

Go-Lab

Global Online Science Labs for Inquiry Learning at School

*Collaborative Project in European Union's Seventh Framework Programme
Grant Agreement no. 317601*



Deliverable D5.2

Specifications of the Go-Lab Portal and App Composer

Editor	Sten Govaerts (EPFL)
Date	30 th October, 2013
Dissemination Level	Public
Status	Final



©2013, Go-Lab consortium

The Go-Lab Consortium

Beneficiary Number	Beneficiary Name	Beneficiary short name	Country
1	University Twente	UT	The Netherlands
2	Ellinogermaniki Agogi Scholi Panagea Savva AE	EA	Greece
3	École Polytechnique Fédérale de Lausanne	EPFL	Switzerland
4	EUN Partnership AISBL	EUN	Belgium
5	IMC AG	IMC	Germany
6	Reseau Menon E.E.I.G.	MENON	Belgium
7	Universidad Nacional de Educación a Distancia	UNED	Spain
8	University of Leicester	ULEIC	United Kingdom
9	University of Cyprus	UCY	Cyprus
10	Universität Duisburg-Essen	UDE	Germany
11	Centre for Research and Technology Hellas	CERTH	Greece
12	Universidad de la Iglesia de Deusto	UDEUSTO	Spain
13	Fachhochschule Kärnten - Gemeinnützige Privatstiftung	CUAS	Austria
14	Tartu Ulikool	UTE	Estonia
15	European Organization for Nuclear Research	CERN	Switzerland
16	European Space Agency	ESA	France
17	University of Glamorgan	UoG	United Kingdom
18	Institute of Accelerating Systems and Applications	IASA	Greece
19	Núcleo Interactivo de Astronomia	NUCLIO	Portugal

Contributors

Name	Institution
Sten Govaerts, Adrian Holzer, Evgeny Bogdanov, Andrii Vozniuk, Na Li, Wissam Halimi, Aubry Cholleton, Denis Gillet	EPFL
Yiwei Cao	IMC
Lars Bollen, Ton de Jong (peer-review)	UT
Pablo Orduña, Luis Rodríguez	UDEUSTO
Antonio Robles Gómez, Miguel Latorre García, Elio San Cristóbal Ruiz	UNED
Panagiotis Zervas, Alexandros Trichos	CERTH
Danilo Garbin Zutin	CUAS
Sven Manske	UDE
Sofoklis Sotiriou (peer-review)	EA

Legal Notices

The information in this document is subject to change without notice. The Members of the Go-Lab Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the Go-Lab Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material. The information and views set out in this deliverable are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

Executive Summary

This deliverable presents the first version of the requirements and their derived specifications for the Go-Lab portal (T5.1) and app composer (T5.3). The final specifications will be presented in D5.6 (M36). The Go-Lab portal enables teachers to search for online labs and create inquiry learning spaces (ILS) to be used in their lessons with students. Such ILS can be shared by the teacher to the community as best practices. The portal also enables search of these shared ILS and relevant applications (apps). Nowadays, these apps are typically created by developers. To foster wide online lab adoption, teachers should be able to aggregate the necessary resources and apps to support their inquiry lessons (or to enrich their lessons) without the need for a software developer. Additionally, we foresee the need of a large number of apps that are domain specific or provide different pedagogical features to support students (for instance differing in age, prior knowledge, followed curriculum and physical abilities). Therefore, Go-Lab wants to enable teachers to create their own apps using the app composer. The app composer is a suite of tools specifically targeted towards teachers to translate, edit and create apps without programming to support their pedagogical scenarios. Additionally, we discuss compatibility of the portal and app composer with mobile devices, which are becoming more popular in the classroom nowadays.

Table of Contents

1	Introduction	8
2	The Go-Lab portal	10
2.1	Introduction & objectives	10
2.2	Terminology	10
2.3	Requirements of the Go-Lab portal	10
2.3.1	User story	10
2.3.2	Functional requirements analysis	11
2.3.3	Non-functional requirements analysis	11
2.4	State of the art	12
2.5	The Go-Lab portal architecture	13
2.5.1	Overall architecture	13
2.5.2	Components and interface specification	14
2.5.3	Portal interoperability	15
2.6	Implementation of the Go-Lab portal	17
2.6.1	The Lab Repository	17
2.6.2	Inquiry learning space platform	19
3	The app composer	22
3.1	Introduction & objectives	22
3.2	Requirements of the App composer	22
3.2.1	User story	22
3.2.2	Functional requirements analysis	23
3.2.3	Non-functional requirements analysis	24
3.3	The App Composer mockups	24
3.3.1	The main window	24
3.3.2	The translator	24
3.3.3	The builder	28
3.4	State of the art	32
3.5	Integration with the Go-Lab portal	33
3.6	Implementation of the App Composer	34
3.6.1	Implementation overview	34
3.6.2	App composer modules	35
4	Mobile devices compatibility	36
5	Conclusion	38
6	Appendix A: Lab metadata	39
6.1	The Metadata survey conducted in WP2	39
6.2	Go-Lab metadata specification	39
6.2.1	Lab metadata	40
6.2.2	App metadata	40
6.2.3	Resource metadata	40
6.2.4	Inquiry learning space template (ILS) metadata	40

6.2.5	Taxonomy	41
7	Appendix B: Go-Lab portal UI functionality	45
7.1	The lab repository	45
7.1.1	The currently implemented features of the lab repository.	45
7.1.2	Future work on the lab repository	45
7.2	The inquiry learning platform (Graasp)	51
7.2.1	The currently implemented features of inquiry learning platform.	51
7.2.2	Future work on the inquiry learning platform	54
7.3	Example inquiry learning spaces	54
8	Appendix C: Graasp development	56
8.1	Performance improvements	56
8.2	Infrastructure improvements	57
8.3	Activity Tracking	59
8.4	OpenSocial 2.5	60
8.5	User Management	62
8.5.1	Login with Facebook & Google+	62
8.5.2	Anonymous login	62
8.6	Other new functionality	63
8.6.1	Google Docs integration	63
8.6.2	Mobile Graasp apps	64
8.6.3	File uploading app	65
8.6.4	Security improvements	66
8.7	Usability improvements	67
8.7.1	Other improvements	67
	References	69

1 Introduction

Although labs for teaching have become available recently, a widely used online lab portal integrated with a ready-to-use learning environment is still missing (see Section 2.4). Usually, individual online labs are operated, maintained and promoted by the lab owners, which causes a high operational cost and limited access. Through the Go-Lab portal, we aim to establish a federation of online labs where lab owners can promote their labs, and teachers can find labs to support their activities and share their resources with others. Online labs consist of remote laboratories, virtual experiments and data sets/analysis tools (as defined in the DoW).

In deliverable D5.1 we have already discussed personalisation techniques that apply to the portal (e.g., recommendation strategies and internationalisation). In this deliverable we present the first version of the specification of the Go-Lab portal (Task 5.1) and the app composer (Task 5.3). The portal (Task 5.1) contains both: (a) a repository of online labs, inquiry learning spaces (ILS), resources and apps, and (b) a platform for inquiry learning. Teachers can use the portal to search for labs, apps and ILS to use in their course and can create an ILS to be used in their lessons. Such ILS can then be used by students and teachers in a STEM course to teach scientific topics following inquiry-based learning methodologies. Inquiry learning typically leads students through various phases, e.g., orientation, conceptualization, investigation, conclusion and discussion, where students create hypotheses, evaluate them through experiments and then reflect on them, possibly repeating the cycle (see D1.1).

To sustain the adoption of online labs and the Go-Lab portal itself, it is important to support teachers with the aggregation of the resources and apps needed to implement their lesson plan. The app composer (Task 5.3) enables teachers looking for missing apps, to adapt existing and create new apps without the need for a software developer. Through a graphical user interface teachers will be able to adapt existing apps to their needs or design new apps that fulfill their requirements. Furthermore, through the app composer teachers can translate the user interface (UI) of existing apps to the mother tongue and language level of their students.

With the recent high adoption of mobile devices in the classroom, it is important for the Go-Lab portal to be compatible with such devices. Students should be able to do investigations in online labs on tablets. To achieve this the app composer should output apps that are compatible with mobile devices.

This deliverable consists of three major sections: one presenting the portal specifications, one the app composer and the last one mobile compatibility issues. The first two sections will be introduced with the objectives of the software, specific terminology and the requirements. Subsequently, related research and platforms will be discussed and contrasted with the Go-Lab approach. Following this, the architecture will be presented and the current status of the implementation. For the app composer, we will discuss the current version of the user interface mockups to elicit a better understanding of its scope. For the mobile

compatibility, we will discuss the implications and the chosen technical solutions. Finally, a conclusion and an outlook on specifications and implementation is presented.

2 The Go-Lab portal

2.1 Introduction & objectives

This section describes the requirements and specification of the Go-Lab portal. Furthermore, since the portal is the central component of the Go-Lab environment and achieving a federation of labs is an important goal, the interoperability aspects of the specification are discussed in more detail. The main part of this section was published in (Govaerts et al., 2013). Finally, the current status of the implementation will be discussed. But first, we will establish some vocabulary.

2.2 Terminology

Online labs are remote laboratories, virtual experiments or data sets accessible from the browser through apps. *Apps* are Web applications (e.g., OpenSocial gadgets), for example to operate a lab or support learning (e.g., via scaffolding or other forms of guidance, see D1.1).

Inquiry learning spaces (ILS) are interactive web pages that can contain labs, resources and apps to enable inquiry learning. Resources are typically texts, videos and other materials to assist students. Teachers usually set up an ILS for their students. An ILS can be shared with other teachers who can repurpose and adapt it to fit their needs.

2.3 Requirements of the Go-Lab portal

We identify three types of portal users: *lab owners*, *teachers* and *students*. A lab owner is a user who operates and publishes a lab and potentially prepares an ILS to support the use of her lab. A teacher is a user who uses, modifies, or creates an ILS and teaches with it. A student is a user who carries out inquiry learning activities in an ILS. The following user story provides more context on the portal use.

2.3.1 User story

On the Go-Lab portal John searches for interesting educational activities including online labs for his physics lesson. He enters 'particle collisions' in the search mechanism of the portal, after which he selects the *Conservation of Momentum* ILS from the search results. The system displays an overview of an ILS using the *HYPATIA* lab that is used to analyse collisions of elementary particles at CERN. He opens the ILS in the Go-Lab environment to try it out. The orientation phase and its resources are shown and he can edit the resources and apps. There are numerous resources along with exact guidelines on how to use this ILS in class. These guidelines were created by the author of the ILS, Denis, a teacher who is working at a school located in Geneva. John finds Denis' approach quite complex and he decides to add a recommended scaffolding app. Additionally, he rewrites some of the prompts and heuristics to better fit the knowledge of his students. He is also using the Go-Lab search facility for online labs and he gets back another CERN resource that is called LHC Game. John integrates this lab to the orientation phase of the inquiry cycle as he believes that it would be an ideal way to introduce his students to the CERN

experiments. After tweaking each inquiry learning phase, he teaches with the resulting ILS in class. Afterwards, he decides to publish his ILS back on the Go-Lab portal and share it on Facebook with his colleagues. He is also sending a message to Denis to initiate a discussion on the use of HYPATIA and the very interesting *plug and play* authoring process in which he was involved.

2.3.2 Functional requirements analysis

In this section we focus on the main functions of the portal, required by the portal users to fulfill their needs.

- *Publishing labs.* Lab owners publish a lab and describe it with metadata.
- *Creating ILS.* Lab owners create ILS to demonstrate a lab and teachers create ILS for students.
- *Modifying ILS.* Teachers adapt existing ILS, e.g., localize the content to the mother tongue of their students or adapt it for a different age group.
- *Publishing ILS.* Teachers and lab owners publish their ILS to enable reuse.
- *Using ILS.* Teachers run activities using ILS. Students use ILS provided by teachers to conduct experiments.
- *Supporting Apps.* Students are supported in their inquiry learning through apps (e.g., a hypothesis app or online lab interfacing apps). Teachers monitor student progress through learning analytics apps.
- *Supporting lesson plans.* Teachers create lesson plans for ILS. Students use lesson plans provided by teacher when studying in ILS.¹
- *Searching Labs & ILS.* Teachers search for labs and ILS using various search filters, e.g., age and scientific domain.
- *User management.* To access all of the portal's functionality, users log in only once. Several login options are provided depending on the acceptable privacy level.
- *Social features.* Teachers and lab owners tag, comment and rate labs and ILS, and share them on social networks.
- *Tracking user activities.* The activities of portal users are tracked and used for learning analytics, recommendation and scaffolding apps.
- *Recommendation.* Recommendation of labs, ILS and apps are provided when searching, creating and editing ILS and labs.
- *Scaffolding.* Students receive assistance from scaffolding apps (e.g., prompts and feedback) based on learning analytics and teacher configurations.

2.3.3 Non-functional requirements analysis

Apart from the previous requirements, there are also non-functional requirements that impact the design.

¹Note that a lesson plan can also include offline activities.

Go-Lab needs to provide a common, ubiquitously accessible ILS platform, so schools do not need to spend resources on installing and administering software. To provide lab federation and to support a common ILS platform, interoperability of the labs is essential. When targeting school students, special attention to usability and data privacy (e.g., anonymizing the tracked user activities) is needed. Additionally, usability is also top priority for school teachers due to varying computer skill levels. The Go-Lab project aims to support 1000 schools in 15 countries, which requires a high scalability and availability of the portal, as well as internationalisation support.

2.4 State of the art

Existing portal solutions for online labs are reviewed and summarized in Table 1 and we evaluate their fit with the requirements presented above. We identified five main solutions in the research literature, namely the GOLC consortium's Lab2Go² portal (Christian Maier, 2010), the Library of Labs (LiLa)³ (Richter, Boehringer, & Jeschke, 2011) portal created by a European consortium using MIT's iLab Shared Architecture (ISA)⁴ (Harward et al., 2008), the LabShare⁵ (David Lowe, 2009) portal initiated by an Australian consortium, the University of Deusto's WebLab-Deusto⁶ (Garcia-Zubia, Ipina, Orduna, & Hernandez-Jayo, 2006), and the University of Colorado's PhET interactive simulations repository⁷.

Table 1 shows that the main requirements which are met by the existing portals are *publishing labs*, *searching for labs & ILS*, *using ILS* and possibly *tracking user activities*. Note that none of the other labs provide explicit ILS with inquiry phases and guidance apps. However, A few labs provide some sort of structured activity that can be created, used, modified or published. As they do not provide complete ILS with inquiry phases and guidance apps we consider that they only partially support these requirements.

Furthermore, several of these portals offer some kind of user management system and provide some social features. Unfortunately, several requirements are not properly supported by any of the portals, such as facilities to create, modify and publish ILS. Furthermore, recommendation and scaffolding are both not supported by any portal. In a nutshell, existing portals mainly work as repositories of labs and provide only support to lab owners to publish labs and to teachers to find and use labs. As they do not provide their own learning spaces, it is difficult for such portals to empower teachers by not supplying them with adequate support for modifying, reusing and publishing ILS. Go-Lab precisely aims to fill this gap by providing support for both ILS usage and ILS creation.

²<http://www.lab2go.net>

³<https://www.library-of-labs.org>

⁴<http://ilabcentral.org>

⁵<http://www.labshare.edu.au>

⁶<https://www.weblab.deusto.es>

⁷<http://phet.colorado.edu>

Functional requirements	Lab2Go	LiLa	ISA	LabShare	WebLab	PhET
Publishing labs	+	+	+	+	+	+
Creating ILS	-	~	-	-	-	~
Modifying ILS	-	~	-	-	-	~
Publishing ILS	-	~	-	-	-	~
Using ILS	-	~	+	+	+	~
Supporting Apps	-	-	-	-	-	-
Supporting lesson plans	-	+	-	?	~	+
Searching Labs & activities	+	+	-	-	-	+
User management	~	~	+	+	+	+
Social features	+	+	-	~	-	+
Tracking user activities	-	~	~	+	+	-
Recommendations	-	-	-	-	-	-
Scaffolding	-	-	-	-	-	-

Table 1: Fit between existing portals and Go-Lab requirements. Legend: requirement supported (+), partially supported (~), not supported (-), unknown (?).

2.5 The Go-Lab portal architecture

Based on the requirements, this section describes the Go-Lab architecture. For its design, we have applied several software design principles (Vogel, Arnold, Chughtai, & Kehrer, 2011). Foremost, the design should be *loosely coupled*, meaning that each component requires little knowledge of the definitions of the other components (Vogel et al., 2011). This principle enables abstraction of each component, which allows a design that can change over time. Additionally, each component should have *high cohesion*, which refers to the degree to which a component is semantically self-contained (Vogel et al., 2011). Applying *'separation of concerns'* enables modularity, as each component takes care of a separate task (Vogel et al., 2011). Finally, the design should support *subsetability*, which is the ability to produce subsets of the system. This allows us to follow an iterative and incremental development strategy and have a basic portal available quickly.

2.5.1 Overall architecture

The high-level Go-Lab architecture, illustrated in Fig. 1, consists of two main components with a graphical user interface (GUI), namely the *lab repository* and the *ILS platform*. Both are supported by components for *user management* and tracking user activities for *learning analytics and recommendation*. By splitting up the portal functionality in this way, each component serves a very different purpose and we aim to satisfy the requirements and design principles stated above. The components have well-specified interfaces and protocols, which allow interchangeability (e.g., the ILS platform could use another repository that implements the same specification of the Publisher & Instantiator interface) and other third-party platforms can make use of each component separately

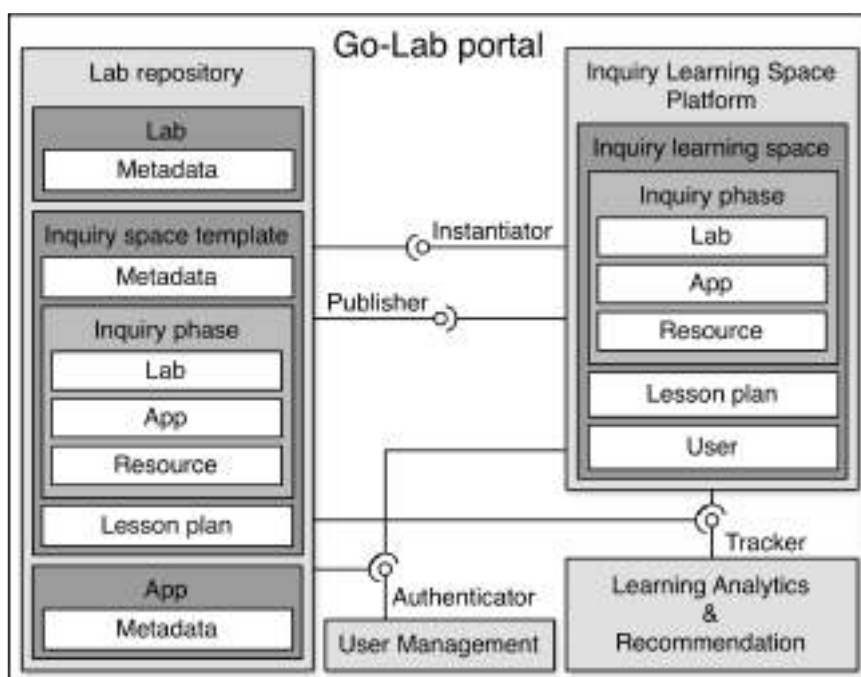


Figure 1: The architecture of the Go-Lab Portal.

enabling wider adoption of Go-Lab technology. The next section elaborates on the components of the architecture.

2.5.2 Components and interface specification

The Lab Repository stores labs, apps and inquiry space templates (or ILS templates), together with their metadata. An ILS template describes the structure and content (i.e., the labs, apps & resources) of an ILS. An ILS template basically functions as a blueprint with all the information to create a new ILS that can be edited by teachers. Additionally, an ILS template can also contain a learning scenario provided by teachers that describes how to use the ILS in a pedagogical context.

The Inquiry Learning Space Platform (ILS platform) allows teachers and students to use labs and apps in an ILS for inquiry learning. Teachers can create an ILS consisting of labs, resources, and apps available from the lab repository through the *Instantiator* interface and enrich the ILS with uploaded or online resources. Afterwards, teachers can provide students with access to the ILS, where students can conduct experiments.

Such an ILS can also be published on the lab repository via the *Publisher* interface. While publishing an ILS, teachers provide metadata (see Appendix A) that describes the ILS together with pedagogical information and possibly a learning scenario. Such published ILS templates should not have to pass a review cycle (which is for example the case for iOS apps). We will rely on peer ratings and reviews to control the quality and sorting. By publishing an ILS template to the lab repository, other teachers can find it there, reuse it in the

ILS platform using the `Instantiator` interface and adapt it to the needs of their students.

The *Learning Analytics and Recommendation* component collects user activities through the `Tracker` interface from the lab repository and the ILS platform that can anonymize the data for privacy reasons. The collected data is used to provide teachers with learning analytics apps to monitor student progress; lab owners can monitor the use of their labs, while students benefit through scaffolding apps. The tracked user activities are also employed for personalization of the portal, e.g., through recommendation of apps, labs and resources. For more information, we reference to deliverable D4.2, which is partially dedicated to a full specification of this component.

The *User Management* component is in charge of user authentication and user profile management through the `Authenticator` interface to the lab repository and ILS platform enabling single sign-on for the portal. Furthermore, the user management component will also be used by other services, such as the app composer and the add-on services (see D4.2).

2.5.3 Portal interoperability

To achieve a federation of labs and to increase the potential uptake of the Go-Lab software, interchangeability of the portal components and interoperability using open specifications is important. The Go-Lab architecture achieves the technical, syntactic, semantic and pragmatic interoperability levels of the Conceptual Interoperability Model (Turnitsa, 2005). This section elaborates on interoperability enabled through the labs, metadata and the interfaces between components.

Lab interoperability enables the integration of labs with an ILS platform, which is often difficult due to the wide variety of labs and their technical differences (e.g., implemented as a Java Applet or a Flash application). To make labs interoperable with learning environments different approaches are possible. For instance, the LiLa project (Richter et al., 2011) bundles labs in SCORM (Bohl, Scheuhase, Sengler, & Winand, 2002) packages, but this does not always enable proper interoperability since SCORM is not designed for interactive labs and the support of the latest versions of SCORM by learning environments is low. In Go-Lab, online labs will be provided as *smart devices* (Gillet, Jong, Sotirou, & Salzmann, 2013) (see D4.1) that make labs more ubiquitous, autonomous and self-aware. The smart device paradigm abstracts the details of each lab on the server-side by providing a specified set of web services (Gillet et al., 2013). This interoperability layer allows the ILS platform to run any lab supporting the smart device paradigm and smart device compatible apps can be reused to operate numerous labs.

Making existing online labs smart device compliant might require the implementation of the specified web services. In some cases it will be impossible to change the lab implementation. To enable interoperability with such labs, we

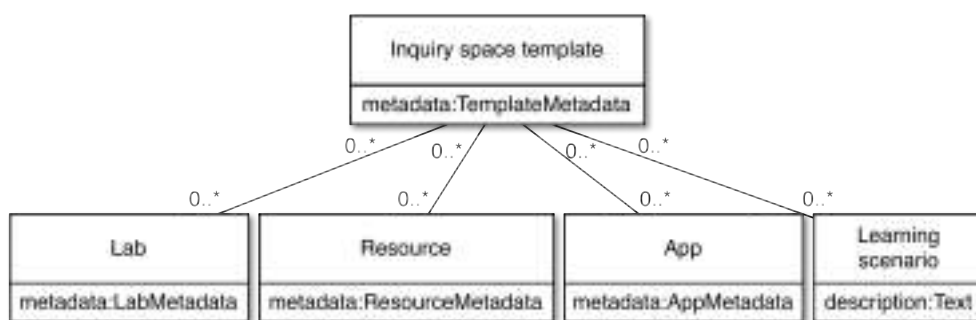


Figure 2: Go-Lab metadata overview.

will provide a *smart gateway* (see D4.1) that transforms existing labs offered by third parties to be conform to the smart device specification (Gillet et al., 2013). Through both the smart device and smart gateway, interoperability between any online lab and the ILS platform is enabled.

Metadata interoperability is the ability to exchange metadata with minimal loss of content and functionality between different systems (NISO Press, 2004). Several initiatives (Christian Maier, 2010; T. Richter & Zutin, 2012) are currently working on metadata specifications to describe online labs and related resources. Their main reason is to allow wider discovery of online labs, but metadata can provide more benefits. Apart from search and discovery of labs and apps in the lab repository, metadata is also used to exchange data between the portal components to enable interoperability and exchangeability. For instance, this interoperability and exchangeability allows exchanging the lab repository with a third-party repository that applies the same interface and metadata specification; similarly the ILS platform could be switched. In the Go-Lab project, metadata is used to describe labs, apps, resources and ILS templates (see Fig. 2) in a linked data approach. Appendix A describes the efforts of the cross-workpackage metadata discussions and the current version of the Go-Lab metadata specification, based on a combination and extension (fitting the Go-Lab requirements) of the ROLE Ontology and the GOLC specification (T. Richter & Zutin, 2012). The reuse of existing open specifications will provide access to existing labs and resources, as well as services.

Interface interoperability allows different implementations of components of the Go-Lab portal to be interchanged. This can be achieved by specifying the component interfaces and the data that is exchanged (metadata interoperability). For instance, the lab repository could use another learning environment that specifies the `Instantiator` and `Publisher` interface. Additionally, the `Authenticator` interface enables the interchangeability of, for instance, the default user management with an LDAP implementation.

2.6 Implementation of the Go-Lab portal

Based on the design of the Go-Lab portal architecture, we implement the portal using an iterative and incremental approach. The portal user interface (UI) has been designed using participatory design among the Go-Lab partners together with teachers using UI mockups⁸ (documented in D1.1). Those mockups have been evaluated in multiple participatory design workshops and the results are documented in deliverable D3.1.

The remainder of this section describes the past and future implementation details per main portal component.

2.6.1 The Lab Repository

The lab repository is implemented on top of Drupal 7⁹. Drupal is a widely-used open-source content management system that allows high scalability. We have applied the experiences learned from the existing ROLE Widget Store (Dahrendorf, Dikke, & Faltin, 2012) of the previous EU FP 7 IP project ROLE. Labs, apps, resources and ILS templates are described with metadata (e.g., functionalities & 'Big Ideas in Science' categories). This metadata is used to describe, organize, manage, and search all types of learning content in the repository. Currently, we have achieved the first implementation phase, where the creation of labs, resources, apps, and ILS templates, and the integration with the ILS platform through the *Instantiator* interface has been implemented. Thus, now the stored ILS templates can be instantiated within the ILS platform with one mouse click. More details on how this process works in the UI of the portal is provided in Appendix B (see Section 7.1). The latest version of the lab repository can be found at <http://go-labz.eu/>.

Types of labs, apps, resources & ILS templates with metadata

In the lab repository, labs, apps, resources, and ILS templates are specified as different content types in Drupal with each a specific set of metadata fields. Some metadata fields are shared among content types, e.g., the ILS template and lab type share the metadata grade level and all four types share the language metadata field. The metadata specification partially reuses existing metadata specifications, such as the metadata schema of the ROLE Widget Store¹⁰. The metadata schema is further detailed in Appendix A.

Inquiry learning spaces with customized inquiry learning phases

An ILS makes use of the inquiry learning cycle, which consists of an ordered set of inquiry learning phases (see deliverable D1.1). As depicted in Figure 3 an inquiry learning space consists of inquiry learning phases, which consist of apps, labs, resources and their metadata.

In the Go-Lab portal, teachers are well supported to personalise an ILS for their course needs (for more personalisation details, we refer to D5.1). The structure

⁸The mockup is available at <http://www.go-lab-project.eu/content/prototypes>

⁹Drupal, <http://www.drupal.org>

¹⁰The ROLE Widget Store, <http://www.role-widgetstore.eu/>

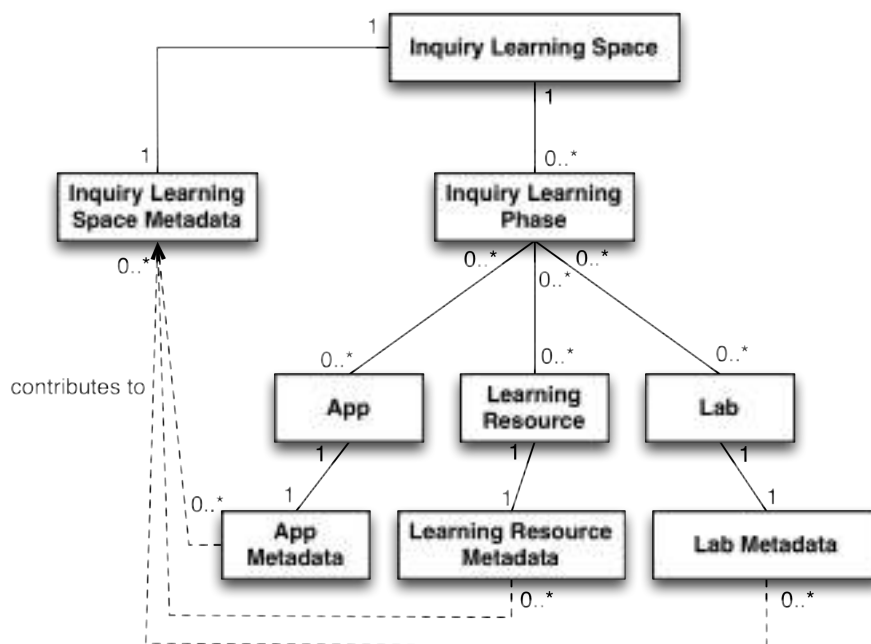


Figure 3: The structure of the inquiry learning space in the lab repository.

of an ILS can be personalised as follows:

- *Inquiry learning phases*: Teachers can organise the inquiry cycle the way they want. Go-Lab provides an inquiry learning cycle as a starting point. The teachers can add or remove inquiry phases to fit their needs.
- *Renaming inquiry learning phases*: The portal will allow teachers to rename inquiry learning phases to provide a term in a language comprehensive to the students. The portal keeps track of the original ‘Go-Lab’ name of the space to provide this information to some apps who can act on knowing in which context they are being used, e.g., a hypothesis tool could allow students to edit the hypothesis in the *conceptualisation* phase but only view the set hypotheses in the *conclusion* phase.
- *Collecting labs, apps, and resources* in each phase allows teachers to create a unique ILS for each course.
- *Translation of an ILS*: Teachers can select the language of the UI of the ILS platform for their students and can provide the content itself in the appropriate language.

Search learning content in the lab repository

The search functionality will be implemented in two steps: metadata-based search and advanced search.

Metadata-based search in the lab repository supports learning content search by single metadata fields. For example, teachers may search all labs related to buoyancy, or in Greek. Since some content types share certain metadata fields, it is possible to search across content types. Currently the metadata-based

search is implemented as faceted search. In the future, we will also provide keyword search.

Advanced search will be implemented in the lab repository, such as search labs by *Big Idea*. The required fields for the advanced search will first be surveyed in participatory design usability workshops. In addition, since the Go-Lab Portal supports internationalisation, teachers should be able to search for learning content in their own language as well as in other languages. For example, a Spanish teacher searches for an ILS in Spanish. However, if there is no suitable ILS, she should be able to search for English content.

2.6.2 Inquiry learning space platform

The ILS platform is implemented on top of the Graasp platform (Bogdanov et al., 2012) (as described in Task 5.1), which is a social media platform that supports personal and collaborative activities using resources and OpenSocial apps. Since the start of the project, Graasp has been under continuous development to mature the platform and integrate Go-Lab specific requirements. Currently, Graasp has already been extended to create a simple ILS and a first version of a dedicated student view of the ILS has also been implemented.

This section will first describe how ILS are created in Graasp. Afterwards, we will discuss the development status and plans of the different components and interfaces of the portal architecture.

Creating an ILS in Graasp

In the current version of Graasp teachers can already create ILS. The implementation is currently a prototype, hence the mechanism to create an ILS is still under discussion and due to change in the future. However, in this section we will discuss the current idea behind creating an ILS in Graasp.

A special type of space for an ILS has been created in Graasp. When a space of the ILS type is created it automatically contains five subspaces, each subspace is one inquiry learning phase of the Go-Lab inquiry cycle. Figure 4 illustrates the ILS in blue and the subspaces in yellow. Teachers can then add apps, labs and resources to each phase subspace. These apps, labs and resources will be shown only in that phase in the student view of the ILS. Teachers can add a description to a phase subspace that provides an explanation and context to students on how to operate the apps, labs and resources in that phase. Appendix B explains how to accomplish this in the Graasp GUI and illustrates the student view of the ILS.

In the portal mockups described in deliverable D1.1, the toolbar concept was introduced as a place to provide apps that should be available in all the inquiry phases. Such toolbar apps can be added to the inquiry learning space itself (the blue space in Figure 4). As briefly mentioned previously, the apps are only available in one specific phase space. However in certain situations such apps want to communicate data. For instance, a hypothesis creation app in the 'Discussion' phase space would like to access the hypotheses created by the same app in the 'Conceptualisation' phase space. To achieve this, we have proposed

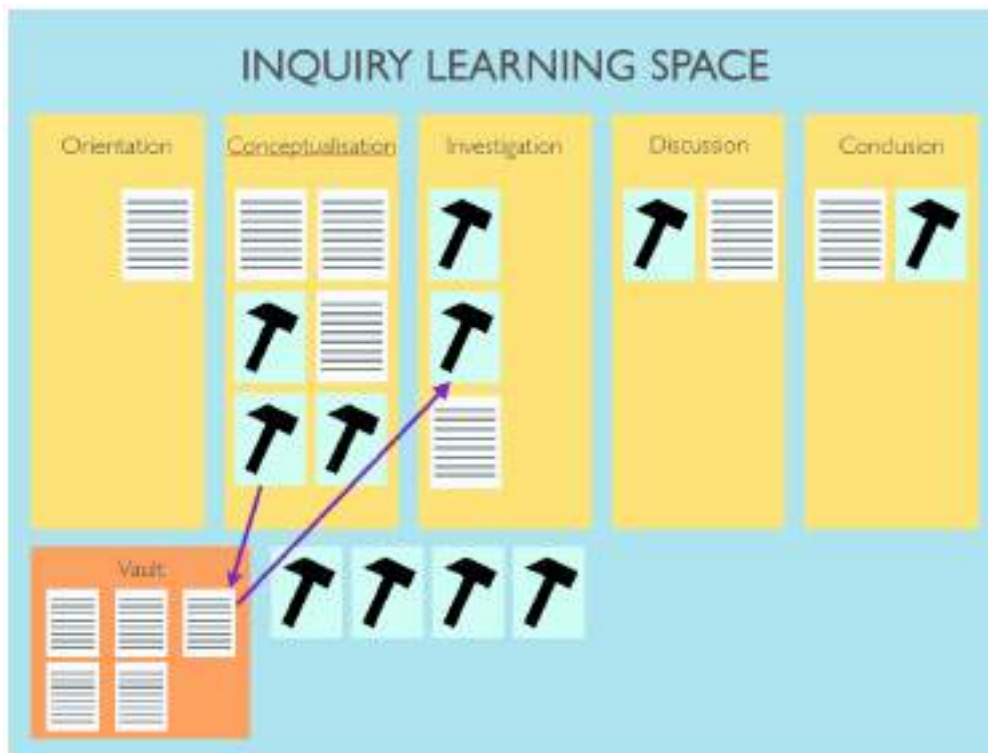


Figure 4: The structure of an ILS in Graasp.

the concept of a data communication space, called 'Vault' in Figure 4. Apps located in one phase space can write data files in this space and other apps from other phase spaces. This way we can provide flexible data communication between phases, while keeping the privacy concepts of spaces. This 'Vault' idea is currently a proposal and in the future we will further investigate this issue before implementing it.

The interfaces between the lab repository and the ILS platform

In order to support the portal interoperability, the portal architecture defines several interfaces between the lab repository and the ILS platform. We summarise the current and future development for an overview below.

A first version of the `Instantiator` interface is implemented in the lab repository and Graasp using JSON. When teachers find a suitable ILS template on the lab repository, they can export the template into Graasp for direct use or further editing.

As the next step, an RDF interface will be created to exchange ILS templates together with their metadata between the lab repository and the ILS platform through the `Instantiator` and `Publisher` interface. In the lab repository, mappings between the metadata of the content types and RDF need to be created. Several RDF namespaces may be used to describe labs, apps, resources, and

ILS templates, e.g., based on FOAF¹¹, the ROLE ontology or Dublin Core¹².

Part of the communication with the learning analytics and recommendation component's *Tracker* interface has also been integrated in Graasp using the OpenSocial specification together with the ActivityStreams specification to represent the user activity data (Vozniuk, Govaerts, & Gillet, 2013). For further details about the implementation status, we refer to Appendix C. The details on the architecture of the Learning Analytics service will be described in deliverable D4.2 and the recommendation service is detailed in D5.1.

In a later phase, the user management service and other more advanced functionality will be implemented. The metadata schema will be gradually implemented and extended if new requirements appear.

¹¹FOAF, <http://xmlns.com/foaf/spec/>

¹²Dublin Core, <http://dublincore.org/>

3 The app composer

3.1 Introduction & objectives

The app composer is a service that enables teachers to develop web applications. In the context of Inquiry Learning Spaces (ILS), teachers typically compose such spaces with existing apps available in the lab repository, but for some specific learning scenarios they can require missing functionality. The app composer is a web tool that allows teachers to create or adapt apps to provide such missing functionality. The adaptation of these apps will include translation of existing apps, as well as the adaptation of existing apps and the creation of new apps.

This section elaborates first on the requirements of the app composer. Afterwards several mockups of the app composer's preliminary UI design are described. After that, the current state of the art related to the app composer is presented. Finally, a section is dedicated to the future implementation options.

3.2 Requirements of the App composer

The app composer will be used mainly by teachers. Students will use the resulting apps and lab owners are typically experts who will create their own apps without the assistance provided by the app composer, except if they are willing to exploit some specific services such as learning analytics.

The app composer assists teachers by making it easy to develop or customize apps. Three types of teachers are identified:

- Teachers who will not need additional apps and who therefore will not use the app composer.
- Teachers who will only need to customize existing apps.
- Advanced teachers who will build their own apps or even simulations.

The following sections first introduce a user story, in order to provide a better understanding of the application composing process. Afterwards, the functional requirements which can be extracted from the story are detailed. Finally, the non-functional requirements are specified as well.

3.2.1 User story

Johanna is a high school teacher from Stockholm. She wants to create an ILS for her physics class. Searching the portal, she finds a pendulum laboratory, which is only available in English, but she doesn't find any proper formula editor that will assist students while using the pendulum. She first goes to the app composer, translates the pendulum laboratory to Swedish for her students and shares her translation with other Swedish teachers on the portal. Then, in the app composer, she builds a new app with which she defines formulas and publishes it in the portal. As a result, she can now build an ILS using a pendulum laboratory in Swedish and a formula system in which students can make calculations easily. She also wants to use an experiment design app adapted for the context of the pendulum. She goes back to the app composer, and se-

lects the “experiment design app” template, she adapts the required fields for the pendulum case, so the students can provide different variables related to the pendulum and check what she expects to happen. The teacher does not need to know how to program to use the app composer and adapt the ILS to the requirements of her class.

3.2.2 Functional requirements analysis

The app composer has certain functional requirements, most of which are derived from the previous user story. These requirements are as follows:

- *Adapting existing apps.*
 - *Listing existing modifiable apps developed by Go-Lab:* A selection of apps will be available for teachers as a basis for adaptation.
 - *Customizing the existing apps to add different models:* Users can configure a selection of existing apps with data or models specific to their learning scenarios.
- *Building new apps.*
 - *Creating completely new apps:* Users can create apps from scratch using a visual editor to draw the UI and interactive tools for developing the logic without needing an external programmer.
 - *Developing graphical simulations:* Providing the users with the capability of defining graphical simulations with relative ease, without requiring the expertise which would be required by programming them from scratch.
- *Publishing modified or new apps.*
 - *Publishing apps on the lab repository:* Teachers can share the apps they have created or translated with the Go-Lab community by publishing them on the lab repository.
 - *Integrating apps with the portal:* The app composer communicates with the portal to allow teachers to search for existing apps to translate and save apps to their ILS or lab repository.
- *Translating apps.*
 - *Different languages:* To support Go-Lab in European schools, the internationalisation of apps is required (see deliverable D5.1).
 - *Different target groups:* Students targeted by Go-Lab range from 10 to 18 years, which means that we have to deal with different language proficiency levels.
- *General management.*
 - *Saving draft versions:* Often users will require several sessions working with previously saved versions before finishing their applications.
 - *Sharing versions with other teachers:* The apps can be published on

the portal and repurposed by other teachers.

3.2.3 Non-functional requirements analysis

Apart from the previous requirements, there are also several non-functional requirements that impact the design.

Although standard internationalisation based on language or region is common practice (as was discussed in Deliverable 5.1), this might not be sufficient for Go-Lab. Go-Lab laboratories and ILS target users with a wide range of ages and varying language proficiencies. In D5.1, we solved this problem by allowing internationalisation for each language and region, but also for each age group.

A single instance of the app composer will be integrated with the portal. Third parties might deploy adapted versions of the app composer. The app composer should be able to serve thousands of teachers. It will thus require high scalability, reliability, and availability. The users of the app composer will come from all countries. Therefore, internationalisation support for the app composer UI will also be required.

The teachers who will use the app composer will have a varying degree of computer expertise. They will not always have the time or willingness to dedicate much resources to learning new applications. Therefore, it is of utmost importance that the app composer is intuitive, user-friendly and easy to use. This is particularly true for the translation and adaptation features of the app composer, which are especially aimed for non-technical teachers.

3.3 The App Composer mockups

To explore the scope and possibilities of the app composer, we have started with creating paper mockups to explore different concepts and designs. The benefits of such mockups are twofold: (i) they allow to discuss something more tangible without investing resources in development and (ii) they allow to have very early prototypes with limited interactivity that can be used in usability evaluations. Such evaluation sessions will be conducted in the near future to illicit further requirements and evaluate the usability of our current design. The mockups basically consist of three modules: one allows users to translate an app, one allows to adapt an existing app, and the third one allows users to build a new app. In the next sections, we will present the current versions of the mockups in a separate section for each module.

3.3.1 The main window

The main window (see Figure 5) provides access to the three different modules. Each button allows the user to navigate to the separate modules. The remainder of this section will describe each module in more detail.

3.3.2 The translator

One of the requirements of the app composer (see Section 3.2) is to enable teachers to translate the UI of an app easily into the language required by the student. This is important because we target schools in many European coun-

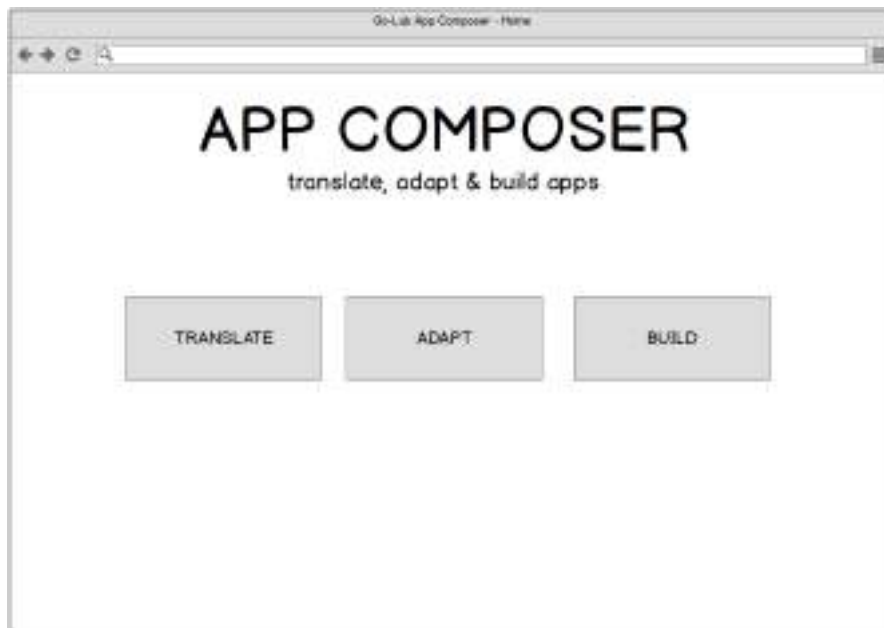


Figure 5: The main starting screen of the app composer.

tries and Go-Lab itself does not have the resources and knowledge to provide appropriate translations. Furthermore, Deliverable 5.1 has described the personalisation techniques related to languages. In this deliverable we discussed the issues of different language proficiency in the age ranges of the students we target. We aim to solve this problem by allowing the translator to enable translations for a specific language and an age group. The following mockups have been designed to solve this task.



Figure 6: The first step to translate an app – select the app.

Once the user selects the 'Translate' option in Figure 5, she is presented with the screen in Figure 6. In this first step, she can provide the URL of an app she

wants to translate or select an app from the lab repository. Once an app has been selected, the user can select the language in Figure 7.



The screenshot shows a web browser window titled "Go-Lab App Composer - k8PFT". The main heading is "TRANSLATE" with the subtitle "the Function Plotter app". Below this, it says "STEP 2: Select the source and target language:". The interface is divided into two sections: "FROM" and "INTO".

FROM

- select language:** A dropdown menu with options: English, French, German.
- select target group:** A dropdown menu with options: General, Preadolescence (age 10-13), Adolescence (age 14-18).

INTO

- select language:** A dropdown menu with options: English, French, German.
- select target group:** A dropdown menu with options: General, Preadolescence (age 10-13), Adolescence (age 14-18).

Below the "INTO" section, a summary line reads: "You want to translate the app from General English into Elementary school German". At the bottom, there is a button labeled "STEP 3 - TRANSLATE =>".

Figure 7: The second step to translate an app – select the languages.

The next step (shown in Figure 7) allows the user to select the target and source language. Since the user might not be comfortable with English as the source language or the app might not provide an English translation yet, we leave the choice up to the user from which language to originate the translation. To accommodate the requirements described in the deliverable D5.1 to support language personalisation for different age groups due to varying language proficiency, we allow next to the language also to specify the target group.

In the final step of the translator (see Figure 8), all the textual UI labels are presented to the user and an automatically generated translation (e.g., using Google Translate¹) into the target language is given. The user can accept the translation by clicking on the 'V'-button next to the label or edit the translation of course. By navigating to the 'Preview' tab, users can preview their translation to validate its correctness in the context of the app, see Figure 9. The translation can then finally be saved and will be available for all users. The translator will also allow users to select languages and target groups that have already been translated to correct existing translations.

¹Google Translate, <http://translate.google.com>



Figure 8: The third step to translate an app – edit the translations.

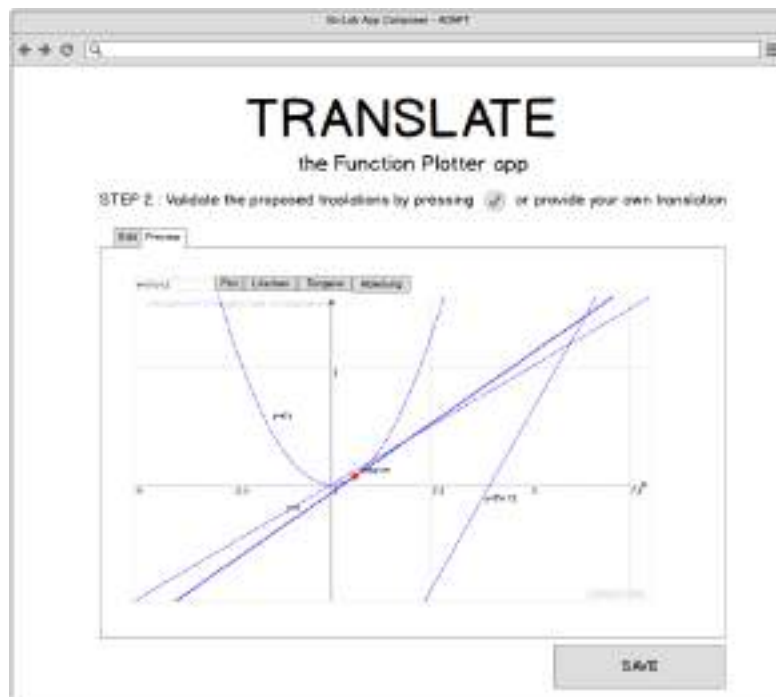


Figure 9: Another view of the second step to translate an app – preview the translated app.

The adaptor.

Inquiry learning apps often require some small adaptations based on the specific ILS and course they are used in. Examples could be teachers who want

to configure a context map with specific concepts and relationships to support the students in creating a specific concept map for the problem at hand. Since quite a few of such inquiry learning apps would benefit from some adaptation to the specific context of the ILS or students, we have decided to support this in the app composer through the adaptor module.

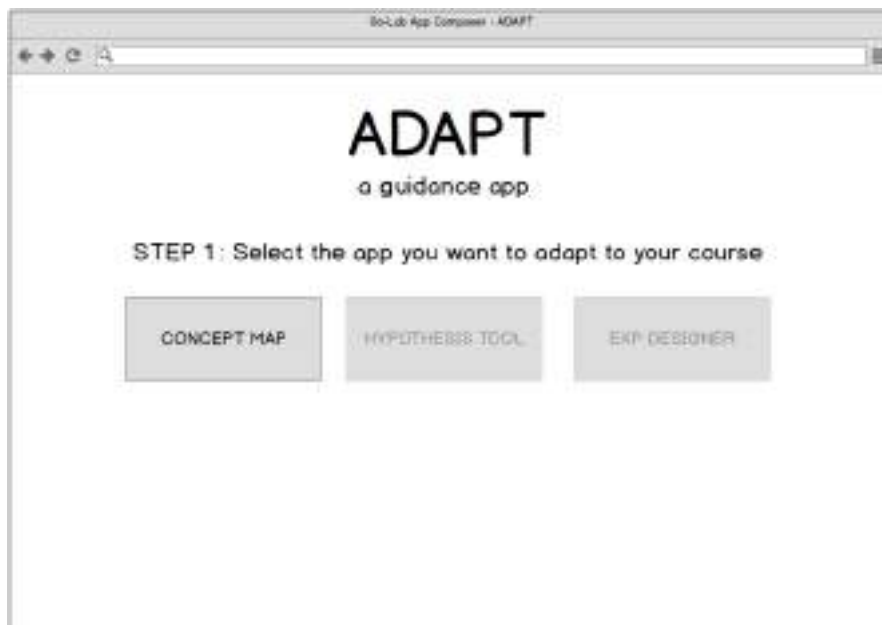


Figure 10: The first step to adapt an app – select the app to adapt.

After selecting the 'Adapt' module in Figure 5, users are asked to select the inquiry learning app that they want to adapt (or configure), see Figure 10. Depending on which app they want to adapt, a different UI will be provided. In our mockups we took the example of adapting the Concept Mapper app that allows students to create concept maps. The Concept Mapper app can be adapted by editing the default concepts and relationships. Figure 11 demonstrates the UI to adapt (remove and add) the concepts and relationships of the Concept Mapper app. Similar to the translator, users can verify if the created concepts and relationships actually allow to build an appropriate concept map in the 'Preview' tab, see Figure 12.

Once the Concept Mapper app has been adjusted, the adapted app can be saved in an ILS created by the user in ILS platform. Figure 13 allows the user to select one of her ILS' and save the adapted Concept Mapper app.

As mentioned the UI to adapt other inquiry learning apps will likely look different, but these mockups illustrate the concept itself.

3.3.3 The builder

The third option of the main screen (see Figure 5) provides the user with the possibility to build her own app from scratch or using basic templates. The app builder allows teachers to create new apps to fit their needs without advanced programming or the help of an external programmer. The builder will enable

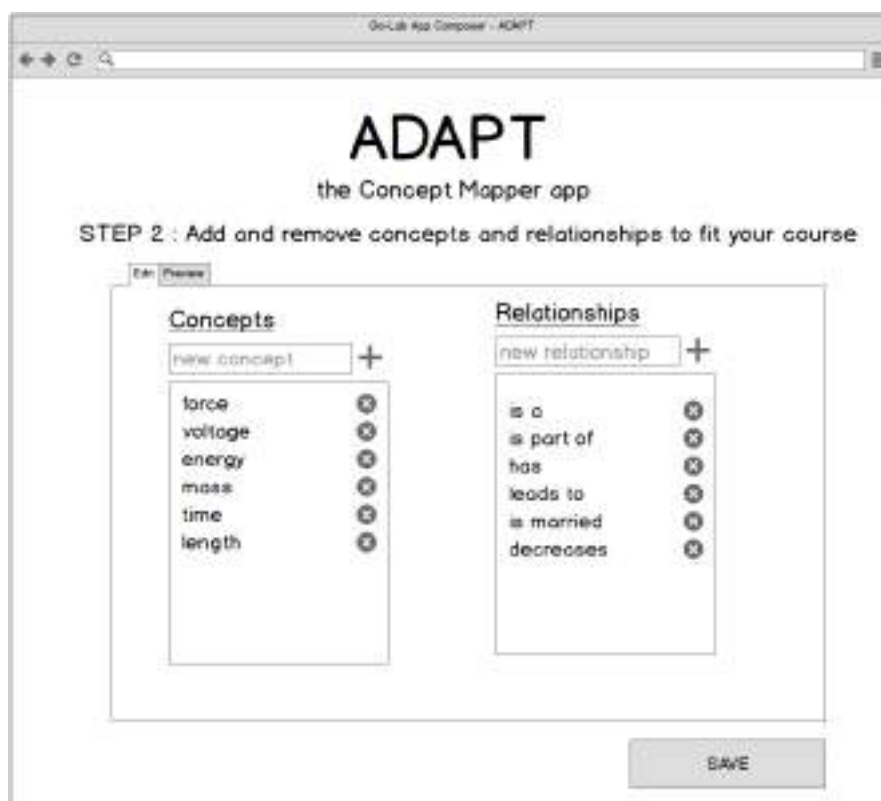


Figure 11: The second step to adapt an app – create the configuration of the app.

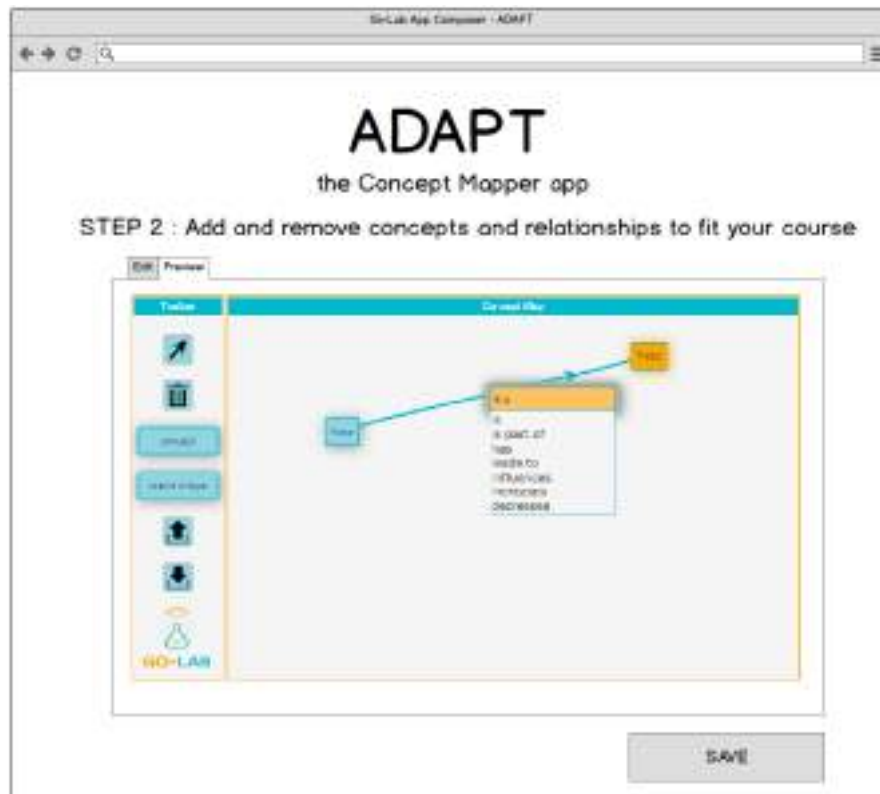


Figure 12: The alternative view of the second step to adapt an app – preview the adapted app.



Figure 13: The third step to adapt an app – save the adapted app.

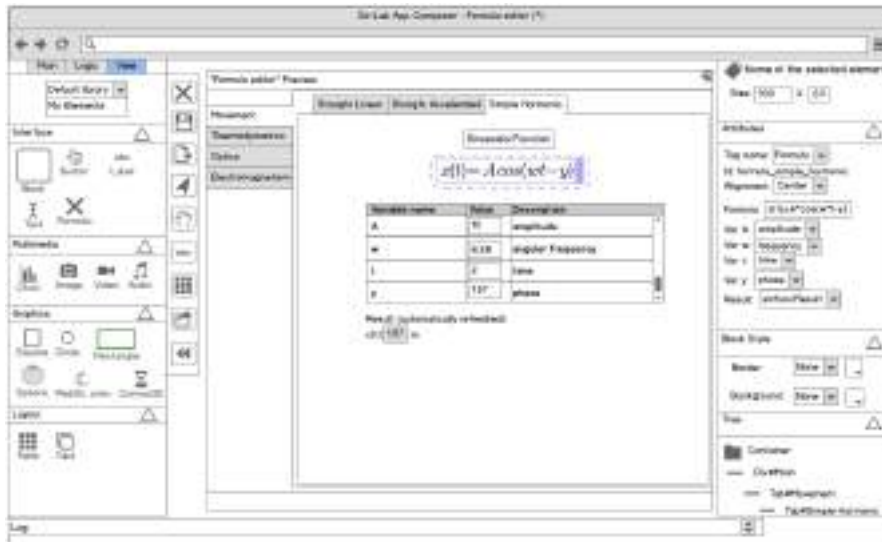


Figure 14: The advanced view editor of the app composer.

teachers to create missing apps to support their learning scenarios.

Most likely, the builder module of the app composer will only be used by a fraction of the teachers since creating an app from scratch will always be a rather complex process. There is a tradeoff: by making systems very simple, the system often becomes less powerful. If we want to create a general app builder, then it will probably require an investment of the teachers to learn how to operate this tool. Furthermore, approaches for non-programmers to adding advanced logic to apps, such as graphical programming, exist, but also often have a learning curve. Therefore, we regard the builder module as a tool for advanced, expert teachers.

Due to the large scope of the builder module we will only discuss two mockups here. Figure 14 illustrates the ‘View’ module of the builder, which allows the user to draw and configure the user interface. In the center of the screen the app’s UI is shown. Such a UI can be assembled by dragging and dropping different components available in the panel on the left side. The column with icons next to the left panel provides some basic functionality to load and save your app and manipulate the graphical components of the app’s UI. On the right-side panel, one can set different properties of the component that is selected in the center part. In Figure 14, the user has selected the formula editor and has configured for instance its size to 100x20px.

Application logic can be added to the UI components composed in the view module. One can configure this in the ‘Logic’ module, which is accessible through the tabs on top of the left-side panel. This logic module allows: (i) to define variables that can be used in the logic and view, (ii) to model progress and animations, (iii) to program the logic using the defined variables and (iv) to describe events and their effect. Figure 15 demonstrates the ‘Variables’ module that allows users to define the variables they want to use in the ‘Progress’, ‘Code’ and ‘Events’ modules. Furthermore, these variables can be used in the

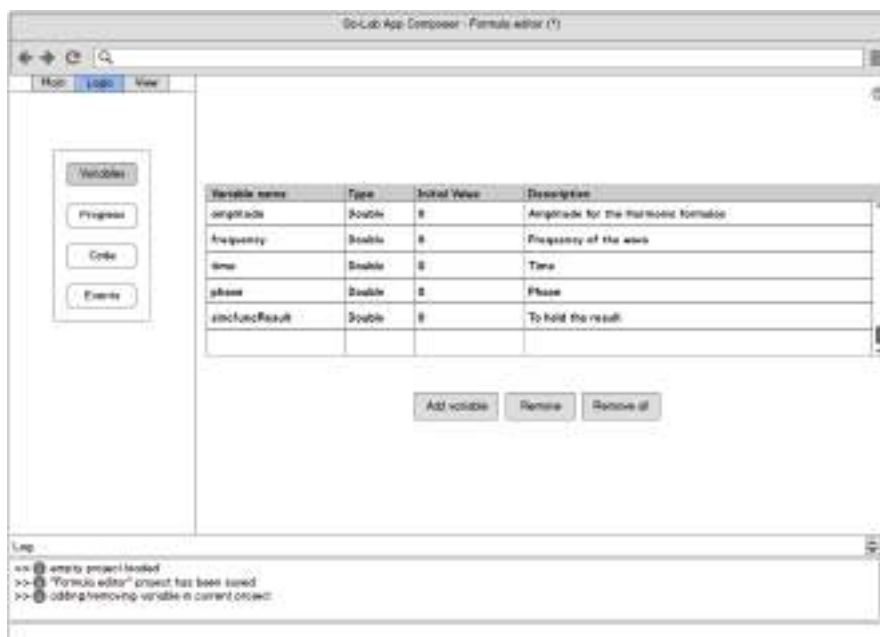


Figure 15: The advanced logic editor of the app composer.

'View' module as is illustrated in Figure 14. On the right panel of the view module one can configure the attributes of the formula editor component. The variables of the formula editor can be linked to the variables defined in the 'Variables' module of Figure 15.

The layout with the 'View' and 'Logic' modules and its submodules 'Variables', 'Progress', 'Code' and 'Events' is based on the design of EasyJava² a tool to help non-programmers to create interactive simulations in Java, which is unfortunately not compatible with mainstream tablets.

3.4 State of the art

The app composer can be defined as a tool to translate, adapt and build web applications tailored to the teachers' needs with just a few clicks. Open standards (e.g., OpenSocial gadgets and W3C widgets) have facilitated the proliferation of software for mobile and desktop platforms. However, current authoring tools for such apps still demand a strong technical background, while teachers do often not have the required skills to create with such authoring tools the apps they need to support their courses.

Through our survey of scientific literature, we found two principal solutions for creating these kinds of apps: the Widget Design Authoring Toolkit (WIDGaT) from JISC and MyCocktail Romulus mashup builder. WIDGaT is considered as "the first online 'code free' widget creation tool" both for teachers and students (Green, Pearson, Gkatzidou, & Perrin, 2012). WIDGaT provides a visual drag-and-drop based UI to create apps based on the W3C widget specification. On the other hand, MyCocktail (Chudnovskyy et al., 2012) focuses on combin-

²EasyJava, <http://fem.um.es/Ejs/>

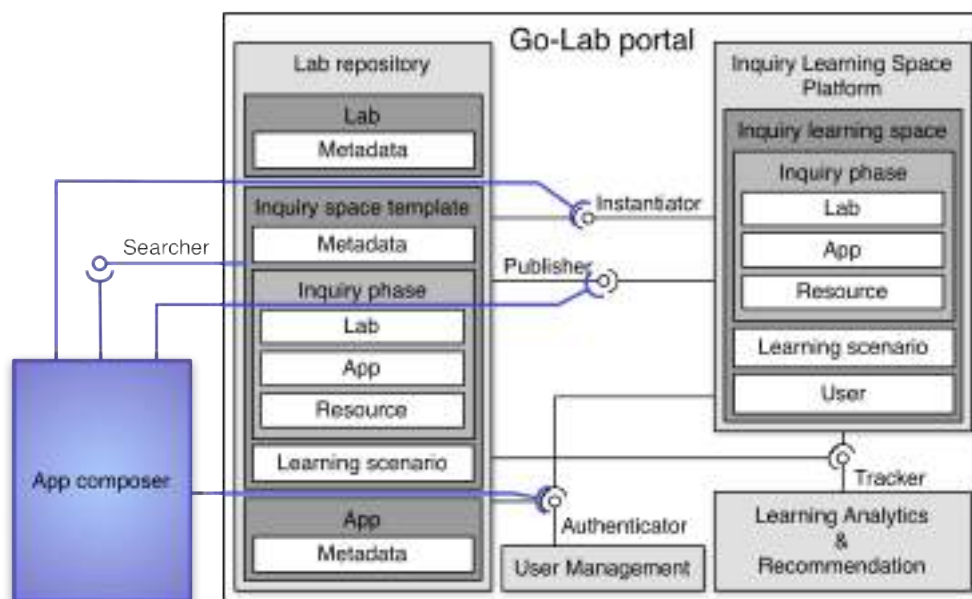


Figure 16: Communication between the app composer and the Go-Lab portal components.

ing (mashing up) information from different data sources through web services using a GUI.

Although the authoring process is described as a straightforward task, both WIDGaT and MyCocktail require moderate to advanced programming skills and the manipulation of services is not easily accessible to non-programmers. Furthermore, internationalisation support is not present. To implement most of the logic, one has to write program code and text editors must be used to check the final result of the app. As a matter of fact, these authoring tools rely on the features provided by social platforms. Apps developed with them can also be tested directly inside the platform with the software provided by the social network, such as the Google Gadget Ad Editor³ or the Open Social Explorer⁴, both of them based on the OpenSocial Dev App⁵. Such text-based editors could also be an option for advanced app composer users, where the development can be guided by providing code templates and samples.

3.5 Integration with the Go-Lab portal

The app composer is a suite of tools (consisting of the Translator, the Adaptor and the Builder) that can be seen as a platform that co-exists next to the two portal platforms, namely the lab repository and the inquiry learning space platform. Nonetheless, the app composer needs to communicate with the portal components to support its requirements.

Figure 16 demonstrates how the app composer interconnects with the Go-Lab

³Google Gadget Ad Editor, <http://www.google.com/ig/modules/gadgetads.html>

⁴Open Social Explorer, <https://github.com/OpenSocial/explorer>

⁵OpenSocial Dev App, <http://osda.appspot.com/>

portal components. The communication is driven by two use cases of the app composer (based on the requirements and mockups):

- *Loading an app from the portal:* When one wants to translate an app, it is possible to select an app from the lab repository (as illustrated in Figure 6).
- *Saving an app to the portal:* When one finishes translating, building or adapting an app, it is saved back to the portal, either to the lab repository to share the app with the Go-Lab community or to the user's ILS.

To support both use cases, the portal architecture has been extended with a `Searcher` interface, as illustrated in Figure 16. The `Searcher` interface allows the app composer to search an app in the lab repository to load it for translation. To save an app, the app composer makes use of the `Publisher` interface to save the new or edited app in the lab repository. If the user wants to make only use of his new or edited app in his own ILS, the app can be saved into the inquiry learning space platform through the `Instantiator` interface.

3.6 Implementation of the App Composer

This section describes the technical options for the future implementation of the app composer. First, an overview of the implementation is presented, then the considerations on the different composer modules are detailed, after which the common base tier is presented and finally considerations for mobile devices are presented.

3.6.1 Implementation overview

The App composer counts three different modules: the translator, the adaptor and the expert mode (see Section 3.3 for details). All of them have a subset of common services (e.g., user management and publishing an app). These features will be implemented in a base tier common to the three composers (see Figure 17). On top of this common base tier, the three composers remain decoupled and no significant amount of code is shared among them (other than the common base tier).

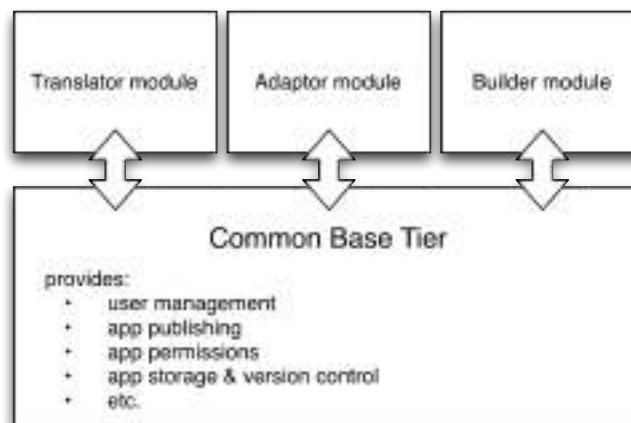


Figure 17: Internal architecture of the app composer

3.6.2 App composer modules

The translator module will extend the OpenSocial internationalisation specification to support target groups (i.e., age ranges). The client side of the translator module is quite straightforward and generally applicable to any OpenSocial app that requires translation. The server side must support our extended OpenSocial internationalisation specification to support app translations for target groups.

The adaptor module will be mostly client-based. It will define a plug-in system on top of which software developers can create new templates to create an app model or configuration and integrate them using the plug-in. The list of apps that can be adapted can be extended by creating a new configurable app and a template to create the model or configuration for that specific app. From that point, the plug-in will define how the customization is shown and will generate the models and configurations to adapt the app.

Finally, the builder modules (the 'expert' module) must support the following components:

- *A rich client that provides a simple IDE⁶*: This could be built from scratch for this particular context, or it could be built on top of the surveyed open source solutions, such as WIDGaT or MyCocktail (see Section 3.4). The former would be more easily extended for Go-Lab specific requirements and can be designed particularly with teachers in mind. The latter would make it possible to advance faster in the first stage of the development.
- *A runtime engine*: The rich client will generate a representation of the app that is translated to an OpenSocial gadget by the runtime engine. After translation, the runtime engine can execute the widget for testing and debugging purposes. This runtime engine will heavily rely on OpenSocial for supporting the features of any OpenSocial API.

The common base layer will integrate Apache Shindig⁷ as app engine.

⁶IDE = Integrated Development Environment

⁷Apache Shindig, <http://shindig.apache.org/>

4 Mobile devices compatibility

With the wider adoption of mobile devices in the classroom recently, support for mobile devices such as smartphones and tablets is very important for Go-Lab. Such compatibility is important in the design of both the portal and the app composer. The portal should allow students and teachers to do experiments on mobile devices in an ILS. To enable this, the app composer should output mobile-ready apps. In this section, we discuss the decisions we have taken so far to ensure mobile compatibility.

In order not only to sustain cross-device compatibility but display a familiar look and feel as well, it is rather important that the Go-Lab Portal, ILS and the app composer-created apps also comply with general UI principles of the Go-Lab project. Regarding the UI layout of the Go-Lab portal and App Composer generated apps, we have decided to use the responsive design¹ version of Twitter Bootstrap². The responsive layout is particularly helpful for rendering apps on smaller screens where the lack of available screen real estate is a serious matter to consider. Responsive design also offers many advantages regarding text re-flowing and image scaling that are mainly based on grid layouts and media queries³. For instance, with this technique, we can easily optimize an app that normally encompasses a horizontal toolbar in a regular viewport (desktop browser) to be presented as a vertical toolbar in portrait screens (tablet devices or mobile phones) or even be completely hidden when not needed.

Moreover, in the general UI layout and design section, the use of the Twitter Bootstrap framework will offer a consistent user experience across all devices and platforms, by using the very same user interface elements like form fields, buttons and Bootstrap utility widgets, when their inclusion is foreseen in the Go-Lab portal or in the app composer. When the use of native or third party Bootstrap widgets is not fulfilling the needs of the particular app or the portal, then jQuery UI⁴ components can be used. jQuery UI can be themed to look similar to Bootstrap with the use of jQuery UI Bootstrap⁵ so that any change in the familiar look and feel will be minor.

In general, the app composer generated apps should solely rely on HTML5-compatible components, such as HTML, JavaScript, CSS3 and HTML5 media. As a matter of fact, streaming media affects the mobile device compatibility. The preferred media streams in Go-Lab are HTML5 compatible media. As far as the streaming video is concerned the HTML5 supported formats are MP4, WebM and Ogg and in the audio streams MP3, WAV and Ogg. Although, any use of video or audio in Adobe Flash or Java Applet format should be discouraged if not avoided, when none of the recommended media types is available as a source for the to-be-created widget or lab then any video or audio stream can be used. The author and users, however, will be notified in the lab repository

¹Responsive design, http://en.wikipedia.org/wiki/Responsive_web_design

²Twitter Bootstrap, <http://getbootstrap.com>

³CSS3 media queries, <http://www.w3.org/TR/css3-mediaqueries/>

⁴jQuery UI, <http://jqueryui.com>

⁵jQuery UI Bootstrap, <http://addyosmani.github.io/jquery-ui-bootstrap/>

that the specific widget might not be mobile friendly or even incompatible with mobile.

In the future, we will further adapt these mobile requirements and continue to investigate mobile technologies to ensure that Go-Lab supports state of the art mobile devices.

5 Conclusion

This deliverable has described the specifications of the Go-Lab portal and the app composer. The portal architecture described above aims to satisfy the requirements and design principles of the Go-Lab portal. Each of the components in the architecture handles a well-defined coherent set of tasks, which fulfils the '*separation of concerns*' design objective. Through well-defined interfaces and metadata specifications, we enable a *loosely coupled* architecture with *high cohesion*. The portal architecture follows the *subsetability* design principle and thus allows an iterative and incremental development. Moreover, this enables early deployment in real-life settings, which will be exploited for participatory design evaluations. Furthermore, a first version of a few of the components of the Go-Lab portal are already available. The lab repository is available at <http://www.golabz.eu> and the ILS platform at <http://graasp.epfl.ch>.

One of the main requirements has been achieved, i.e., to provide a common, ubiquitously accessible inquiry learning environment without any installation hassle. Various design decisions contribute to this. First, the smart device specification and smart gateway for online labs empower interoperability between any lab and the ILS platform. Second, the Go-Lab portal provides a seamless integration between a repository of labs and an ILS. The integration is supported by the well-defined interfaces and metadata specifications to exchange data.

In the second part of this deliverable, we have presented the app composer, its requirements, the current mockups and future implementation plans. The mockups were used as a tool to gather requirements and discuss ideas internally in the project. In the near future, we want to check these ideas with teachers in participatory design sessions to assess our ideas and validate whether teachers need such tools, whether the proposed designs are usable and especially whether the builder module or the expert mode of the app composer is something that teachers are willing to use. Based on these participatory design and usability results we will adjust our requirements and designs. For instance, if only teachers with a background in programming are willing to create their own apps with the expert mode, they might be more efficient with a more expert integrated development environment (IDE) than with a beginner-friendly IDE. The results and the updated specifications will be described in deliverable D5.6 in M36.

6 Appendix A: Lab metadata

This appendix describes the efforts of the cross-workpackage metadata discussions and the current version of the Go-Lab metadata specification. The Go-Lab metadata specification is based on a combination and extension (fitting the Go-Lab requirements) of the ROLE Ontology and the GOLC specification (T. Richter & Zutin, 2012). Primarily, the Go-Lab metadata is implemented and used in the lab repository of the Go-Lab Portal. It describes and indexes all learning content stored in the repository: labs, apps, resources, and inquiry learning spaces.

6.1 The Metadata survey conducted in WP2

Work Package 2 has conducted a survey of the existing online labs (see Deliverable D2.1). The related requirements analysis is one of the starting points for the design and realisation of the Go-Lab portal. Work Package 2 also specifies some useful taxonomies for the Go-Lab metadata. Examples are:

- *Lab types*: remote labs, virtual labs and data sets.
- *Grade levels*: primary education (10-12 years old), lower secondary education (12-15 years old), upper secondary education (15-18 years old), higher education bachelor, higher education master.
- *Inquiry learning phases*: orientation, conceptualisation, investigation, conclusion & discussion.

6.2 Go-Lab metadata specification

To be able to describe an inquiry learning space, different components need to be described, for instance labs and apps. It is hard to capture this in one metadata schema, therefore we have decided to offer a schema for each component.

In addition, we specify six groups of metadata types for various use purpose:

- *General metadata* describes the overall information such as titles or descriptions and the content-specific metadata group, such as subject domain(s) and languages.
- *Pedagogical metadata* is metadata for educational purposes, such as grade level. It is not inherent to the content, but for content management based on pedagogical requirements.
- *Organisational metadata* is administrative metadata for content management like access rights, licensing info, etc.
- *Technical metadata* describes technical information of the content, such as URL. User-specific metadata is metadata related to users and includes social metadata such as users' rating and ranking.

In the remainder of this appendix, we present the preliminary metadata schema that describes each component that is needed to describe an ILS. Furthermore, we present the relationship of these schema with taxonomies, other components, as well as a mapping to the related existed metadata standards.

6.2.1 Lab metadata

Labs, whether they are remote labs, virtual experiments, or data sets, contain a large set of metadata. Teachers search labs for their courses based on the metadata in the portal. We specify the following metadata with the reference to the existing GOLC¹ specification (T. Richter & Zutin, 2012) in Table 2. The GOLC has created a comprehensive metadata set for labs.

This deliverable presents the lab metadata necessary for the portal. It contains a subset of the lab metadata inventory presented in D2.1. Besides the aforementioned metadata for labs, some other metadata maybe be further specified in the future. For technical metadata, *compatibility* and *additional software request* could be useful to describe labs as well. *Compatibility* refers to mobile device and browser compatibility (e.g., HTML5). For user-specific metadata, *teachers' ICT competency level* tells school teachers whether it requires a lot of ICT competency to operate an online lab, to design lessons and apps. *Accessibility* describes the support to students with special needs. Social metadata like *rating* will be included to rank the labs.

6.2.2 App metadata

Apps (in our case often OpenSocial gadgets) provide support for inquiry based learning using online labs. Most of the apps will not be built by teachers. Nevertheless, the app composer will allow teachers to translate adapt and build apps. There is few metadata standards to describe apps from the online lab communities. We have developed the metadata for apps based on the research results of the previous EU IP project ROLE (Govaerts et al., 2011), as listed in Table 3. This set of metadata is rich enough to cover the app categories we focus on in Go-Lab, i.e., interfacing apps for remote labs, simulation apps (virtual labs), analysis and visualisation apps for data sets, guidance apps, as well as links to external resources.

6.2.3 Resource metadata

Additional resources besides apps and labs have versatile content types. Videos, Images, text, eBooks, websites and any other kind of multimedia content contribute to resources. Thus, their metadata is diverse. Such resources will be embedded as content in ILS templates and dedicated apps, as well as lab metadata as URLs.

6.2.4 Inquiry learning space template (ILS) metadata

Inquiry learning space templates consist of at least one online lab and a bundle of apps and resources to help teachers prepare and conduct lab-based school courses for the school students. Since ILS templates make use of online labs, their metadata is strongly based on the lab metadata. To support inquiry learning, ILS templates consist of different inquiry phases (see Section 2 and deliverable D2.2). Each inquiry learning phase has the metadata *title*, *ILS term*, and *content*. An *ILS term* is a standardised inquiry learning phase term. Teachers

¹Global Online Laboratory Consortium (GOLC), <http://www.online-lab.org/>

are able to edit the title of each inquiry learning phase. To enable apps to query in which phase they are running, ILS terms keep these personalised inquiry learning phase titles consistent, and also searchable in the portal. Each inquiry learning phase consists further apps, labs, and resources.

In Graasp, teachers can further adapt and modify the inquiry learning spaces and publish them back to the lab repository. So the ILS metadata will be updated accordingly. The metadata set of ILS is listed in Table 5.

Similar to the lab metadata, *big ideas*, *compatibility*, *additional software request*, *teachers' ICT competency level*, *accessibility* and *rating* will be further specified in the future. On the other hand, ILS templates could be further developed and referenced to their previous versions. Thus, *isBasedOn* will be used to show the evolution of the ILS templates.

6.2.5 Taxonomy

All learning content (labs, apps, resources, and inquiry learning space templates) share the taxonomies of the subject, grade level, language, level of difficulty, level of interaction, inquiry learning phase fields. This enables cross-content search by metadata. In the lab repository, search for a certain subject or grade level could deliver a set of results consisting of labs, apps, and inquiry learning space templates. Work package 2 has already surveyed metadata taxonomies for some of these fields (see deliverable D2.1). Which taxonomies we will finally adopt in the Go-Lab metadata specification is still under discussion and will be further detailed in deliverable D5.6.

Metadata groups	Metadata	Taxonomy	Relation	GOLC element
General	Lab title			InteractionPackage:Title
	Lab owner			InteractionPackage:RightsHolder
	Lab description			InteractionPackage:Description
	Lab category	x		InteractionPackage:interactionPackage Type
	Status			
	License	x		Rig::License / interactionPackage:License
	Language	x		interactionPackage::language / lesson:language / activity::language
	Keyword			
	Booking required		smart device	Rig:bookingSystemURL
	Contact details			Rig::rightsHolder::Name
Pedagogical	Grade level	x		
	Big ideas			
	Subject domain			
	Educational objective			activity::learningObjective / lesson::learningObjective
	Inquiry learning phase	x		
	Level of difficulty	x		
Technical	Level of interaction	x		
	Lab URL			
Additional materials	Resource type			
	Resource URLs			
	Lesson plan			
	Supportive Apps		apps	

Table 2: Online lab metadata

Metadata groups	Metadata	Taxonomy	ROLE Ontology
General	App title		dcterms:title
	App category	x	role:category
	App description		dcterms:description
	App creator		foaf:maker (foaf Person type)
	Subject domain	x	
Padagogical	Inquiry learning phase	x	
Organisational	Contact details		
	License	x	
Technical	App URL		dcterms:source
	App functionality	x	role:functionality
	App type	x	rdf:type
User-specific	Lesson plan		
	Keyword		

Table 3: App metadata

Metadata groups	Metadata	Example: Go-Lab Concept Mapper
General	App Title	'Go-Lab Concept Mapper'
	App category	'Plan and organise ; Collaborate and communicate; Explore and view'
	App description	'A concept mapping tool to support learners in various inquiry phases, e.g., Orientation and Conceptualization.'
	App creator	'Lars Bollen, UTwente, l.bollen@utwente.nl'
	Subject domain	'n/a; domain independent'
Pedagogical	Inquiry Learning phase	'Orientation; Conceptualization'
Organisational	App licence	'free'
	Contact details	'Lars Bollen; l.bollen@utwente.nl'
Technical	App URL	'http://go-lab.gw.utwente.nl/sources/tools/conceptmap/src/main/webapp/conceptmap0.6.html'
	App functionality	Modelling, Inquiry support
	App type	'Open Social Widget'
User Specific	Lesson plan	//dynamic data
	Keyword	//dynamic data

Table 4: Metadata set of the app 'Go-Lab Concept Mapper'

Metadata groups	Metadata	Taxonomy	Relation	GOLC element
General	ILS title			InteractionPackage:Title
	ILS owner/creator			InteractionPackage:RightsHolder
	ILS description			InteractionPackage:Description
	Subject domain	x		interactionPackage::scientificField / lesson::scientificField / activity::scientificField
	Language	x		interactionPackage::language / lesson:language / activity::language
	Additional supportive materials		resource	interactionPackage::supportingMaterial / Rig::supportingMaterial
	Supportive app		apps	
	Used labs		labs	
	Grade level	x		
	Educational objective			activity::learningObjective / lesson::learningObjective
Pedagogical	Inquiry learning phase	x		
	Level of difficulty	x		
Organisational	License	x		Rig::License / interactionPackage::License
	Lab URL			
Technical	Level of interaction	x		
	Lesson plan			
User-specific	Keyword			

Table 5: Inquiry learning space metadata

7 Appendix B: Go-Lab portal UI functionality

This appendix presents the UI of the Go-Lab portal in more detail. Its goal is to provide a walkthrough on how portal users can create, edit and reuse inquiry learning spaces. Additionally, this appendix will also briefly present ideas for future UI features. Both platforms of the portal and their interplay will be discussed. Finally, we will present the demo inquiry learning spaces we have created so far applying the current version of the portal.

7.1 The lab repository

The lab repository (see Section 2.5.2), available at <http://go1abz.eu/>, enables the storage of labs, apps and ILS templates and provides various features for teachers to browse and find such labs, apps and ILS templates. This section will highlight the currently implemented features and their implementation details briefly. Since this is the first iteration of the lab repository, keep in mind that many UI features and architecture interfaces are not yet implemented.

7.1.1 The currently implemented features of the lab repository.

Figure 18 illustrates the front page of the lab repository, where users start their search for resources for their course. From that page they can access the main currently implemented features. The top menu provides access to pages that contain a list of all online labs (see Figure 19), a list of all apps (see Figure 20) and a list of all shared ILS templates (see Figure 21)

Each content type stored in the lab repository, i.e., online labs, apps and ILS templates, has its own listing page which is available through the top menu. For instance, Figure 19, Figure 20 and Figure 21 illustrate the listing pages for respectively online labs, apps and ILS templates. As one can see, faceted search functionality is again provided. One thing should be pointed out: in the UI we call the ILS templates '*inquiry spaces*'. We hope that this makes the concept of ILS templates clearer to teachers and lab owners. Whether this is really the case will be shown in future usability studies of the portal.

Each of the items in the listing pages described above can be selected, which will lead the user to a detailed information page. Figure 22 shows an example for an ILS template, 'Buoyancy Inquiry Learning Space'. In the right column of Figure 22 one can see the export to Graasp functionality. By clicking on this button the ILS template is instantiated in Graasp and teachers can re-use and adapt the ILS to their specific needs in Graasp, which will be described in more detail in the next section.

7.1.2 Future work on the lab repository

There is one caveat with regard to the ILS template export feature, currently the user has to be logged into Graasp before attempting to export an ILS template. In the near future we plan to implement a common identity between Graasp and the lab repository, which will resolve this issue. Furthermore, we plan to update the graphical design of the lab repository and add search features, as well as user-friendly ways to add apps, online labs and ILS templates to the lab



Figure 18: The front page of the lab repository.



GO-LAB
CLEAN ONLINE SCIENCE LABS
FOR THE UNIVERSITY OF SCIENCE

Home Online Labs Apps Inquiry Spaces

Online Labs

The online labs aim at supporting inquiry-based learning and providing the possibility to conduct scientific experiments in a virtual environment. Importantly, the inquiry process should be well structured and scaffold to achieve optimal learning results. Scaffolding refers to support (dedicated software tools) that helps students with tasks that they cannot complete on their own. For example, they can help students to create hypotheses, design experiments, make predictions, and formulate interpretations of the data.

Online laboratories can be of two kinds. Remotely-operated educational labs (remote labs) provide students with the opportunity to collect data from a real physical laboratory setup, including real equipment, from remote locations. As an alternative there are virtual labs that simulate the real equipment. Remote and virtual labs both have specific advantages for learning and can be combined to support specific learning activities. Additionally, the Go-Lab project offers access to scientific databases, tools, and resources supporting inquiry learning activities of the students.

Please use the filters on the right to find appropriate online labs and resources for your class.

HYPATIA

Subject: Particle physics
Lab type: Analysis tools
Lab owner: University of Athens, department of Physics / Institute of A
 Christine Kourkouris
Web link: <http://hypatia.lss.fnf.gr/>
Languages: English
 Greek
Grade level: Secondary Education (15-18 years old)
 Higher education bachelor

HYPATIA is an event analysis tool for data collected by the ATLAS experiment of the LHC at CERN. Its goal is to allow high school and university students to visualize the complexity of the

Difficulty Levels
 Medium (2)
 High (1)

Grade levels
 Secondary Education (15-18 years old) (3)
 Higher education bachelor (2)
 Higher education master (1)
 Secondary education (12-15 years old) (1)

Filter by inquiry learning phases:
 Investigation (3)
 Conclusion (2)
 Conceptualization (1)
 Discussion (1)
 Orientation (1)

Interaction levels
 Medium (2)
 High (1)

Filter by keywords:
 quantum (1)
 electricity (1)
 engineering (1)
 lab (1)
 HEP (1)

Lab types
 Remote labs (2)
 Analysis tools (1)

Figure 19: The 'Online labs' page lists all online labs in the lab repository.



GO-LAB
GLOBAL ONLINE SCIENCE LABS
WEB AND MOBILE UNIVERSAL BY DESIGN

Home Online Labs **Apps** Inquiry Spaces

Apps

Apps, also known as tools or widgets, are small web based software applications supporting specific learning or teaching goals and tasks in online labs. Apps can be added to a Inquiry Learning Space together with online labs. Apps are grouped within inquiry learning spaces according to their functionalities and purposes, and used to support particular experimenting and learning activities in online labs.

Go-Lab Concept Mapper

 Category: Plan & organise
Collaborate & communicate
Explore & view
App type: Open Social widget

A Concept Mapping tool to support learners in... working with concepts for online experiments.

Periodic Table

 Category: Explore & view
Functionality: test
App type: Open Social widget

App types
Open Social widget (6)

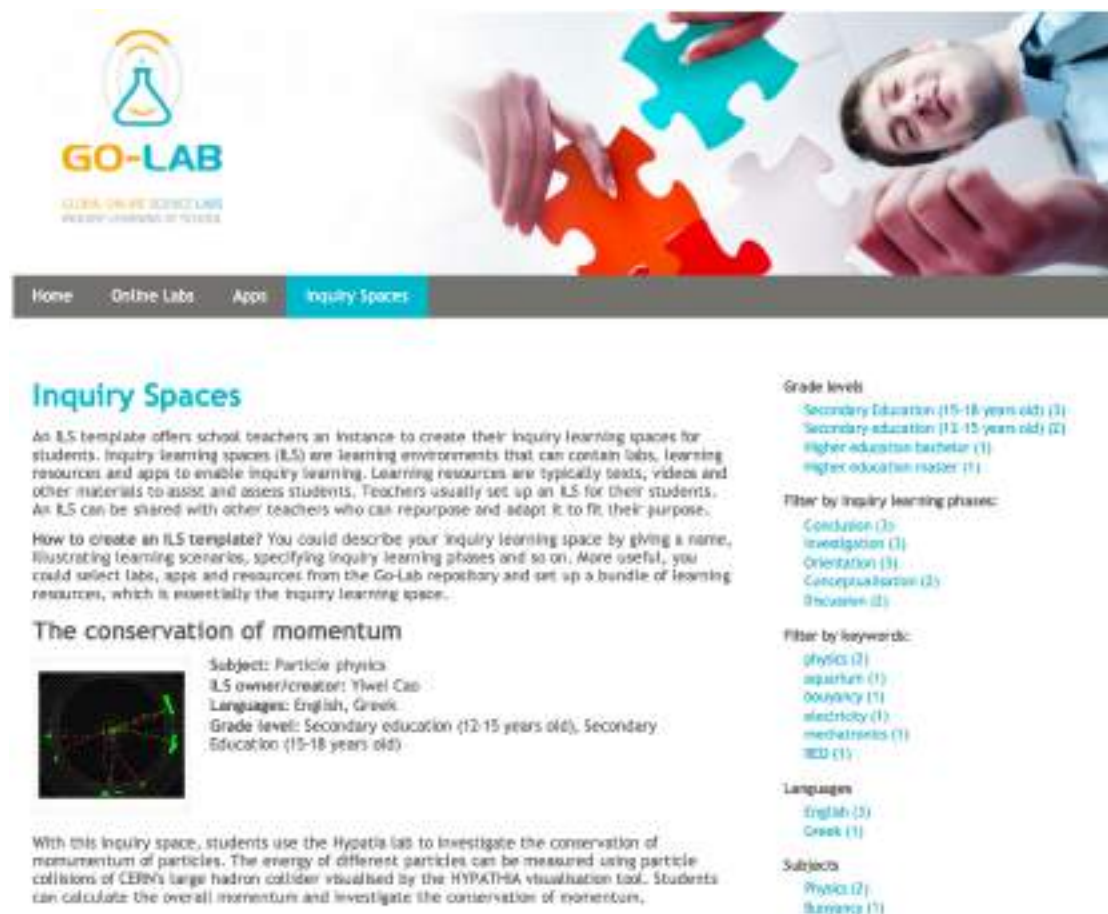
Categories
Explore & view (6)
Plan & organise (3)
Collaborate & communicate (1)
Create & manipulate (1)

Functionality
calculation (1)
editing (1)
test (1)
view (1)
write (1)
writing (1)

Filter by inquiry learning phases:
Conceptualisation (5)
Orientation (2)
Conclusion (1)

Filter by keywords:
chart (2)
graph (2)
calculation (1)
collaborate (1)
editing (1)
equation (1)
function (1)
Apparents (1)
mendeleev (1)

Figure 20: The 'Apps' page lists all apps in the lab repository.



GO-LAB
GLOBE ONLINE SCIENCE LABS
FOR EDUCATIONAL USE

Home Online Labs Apps **Inquiry Spaces**


Inquiry Spaces

An ILS template offers school teachers an instance to create their inquiry learning spaces for students. Inquiry learning spaces (ILS) are learning environments that can contain labs, learning resources and apps to enable inquiry learning. Learning resources are typically texts, videos and other materials to assist and assess students. Teachers usually set up an ILS for their students. An ILS can be shared with other teachers who can repurpose and adapt it to fit their purpose.

How to create an ILS template? You could describe your inquiry learning space by giving a name, illustrating learning scenarios, specifying inquiry learning phases and so on. More useful, you could select labs, apps and resources from the Go-Lab repository and set up a bundle of learning resources, which is essentially the inquiry learning space.

The conservation of momentum

Subject: Particle physics
ILS owner/creator: Yiwei Cao
Languages: English, Greek
Grade level: Secondary education (12-15 years old), Secondary Education (15-18 years old)



With this inquiry space, students use the Hypatia lab to investigate the conservation of momentum of particles. The energy of different particles can be measured using particle collisions of CERN's large hadron collider visualised by the HYPATHIA visualisation tool. Students can calculate the overall momentum and investigate the conservation of momentum.

Grade levels

- Secondary Education (15-18 years old) (3)
- Secondary education (12-15 years old) (2)
- Higher education bachelor (1)
- Higher education master (1)

Filter by inquiry learning phases:

- Conclusion (3)
- Investigation (3)
- Orientation (3)
- Conceptualisation (2)
- Discussion (2)

Filter by keywords:

- physics (2)
- experiment (1)
- oscillancy (1)
- electricity (1)
- mechanisms (1)
- IBD (1)

Languages

- English (2)
- Greek (1)

Subjects

- Physics (2)
- Language (1)

Figure 21: The 'Inquiry Spaces' page lists all ILS templates in the lab repository.



GO-LAB
GLOBAL ONLINE SCIENCE LABS
INQUIRY LEARNING AT SCHOOL

Home Online Labs Apps Inquiry Spaces

Sinking or floating?


Use this Inquiry Space in your course!

[Create Inquiry Space](#)

ILS owner/creator: Sten Govaerts
Learning objective: In science, buoyancy is an upward force exerted by a fluid that opposes the weight of an immersed object. In a column of fluid, pressure increases with depth as a result of the weight of the overlying fluid.
Grade level: Secondary education (12-15 years old), Secondary Education (15-18 years old)
Subject: Physics, Buoyancy
Languages: English
Inquiry learning phases: Orientation, Conceptualisation, Investigation, Conclusion, Discussion



Supportive apps:
[Periodic Table](#)



Category: Explore & view
Functionality: test
App type: Open Social widget
 A widget to see the chemical elements in the standard periodic table. The elements are shown with their atomic number and their symbol, which is a one to three letter long abbreviation for instance, Fe for Iron.

Used labs:
[WebLab Deusto Aquarium](#)



Subject: Physics
 Buoyancy
Lab type: Remote labs
Lab owner: Web Lab Deusto
 Pablo Orduña
Web link: <https://www.weblab.deusto.es/w>
Language: English
Grade level: Secondary education (12-15 years old)
 Secondary Education (15-18 years old)
 The lab allows students to drop and raise balls in an aquarium and calculate the buoyancy of the balls.

Function plotter



Figure 22: This is a detail page of an ILS template. The orange button on the right provides import of the ILS template in Graasp.



Figure 23: A screenshot of the inquiry learning space that was created from the ILS template in the lab repository shown in Graasp.

repository.

7.2 The inquiry learning platform (Graasp)

The inquiry learning space platform (see Section 2.5.2) is making use of the Graasp platform, available at <http://graasp.epfl.ch>. Graasp provides an environment to author ILS by teachers and lab owners, and allows teachers to share such ILS with their students who use Graasp to operate the online labs and apply inquiry-based learning. This section will highlight the currently implemented features and their implementation details briefly. We will continue to explain the features of Graasp for authoring ILS and inquiry learning from where we left off with the explanation of the lab repository.

7.2.1 The currently implemented features of inquiry learning platform.

To recapitulate, after clicking the orange 'Create your own' button in Figure 22, an ILS is created in Graasp based on the ILS template we have selected in the lab repository. Figure 23 shows the inquiry learning space in Graasp that has been created based on the 'Buoyancy Inquiry Learning Space' ILS from the lab repository. One can see that each subspace in this ILS corresponds to one inquiry phase. Inside these spaces the apps and resources related to the corresponding inquiry phase are stored. Furthermore, the description of each inquiry phase space is used as guidelines for students. Teachers can edit the ILS (e.g., change the space description, remove an inquiry phase space or add apps) using regular Graasp features. Teachers can then share the ILS with their students by clicking on the 'Share' button on the right of the space title 'Buoyancy Inquiry Learning Space'. This opens up the modal dialogue seen in Figure 24. The dialogue window provides a URL to the student mode of the



Figure 24: A screenshot of the ‘Share’ dialog that provides a secret URL and social media share buttons.

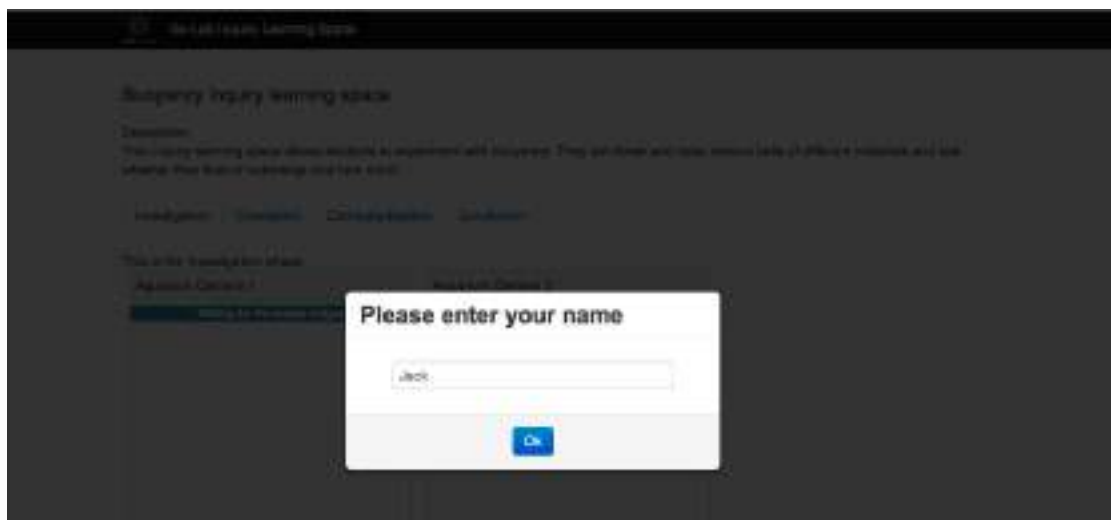


Figure 25: A screenshot of the ‘Login’ dialog of the student mode of the ILS.

ILS, which can be shared with the students and a set of social media icons for sharing the ILS. When the students browse to this URL, they are asked to provide a nickname, as illustrated in Figure 25.

Once the student provides a nickname, she can access the ILS in the student mode (see Figure 26). The students are actually not logged into Graasp and therefore do not need a Graasp account. This mechanism allows teachers to more quickly deploy an ILS in the classroom without asking each student to create a Graasp account and ensure privacy (only the teacher knows who is hiding behind a nickname). Additionally this allows for more anonymous user tracking. From Figure 26, one can see that the inquiry phase spaces from the regular Graasp view or teacher mode (see Figure 23) are represented here as tabs. Furthermore, the description of the ILS is presented above the tabs and the description provided in each inquiry phase space is provided under each tab.

The screenshot shows the 'Go-Lab Inquiry Learning Space' interface. At the top, there is a header with the Go-Lab logo and the text 'Go-Lab Inquiry Learning Space' on the left, and 'Hello Jackie!' on the right. Below the header, the title 'Buoyancy inquiry learning space' is displayed. A 'Description' section follows, stating: 'This inquiry learning space allows students to experiment with buoyancy. They can lower and raise various balls of different materials and see whether they float or submerge and how much.' Below the description are four tabs: 'Investigation' (selected), 'Orientation', 'Conceptualization', and 'Conclusion'. A message reads: 'This is the Investigation phase.' Below this, there are two camera viewports labeled 'Aquarium Camera 1' and 'Aquarium Camera 2'. Each viewport shows a video feed of an aquarium experiment with a red timer at the top center of each frame showing '04:18'. Below each video frame are options for 'jpg' and 'video (mjpeg)', and a 'Finish' button.

Figure 26: A screenshot of the student mode of the ILS.



Figure 27: Creating a new ILS in the teacher mode in Graasp.

Teachers can also create their ILS independently of the ILS templates available on the lab repository. This can be done in the same way as creating a regular Graasp space and then clicking on the *'This is an inquiry learning space'* checkbox, see Figure 27. This will create a space with by default the five inquiry phases (proposed in D1.1) set. From this empty workspace, they can start constructing their vision.

7.2.2 Future work on the inquiry learning platform

One of the features we plan to implement next, would be the ILS export, where an ILS created in Graasp can be exported to the lab repository as an ILS template. One of the requirements for this is to have a common user identity between the lab repository and Graasp to determine the creator of the ILS template. Visually the export feature could be implemented as a button on each ILS page in Graasp. Once this button is clicked the ILS content and structure is sent to the lab repository, where the teacher can describe the ILS with metadata according to our metadata schema (see Appendix A).

7.3 Example inquiry learning spaces

By applying the current portal UI functionality described in the previous sections, we have created a few ILS for demonstration purposes. Some of these ILS have been used to present the Go-Lab project in conferences and participatory design sessions with teachers. This section provides a list of these ILS and their respective URLs. These spaces are live and subject to change. The scenarios used for these ILS come from those applied in the mockups (see D1.1).

- *Buoyancy lab*: This lab uses the aquarium-based remote lab that allows buoyancy experiments by controlling balls of various materials that can be submerged in a fishtank.
 - The teacher mode: https://graasp.epfl.ch/#item=space_4648

- The student mode: <https://graasp.epfl.ch/metawidget/1/8ce2e500\f04254a531ae7a2f675623c471138dfc>
- *Interacting galaxy lab*: This lab uses the Faulkes telescope remote lab that allows the investigation of varying galactic morphologies.
 - The teacher mode: https://graasp.epfl.ch/#item=space_4642
 - The student mode: <https://graasp.epfl.ch/metawidget/1/60a035be\c69cc0d4194799eaeb0a3e9ad5fd2b48>
- *Hypatia lab*: This lab uses the Hypatia visualisation tool and data sets of the large hadron collider to determine the total momentum of all particles that are tracked after a particle collision.
 - The teacher mode: https://graasp.epfl.ch/#item=space_4634
 - The student mode: <https://graasp.epfl.ch/metawidget/1/88658671\fff4cfe39b2febb78b481d47763ec94b>

All these demo spaces are available in the Go-Lab Prototypes space in graasp, <https://graasp.epfl.ch/#url=Protos>.

8 Appendix C: Graasp development

This appendix will highlight the changes made to the Graasp platform developed in the framework of the ROLE IP since the start of the Go-Lab project (November, 2013). These changes were driven by Go-Lab requirements or to ensure future growth, scalability and stability of Graasp.

Since the beginning of the Go-Lab project, we have started using GitHub¹ to source code versioning control and issue tracking. Related to the Graasp development, there are three different repositories: the Graasp repository², the Graasp Shindig repository³, and Graasp gadgets repository. Up till now, we have resolved 256 issues on Graasp⁴, 19 issues on the Graasp Shindig repository⁵, and 16 issues on Graasp gadgets repository.

This appendix is structured as follows: first, the work on performance improvements is discussed, after which the development infrastructure is presented. Then, the development of activity tracking, OpenSocial and user management is highlighted. Finally a section with smaller improvements and new features concludes this appendix.

8.1 Performance improvements

The responsiveness and the scalability of Graasp has been improved to enable a better user experience and to provide a stable platform to build the Go-Lab portal upon. To be able to do meaningful performance improvements, the source code was analysed and profiled to identify performance bottlenecks. Afterwards, critical parts have been removed, redesigned and reimplemented. Two main metrics were used to measure the responsiveness: *the average first load time* and *the average transition time between spaces*. For Graasp this is almost equal to the app server response time (see Figure 28). The measurements were done with the help of the NewRelic web application monitoring tool⁶.

To achieve better responsiveness and scalability, the following improvements have been made:

- *Code Refactoring*: A result of Graasp profiling was the discovery that Graasp's notification subsystem contained a performance bottleneck. Inefficient code has been removed and part of it has been reimplemented.

Another performance-related issue was located in the network communication between the front-end (the presentation tier) and the back-end (the logic tier) (see deliverable G5.2 for the technical documentation of Graasp). The presentation tier and the logic tier communicate using JSON

¹GitHub, <https://github.com>

²Graasp GitHub repository is private due to source code licensing restrictions.

³Graasp Shindig repository, <https://github.com/react-epfl/shindig-react>

⁴Graasp issues page, <https://github.com/react-epfl/graasp/issues?page=1&state=closed>

⁵Graasp Shindig issues page, <https://github.com/react-epfl/shindig-react/issues?page=1&state=closed>

⁶New Relic, <http://newrelic.com/>

objects. The creation of these objects was inefficient, often due to redundant JSON fields. Such fields were identified and removed. This enabled a reduction in network traffic by a factor of 2 to 10 times providing an important responsiveness improvement.

- *Loading members asynchronously:* Members of a space are now loaded asynchronously in a separate call, which prevents blocking the UI and increases the responsiveness.
- *Memcached caching:* A two-level caching for item sequences has been implemented in Graasp. When a user enters a space, first a general sequence of items is cached per space without taking into account any access control permissions. Afterwards, a sequence for a specific user based on the user's access control permissions is cached for each space. The caching enables to load spaces without accessing the MySQL database enabling high scalability.
- *Image optimization and caching:* By compressing the icons used in Graasp, we were able to reduce their total size by 60%. Additionally, the web server has been configured to force clients to cache images for longer time periods.

Overall, the changes resulted in the following improvements of the measured metrics:

- *The average first load time:* Initially, the first loading was taking on average 10-15 seconds. Now, the first load time takes 3.1 seconds on average.
- *The average transition time:* Initially, making a transition from one space to another was taking from 3 to 60 seconds (in some cases even more depending on the number of items in the space, and the user). Now, space transitions happen under 100 ms.

The current Graasp performance is presented in Figure 28. From the top chart, one can see that at the moment the network loading (brown) is very low and that the main time is spent on rendering the page (blue) and processing the DOM tree (yellow). From the bottom chart one can see that memcached (the tiny dark blue line on the top) is very efficient in serving requests and that most of the time is spent on querying the database. If further performance improvements become necessary, the page rendering and DOM processing should be further improved as these are currently the most time-consuming.

8.2 Infrastructure improvements

To enable collaboration between the partners and guarantee high quality of the software artefacts produced, the following infrastructure has been set up:

- *GitHub version control repository:* Originally, the Graasp source code was hosted on a private Subversion repository. We migrated the source code to GitHub and access is available to all technical partners (see the Appendix of deliverable G5.2). All scripts and services (e.g., building, deployment)

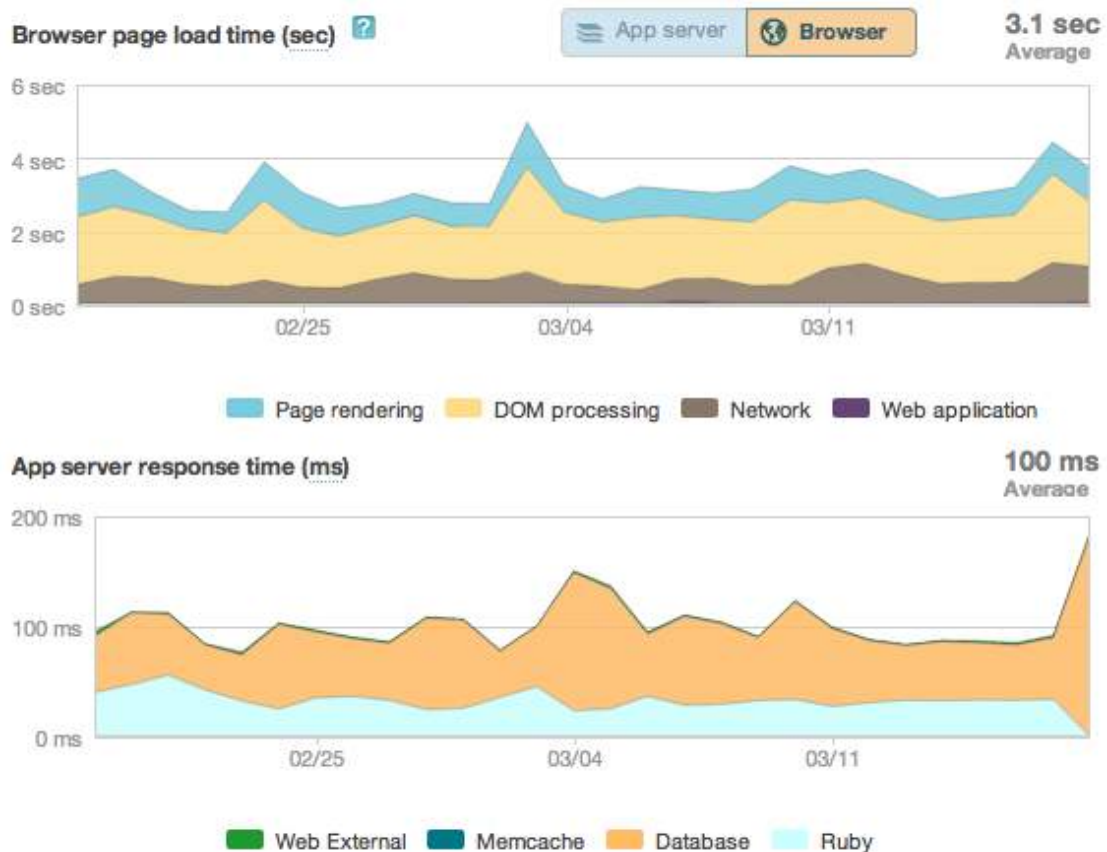


Figure 28: Graasp performance as monitored by NewRelic in March 2013.

are adapted to GitHub as the version control system. Additionally, the features of GitHub are used in the project for code review.

- *GitHub issue tracker*: To track issues related to the development of Graasp, the GitHub issue tracker is used.
- *Jenkins continuous integration server*: The Jenkins server⁷ is used for continuous integration. The continuous integration server is configured as such, that when new code is merged into the source code on Github, an updated version of Graasp gets automatically deployed to the test server⁸. We plan to integrate automated tests into the continuous integration to ensure high software quality.
- *New Relic Web Application Monitoring*: Through the use of the NewRelic web application monitoring tool, we are able to observe the performance of Graasp almost in real time, and react on and analyse issues at the time they occur.
- *Pingdom uptime monitoring*: By using the Pingdom service⁹, Graasp team members will receive a notification when the Graasp server is down in

⁷Jenkins, <http://jenkins-ci.org/>

⁸The Graasp test server, <http://reacttest.epfl.ch/>

⁹Pingdom, <https://www.pingdom.com/>

order to react fast. Note that in recent months, Graasp's uptime has been close to 100%.

- *HipChat integration with the Graasp development infrastructure:* HipChat¹⁰ is a hosted group chat and instant messaging service for teams. It allows real-time collaboration and saves conversation histories. We have integrated HipChat with other components of our infrastructure (e.g., GitHub, NewRelic & Capistrano). Through HipChat we provide a communication and awareness channel on the project development.
- *User feedback mechanism:* We have set up UserVoice service¹¹ to gather feedback from Graasp users. Every page has a Feedback button on the left, that allows user to leave their opinions or suggestions.
- *Bundler for dependency management:* We have moved to Bundler¹² for dependency management in Graasp. Now deploying Graasp to a server or setting it up on a developer's computer is much more convenient and takes much less time.
- *Maintenance scripts:* A set of Graasp database administration scripts has been developed, e.g., for fixing dangling links and regenerating sequences of items in a space. These scripts enable the swift execution of laborious maintenance tasks.

8.3 Activity Tracking

To support learning analytics, user action tracking has been implemented in Graasp. We have decided to use the ActivityStreams specification¹³ to represent data about user activities in the Go-Lab project (for more information on learning analytics specifications, see D4.2). The ActivityStreams specification provides a vocabulary list of verbs to track actions of users¹⁴. Currently, the following user actions are tracked in Graasp (note that an actor is a Graasp user in this case):

- *Add:* An actor has added an item to a space.
- *Update:* An actor has modified an item.
- *Invite:* An actor has invited another actor into a space.
- *Access:* An actor has opened a space or a resource or launched an application.
- *Join:* An actor has joined a space.
- *Remove:* An actor has removed an object from a space.

¹⁰HipChat, <http://www.hipchat.com>

¹¹UserVoice, <https://www.uservoice.com/>

¹²Bundler, <http://gembundler.com/>

¹³The ActivityStreams specification, <http://activitystrea.ms>

¹⁴The ActivityStreams verbs, <http://activitystrea.ms/registry/verbs/>

- *Delete*: An actor has deleted an item (space, resource, widget).
- *Access*: An actor has opened a space or a resource or launched an application.

We have implemented the ActivityStreams API in Graasp, which allows external services, when properly authenticated, to query activities stored in Graasp and to create new activities. The ActivityStreams API of Graasp itself was used to implement the ActivityStreams API in Shindig (as part of OpenSocial 2.5) to support OpenSocial widgets, that submit or query user activities in Graasp (e.g., for learning analytics). Most of the implementation has been done directly in the activity stream controller of Graasp, and the role of Shindig is mainly to forward ActivityStreams REST requests to Graasp. Through redirecting the Shindig APIs to the Graasp APIs, we aim to improve the maintainability and development of these APIs.

Both GET and POST requests have been implemented :

- *Get activities*: It is possible to retrieve the activity stream of a given user in any given space. The activities are retrieved as a collection of activityEntries. Every activityEntry consists of a verb, as well as an actor, an object and possibly a target (which are activityObjects). The standard OpenSocial request parameters have been implemented and allow to filter and paginate activities.
- *Create an activity*: Similarly, one can post a new activity to Graasp, by specifying the structure and the content of the corresponding activityEntry (verb, target, object, actor...).

Additional documentation about the ActivityStream implementation in Graasp and Shindig can be found at https://github.com/react-epfl/shindig-react/blob/activity_streams/open_social_api.md#activities-activity-streams.

We have developed and implemented a clear activity access policy:

- Every tracked action is shown in the activity stream with a few exceptions: accesses are shown as aggregate data and add actions are shown only to the actor when the space is the object.
- Activities inside a private space are shown only to the members of this space.
- Activities involving a hidden item are shown only to the users who can see it.

Based on the ActivityStreams API, available in Shindig, we have developed widgets and a portable dashboard for basic learning analytics (Vozniuk et al., 2013), shown in Figure 29.

8.4 OpenSocial 2.5

At the start of the Go-Lab project, Graasp used Apache Shindig 2.0 to render OpenSocial gadgets. Apache Shindig 2.0 supports version 1.1 of the OpenSo-

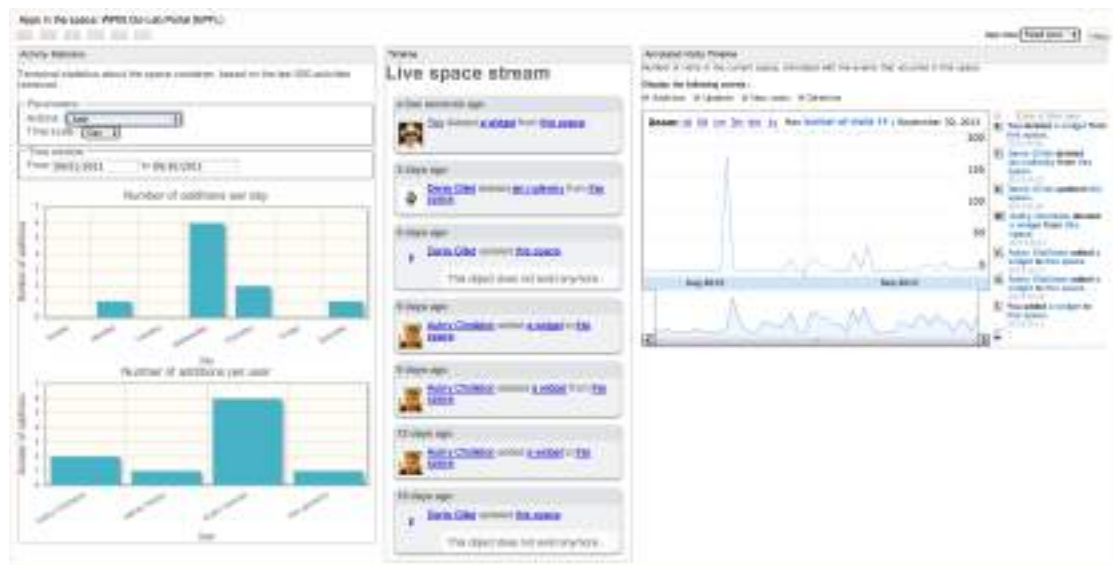


Figure 29: A portable learning analytics dashboard based on ActivityStreams API.

cial specification.

Since Graasp relies heavily on the usage of spaces, the Space concept was proposed to OpenSocial¹⁵. Currently, the Space extension is not in the OpenSocial specification 2.5, however, it is on the roadmap of OpenSocial 3.0¹⁶.

To be able to use the OpenSocial space APIs in Graasp, we extended Apache Shindig 2.0 to support spaces.

Since the ActivityStreams APIs were not supported in OpenSocial 1.1, we decided to upgrade the Apache Shindig to the newest version 2.5. This version implements OpenSocial 2.5. The major changes¹⁷ from OpenSocial 1.1 to OpenSocial 2.5 are listed below:

- Activity Streams support
- Deprecated support for ATOM
- Simplified gadget format
- Embedded Experiences
- OAuth 2 support
- Common Container
- Person Service - update APIs

¹⁵The Space concept proposal, <https://opensocial.atlassian.net/wiki/display/OSD/Space+Proposal>

¹⁶OpenSocial 3.0 roadmap, <https://opensocial.atlassian.net/wiki/display/OSD/Spec+Changes+-+v3.0>

¹⁷Release notes of OpenSocial 2.0 and OpenSocial 2.5: <http://blog.opensocial.org/2011/08/announcing-release-of-opensocial-20.html>, <https://opensocial.atlassian.net/wiki/display/OSD/Spec+Changes+-+v2.5>.

- Additional group models: @followers/@following/@colleagues/@reports/@manager

To achieve this upgrade, we have forked the latest version of Apache Shindig (namely 2.5 beta 6 at the time). Afterwards, we have migrated into this fork the OpenSocial space extension patch and the database mapping that maps OpenSocial database tables on to Graasp's database. In total, there were more than 20000 lines of code migrated. We tested and fixed many issues that appeared during the migration process and now we have a stable version that implements the same functionality that was available previously in Apache Shindig 2.0.

8.5 User Management

8.5.1 Login with Facebook & Google+

To simplify the sign-up procedure for users, we have implemented a social login service that allows users to sign into Graasp using their existing accounts on popular social networking services. Currently, Graasp supports signing up with Facebook and Google+ accounts in addition to the regular email-based sign-up. Other than speeding up the sign-up procedure, the profile and social graph information obtained from the identity providers (Facebook or Google+) could be used later to provide personalised content for users.

Figure 30 shows how the signup process with a social network login works. After choosing to log in with Facebook or Google+, we ask the user's permission to access her profile information. Once the user grants the permission, she is redirected to the Facebook or the Google+ login page (if she is not already logged in to Facebook or Google+). Using her login information of Facebook or Google+, the user can directly sign into Graasp.

To implement the social network login service, we make use of the OAuth standard for secure authorization. After the user grants Graasp the permission to access her profile information, a session token is returned from Facebook or Google+. Using the session token, Graasp makes API calls to Facebook or Google+ on behalf of the user, and retrieves the profile information of the user including email, name, picture, and so on. If the email already exists in Graasp, the user automatically signs into Graasp with the existing account. Otherwise, a new Graasp account is created for the user.

8.5.2 Anonymous login

As mentioned, Graasp provides an email-based or social network-based login scheme, where the user logs in using her email and password or social network credentials. To comply with the preliminary participatory design recommendation of easing the login procedure for students, we have implemented a new mechanism that allows students to access Graasp spaces without providing credentials.

Such an anonymous login could be applied in the following use case. A teacher creates an inquiry learning space for her course and populates the space with all the apps needed for the inquiry-based learning activities, such as hypothesis



Figure 30: Facebook and Google+ log in process

tools, simulators, and so on. Then the teacher shares the space through a secret URL with her students, as shown in Figure 31. Once a student loads the url, she is asked to enter her name to identify herself. A nickname can be provided to preserve anonymity. After entering her name, the student can use the apps in the space and her actions are saved in Graasp. Figure 32 illustrates an inquiry learning space for an experiment on the collision of galaxies where students can observe galaxies through telescopes, conduct simulations, and upload their reports into the space.

From a technical point of view, when the teacher shares the space, an encrypted url is generated using a hash value that contains information about the creation time, the corresponding space, and the space creator. When the url is loaded, a metawidget that consists of all the apps in the space is rendered. As the teacher might need to monitor students' interaction with the apps, each student can be identified by the name entered by the student. Students' names are saved via cookies so that they do not need to enter their names every time they load the page. Furthermore, students' actions are saved through AppData for further analytics. We are now working a scheme to identify contributions from students using their nickname for retrieval in other sessions or as information for the teacher.

8.6 Other new functionality

8.6.1 Google Docs integration

Graasp allows users to integrate *Google Docs* as a resource into a space. When a user creates a new resource, she can select 'Google Doc' as *resource type* and then specify the URL of the Google Doc or pick the appropriate Google Doc from your Google Drive by clicking the upload button next to the *url* field as shown in Figure 33.

The Google Doc then appears embedded in Graasp as shown in Figure 34. Note that this integration is not limited to text documents, but includes all types

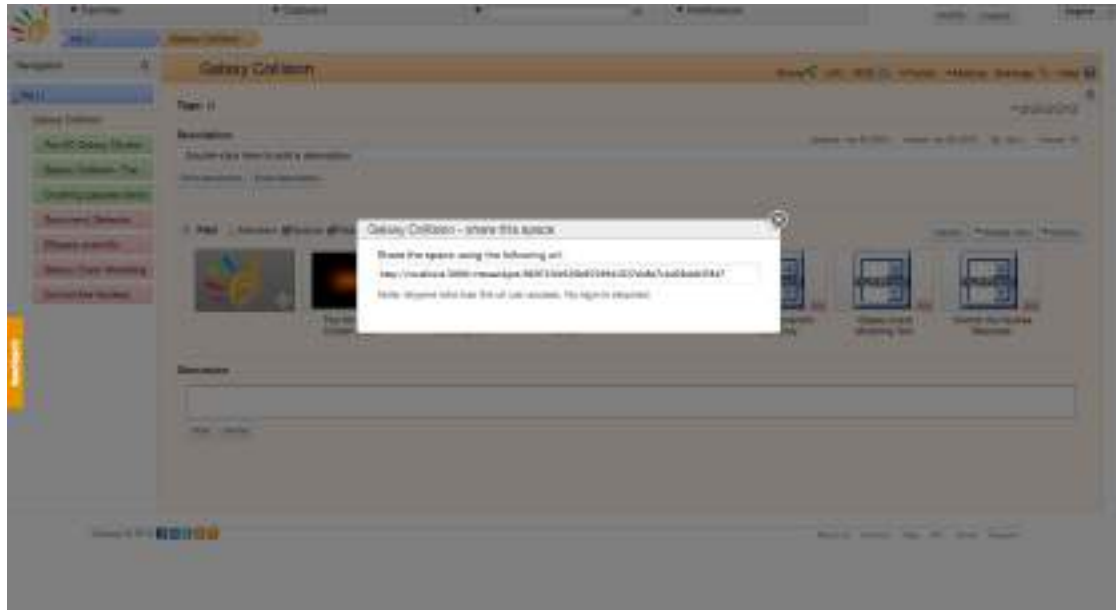


Figure 31: Sharing a space through an encrypted url in Graasp.



Figure 32: A learning space loaded as a metawidget.

of Google Docs, i.e., spreadsheets, drawings, presentations, etc.

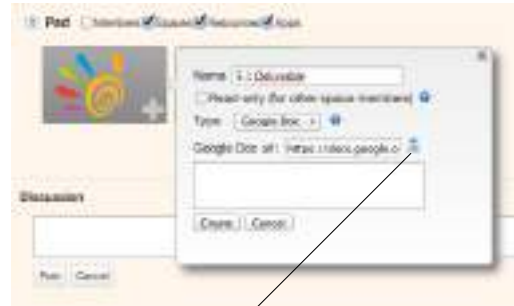
8.6.2 Mobile Graasp apps

In order to be widely accessible and run smoothly on mobile devices, we are planning to create a mobile-friendly version of Graasp accessible through a Web app. However, as the most efficient way to reach mobile app users is through centralized native app stores, such as Apple's AppStore and Google's Play Store, we have developed hybrid mobile apps for Android and iOS and are currently in a first testing phase. Hybrid apps are basically Web apps with a small native wrapper. In our case it can be described as native apps with basically only a Web view. Figure 35 shows the mobile version of Graasp running on iOS.

First, select Google Doc as Resource type



Then, Specify the Google Doc URL



...or pick Google Doc from your Google Drive



Figure 33: Creating a Google Doc resource in Graasp

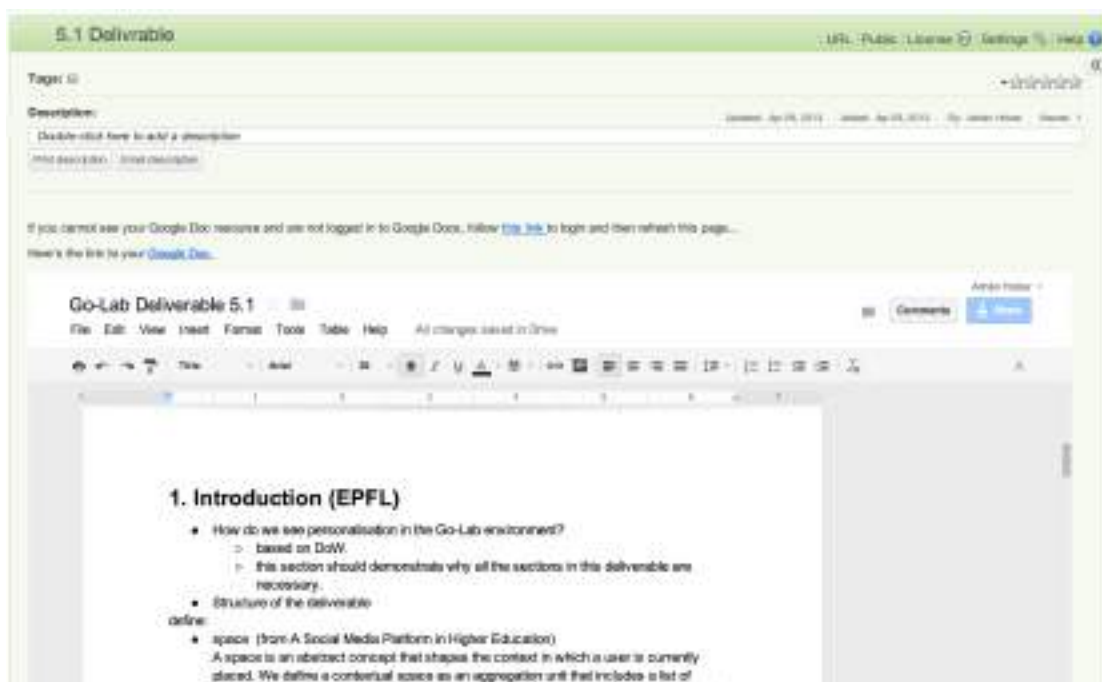


Figure 34: Viewing a Google Doc resource

8.6.3 File uploading app

In order to allow students to upload files such as reports into the inquiry learning space, a file uploading app has been developed¹⁸. The screenshot of the app

¹⁸The file uploading app is available at http://graasp.epfl.ch/gadget/1a/file_drop.xml

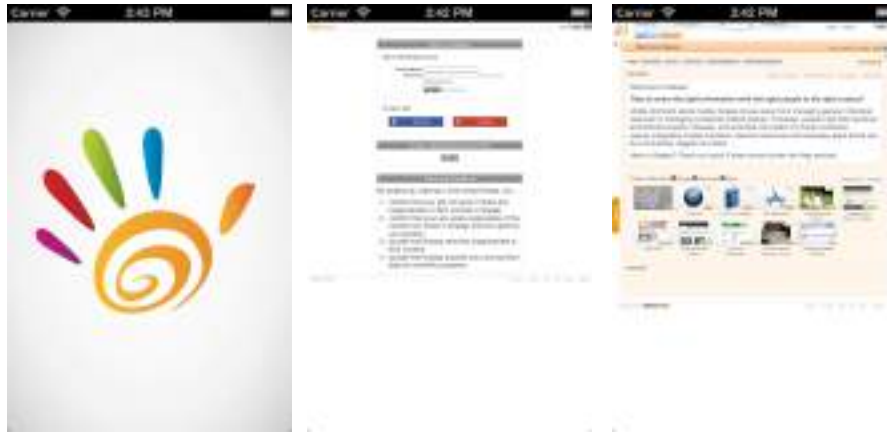


Figure 35: Screenshots of the Graasp mobile app on iOS

is shown in Figure 36. The app allows students to drag and drop files from Desktop and upload them into the space where the app is. The app also lists the existing files in the space, and provides a download link for each file.

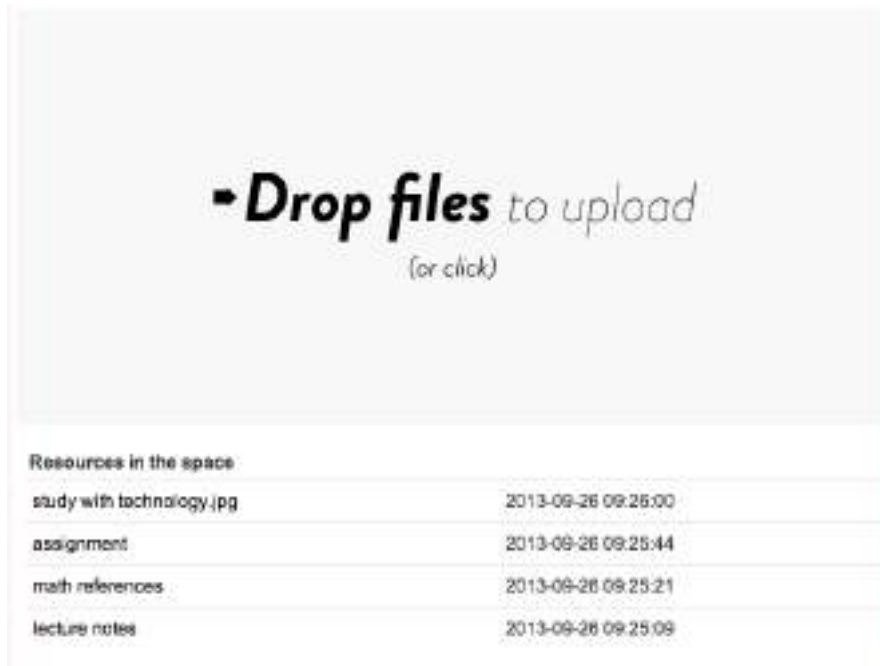


Figure 36: Screenshots of the file upload app

8.6.4 Security improvements

To enhance security of the Graasp platform we have enabled SSL encryption of the communication channel between the user's machine and the platform. We also have established redirection of all HTTP requests to HTTPS to force usage of the encrypted channel. In this way all users' content is served in a secure way.



Figure 37: “Spaces shared with me” pad provides a quick access to spaces shared with them.

8.7 Usability improvements

We have gathered feedback from users concerning Graasp usability issues and addressed the most important ones. For instance, often users were not able to find a specific space that they are members of. Now we show all spaces that a user is a member of on a dedicated pad named “Spaces shared with me” in their profile page (see Figure 37), so it is easy to find a space.

Another issue was related to the visibility of the filter panel. As this panel was hidden by default, users were not able to find it and perform filtering. Now the panel is always visible in the space (see Figure 38) so it is easy to leave only necessary items on the pad.

8.7.1 Other improvements

In this section, we list an incomplete set of other improvements that have been achieved in Graasp. As mentioned in the introduction, over 250 issues have been closed on GitHub since the start of the project. The complete list can be found on GitHub at <https://github.com/react-epfl/graasp/issues?page=1&state=closed>.

- *Improvements in the relevancy ranking of search results:* Relevant items in the search results are ranked based on the total number of views they have received.
- *Portal and Inquiry Learning spaces:* Two new type of space have been implemented: (i) a portal space and (ii) an inquiry learning space. A portal



Figure 38: Always visible filter panel in Graasp.

space is a space that is not a part of user's space hierarchy but is a separate item. It is targeting a group of users and, as such, it is not explicitly linked to its creator. The inquiry learning space has been added to

- *Notifications changes:* Notifications were sent separately for various user actions in spaces that a user was part of (e.g., when a resource is added to a space). This resulted often in large quantities of emails sent to users, which resulted in complaints. Based on this feedback we have turned off email notifications for all events except comments.
- *Fixed back and forward buttons for alias URLs:* There are two types of URLs in Graasp: (i) ordinary URLs (e.g., http://graasp.epfl.ch/#item=space_2602) and (ii) alias URLs (e.g., <http://graasp.epfl.ch/#url=GoLab>). Alias URLs were introduced for usability reasons: they are much easier to remember. Graasp contained a bug in managing browser history when navigating from pages with ordinary URLs to pages with alias URLs and vice versa. Graasp, being a single-page web application, changes URL in the address bar by calling `unset` and `set` function of the Prototype Xensions library. Each of the calls puts an address to the browser's history, while in the case of Graasp, only the new address should be put. We have extended the library with a new function `unset_set`, which adds only the new address to the browser's history.
- *Upload file size increased to 100Mb:* Users are now allowed to upload files up to 100Mb to Graasp. The usability of the file upload has also improved: an error message will be shown when the user tries to upload larger files and more meaningful error messages are used when the file upload fails.
- *Permission changes:* Now a confirmation is required to join public spaces.
- *Improved identity check:* When an owner receives a request to accept a new member, she can see the member's email address to be able to perform a basic identity check.
- *Minor interface improvements:* The notifications panel has been removed due to its limited functionality, e.g., users were unable to mark the grow-

ing number of notifications as read. Additionally, loading the notifications panel containing many messages slowed down Graasp.

- *Search improvements:* We implemented the search query as a bag of words and improved the performance of search. For example, the queries “go,lab”, “gO LAB”, etc. should find appropriate results.
- *Social sharing:* Sharing of space URL via facebook, twitter and google+ has been added. The Open Graph protocol is supported.

References

- Bogdanov, E., Limpens, F., Li, N., El Helou, S., Salzmann, C., & Gillet, D. (2012). A social media platform in higher education. In *Global engineering education conference (educon), 2012 ieee* (pp. 1–8).
- Bohl, O., Scheuhase, J., Sengler, R., & Winand, U. (2002). The sharable content object reference model (scorm) - a critical review. In *Computers in education, 2002. proceedings. international conference on* (p. 950-951 vol.2).
- Christian Maier, M. N. (2010). Lab2go: A repository to locate online laboratories. *International Journal of Online Engineering (iJOE), Vol 6*(No. 1).
- Chudnovskyy, O., Nestler, T., Gaedke, M., Daniel, F., Fernández-Villamor, J. I., Chepegin, V., et al. (2012). End-user-oriented telco mashups: the omelette approach. In *Proceedings of the 21st international conference companion on world wide web* (pp. 235–238).
- Dahrendorf, D., Dikke, D., & Faltin, N. (2012). Sharing personal learning environments for widget based systems using a widget marketplace. In *Proceedings of the ple conference 2012 in aveiro, portugal, july 11-13, 2012. aveiro, portugal*.
- David Lowe, L. W. M. d. I. V., Steve Murray. (2009). Labshare: Towards a national approach to laboratory sharing.
- Garcia-Zubia, J., Ipina, D. Lopez-de, Orduna, P., & Hernandez-Jayo, U. (2006). Experience with weblab-deusto. In *Industrial electronics, 2006 ieee international symposium on* (Vol. 4, p. 3190-3195).
- Gillet, D., Jong, T. de, Sotirou, S., & Salzmann, C. (2013). Personalised Learning Spaces and Federated Online Labs for STEM Education at School: Supporting Teacher Communities and Inquiry Learning. In *Proceedings of the 4th IEEE Global Engineering Education Conference (EDUCON)* (pp. 769–773). IEEE.
- Govaerts, S., Cao, Y., Vozniuk, A., Holzer, A. C., Garbi Zutin, D., San Cristóbal Ruiz, E., et al. (2013). Towards an Online Lab Portal for Inquiry-based STEM Learning at School. In *The 12th International Conference on Web-based Learning (ICWL 2013)*. Springer.
- Govaerts, S., Verbert, K., Dahrendorf, D., Ullrich, C., Schmidt, M., Werkle, M., et al. (2011, September). Towards Responsive Open Learning Environments: the ROLE Interoperability framework. In C. Delgado Kloos, D. Gillet, R. M. Crespo Garcia, F. Wild, & M. Wolpers (Eds.), *Towards ubiquitous learning - proceedings of 6th european conference of technology enhanced learning, ec-tel 2011*, (pp. 125–138). Springer. Available

- from <https://lirias.kuleuven.be/handle/123456789/319048>
- Green, S., Pearson, E., Gkatzidou, V., & Perrin, F. O. (2012). A community-centred design approach for accessible rich internet applications (aria). In *Proceedings of the 26th annual bcs interaction specialist group conference on people and computers* (pp. 89–98).
- Harward, V., Alamo, J. del, Lerman, S., Bailey, P., Carpenter, J., DeLong, K., et al. (2008). The ilab shared architecture: A web services infrastructure to build communities of internet accessible laboratories. *Proceedings of the IEEE*, 96(6), 931-950.
- NISO Press. (2004). *Understanding Metadata*. National Information Standards Organization Press. Available from <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Understanding+Metadata#2>
- Richter, T., Boehringer, D., & Jeschke, S. (2011). Lila: A european project on networked experiments. In S. Jeschke, I. Isenhardt, & K. Henning (Eds.), *Automation, communication and cybernetics in science and engineering 2009/2010* (p. 307-317). Springer Berlin Heidelberg. Available from http://dx.doi.org/10.1007/978-3-642-16208-4_27
- T. Richter, P. G., & Zutin, D. (2012). *A standardized metadata set for annotation of virtual and remote laboratories*.
- Turnitsa, C. (2005). Extending the levels of conceptual interoperability model. IEEE CS Press.
- Vogel, O., Arnold, I., Chughtai, A., & Kehrer, T. (2011). *Software architecture - a comprehensive framework and guide for practitioners*. Springer.
- Vozniuk, A., Govaerts, S., & Gillet, D. (2013). Towards portable learning analytics dashboards. In *Proceedings of the 13th IEEE International Conference on Advanced Learning Technologies*.