuropeanaSearch/record

Description: Retrieves Europeana item description

Parameters:

Parameter Name	Mandatory	Description
recid	yes	The item id

Return:

- an http code 200 if the operation was successful with a JSON response containing the item description:

Example reply:

```
{"description" : "Selling pieces of the wall."}
```

- an http code 400 on malformed input parameters
- an http code 500 on error getting results.

http://{hostname}/awareness/Themes/save

Description: Saves the theme json object to MongoDB.

Parameters: none, the actual object is part of the body of the HTTP Post request.

Return:

- an http code 200 if the operation was successful with a JSON response containing the theme details:

Example reply:

{"id":"50fff30ca3d5be03a00bcd5f","title":"Test theme","description":"A theme description","wallpaper":"50fff30aa3d5be03a00bcd59","banner":"50fff30ba3d5be03a00bcd5b","minibanner":"50fff30ca3d5be03a00bcd5d","background":"FFA578","defaultTheme":false}

*images from can be displayed as src='http://{host}/awareness/images/{id}'

- an http code 400 on malformed/missing input parameters

http://{hostname}/awareness/Themes/delete

Description: Deletes an existing theme.

Parameters:

Parameter Name	Mandatory	Description
id	Yes	The id of the theme object to be deleted

Return:

- an http code 200 if the operation was successful.
- an http code 400 if the theme is used in stories
- an http 500 code on system error

http://{hostname}/awareness/Themes/default

Description: Returns the json of the default theme.

Parameters: none

Return:

- an http code 200 if the operation was successful.

Example reply:

{"id":"50fff30ca3d5be03a00bcd5f","title":"All themes","description":"","wallpaper":"50fff30aa3d5be03a00bcd59","banner":"50fff30ba3d5be03a00bcd5b","minibanner":"50fff30ca3d5be03a00bcd5d","background":"FFA578","defaultTheme":yes}

http://{hostname}/awareness/Themes/list

Description: Returns a list of all theme ids with their titles.

Parameters: none

Return:

- an http code 200 if the operation was successful.

Example reply:

{ "totalSize" : 3 , "values" : [{ "id" : "50fe79f70f44cff3684af1ce" , "title" : "World War II"} , { "id" : "50fe8eeb74d45000b2c1563f" , "title" : "Test theme 2"} , { "id" : "50fff30ca3d5be03a00bcd5f" , "title" : "Test theme"}]}

http://{hostname}/awareness/Themes/{id}

Description: Retrieves a specific theme based on its unique ID.

Parameters: none

Return:

- An http code 200 with the complete JSON object of a theme.
- an http code 404 (not found) if no theme was found

http://{hostname}/awareness/Themes/{id}/stories

Description: Returns all the published stories that belong to a theme with the given id

Parameters:

Parameter Name	Mandatory	Description
from	No	Indicates the starting point from where the data will be fetched. Starting value=0
to	no	Indicates the ending point for the returned objects.

Return:

- A json document containing an array of stories and their story objects.Every story includes the following parameters: id, title, coverimage, published and an array of story objects. Every story object in array includes the following parameters: id, position, thumbnail.

Example reply:

{ "from" : 1 , "to" : 2 , "totalSize" : 2 , "stories" : [{ "id" : "50fe5e340364dcb5a18bb295" , "creator" : "50aa32330364256730e6e48b" , "creator_uname" : "jokke" , "coverimage" : null , "title" : "Some updated story" , "published" : true , "storyobjects" : [{ "storyobject_id" : "50fd1bc803642ca1e71f4d90" , "position" : 4 , "thumbnail" : "images/cubes/image-eu.png"} , { "storyobject_id" : "50b4bbe4e4b0a2be4f2ed1c8" , "position" : 6 , "thumbnail" : "images/cubes/youtube.png"}]}]}

*number of stories returned=to-from, totalSize=total story count

http://{hostname}/awareness/Users/{id}/stories

Description: Returns all the stories where creator id={id}

Parameters:

Parameter Name	Mandatory	Description
from	No	Indicates the starting point from where the data will be fetched. Starting value=0
to	no	Indicates the ending point for the returned objects.

Return:

- A json document containing an array of stories and their story objects.Every story ‘s complete json is returned.

Example: http://localhost:9000/awareness/Users/50d44598330164948796 37c2/stories?from=1&to=2

{ "totalSize" : 3 , "start" : 1 , "to" : 2 , "values" : [{ "className" : "gr.ntua.ivml.awareness.persistent.DigitalStory" , "_id" : { "$oid" : "50b4c148e4b0a2be4f2ed1cc"} , "title" : "That's one small step for man, one giant leap for mankind" , "coverImage" : "none" , "isPublished" : false , "isPublishable" : false , "forReview" : false , "storyObjects" : [{ "StoryObjectID" : "50b4bd68e4b0a2be4f2ed1c9" ,

"scriptPart" : "Why i use this" , "position" : 1 , "order" : 1} , { "StoryObjectID" : "50b4bbe4e4b0a2be4f2ed1c8" , "scriptPart" : "Why i use that" , "position" : 2 , "order" : 2}] , "dateCreated" : { "$date" : "2013-01-24T15:44:36.547Z"}}]}

*number of stories returned=to-from, totalSize=total story count

http://{hostname}/awareness/Users/{id}/objects

Description: Returns all the story objects where creator id={id}

Parameters:

Parameter Name	Mandatory	Description
from	No	Indicates the starting point from where the data will be fetched. Starting value=0
to	no	Indicates the ending point for the returned objects.

Return:

- A json document containing an array of story objects. Every story object complete json is returned.

Example: http://localhost:9000/awareness/Users/50d44598330164948796 37c2/stories?from=1&to=2

{ "totalSize" : 1 , "start" : 0 , "to" : 1 , "values" : [{ "className" : "gr.ntua.ivml.awareness.persistent.StoryObject" , "_id" : { "$oid" : "50b4c096e4b0a2be4f2ed1cb"} , "title" : "DE EERSTE MENS OP DE MAAN" , "description" : "Vanuit een studio van de NOS in Hilversum doet presentator Henk Terlingen verslag van de maanreis van de Apollo-11." , "provider" : "http://www.beeldengeluid.nl" , "creator" : "50d44598330164948796 37c2" , "source" : "http://www.europeana.eu/portal/record/09209/73F5B38795F46D6D0F9D17C3A0194B280D63374D.html" , "type" : "video" , "url" : "http://www.europeana.eu/portal/record/09209/73F5B38795F46D6D0F9D17C3A0194B280D63374D.html" , "language" : "nl" , "dateCreated" : { "$date" : "2012-11-27T13:31:02.047Z"} , "sosource" : "Europeana" , "thumbnail" : "images/cubes/video-eu.png"}]}

*number of objects returned=to-from, totalSize=total object count for user

http://{hostname}/awareness/StoryImages/save

Description: Saves the user uploaded image to GridFS, does all needed transformations, then saves all derived images to MongoDB in StoryImages collection and creates a new StoryObject containing the image.

Parameters: none, the actual object is part of the body of the HTTP Post request.

Return:

- an http code 200 if the operation was successful with a JSON response containing the story object details:

Example reply:

{ "className" : "gr.ntua.ivml.awareness.persistent.StoryObject" , "_id" : { "$oid" : "510bb32034c356dd95836fac"} , "title" : "Testing upload 3" , "description" : "" , "creator" : "50d4459833016494879637c2" , "source" : "localhost" , "type" : "image" , "url" : "510bb31f34c356dd95836fa5" , "language" : "en" , "dateCreated" : { "$date" : "2013-02-01T12:20:48.754Z"} , "sosource" : "local" , "thumbnail" : "510bb31f34c356dd95836fa7" , "storyImage" : "510bb31f34c356dd95836fab"}

*thumbnail can be displayed as src='http://{host}/awareness/images/{id}'

an http code 400 on malformed/missing input parameters

http://{hostname}/awareness/StoryImages/{id}

Description: Retrieves a specific story image based on its unique ID.

Parameters: none

Return:

- An http code 200 with the complete JSON object of a story image.
- an http code 404 (not found) if no theme was found

http://{hostname}/awareness/StoryImages/delete

Description: Permanently removes a user uploaded image from DSP. To be used by admins when searching through DSP image files to remove them permanently.It will remove all story objects that use that image and update any stories using those story objects.

Parameters:

Parameter Name	Mandatory	Description
id	Yes	The id of the user uploaded image to be deleted

Return:

- an http code 200 if the operation was successful.
- an http code 400 if id was not found
- an http 500 code on system error

http://{hostname}/awareness/session/user

Description: Retrieves the user object from the session uid.

Parameters: none

Return:

- An http code 200 with the complete JSON object of a user.
- an http code 404 (not found) if no user was attached to session

http://{hostname}/awareness/Admin/deleteObject

Description: Permanently removes story object from DSP.

If the story object was generated from user uploaded image then the image will be deleted from the image service and all story objects that were created from that image will also be deleted. Stories that used that story image will be updated and will no longer include the story objects generated from the image.

If the story object was generated from an external datasource the object will be removed from all the stories it was used in and it will be deleted from DB. (any copies of that object were created they will not be removed from DSP)

Parameters:

Parameter Name	Mandatory	Description
id	Yes	The id of the story object to be deleted

Return:

- an http code 200 if the operation was successful.
- an http code 400 if id was not found
- an http 500 code on system error

http://{hostname}/awareness/Users/logout

Description: Clears the session parameters (user id is removed)

Parameters: none

Return:

- An http code 200

http://{hostname}/awareness/Users/{id}/comments

Description: Returns all the user comments where creator userid={id}

Parameters:

Parameter Name	Mandatory	Description
from	No	Indicates the starting point from where the data will be fetched. Starting value=0

to	no	Indicates the ending point for the returned objects.

Return:

- A json document containing an array of story comments.Every story comment's complete json is returned.

Example: http://localhost:9000/awareness/Users/ 50810932bf9594d11fa039bd/comments

{ "totalSize" : 2 , "start" : 0 , "to" : 2 , "values" : [{ "className" : "gr.ntua.ivml.awareness.persistent.DigitalStoryComment" , "_id" : { "$oid" : "51266d919614e937b69a5d3b"} , "userId" : "50810932bf9594d11fa039bd" , "storyId" : "510ce375e4b02307557b9ae1" , "text" : "this is my second comment!!!" , "dateCreated" : { "$date" : "2013-02-21T18:55:13.832Z"}} , { "className" : "gr.ntua.ivml.awareness.persistent.DigitalStoryComment" , "_id" : { "$oid" : "51266d5d9614e937b69a5d3a"} , "userId" : "50810932bf9594d11fa039bd" , "storyId" : "510ce375e4b02307557b9ae1" , "text" : "this is my very first comment!!!" , "dateCreated" : { "$date" : "2013-02-21T18:54:21.347Z"}}]}

*number of objects returned=to-from, totalSize=total comment count for user

http://{hostname}/awareness/digitalStories/{id}/comments

Description: Returns all the story comments where story id={id}

Parameters:

Parameter Name	Mandatory	Description
from	No	Indicates the starting point from where the data will be fetched. Starting value=0
to	no	Indicates the ending point for the returned comments.

Return:

- A json document containing an array of story comments.Every story comment's complete json is returned.

Example: http://localhost:9000/awareness/digitalStories/ 510ce375e4b02307557b9ae1/comments

{ "totalSize" : 2 , "start" : 0 , "to" : 2 , "values" : [{ "className" : "gr.ntua.ivml.awareness.persistent.DigitalStoryComment" , "_id" : { "$oid" : "51266d919614e937b69a5d3b"} , "userId" : "50810932bf9594d11fa039bd" , "storyId" : "510ce375e4b02307557b9ae1" , "text" : "this is my second comment!!!" , "dateCreated" : { "$date" : "2013-02-21T18:55:13.832Z"}} , { "className" : "gr.ntua.ivml.awareness.persistent.DigitalStoryComment" , "_id" : { "$oid" : "51266d5d9614e937b69a5d3a"} , "userId" : "50810932bf9594d11fa039bd" , "storyId" : "510ce375e4b02307557b9ae1" , "text" : "this is my very first comment!!!" , "dateCreated" :

{ "$date" : "2013-02-21T18:54:21.347Z"}}]}

*number of objects returned=to-from, totalSize=total comment count for story

http://{hostname}/awareness/digitalStories/comments/save

Description: Saves the comment json object to MongoDB.

Parameters: none, the actual object is part of the body of the HTTP Post request. (must include userid, storyid and text).

Return:

- an http code 200 if the operation was successful with a JSON response containing the comment details:

Example reply:

{"_id" : ObjectId("51266d5d9614e937b69a5d3a"),
"className" : "gr.ntua.ivml.awareness.persistent.DigitalStoryComment",
"userId" : "50810932bf9594d11fa039bd",
"storyId" : "510ce375e4b02307557b9ae1",
"text" : "this is my very first comment!!!",
"dateCreated" : new Date("2/21/2013 20:54:21")
}

- an http code 400 on malformed/missing input parameters

http://{hostname}/awareness/digitalStories/comments/delete

Description: Permanently removes a story comment from a story.

Parameters:

Parameter Name	Mandatory	Description
id	Yes	The id of the user comment to be deleted

Return:

- an http code 200 if the operation was successful.

- an http code 400 if id was not found
- an http 500 code on system error

http://{hostname}/awareness/digitalStoryComment/{id}

Description: Retrieves a specific user comment on its unique ID.

Parameters: none

Return:

- an http code 200 with the JSON object of a user comment.
- an http code 404 (not found) if no user comment with that id was found