EUROPEAN LANGUAGE GRID

D3.4 Platform GUI (final release)

Author:	Andis Lagzdiņš (Tilde), Uldis Siliņš (Tilde), Andrejs Vasiļjevs (Tilde), Aivars Bērziņš (Tilde), Indra Sāmīte (Tilde), Athanasia Kolovou (ILSP), Dimitris Gkoumas (ILSP), Penny Labropoulou (ILSP), Stelios Piperidis (ILSP)
Dissemination Level:	Public
Date:	28-02-2022

About this document

Project	ELG – European Language Grid
Grant agreement no.	825627 – Horizon 2020, ICT 2018-2020 – Innovation Action
Coordinator	Prof. Dr. Georg Rehm (DFKI)
Start date, duration	01-01-2019, 42 months (GA amendment version: AMD-825627-7)
Deliverable number	D3.4 Grid Platform Interactive Interface and Information System
Deliverable title	Platform GUI (final release)
Type	Report
Number of pages	62
Status and version	Final – Version 1.0
Dissemination level	Public
Date of delivery	Contractual: 28-02-2022 – Actual: 28-02-2022
WP number and title	WP3: Grid Platform – Interactive Interface and Information System
Task number and title	Task 3.3: Set up and maintenance of CMS for the ELG; Task 3.4: Development of GUIs for portal services; Task 3.6: Development of Catalogue GUI
Authors	Andis Lagzdiņš (Tilde), Uldis Siliņš (Tilde), Andrejs Vasiļjevs (Tilde), Aivars Bērziņš (Tilde), Indra Sāmīte (Tilde), Athanasia Kolovou (ILSP), Dimitris Gkoumas (ILSP), Penny Labropoulou (ILSP), Stelios Piperidis (ILSP)
Reviewers	Khalid Choukri (ELDA), Ulrich Germann (UEDIN)
Consortium	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Germany
	Institute for Language and Speech Processing (ILSP), Greece
	University of Sheffield (USFD), United Kingdom
	Charles University (CUNI), Czech Republic
	Evaluations and Language Resources Distribution Agency (ELDA), France
	Tilde SIA (TILDE), Latvia
	Hensoldt Analytics (HENS), Austria
	Expert System Iberia SL (EXPSYS), Spain
	University of Edinburgh (UEDIN), United Kingdom
EC project officers	Philippe Gelin, Miklos Druskoczi
For copies of reports and other ELG-related information, please contact:	DFKI GmbH European Language Grid (ELG) Alt-Moabit 91c D-10559 Berlin Germany Prof. Dr. Georg Rehm, DFKI GmbH georg.rehm@dfki.de Phone: +49 (0)30 23895-1833 Fax: +49 (0)30 23895-1810 http://european-language-grid.eu © 2022 ELG Consortium

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
CMS	Content Management System
CSS	Cascading Style Sheets
ELG	European Language Grid
GPL	GNU General Public License
GUI	Graphical User Interface
HTML	Hypertext Markup Language
JWT	JSON Web Token
LT	Language Technologies
MVP	Minimum Viable Product
POS	Part of Speech
REST	Representational State Transfer
SEO	Search Engine Optimization
SME	Small and Medium Size Enterprises
SPA	Single Page Application
SSR	Server-Side Rendering
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
W3C	World Wide Web Consortium
WCAG	Web Content Accessibility Guidelines
WYSIWYG	What You See Is What You Get

Abstract

This document provides a description of the ELG platform graphical user interfaces. We present an overview of the overall frontend architecture and provide a technical description and implementation details for the solutions designed for the central portal, the CMS, the catalogue UI, the implementation of UX and the authentication solution, and the billing module.

1 Introduction

This document describes the state of development of the ELG platform graphical user interface (GUI), as deployed in Release 3.0 of the ELG platform at https://live.european-language-grid.eu. Previous deliverables D3.2 and D3.3 have been taken as a base by incorporating new changes since release 2.0.

ELG targets a wide range of users with different requirements and expectations, broadly divided into the following groups (cf. Deliverables D2.1, D2.3 and D7.1 for more information):

- **content providers** and **developers and LT integrators** are providers and consumers of LT resources, as described above,
- **information providers** and **information seekers** are providers and consumers of LT-related (meta)-information,
- **citizens**, i.e., individuals who want to inform themselves about LT and ELG,
- **ELG technical users**, including (a) the **validators** who are responsible for supervising the publication of resources in the platform, and (b) platform and content **administrators**, i.e., the ELG technical team that administers and monitors the day-to-day operation and performance of the platform.

The Platform GUI and the backend that drives it comprises the following components:

- The *Central Portal* is the network of web pages that provide user role-specific journeys through the ELG's offerings. User roles include, among others, *Consumers* who use the ELG to search for information or use services, and *Providers*, who supply resources or services. A journey is a use-case specific network of paths that describes the user experience as they navigate through the Platform in order to achieve their objectives. Journey design is a core component of UX design. In particular, we added a Provider role and Journey to facilitate contributions to the ELG.
- The *Content Management System* (CMS) drives the dynamic generation of web pages that users of the portal interact with.
- The Catalogue UI allows users to interact with the ELG Catalogue of resources, services, technologies, etc.
- The *Authentication Mechanism* authenticates users and grants or denies access rights to the platform depending on the user's role within the platform.
- *Try-out GUIs* have been developed for selected service types.

An external online billing platform Chargebee[1] has been chosen to integrate with the ELG platform to provide a billing prototype.

2 Key Developments since the Interim Release

During the last year we have been stabilizing the existing functionality, validated discoverability against search engines and analysed performance with online tools. We have also worked on the following new functionalities:

- The Development infrastructure of the Central Portal and Drupal[2] has been moved to ELG Gitlab.
- The Central portal has a new layout and visual identity.
- The CMS has been adjusted to provide missing functionality for static content migration from the project's website.
- The Keycloak[3] authentication system has been improved and stabilized.
- We discussed the requirements for the Billing module, performed market research about existing billing platforms, selected the most suitable platform, and configured and set up a prototype for demonstrating the billing scenario. Thus, the Billing module is ready to launch the commercialization strategy.

3 Front-end Architecture Overview

The GUI (or frontend) communicates with end users and clients directly, so it is crucial to provide the best possible solution not only from the usability and information architecture point of view, but also from a technological perspective. As both search engines and information sharing in social portals are essential for the European Language Grid to attract more attention and visitors, the solution must provide technical support for them.

On the other hand, there are different teams from the consortium bringing different software and product backgrounds. The GUIs are developed by fulfilling various tasks and timelines, so this challenge must be taken into consideration when providing a good balance between the development process and the result.

With the purpose of creating a single integrated GUI platform for end-users, search engines and social platform agents (content scrappers), a dedicated architecture has been created and implemented that provides good user experience and that is suitable for different development teams and processes.

[1] https://www.chargebee.com
[2] https://www.drupal.org
[3] https://www.keycloak.org

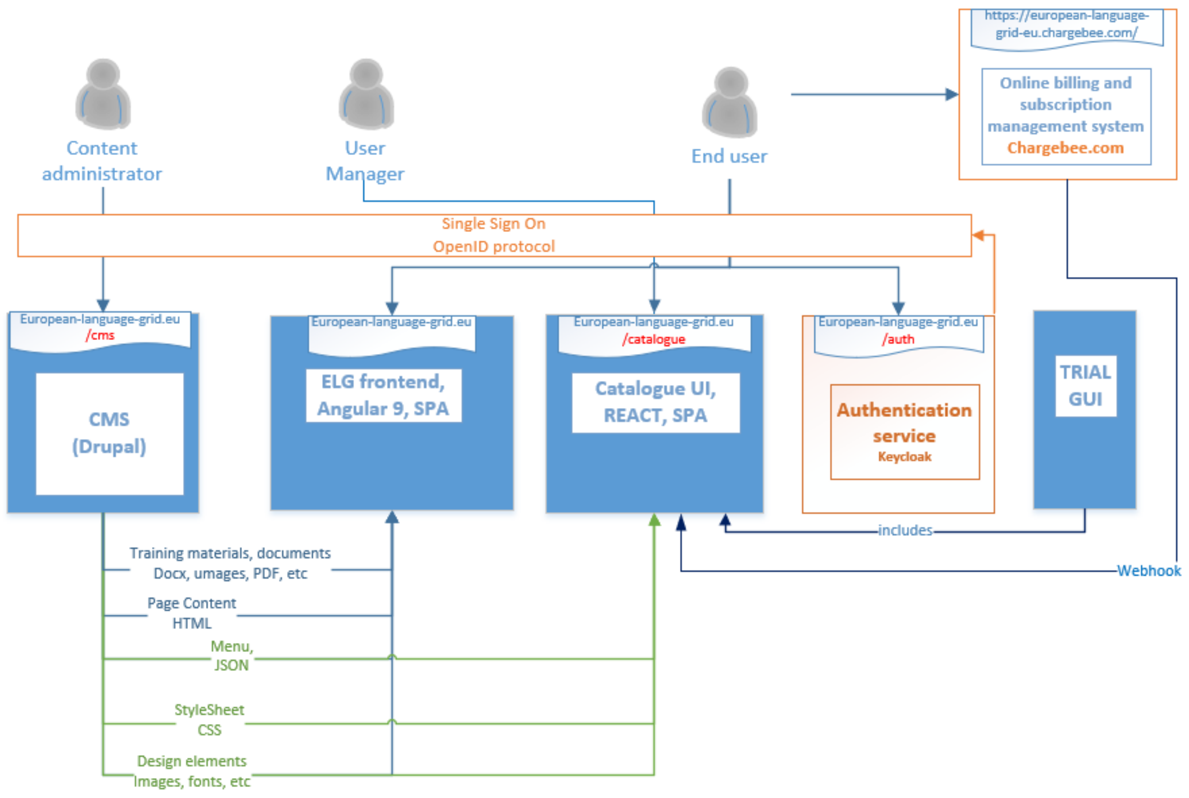


Figure 1: GUI architecture overview

The frontend architecture consists of the following independent software solutions (see Figure 1):

- ELG frontend/Central Portal – central entry point for ELG.
- Content Management System (CMS) – provides back-office functionality for static content management.
- Catalogue UI – provides GUI for searching, opening LT services, tools, data, projects, and other dynamic content.
- Authentication service – an authentication service provider that is shared across all GUIs and backend services.
- Trial GUIs – multiple small graphic user interfaces that are dedicated to letting the end-users to try-out specific types of ELG services.
- Integration with Chargebee, an external online billing solution.

Both Drupal (a content management system) and Keycloak (an authentication solution) are cross-functional providers for different GUI solutions. The content management system shares the design and content information that is reused in different GUI parts, while the authentication solution provides centralized user management and a single sign-on capability to ensure smooth switching between different GUI parts of the ELG platform.

The previous figure shows that the CMS is used to share different assets for all GUIs:

- User training materials – this section is mainly dedicated to the central ELG frontend, a user with appropriate rights is able to upload and manage training materials and documents in various formats, e.g., docx, images, PDFs. These materials will be published by the central portal, but if needed, can be also published using the Catalogue UI.

- Page content as HTML – these html documents or pages will also be published in the central ELG portal, e.g., content for the first page, about, contacts etc.
- Menus – different menus are stored and managed centrally, every ELG platform component can access them and download in JSON format using REST API
- Stylesheets – CMS is used as central point for storing the ELG theme designs that are used in various parts of the GUI, also by the Language Resource (LR) service specific try-out GUIs. These stylesheets can be downloaded as CSS files.
- Design elements – the CMS is also used to centrally store design and page elements, such as images, fonts, logos.

4 Central Portal

4.1 Technical Description

The central entry point of the European Language Grid is served by a single page application (SPA), which is built using Angular framework. This software retrieves content from the Content Management System (CMS), redirects the user to the Catalogue, and allows the user to login.

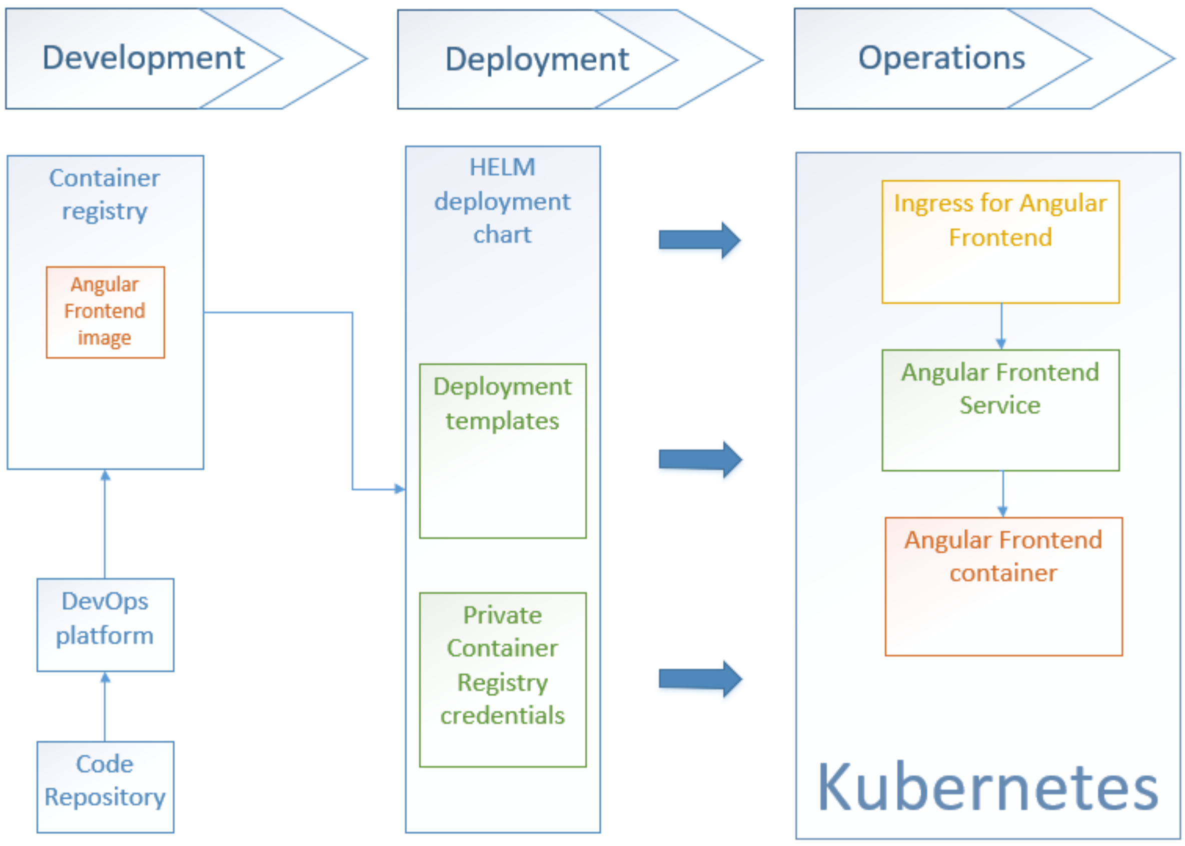


Figure 2: Angular application life cycle

Figure 2 above describes how the Angular application development process is organised. Angular frontend developers upload their code repository to the cloud platform Gitlab. After that, a GitLab Pipeline is automatically triggered that pulls the source code for the Angular application, adds configuration values, builds, and tags Docker images, and finally pushes newly created images to the private Gitlab Container Registry. Each Docker image version is tagged automatically with an auto-increasing build number and with a static tag to control release versions manually.

One of the challenges of modern single page application (SPA) that are built using JavaScript technologies is accessibility to search engine crawlers – tools that read information from the application and put it into search engines. Search engine crawlers can only access static HTML files. They do not run on the browser, and thus cannot access web pages which are generated dynamically during execution in the browser only when this is accessed. In these cases, most of the important information from a page is not read. However Angular can handle this issue. There is a way to develop and deploy the Angular application so that it can provide whole html files to search engine crawlers. This approach is called Angular Universal and an application built this way runs on the browser and renders the application on the server as well. This technology represents a good approach to meet all search engine optimisation requirements as well as usability best practices. It works as follows: whenever a user or a web crawler requests a web page, instead of doing it the usual way – answering user requests with html and all the JavaScript files – an intermediate step is introduced that renders the answer for the user and sends whole html files as an answer to the user. As soon as this html is delivered to the user's browser, the user can see the result immediately, and when the JavaScript is delivered, the application becomes interactive.

There is a need for a uniform web design implementation across the whole portal. Recreating every design element from scratch would be unnecessary as there are design systems available that operate with pre-created design element libraries that can be adjusted to the needs of the project. The Google Material[4] design system was chosen as a starter theme for ELG and it has been useful for different parts of the project. As the application grows and is put together with multiple tools and project parts, there is a need for a uniform design as well. Sharing Material design with CMS, service GUIs and the catalogue has helped to ensure a seamless look on the application's visual parts (see Figure 3 and Figure 4).

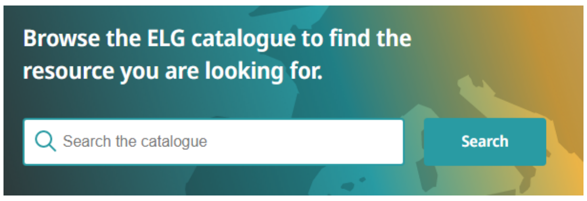

Browse the ELG catalogue to find the resource you are looking for.
Search the catalogue
Search

Figure 3: Same style of the different solutions – front page search bar

[4] https://material.io

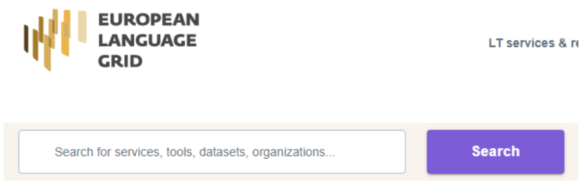


Figure 4: Same style of the different solutions – catalogue search bar

4.2 Functionality and Web Interfaces

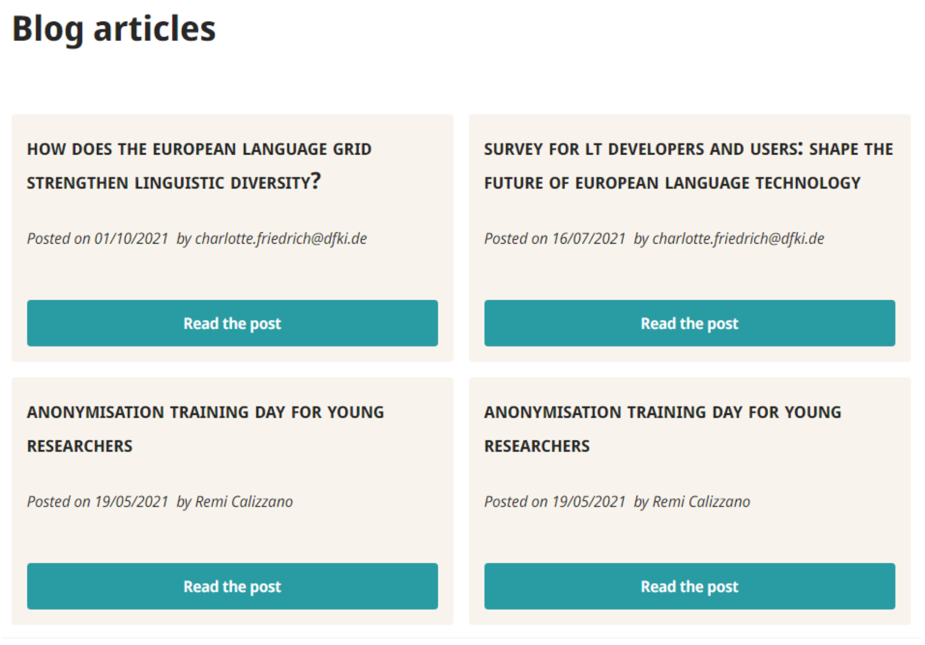
The landing page is built as the main entry point where the user that is visiting the portal for the first time can have a full idea of what kind of resources and services one can find. Just after the search bar that leads to the catalogue is a statistics section that is dynamically updated with fresh data on how many corpora, tools/services, conceptual resources, models and grammar, organizations and projects can be found in the catalogue. Right after the statistics is a section with different widgets that display featured, most popular and most recent catalogue resources.

The main menu is the way to quickly navigate to the most important parts of the portal.

"LT services & resources" and "LT community" points to the catalogue collection of resources.

"About" is a collection of various CMS articles about portal and community. Information on these pages contain the same information as pages of the initial project website (currently available at https://www.european-language-grid.eu) to provide a smooth transition that goes unnoticed for the users.

There is a blog-style news page (see Figure 5) that is used for collecting recent news posts and for keeping visitors up to date about the European Language Grid.


Blog articles

HOW DOES THE EUROPEAN LANGUAGE GRID STRENGTHEN LINGUISTIC DIVERSITY?

Posted on 01/10/2021 by charlotte.friedrich@dfki.de

Read the post

SURVEY FOR LT DEVELOPERS AND USERS: SHAPE THE FUTURE OF EUROPEAN LANGUAGE TECHNOLOGY

Posted on 16/07/2021 by charlotte.friedrich@dfki.de

Read the post

ANONYMISATION TRAINING DAY FOR YOUNG RESEARCHERS

Posted on 19/05/2021 by Remi Calizzano

Read the post

ANONYMISATION TRAINING DAY FOR YOUNG RESEARCHERS

Posted on 19/05/2021 by Remi Calizzano

Read the post

Figure 5: News page articles

"Documentation" is an extensive collection of documentation on how to use the portal and how to contribute to ELG as well as integrated all training materials.

As some of the menu links lead to a broad collection of documents, there a secondary menu is available that helps make navigation easier and the information structured. As you can see in Figure 6 there is a way to add a third level menu ensuring that it will always be possible to add new links and the menu won't run out of space.

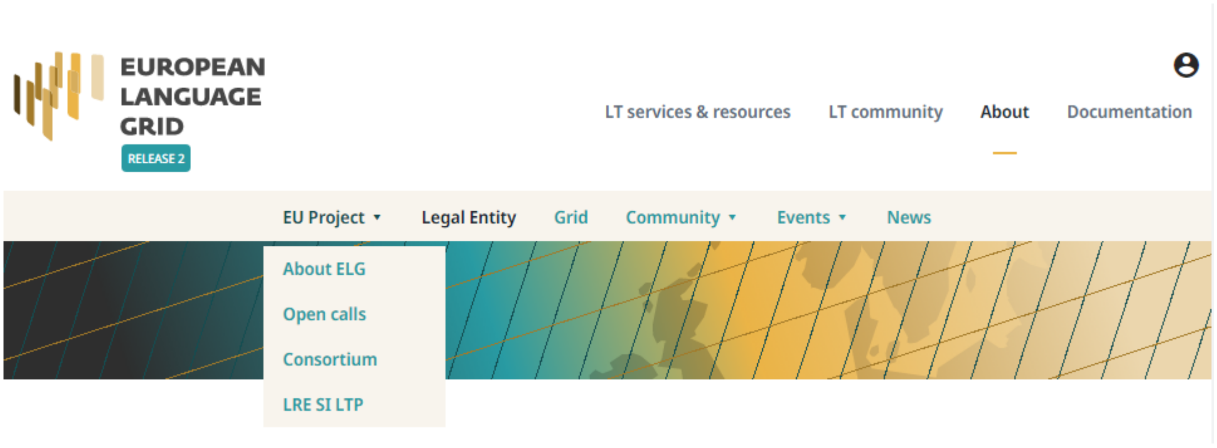


Figure 6: Portal menu and submenu

When a user reaches the main page, the single sign-on mechanism checks whether the user has been signed in before and if they have – it signs them on. Some pages are restricted to unauthorized users (e.g., upload of the resource) and some are not (such as catalogue search results). Depending on whether only authorized users can see the content, and following the initial checking, an unauthorized user may be sent to the login screen.

The frontend portal is built according to all search engine optimization guidelines[5] so that all necessary prerequisites are met for the portal to be highly ranked in various search engine results.

5 Content Management System

5.1 Technical Description

Drupal was chosen for static content management, as it is one of the leading and one of the most popular Content Management Systems worldwide and it provides good support for functionality, management, and security. The most crucial argument for Drupal and not WordPress that is a market share leader among content management systems was security issues and Drupal is considered much secure than WordPress. Drupal acts as a ‘headless’ solution, which means that it is mainly used as a back-office application and portal users do not see anything directly from the CMS. For external access, the Drupal server generates menus and html content pages, and it allows for configuration using the REST API.

[5] Search Engine Optimization (SEO) Starter Guide – https://developers.google.com/search/docs/beginner/seo-starter-guide

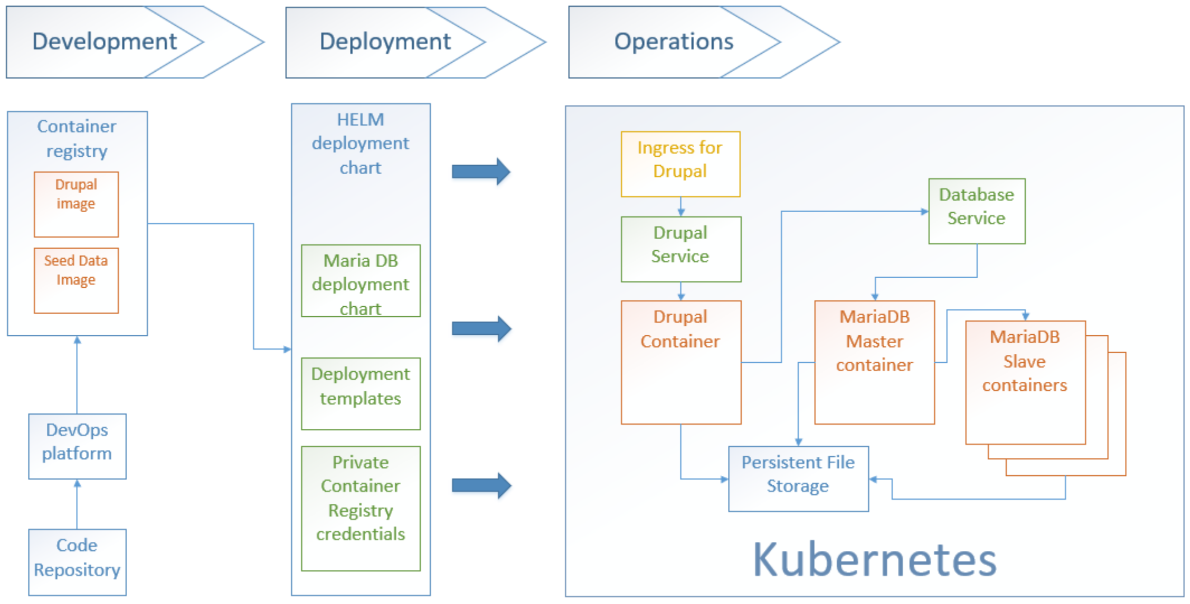


Figure 7: CMS life cycle

Figure 7 above describes how the CMS development process works. This process is very similar to the Angular application development – developers upload their code and seed data to the code repository where automatic pipelines push Docker images to the private Azure Container Registry. For the CMS there are two parts and two Docker images – one is for the CMS instance and another – for seed data. The seed data container holds:

- initial data for the CMS database, e.g., site configuration, html content.
- files for the CMS website, e.g., logos, favicon, images, pdf, other downloadable files.

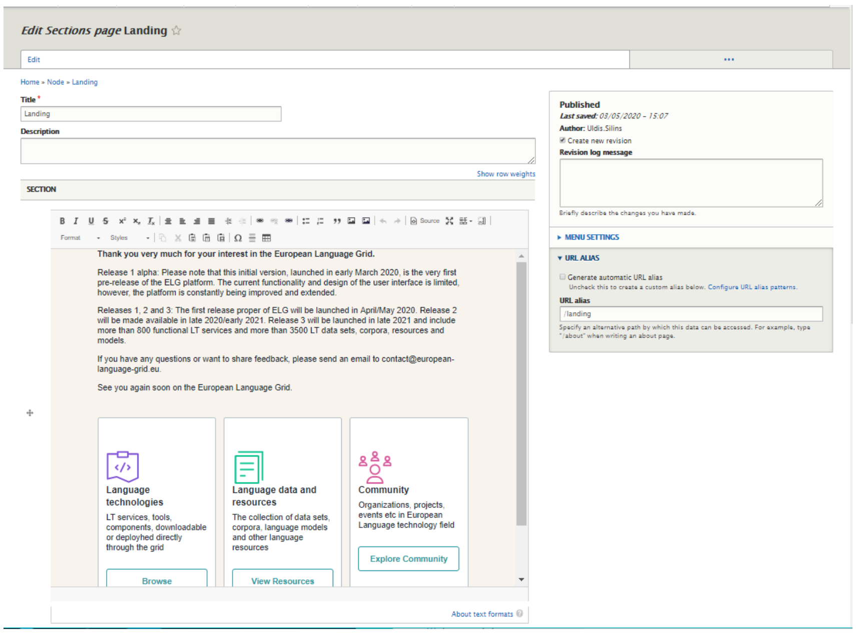
The ELG platform deployment pulls those Docker images by providing image pull secrets.

For our benefit the Drupal instance needed to be extended with several modules:

- CKEditor: This module lets Drupal replace text area fields with a visual HTML editor, usually called a WYSIWYG editor. This HTML text editor brings many of the powerful WYSIWYG editing functions of known desktop editors like Word to the web.
- Menu Item Extras: This module lets extra configuration fields for menu items to be added.
- Rest menu Items: This module allows accessing menu items via the Rest protocol.
- Keycloak integration: Integration of our Single Sign On solution.
- Webforms and webforms REST module: This module creates web forms as well as their obtaining and posting via the REST protocol.
- Page serializer. This module allows to include paging information in news HTTP request.

5.2 Functionality and Web Interfaces

There are multiple content types served by the CMS to the portal. The most basic one is a simple page, for example the ELG landing page (see Figure 8). Its content is prepared in the CMS and transferred to the portal in the form of JSON when requested. To be able to create content that looks the same in the CMS and in the ELG portal, a portion of the style configuration should be shared between them. A Google Material stylesheet file that is exported from the ELG portal and imported into the CMS ensures a common ground between these two components.



Edit Sections page **Landing** ☆

Edit | •••

Home » Node » Landing

Title *

Landing

Description

Show row weights

SECTION

Format · Styles · Source

Thank you very much for your interest in the European Language Grid.

Release 1 alpha: Please note that this initial version, launched in early March 2020, is the very first pre-release of the ELG platform. The current functionality and design of the user interface is limited, however, the platform is constantly being improved and extended.

Releases 1, 2 and 3: The first release proper of ELG will be launched in April/May 2020. Release 2 will be made available in late 2020/early 2021. Release 3 will be launched in late 2021 and include more than 800 functional LT services and more than 3500 LT data sets, corpora, resources and models.

If you have any questions or want to share feedback, please send an email to contact@european-language-grid.eu.

See you again soon on the European Language Grid.

Language technologies
LT services, tools, components, downloadable or deployhed directly through the grid
Browse

Language data and resources
The collection of data sets, corpora, language models and other language resources
View Resources

Community
Organizations, projects, events etc in European Language technology field
Explore Community

About text formats

Published
Last saved: *03/05/2020 - 15:07*
Author: Uldis.Silins
☑ Create new revision
Revision log message
Briefly describe the changes you have made.

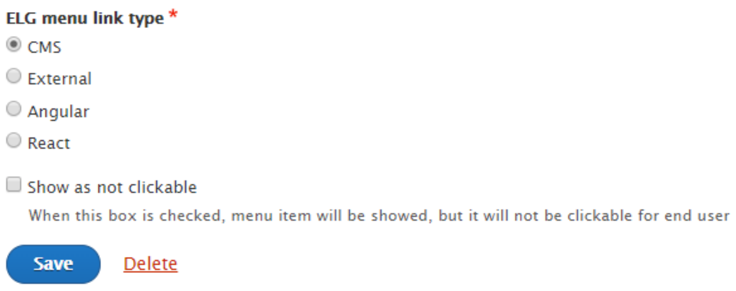
▸ MENU SETTINGS

▾ URL ALIAS
☐ Generate automatic URL alias
Uncheck this to create a custom alias below. Configure URL alias patterns.
URL alias
/landing
Specify an alternative path by which this data can be accessed. For example, type "/about" when writing an about page.

Figure 8: Landing page in the CMS

There are three menus that are created in the CMS and displayed in the ELG portal. These are one main in the header menu and two secondary menus in the footer – one on the left side, that mimics header menu and one on the right side. There is one link (Contact us) on the right-side footer menu at the moment.

Each menu item can be configured individually by indicating target links of the menu item and various other options (see Figure 9).



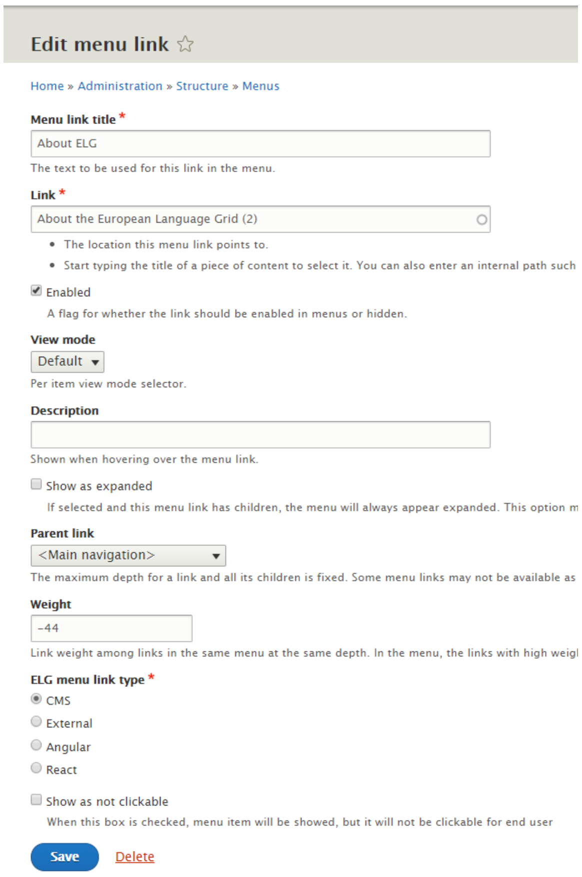
ELG menu link type *
[✓ CMS]
External
Angular
React

☐ Show as not clickable
When this box is checked, menu item will be showed, but it will not be clickable for end user

Save Delete

Figure 9: Menu link editing

There are additional parameters added to the menu item configuration, such as the ELG menu link type (see Figure 10). This is added via the Drupal module *Menu items extras* and it allows to configure the type of link and prepend the URL depending on this type. When the menu info is received in the frontend, there are parameters that mark the type of link (see Figure 10).



Edit menu link ☆

Home » Administration » Structure » Menus

Menu link title *

About ELG

The text to be used for this link in the menu.

Link *

About the European Language Grid (2)

- The location this menu link points to.
- Start typing the title of a piece of content to select it. You can also enter an internal path such

[✓] Enabled

A flag for whether the link should be enabled in menus or hidden.

View mode

Default

Per item view mode selector.

Description

Shown when hovering over the menu link.

[ ] Show as expanded

If selected and this menu link has children, the menu will always appear expanded. This option m

Parent link

<Main navigation>

The maximum depth for a link and all its children is fixed. Some menu links may not be available as

Weight

-44

Link weight among links in the same menu at the same depth. In the menu, the links with high weig

ELG menu link type *

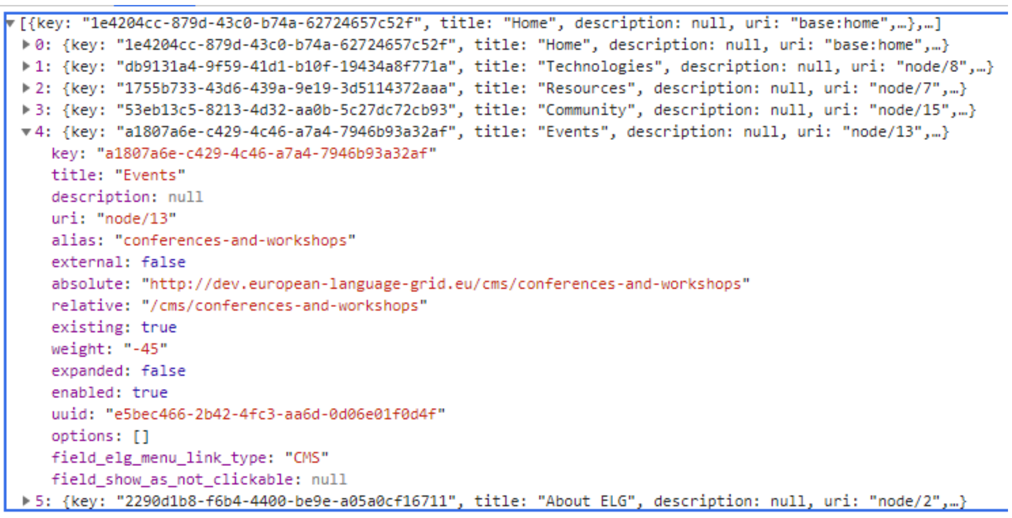
[✓ CMS]
External
Angular
React

[ ] Show as not clickable

When this box is checked, menu item will be showed, but it will not be clickable for end user

Figure 10: Adding menu item

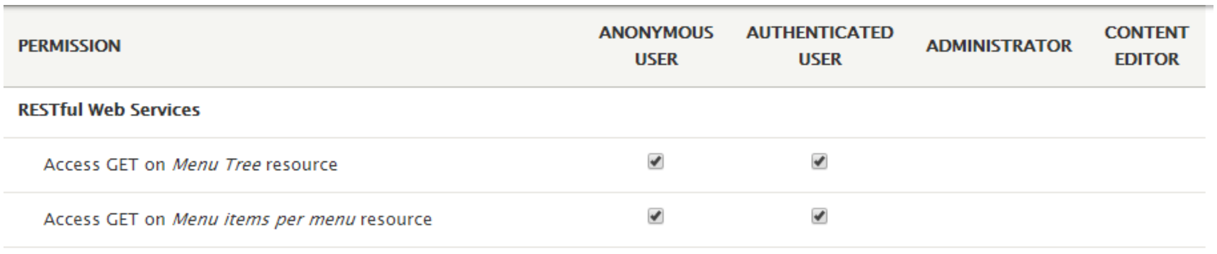
For example, CMS links are prepended with “/page/” and React links are prepended with “/catalogue/”.



```
▼[{key: "1e4204cc-879d-43c0-b74a-62724657c52f", title: "Home", description: null, uri: "base:home",…},…]
 ▶0: {key: "1e4204cc-879d-43c0-b74a-62724657c52f", title: "Home", description: null, uri: "base:home",…}
 ▶1: {key: "db9131a4-9f59-41d1-b10f-19434a8f771a", title: "Technologies", description: null, uri: "node/8",…}
 ▶2: {key: "1755b733-43d6-439a-9e19-3d5114372aaa", title: "Resources", description: null, uri: "node/7",…}
 ▶3: {key: "53eb13c5-8213-4d32-aa0b-5c27dc72cb93", title: "Community", description: null, uri: "node/15",…}
 ▼4: {key: "a1807a6e-c429-4c46-a7a4-7946b93a32af", title: "Events", description: null, uri: "node/13",…}
    key: "a1807a6e-c429-4c46-a7a4-7946b93a32af"
    title: "Events"
    description: null
    uri: "node/13"
    alias: "conferences-and-workshops"
    external: false
    absolute: "http://dev.european-language-grid.eu/cms/conferences-and-workshops"
    relative: "/cms/conferences-and-workshops"
    existing: true
    weight: "-45"
    expanded: false
    enabled: true
    uuid: "e5bec466-2b42-4fc3-aa6d-0d06e01f0d4f"
    options: []
    field_elg_menu_link_type: "CMS"
    field_show_as_not_clickable: null
 ▶5: {key: "2290d1b8-f6b4-4400-be9e-a05a0cf16711", title: "About ELG", description: null, uri: "node/2",…}
```

Figure 11: Menu items received as JSON object through REST request

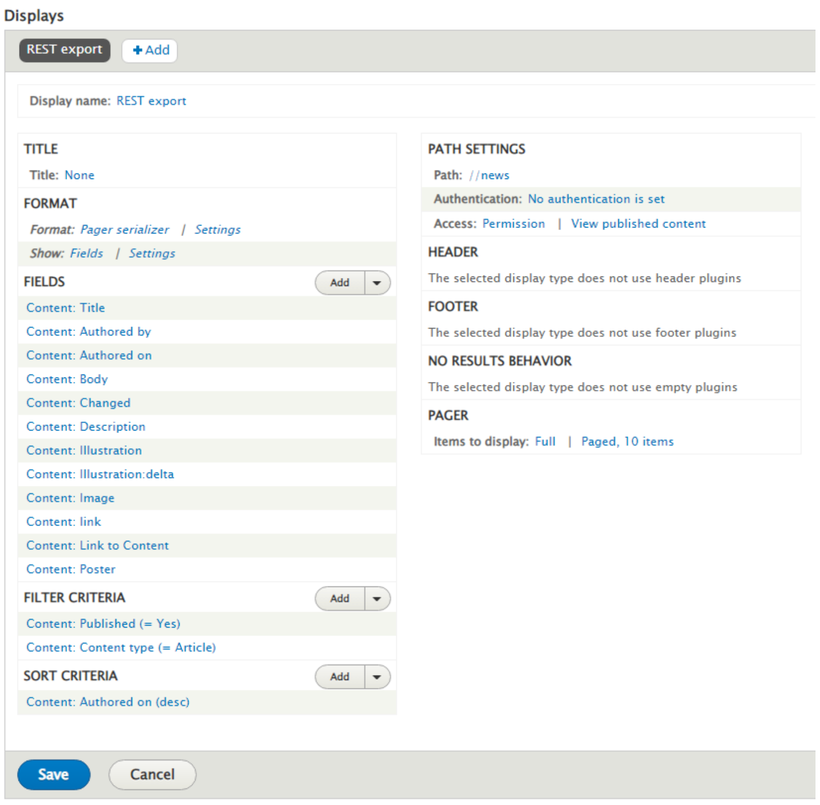
Access to various CMS actions is granted through a permission mechanism which can be assigned to different roles (see Figure 12). For example, access to menu items is not restricted to logged-in users, so every page visitor can see the menu.



PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR	CONTENT EDITOR
RESTful Web Services				
Access GET on *Menu Tree* resource	✓	✓		
Access GET on *Menu items per menu* resource	✓	✓		

Figure 12: Menu permission mechanism

There is a special view for the news page. As it is seen in Figure 13 it gathers all the necessary information about news posts and serves it in an HTTP request.



Displays

REST export | + Add

Display name: REST export

TITLE
Title: None

FORMAT
Format: *Pager serializer* | *Settings*
Show: *Fields* | *Settings*

FIELDS
Content: Title
Content: Authored by
Content: Authored on
Content: Body
Content: Changed
Content: Description
Content: Illustration
Content: Illustration:delta
Content: Image
Content: link
Content: Link to Content
Content: Poster

FILTER CRITERIA
Content: Published (= Yes)
Content: Content type (= Article)

SORT CRITERIA
Content: Authored on (desc)

PATH SETTINGS
Path: /news
Authentication: No authentication is set
Access: Permission | View published content

HEADER
The selected display type does not use header plugins

FOOTER
The selected display type does not use footer plugins

NO RESULTS BEHAVIOR
The selected display type does not use empty plugins

PAGER
Items to display: Full | Paged, 10 items

Figure 13: News page view

6 Catalogue UI

6.1 Technical description

For the implementation of the catalogue UI, we moved away from the traditional monolithic architecture which dictates that backend and frontend are tightly connected. Instead, we follow the emerging micro-frontend architecture (https://martinfowler.com/articles/micro-frontends.html) which separates the two. This makes each codebase simpler and easier to maintain. Also, by decoupling the two concepts, the backend and frontend developers' team can take independent decisions, which makes the development process faster and more versatile. For the development of the catalogue UI we use React (https://reactjs.org), a JavaScript library for user interfaces.

The catalogue UI is implemented as a Single Page Application (SPA). In this way, the end-user experiences instant page updates instead of full page reloads. This is made possible due to AJAX requests, the Virtual Document Object Model (VDOM) and the Reconciliation heuristic algorithm O(n) that the library uses to update the DOM of the page. In order to deliver a robust and easy to maintain application, we exploit mature and widely adopted technologies and libraries that are well documented and community supported.

The navigation from one page to the other is made with the concept of client-side routing. In the traditional server-side routing approach, every time a user navigates to a new page the entire webpage is downloaded from the server. In contrast to this, with client-side routing only the required data are downloaded, being processed and displayed to the user, thus speeding up the process and providing seamless user experience. In addition, the URL that is displayed in the browser is changed manually by the routing library. For client-side routing, we use the react-router-dom (https://reacttraining.com/react-router/) library. For the authentication and authorization part we use Keycloak’s JavaScript library (https://www.npmjs.com/package/keycloak-js). Following the registration of our application to Keycloak, we are able to access user information (username, e-mail) and acquire authentication tokens. This information feeds into policy-related actions; thus, we can assess if a certain user is allowed to view restricted pages and if she should be granted access to specific functionalities, like the try out GUIs, for example.

One of our primary concerns is to develop and deliver a catalogue that provides a pleasant experience to the end-users independently of the browser or device they are using. In order to cover a wide range of browsers, we transform the code into ECMAScript 5 code that is supported by many browser versions. We specifically target a broad range of browsers based on global usage of production builds.

To accomplish these functionalities, we use the official create-react-app frontend build pipeline (https://github.com/facebook/create-react-app), which, among others, utilizes webpack (https://github.com/webpack/webpack) for the bundling process and Babel (https://github.com/babel/babel) for the transpilation stage. By using this pipeline tool, we also get an optimized production build which is produced each time the GitLab Continuous Integration, Continuous Delivery, and Continuous Deployment (CI/CD) is running. GitLab CI/CD is an iterative process that helps developers to build, test and deploy their applications every time a code change is made. In our case, each time we commit a change, a new build is made that results to a new Docker image that we deploy to Kubernetes. This Docker image contains an Apache server which serves our production-ready application.

This production-ready application in the beginning was one large bundle file. All the code was merged into one single file and this file was served to the clients. Whilst this is a great and acceptable approach for small applications, it is by no means ideal for larger applications. As the ELG catalogue pages started to grow, the size of the bundle file also grew, making this approach inefficient. When users access the ELG catalogue pages they had to wait until this file is fully downloaded and only then they could interact with the page.

One more drawback of this approach is that, because all of the code is merged into one single file, users have to download portions of code that they will never make use of. For instance, unregistered users that do not have access to dashboard pages and will never visit them still have to wait and download the related code even though they will never be allowed to use it. This creates a bad user experience especially for users that are at slow congested networks.

In order to mitigate all of these issues we utilized the notion of code splitting technique (https://reactjs.org/docs/code-splitting.html). The idea behind this is that the code is split into smaller autonomous files and the users get served only the files that are required to load the pages they are accessing. Thus, users that navigate to the catalogue main page will have to download only the files that are needed to interact with this page. Moreover, most browsers would cache these files so any subsequent visit to these pages would make use of the cached version of the files.

In addition, in order to make the size of the transferred files (images, CSS style sheets, fonts, JavaScript files) as small as possible we configured the Apache server to compress them. As a first option the Google's brotli (https://github.com/google/brotli) lossless compression algorithm is used before the assets are sent to the client over the network and as a fallback in case that the client doesn't support it, the gzip compression format is used.

The following chart (Figure 14) depicts the total amount of data in MB that the user of a mobile device will have to use in order to load the ELG catalogue main page when code splitting is used and when it is not. The chart depicts the transferred resources when they are compressed and when they are not. As we can see the use of code splitting has a huge impact and dramatically improves the initial load size.

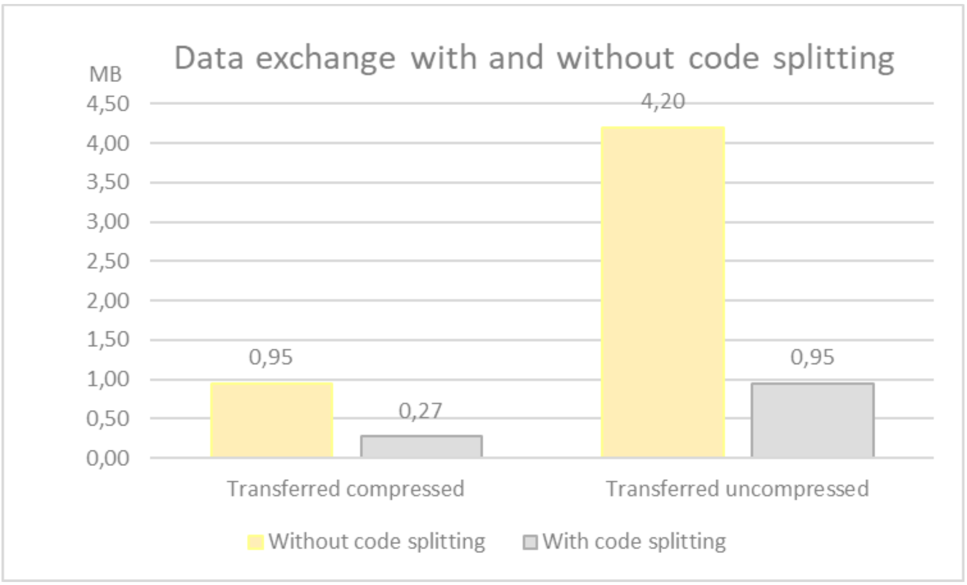


Figure 14: Data exchange with and without code splitting

In this release (Release 3, R3), we have introduced some changes to improve the indexing and discoverability of the landing pages of the metadata records by search engines. Since the catalogue pages are implemented as single page application with client-side rendering, the search engines need to be able to execute JavaScript in order to render and index the pages[6]. Following recommendations for such cases, we have implemented the following:

- Use the History API instead of fragments. In previous releases, the ELG catalogue URLs contained the fragment '#' symbol. In order to implement this, we had to change the URLs by removing the fragment

[6] See for more information the Google's developers guide with the relevant recommendations.

('#') from them. Also, we had to configure the server that serves the catalogue's assets to be able to handle each route request. Finally, in order to be backwards compatible, we configured the server to redirect any old traffic to the new URLs. Thus, any user that accesses the old URL (e.g., https://live.european-language-grid.eu/catalogue/#/resource/service/tool/479) she will be redirected to the new one (https://live.european-language-grid.eu/catalogue/tool-service/479).

- Embed structured data in our pages. This feature improves the discovery of our datasets by Google's dedicated search engine for datasets platform. In order to implement this, we embedded in the page of all items the appropriate JSON-LD (https://json-ld.org) metadata.
- Make sure that pages are linked together with hyperlinks and not buttons. Search engines, in order to discover new pages, search for links, they are not designed to search and click buttons. Thus, we made sure that navigation from one page to another is made through hyperlinks.
- Finally, we enriched catalogue pages with search engines optimization meta tags. We provide meta tags for title, description, keywords and the canonical URL for each item page.

In order to track our performance in Google search results, we have utilized the Google search console. This is a platform that enables owners to monitor their site's traffic, search performance and fix potential problems with indexing. Figure 15 depicts the ranking metrics for the ELG catalogue's pages at Google's search results.

- "Average position" is the average position at Google search results for ELG
- "Average CTR" is the percentage of impressions that resulted in a click
- "Total impressions" refers to the number of times a user saw a link to ELG in search results
- "Total clicks" refers to the count of times a user clicked through to ELG.

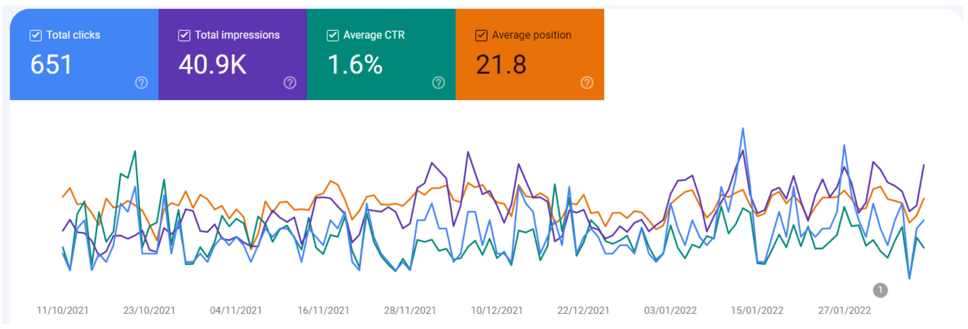
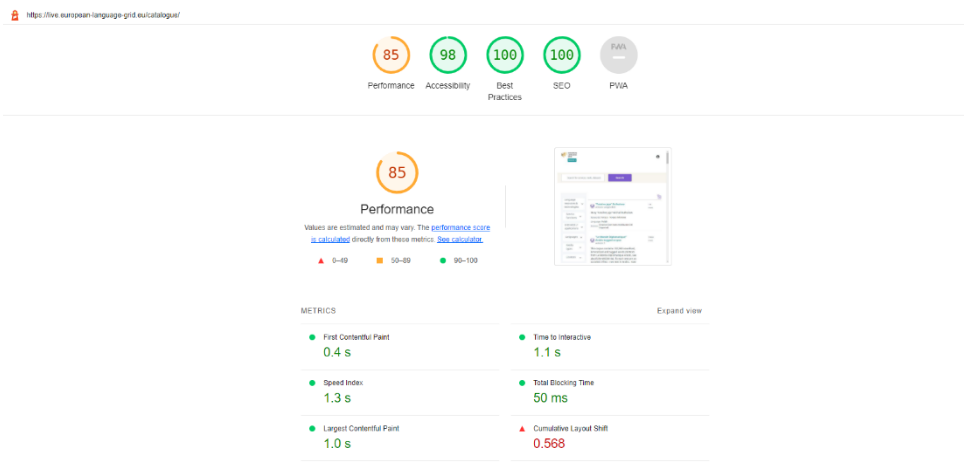


Figure 15: Ranking metrics for ELG at Google search results

Finally, in order to have some baseline of quality measures, Figure 16 shows the performance scores of Google's Lighthouse (https://developers.google.com/web/tools/lighthouse/), an Open Source auditing tool for performance and accessibility.



https://live.european-language-grid.eu/catalogue/

85	98	100	100	PWA —
Performance	Accessibility	Best Practices	SEO	PWA

85

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS Expand view

● First Contentful Paint 0.4 s	● Time to Interactive 1.1 s
● Speed Index 1.3 s	● Total Blocking Time 50 ms
● Largest Contentful Paint 1.0 s	▲ Cumulative Layout Shift 0.568

Figure 16: Performance metrics for R3

6.2 Functionality and Web Interfaces

The catalogue UI is the one of the main access points through which users interact with resources hosted or described in the ELG[7]. These include:

- LT services and applications that are fully integrated into the platform and that can be used through trial UIs supplied by ELG and executed following ELG specifications,
- metadata records for data resources (e.g., corpora, datasets, lexica, terminologies, Machine Learning models, computational grammars, etc.) that are available through ELG and/or other catalogues, and for tools/services that are available in a downloadable form from the ELG platform or that are available through other catalogues or websites,
- Metadata records for companies, research organisations, projects, etc. that are active in the LT domain and can promote themselves through ELG.

The platform architecture (D2.2), and the design of the frontend take into account the needs and requirements of the user groups described in the introduction.

The final release (R3) of the Catalogue UI builds upon, enhances and extends the functionalities and features of the previous releases. In the initial release (R1, delivered in April 2020) the focus was on the consumer's side, the interim release (R2, delivered in March 2021) incorporated functionalities for the provider and, to a lesser degree, the ELG administrators and, finally, R3 enhances and improves previous functionalities, and introduces new features for consumers.

Keeping in mind the different applications that should be integrated, the catalogue UI follows the overall design concept of the ELG frontend Angular 8 SPA application and corporate identity. Accordingly, it features the same header and footer areas, icons, typeface and colour scheme to achieve a consistent look and feel.

The following subsections describe the main pages of the catalogue UI and their features.

[7] REST APIs and the Python SDK that are also used for accessing ELG are described in Deliverable D2.6.

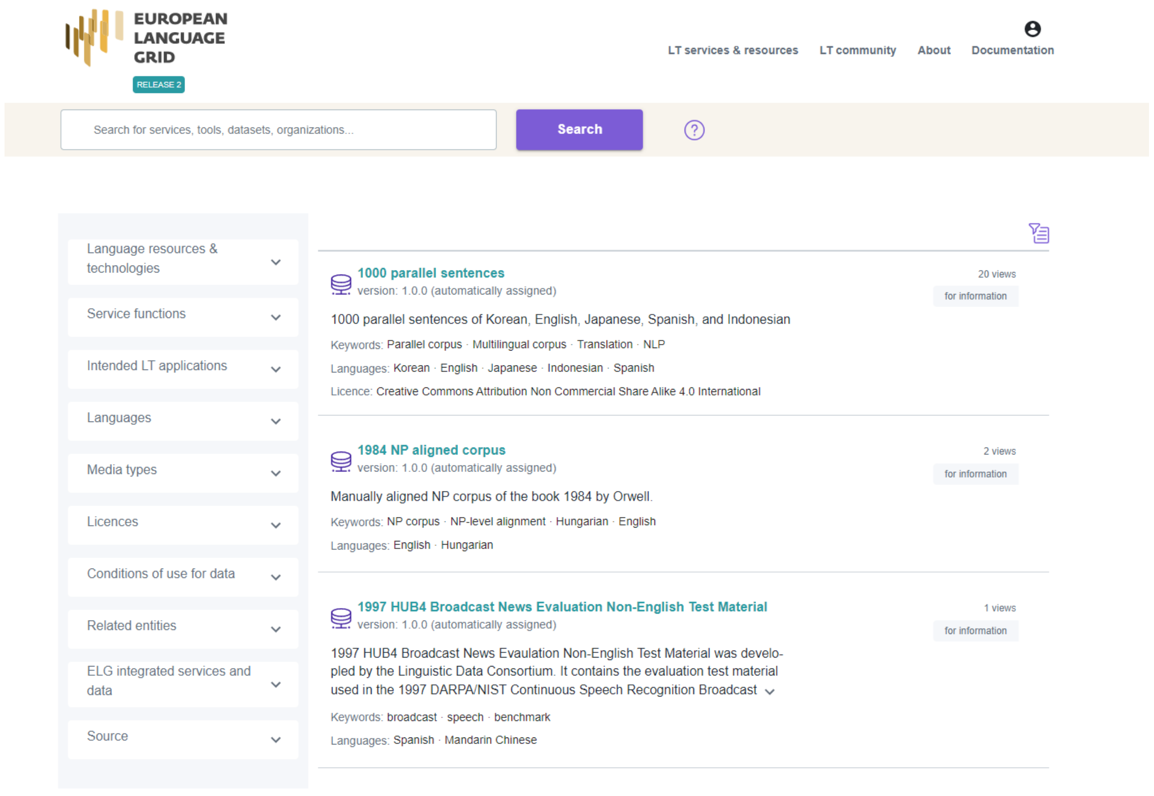
6.2.1 Catalogue Browse and Search Page

The catalogue main page (Figure 17) displays minimum information, allowing users coming from the ELG main website to get a clear picture of the contents of the ELG platform. The top menu items are inherited from the main ELG website and redirect the user to the corresponding pages.

The structure of the page follows the standard way of presenting resources in other catalogues, enabling users to quickly familiarize themselves with ELG.

The main section of the page lists the catalogue contents sorted by resource type and in alphabetical order. Users can also select to sort them by update date in ascending or descending order by clicking on the sorting button at the top of the catalogue.

The information displayed for each item is carefully selected to serve as a preview of the full description and to help users decide whether they want to explore the item further. Following the main findings of the user survey (see Deliverable D3.1), this information includes: the title of the resource (which is a link to the landing page of the item), the version number, an excerpt from its description, the licence(s) under which the resource is available, the language(s) of the resource or supported by the tool/service and keywords. The item type is displayed in the form of an icon next to the title of the item. On the right side of each snippet, further information is provided: mainly statistical information (views for all records, downloads for ELG-hosted data and times used for ELG-compatible services), and appropriate marks for ELG-hosted data and ELG-compatible services.



1000 parallel sentences
version: 1.0.0 (automatically assigned)
20 views
for information
1000 parallel sentences of Korean, English, Japanese, Spanish, and Indonesian
Keywords: Parallel corpus · Multilingual corpus · Translation · NLP
Languages: Korean · English · Japanese · Indonesian · Spanish
Licence: Creative Commons Attribution Non Commercial Share Alike 4.0 International

1984 NP aligned corpus
version: 1.0.0 (automatically assigned)
2 views
for information
Manually aligned NP corpus of the book 1984 by Orwell.
Keywords: NP corpus · NP-level alignment · Hungarian · English
Languages: English · Hungarian

1997 HUB4 Broadcast News Evaluation Non-English Test Material
version: 1.0.0 (automatically assigned)
1 views
for information
1997 HUB4 Broadcast News Evaulation Non-English Test Material was developed by the Linguistic Data Consortium. It contains the evaluation test material used in the 1997 DARPA/NIST Continuous Speech Recognition Broadcast
Keywords: broadcast · speech · benchmark
Languages: Spanish · Mandarin Chinese

Figure 17. List view of catalogue contents

Users can browse through the catalogue or search for items in two ways: (1) using the bar at the top of the catalogue page for free text search, and (2) using the filters (on the left-hand side of the page) to narrow down search results using specific criteria that give a faceted view on the catalogue. The search categories both for free text and facets have been carefully selected to reflect user preferences from the initial survey and follow-up interactions.

An effort to keep the number of facets restricted so as not to overwhelm the users with too much information has been made. The facets include: resource type and related entity type, language, intended LT application, media type, function (for LT services), licence, condition of use, and source of the item. A facet for selecting ELG integrated services and ELG hosted data is also at the user's disposal.

Given the list of facets, accordions (expansion panels) are used to collapse them; this allows us to make all facets visible at once. Facets with long lists of values (e.g. languages, service functions, licences) are endowed with a simple string-based lookup functionality to help users easily find the values they seek for. In addition, a second level ("subfacets") of display is used for the "language" facet (Figure 18), bringing the EU official languages at the top of the facet, since they are of most interest to the ELG users.

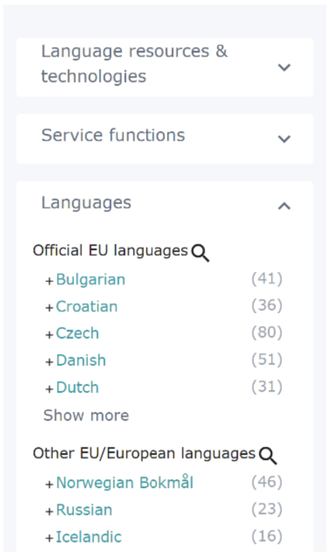
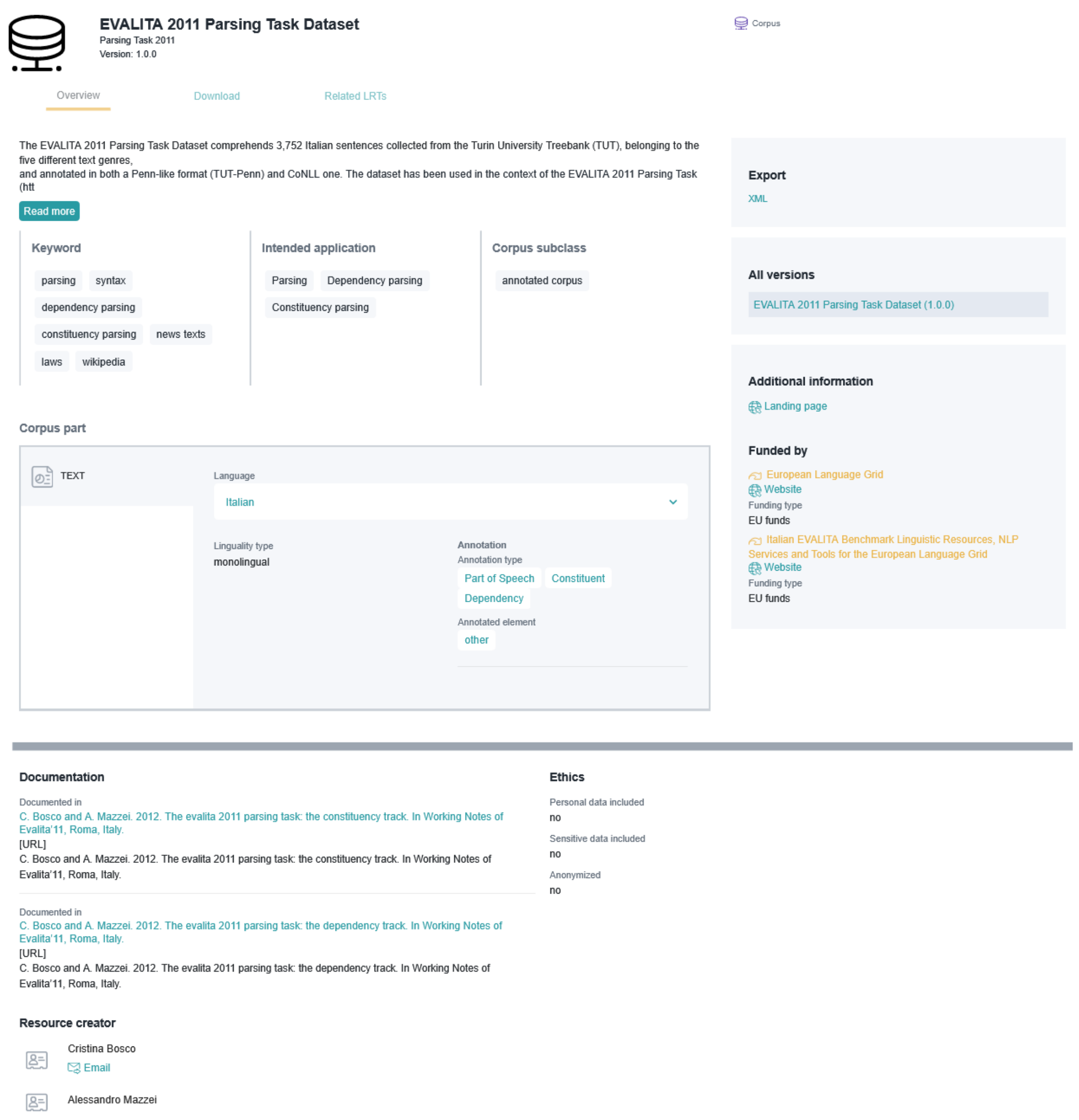


Figure 18: Facet for language

6.2.2 Landing page for items

Once users have spotted an item they find interesting, they can click on its title and be directed to a page that describes it in detail. We developed landing pages for each item category (i.e., LT services, data resources and their subtypes, organizations, projects, etc.) included in the catalogue. They allow users to get detailed information on an item, preview its contents or functionality, and, finally, obtain and use it for their purposes. Figure 19 shows the landing page for a corpus (which is available at https://live.european-language-grid.eu/catalogue/corpus/8057).



EVALITA 2011 Parsing Task Dataset
Parsing Task 2011
Version: 1.0.0

Corpus

Overview Download Related LRTs

The EVALITA 2011 Parsing Task Dataset comprehends 3,752 Italian sentences collected from the Turin University Treebank (TUT), belonging to the five different text genres,
and annotated in both a Penn-like format (TUT-Penn) and CoNLL one. The dataset has been used in the context of the EVALITA 2011 Parsing Task (htt

Read more

Keyword: parsing, syntax, dependency parsing, constituency parsing, news texts, laws, wikipedia

Intended application: Parsing, Dependency parsing, Constituency parsing

Corpus subclass: annotated corpus

Export
XML

All versions
EVALITA 2011 Parsing Task Dataset (1.0.0)

Additional information
Landing page

Funded by
European Language Grid
Website
Funding type
EU funds
Italian EVALITA Benchmark Linguistic Resources, NLP Services and Tools for the European Language Grid
Website
Funding type
EU funds

Corpus part

TEXT

Language
Italian

Linguality type
monolingual

Annotation
Annotation type
Part of Speech, Constituent, Dependency
Annotated element
other

Documentation

Documented in
C. Bosco and A. Mazzei. 2012. The evalita 2011 parsing task: the constituency track. In Working Notes of Evalita'11, Roma, Italy.
[URL]
C. Bosco and A. Mazzei. 2012. The evalita 2011 parsing task: the constituency track. In Working Notes of Evalita'11, Roma, Italy.

Documented in
C. Bosco and A. Mazzei. 2012. The evalita 2011 parsing task: the dependency track. In Working Notes of Evalita'11, Roma, Italy.
[URL]
C. Bosco and A. Mazzei. 2012. The evalita 2011 parsing task: the dependency track. In Working Notes of Evalita'11, Roma, Italy.

Ethics
Personal data included
no
Sensitive data included
no
Anonymized
no

Resource creator
Cristina Bosco
Email
Alessandro Mazzei

Figure 19: Landing page for a corpus

Even though the landing pages contain information that differ for each category, our main efforts aim to provide a consistent visual look and feel. Landing pages share a common layout that consists of a header, a right-hand sidebar, a main content area and a bottom content region, described in the following subsections. Each page uses the same principles customized for the different categories of content (i.e., Project, Tool/Service, Organization, Corpus etc.). The pages apply the general principles of content organization such as grouping similar items together, numbering items, and using headings and prompt text.

The information displayed on the landing page of each item comes from the respective metadata record. Taking into consideration the specificities and richness of the metadata schema (D2.3), but also user-friendliness, we opted for splitting the information along specific sections of the page and organizing elements into different areas. Tabs are also used for splitting information into smaller views and enabling users to navigate through the item features: view its general information, try out the service, view information on how it is distributed and obtain the item (run the service or download the data resource), etc. We also use accordions to group similar information and keep the landing page at a reasonable length.

The main challenge in designing landing pages for the catalogue resources is that the metadata schema includes a lot of elements that take values from controlled vocabularies, rather than free-text items. This poses a problem since the accumulation of many elements and values together can become tiring to the eye. The proposed design displays the keys (element labels) in small font size and places more emphasis on the values, which makes them visually more prominent and hence better identified. In addition, the positioning of the elements on the page is carefully thought through to draw users' attention to the most important information.

6.2.2.1 Header

The header area displays the main information for each item. Its purpose is to introduce the user to the current page. The left header region is reserved for the logo of the item (Figure 20); if none is provided by the metadata creator, then the default icon associated with its category is displayed (Figure 21).

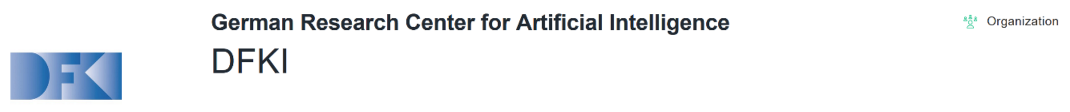


Figure 20: Header for an organization with the organization logo

University of Edinburgh
UEDIN
Organization

Figure 21: Header for an organization with default icon

6.2.2.2 Main Content Area

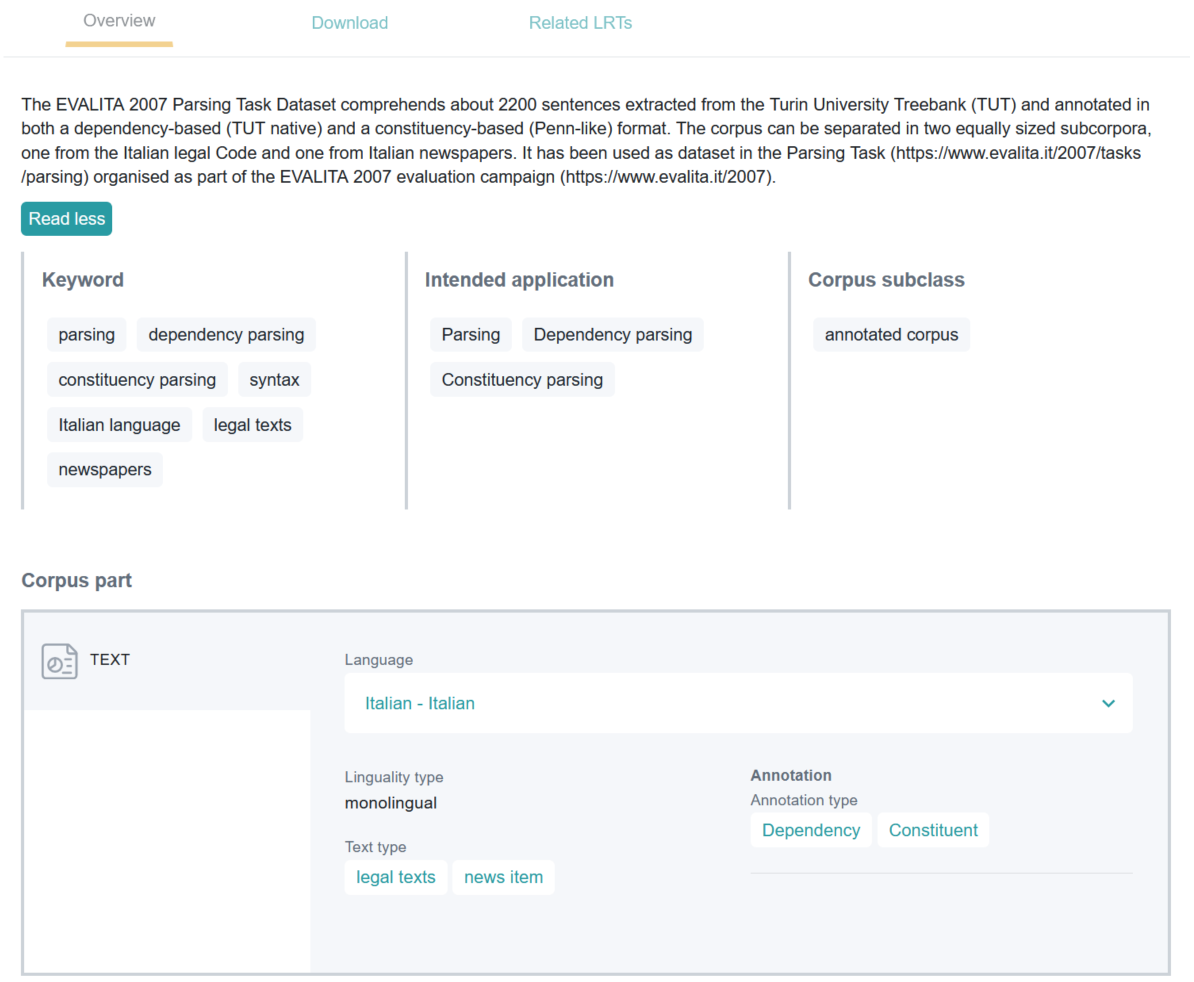
The main region contains the information that is of most interest to users. Figure 22 shows the main region for the item displayed on Figure 22.

The top part of this region is devoted to the "description", a human readable summary of the item's main characteristics, as provided by the metadata creator. Since we do not have control over free text items, like, for instance, the description of a resource, we ensure readability by trimming texts to a maximum of five lines of text. A "Read more" button expands the text to the remaining lines. In addition, in this release, we have started supporting a subset of HTML tags[8] for this metadata element; users can add the HTML tags in the element in the XML metadata record they upload in the platform or use the rich text editor that has been introduced for this reason in the metadata editor. The use of HTML tags in the description text makes the text easier to read.

The description is followed by the "classification section" in which we group information that classifies each item according to different parameters, such as keywords, domain, relevant LT area, resource subtype, etc.

[8] The subset has been selected so as to avoid security issues.

The bottom section of the main content region is used for the technical metadata and is different for each item type. For corpora, for instance, this page includes information on the language(s) and contents grouped in tabs corresponding to the corpus parts.



The EVALITA 2007 Parsing Task Dataset comprehends about 2200 sentences extracted from the Turin University Treebank (TUT) and annotated in both a dependency-based (TUT native) and a constituency-based (Penn-like) format. The corpus can be separated in two equally sized subcorpora, one from the Italian legal Code and one from Italian newspapers. It has been used as dataset in the Parsing Task (https://www.evalita.it/2007/tasks/parsing) organised as part of the EVALITA 2007 evaluation campaign (https://www.evalita.it/2007).

Keyword: parsing, dependency parsing, constituency parsing, syntax, Italian language, legal texts, newspapers

Intended application: Parsing, Dependency parsing, Constituency parsing

Corpus subclass: annotated corpus

Corpus part

TEXT

Language: Italian - Italian

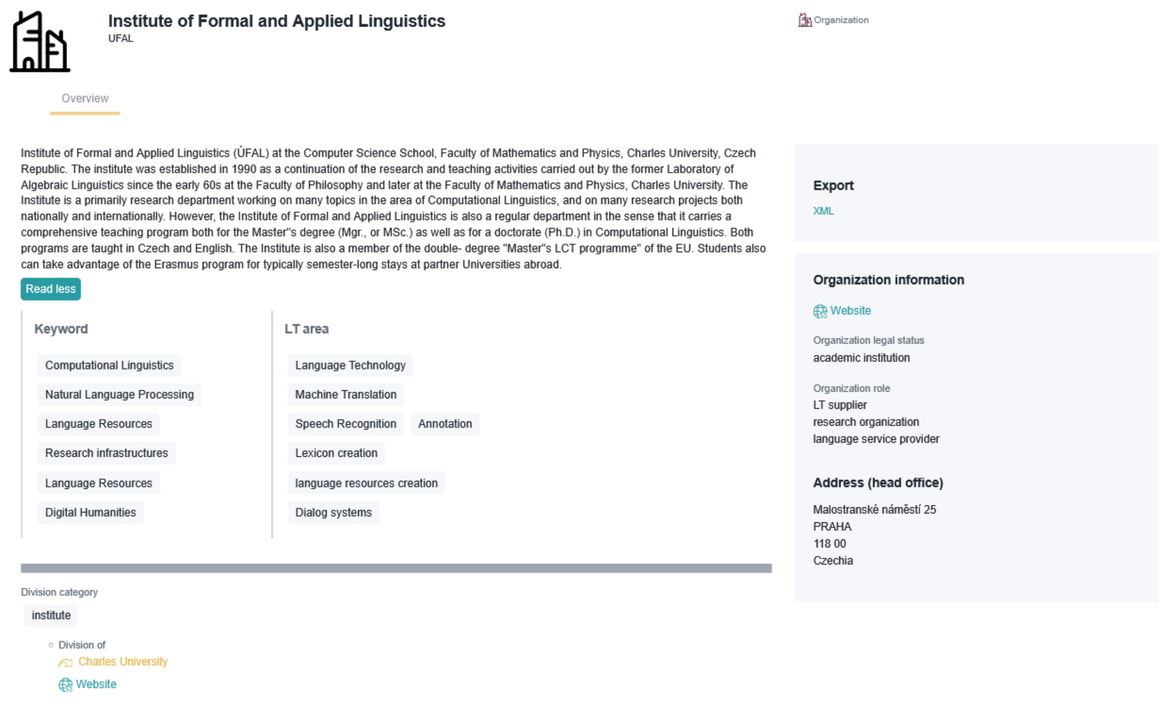
Linguality type: monolingual

Text type: legal texts, news item

Annotation
Annotation type: Dependency, Constituent

Figure 22: Main area for a corpus

For Projects and Organizations, the main content area is similar. Still, fine tuning to accommodate the different metadata elements was made. For instance, while the description and classification sections follow the same principles, the Organization page includes a section for the link to the parent organization, in the case of divisions (Figure 23 showing the organization page at https://live.european-language-grid.eu/catalogue/organization/385).


Institute of Formal and Applied Linguistics
UFAL

Organization

Overview

Institute of Formal and Applied Linguistics (ÚFAL) at the Computer Science School, Faculty of Mathematics and Physics, Charles University, Czech Republic. The institute was established in 1990 as a continuation of the research and teaching activities carried out by the former Laboratory of Algebraic Linguistics since the early 60s at the Faculty of Philosophy and later at the Faculty of Mathematics and Physics, Charles University. The Institute is a primarily research department working on many topics in the area of Computational Linguistics, and on many research projects both nationally and internationally. However, the Institute of Formal and Applied Linguistics is also a regular department in the sense that it carries a comprehensive teaching program both for the Master"s degree (Mgr., or MSc.) as well as for a doctorate (Ph.D.) in Computational Linguistics. Both programs are taught in Czech and English. The Institute is also a member of the double- degree "Master"s LCT programme" of the EU. Students also can take advantage of the Erasmus program for typically semester-long stays at partner Universities abroad.

Read less

Keyword	LT area
Computational Linguistics	Language Technology
Natural Language Processing	Machine Translation
Language Resources	Speech Recognition, Annotation
Research infrastructures	Lexicon creation
Language Resources	language resources creation
Digital Humanities	Dialog systems

Division category
institute

Division of
Charles University
Website

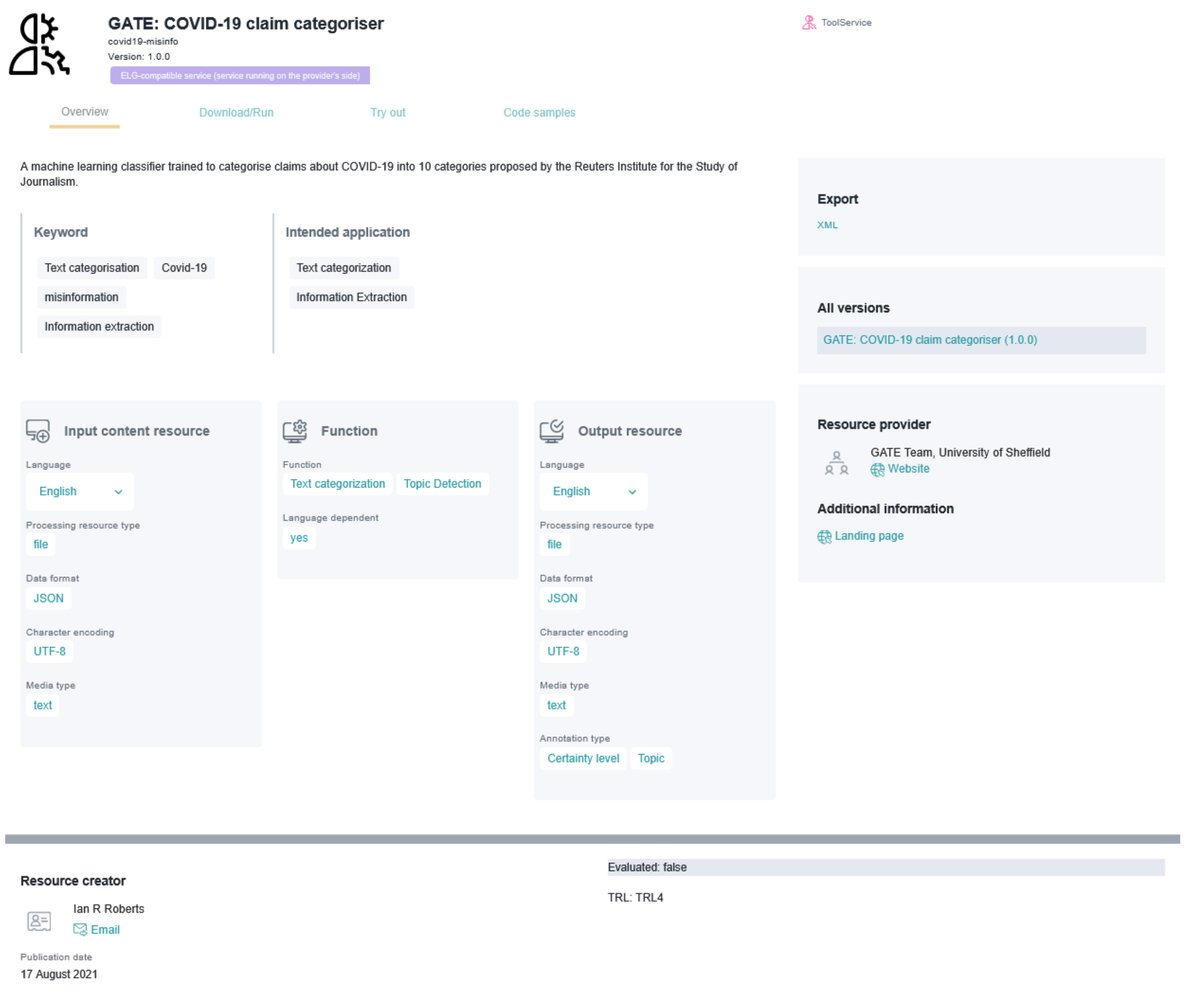
Export
XML

Organization information
Website
Organization legal status
academic institution
Organization role
LT supplier
research organization
language service provider

Address (head office)
Malostranské náměstí 25
PRAHA
118 00
Czechia

Figure 23: Landing page for an organization

Other categories, on the other hand, include more information and thus additional layouts and styles. One example is the category of Tool/Services. Figure 24 displays the landing page of a Tool/Service, which is available at https://live.european-language-grid.eu/catalogue/tool-service/17408. Each Tool/Service is described according to its specifications, i.e., a tool expects an input, performs an operation on this input and results in the expected output. In order to visually display this procedure, we split the area into three different columns.



GATE: COVID-19 claim categoriser
covid19-misinfo
Version: 1.0.0
ELG-compatible service (service running on the provider's side)

ToolService

Overview | Download/Run | Try out | Code samples

A machine learning classifier trained to categorise claims about COVID-19 into 10 categories proposed by the Reuters Institute for the Study of Journalism.

Keyword
Text categorisation · Covid-19 · misinformation · Information extraction

Intended application
Text categorization · Information Extraction

Input content resource
Language: English
Processing resource type: file
Data format: JSON
Character encoding: UTF-8
Media type: text

Function
Function: Text categorization · Topic Detection
Language dependent: yes

Output resource
Language: English
Processing resource type: file
Data format: JSON
Character encoding: UTF-8
Media type: text
Annotation type: Certainty level · Topic

Export
XML

All versions
GATE: COVID-19 claim categoriser (1.0.0)

Resource provider
GATE Team, University of Sheffield
Website

Additional information
Landing page

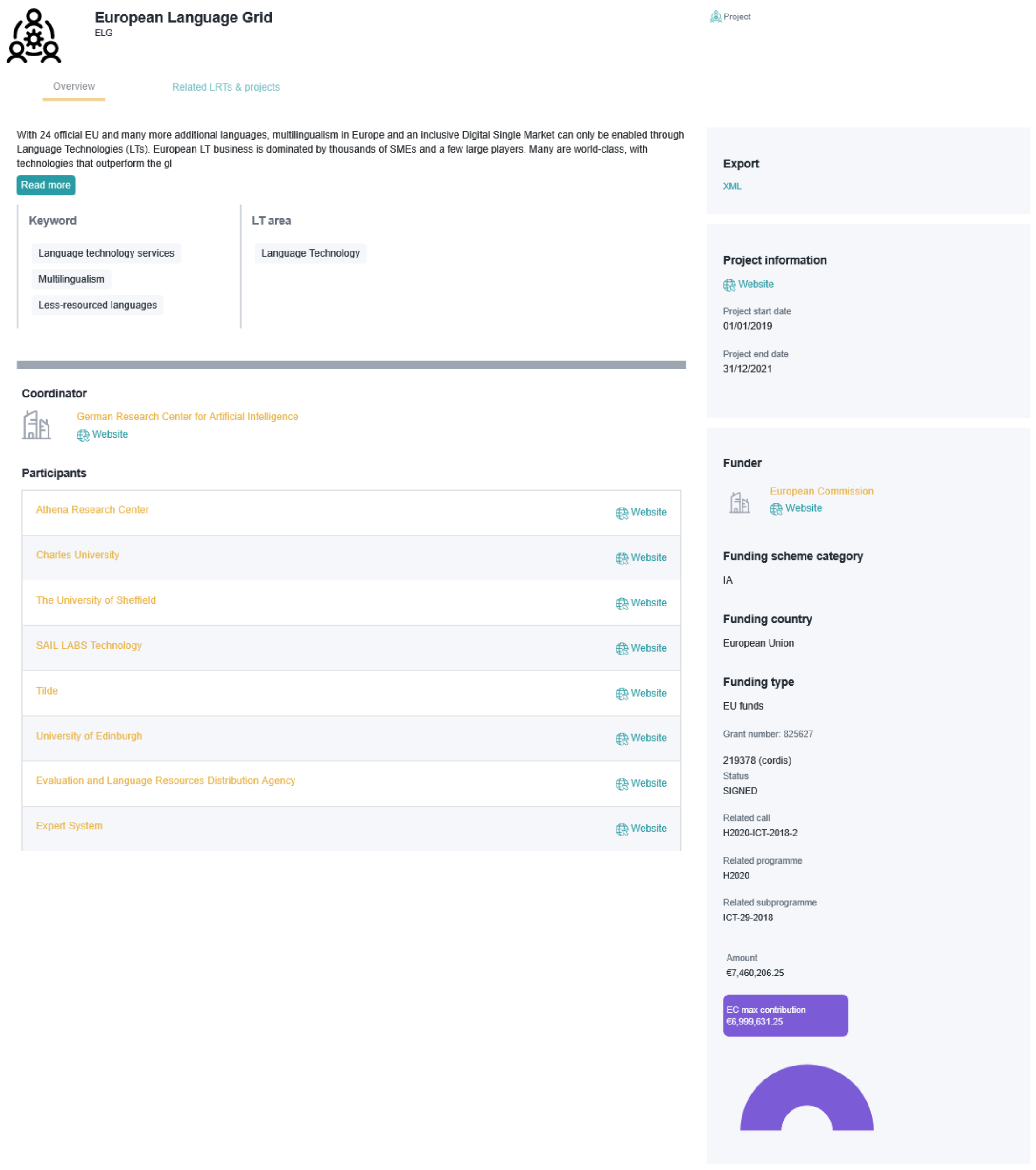
Resource creator
Ian R Roberts
Email

Publication date
17 August 2021

Evaluated: false
TRL: TRL4

Figure 24:Landing page for a tool/service

In a similar way, the landing page for a Project (Figure 25, available at https://live.european-language-grid.eu/catalogue/project/395) also engages users with areas that are visually the same but alters the main content area focusing on a project's relevant content.



European Language Grid
ELG

Project

Overview   Related LRTs & projects

With 24 official EU and many more additional languages, multilingualism in Europe and an inclusive Digital Single Market can only be enabled through Language Technologies (LTs). European LT business is dominated by thousands of SMEs and a few large players. Many are world-class, with technologies that outperform the gl

Read more

Keyword	LT area
Language technology services	Language Technology
Multilingualism	
Less-resourced languages	

Coordinator

German Research Center for Artificial Intelligence
Website

Participants

Athena Research Center	Website
Charles University	Website
The University of Sheffield	Website
SAIL LABS Technology	Website
Tilde	Website
University of Edinburgh	Website
Evaluation and Language Resources Distribution Agency	Website
Expert System	Website

Export
XML

Project information
Website
Project start date
01/01/2019
Project end date
31/12/2021

Funder
European Commission
Website

Funding scheme category
IA

Funding country
European Union

Funding type
EU funds

Grant number: 825627

219378 (cordis)
Status
SIGNED

Related call
H2020-ICT-2018-2

Related programme
H2020

Related subprogramme
ICT-29-2018

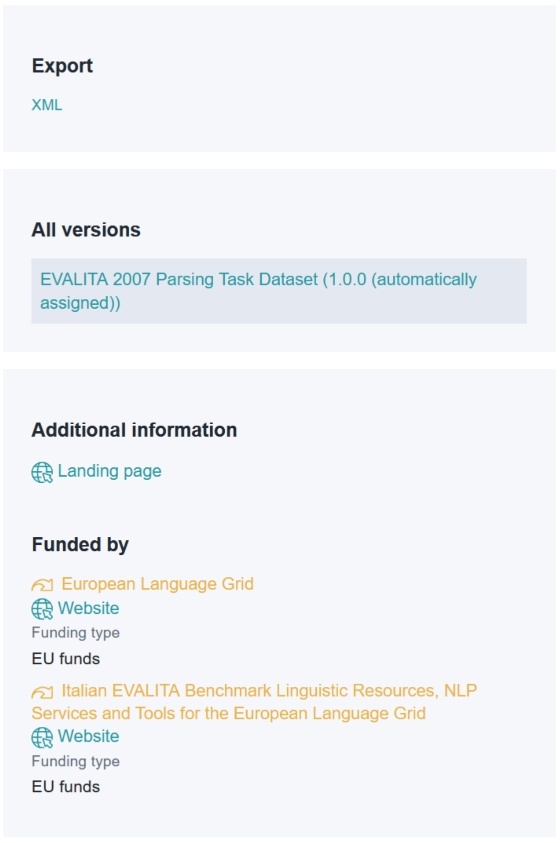
Amount
€7,460,206.25

EC max contribution
€6,999,631.25

Figure 25: Landing page for a project

6.2.2.3 Right sidebar

This area was created in order to show information deemed important for each category while keeping the page short enough so that users do not have to scroll down to view all the information. For instance, for resources, this section includes information on funding projects, contact points where users can ask for more information, links to all versions of the item, etc. (see Figure 26 with the right sidebar of the resource on Figure 20). A grey background colour distinguishes the sidebar area from the rest of the content.



Export

XML

All versions

EVALITA 2007 Parsing Task Dataset (1.0.0 (automatically assigned))

Additional information

Landing page

Funded by

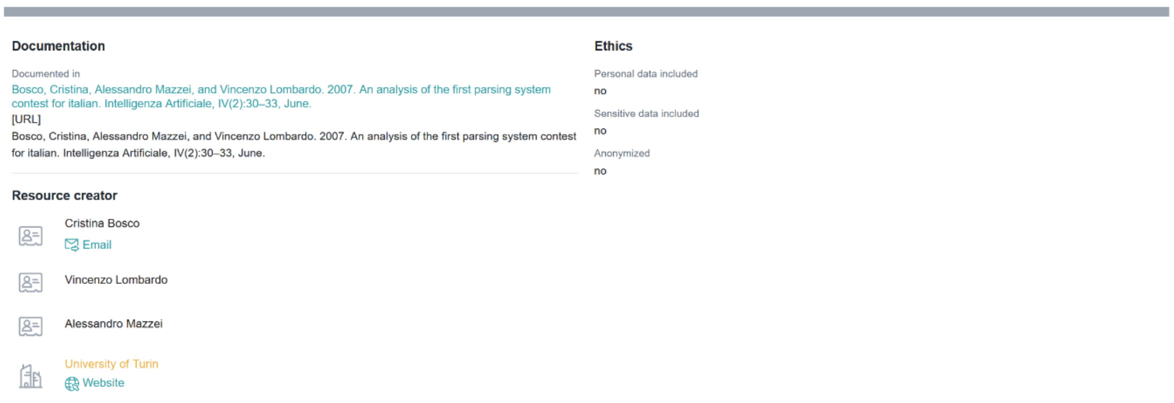
European Language Grid
Website
Funding type
EU funds

Italian EVALITA Benchmark Linguistic Resources, NLP Services and Tools for the European Language Grid
Website
Funding type
EU funds

Figure 26: Sidebar for a resource

6.2.2.4 Bottom Content

The bottom content area brings together information types that are common to all resource types and considered important enough to show on the main tab. Such information includes documents, such as user manuals, publications, etc. where users can learn how to use the resource, have access to information on the resource creator and creation details, evaluation of the resource, etc. Figure 27 shows the bottom area of the resource on Figure 19.



Documentation

Documented in
Bosco, Cristina, Alessandro Mazzei, and Vincenzo Lombardo. 2007. An analysis of the first parsing system contest for italian. Intelligenza Artificiale, IV(2):30–33, June.
[URL]
Bosco, Cristina, Alessandro Mazzei, and Vincenzo Lombardo. 2007. An analysis of the first parsing system contest for italian. Intelligenza Artificiale, IV(2):30–33, June.

Resource creator

Cristina Bosco
Email

Vincenzo Lombardo

Alessandro Mazzei

University of Turin
Website

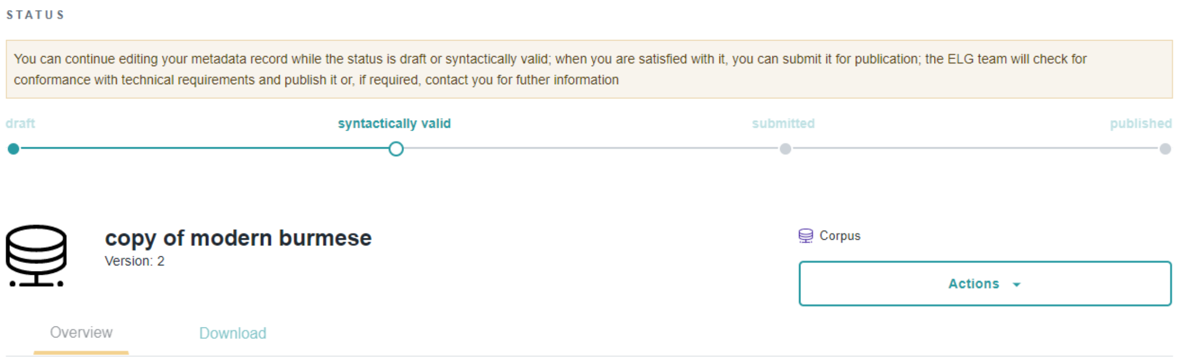
Ethics

Personal data included
no
Sensitive data included
no
Anonymized
no

Figure 27: Bottom content area for resources

Finally, the landing page of items for logged-in providers and administrators (Figure 28) includes two additional features:

- a dedicated area for "Actions" which allows them to manage the metadata records;
- an area showing a bar with the publication status of the item, in accordance with the ELG publication lifecycle (for more information, see Deliverable D2.4).


STATUS

You can continue editing your metadata record while the status is draft or syntactically valid; when you are satisfied with it, you can submit it for publication; the ELG team will check for conformance with technical requirements and publish it or, if required, contact you for futher information

Figure 28: Publication status bar and Actions areas

6.2.2.5 Tabs

Tabs are used to separate information into appropriate groups and draw users' attention to particular details. All categories include a default tab "Overview" which contains the main descriptive and technical metadata. Additional tabs are used depending on the category.

All resources include a tab ("Download" or "Download/Run") with information on their distributable form(s), i.e., the licensing conditions under which a resource may be acquired, technical details (such as format) of the physical files. This tab includes the button for downloading a data resource, if hosted in ELG.

ELG-compatible services include two more tabs, as shown on Figure 29 (top part of the tool on Figure 24):

- a "Test/Try out" Tab, which contains the trial UI implemented by ELG so that users can test the service and which is specific to broad categories, namely, Information Extraction (Figure 30), Machine Translation, Speech Synthesis, Automatic Speech Recognition;
- the "Code samples" tab, which allows end-users with technical knowledge to test a tool in command mode or through the Python SDK (for ORE information, see Deliverable D2.4).

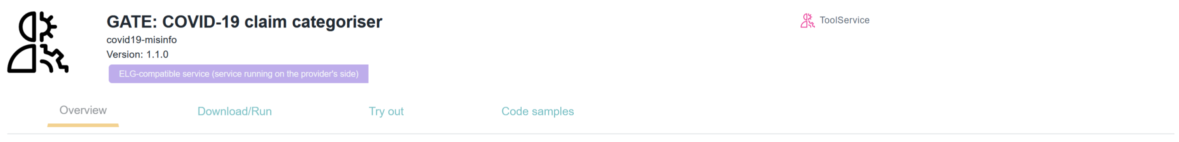

GATE: COVID-19 claim categoriser
covid19-misinfo
Version: 1.1.0
ELG-compatible service (service running on the provider's side)
ToolService
Overview
Download/Run
Try out
Code samples

Figure 29: Tabs for ELG-compatible services

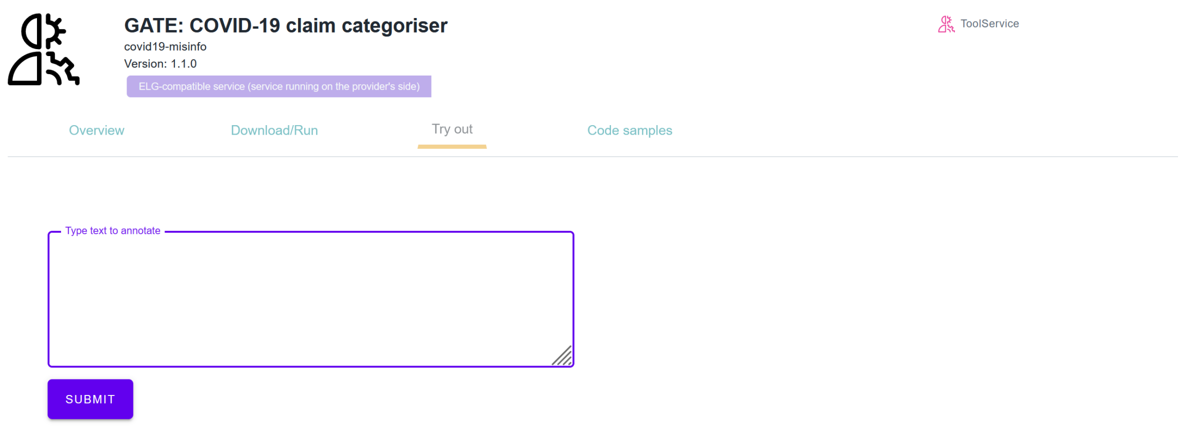
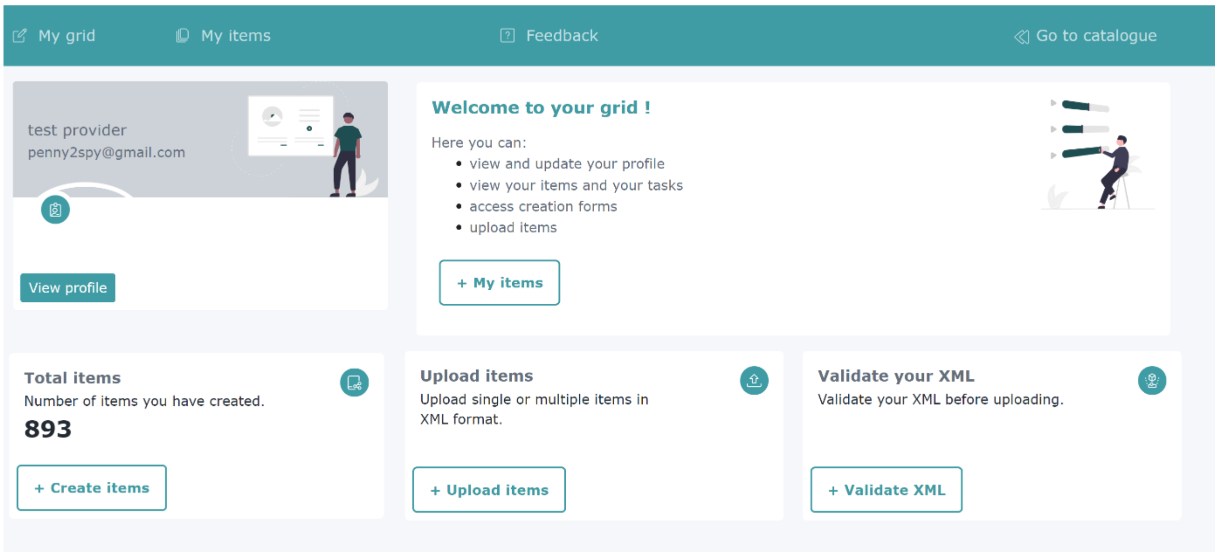


Figure 30: Test/Try out tab for a sample Information Extraction service

6.2.3 Dashboard

To support users in accessing and managing the catalogue items and performing the actions that are allowed depending on their user role, we make use of a dashboard, which in ELG we call "**grid**" (Figure 31). The grid is accessible to all logged-in users through the menu at the top of the ELG page.



Welcome to your grid !

Here you can:
- view and update your profile
- view your items and your tasks
- access creation forms
- upload items

Total items
Number of items you have created.
893

Upload items
Upload single or multiple items in XML format.

Validate your XML
Validate your XML before uploading.

Figure 31: Provider's grid

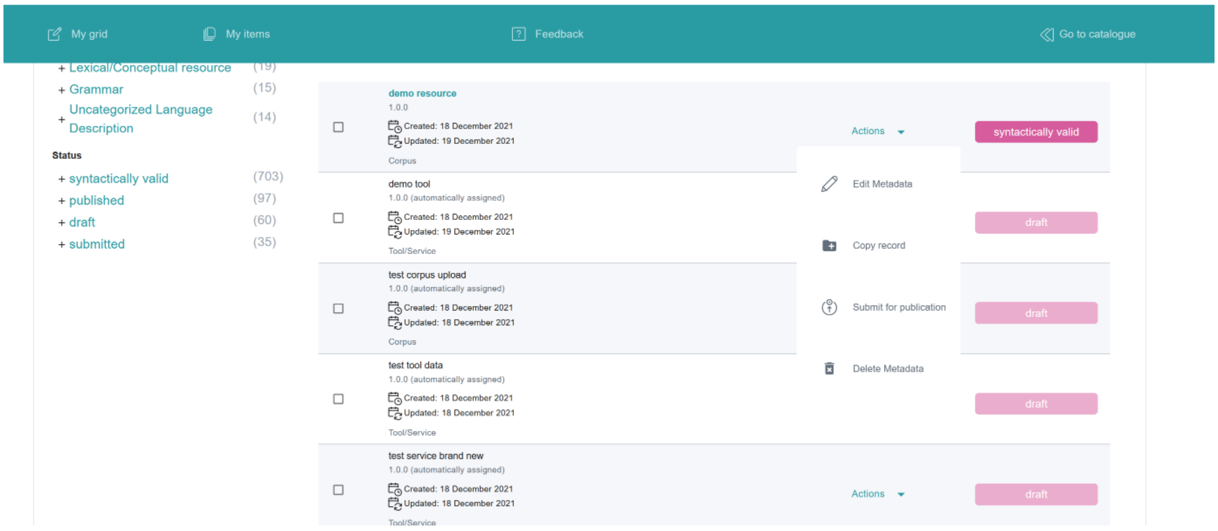
The grid is the central place through which all users have access to various actions and resources, depending on their role. Although the look and feel is the same across user roles, the menu items, pages and actions differ depending on the user role. For the sake of convenience, in the following subsections, we use the terms "consumer's grid", "provider's grid", and "validator's grid" to refer to the pages offered for the respective user role, although users with multiple roles have access to all functionalities through a single grid[9].

[9] It should be noted that administrators perform all monitoring actions through pages created with Django. Given their familiarity with technical environments, the creation of dedicated pages and forms was considered unnecessary for the project duration.

The main page of the grid (Figure 31) is made up of cards offering an overview of the user's data, and quick access to the main functionalities of their user role.

In addition, they have access to dedicated catalogue pages that serve as a focused version of the catalogue, filtering records according to each user's role (e.g., for providers, a list of the items they have created, for validators, a list of the items they have been assigned for validation). Figure 32 shows the relevant page for providers called "my items". This type of page implements browse and search functionalities like the main catalogue page, i.e.

- browse through all the items
- sort items by name or date of update
- search for specific items using the free text bar (e.g., search for a language processing service, a corpus, a project or an organization by its name; or search for all entries that mention "machine translation" or "English")
- filter the catalogue or refine the search results through the facets on the left side bar, which are customized depending on the user role
- perform actions on single or multiple items.

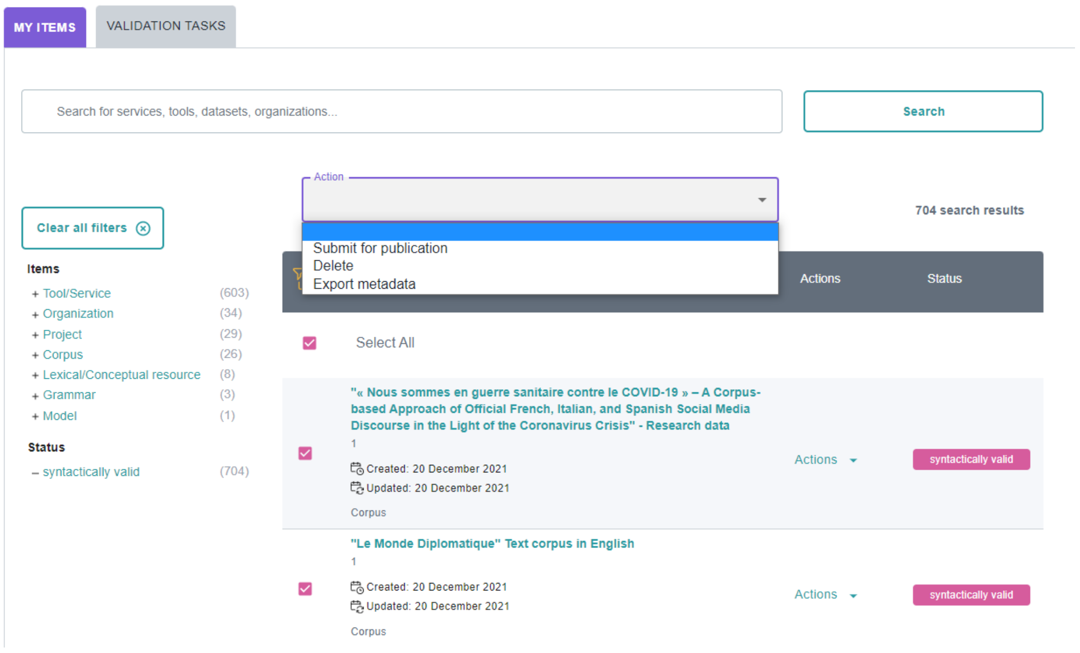

My grid · My items · Feedback · Go to catalogue

+ Lexical/Conceptual resource	(19)
+ Grammar	(15)
+ Uncategorized Language Description	(14)
Status	
+ syntactically valid	(703)
+ published	(97)
+ draft	(60)
+ submitted	(35)

☐	demo resource 1.0.0 Created: 18 December 2021 Updated: 19 December 2021 Corpus	Actions	syntactically valid
☐	demo tool 1.0.0 (automatically assigned) Created: 18 December 2021 Updated: 19 December 2021 Tool/Service		draft
☐	test corpus upload 1.0.0 (automatically assigned) Created: 18 December 2021 Updated: 18 December 2021 Corpus		draft
☐	test tool data 1.0.0 (automatically assigned) Created: 18 December 2021 Updated: 18 December 2021 Tool/Service		draft
☐	test service brand new 1.0.0 (automatically assigned) Created: 18 December 2021 Updated: 18 December 2021 Tool/Service	Actions	draft

Figure 32: My items page (for providers)

To do this, they simply click on "Actions" of each record and they are presented with a choice of options (Figure 32). Single actions are straightforward from this interface, but one of the major benefits of this page is that a user can perform bulk actions on selected items. Bulk actions are performed by simply clicking the checkboxes on the left of each element, and then a dropdown menu appears on the top that lists all the allowed actions for the selected items (Figure 33).


MY ITEMS | VALIDATION TASKS

Search for services, tools, datasets, organizations...

Search

Clear all filters

Action

Submit for publication
Delete
Export metadata

704 search results

Items	
+ Tool/Service	(603)
+ Organization	(34)
+ Project	(29)
+ Corpus	(26)
+ Lexical/Conceptual resource	(8)
+ Grammar	(3)
+ Model	(1)
Status	
– syntactically valid	(704)

		Actions	Status
☑	Select All		
☑	"« Nous sommes en guerre sanitaire contre le COVID-19 » – A Corpus-based Approach of Official French, Italian, and Spanish Social Media Discourse in the Light of the Coronavirus Crisis" - Research data 1 Created: 20 December 2021 Updated: 20 December 2021 Corpus	Actions	syntactically valid
☑	"Le Monde Diplomatique" Text corpus in English 1 Created: 20 December 2021 Updated: 20 December 2021 Corpus	Actions	syntactically valid

Figure 33: Bulk actions on my items

In the following subsections, we present the functionalities offered through the ELG platform for each user role.

6.2.3.1 Consumer's grid and functionalities

Through the grid, registered consumers have access to the following functionalities:

- Information on the remaining quotas they can use for calling services, in accordance with the ELG policies that restrict the daily usage of services to a specific number of requests/calls and size (in bytes)[10].
- View through their dedicated catalogue pages
- "my downloads": the list of resources they have previously downloaded, with accompanying information on the date and the licence selected for the download;
- "my usage": the list of the services they have used before, with information on the status of the call, the call type, used bytes, start time and duration of the call.

Through this page, consumers can download again a resource with the same licence.

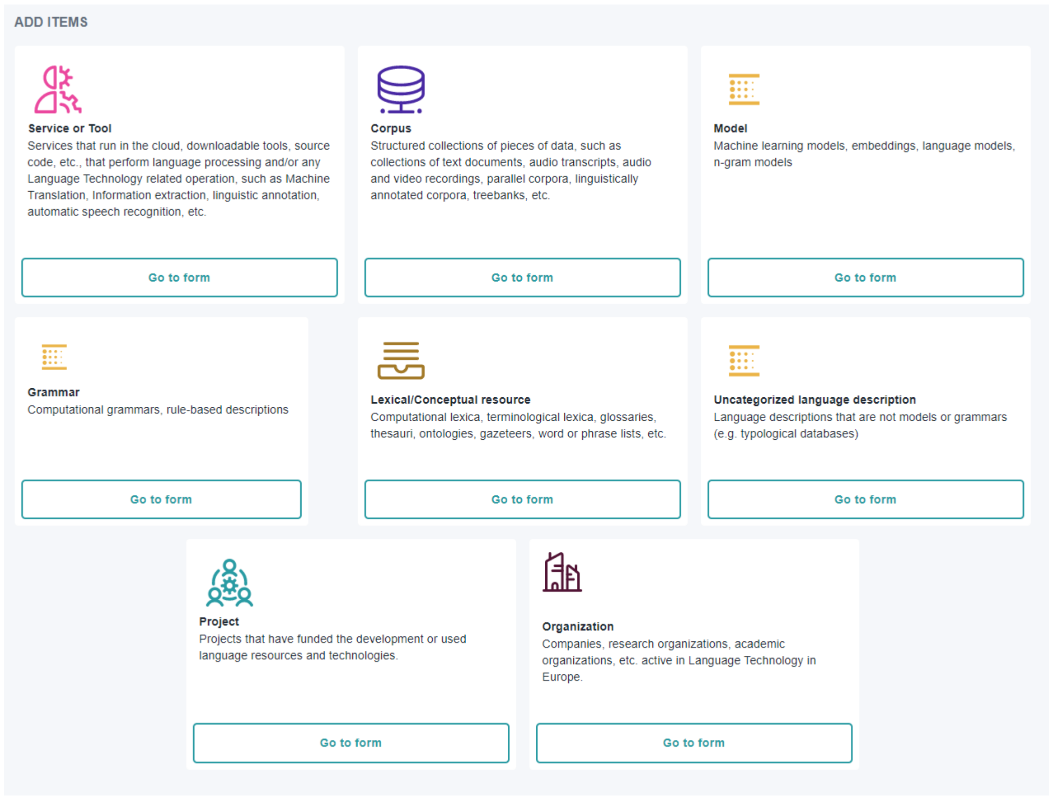
6.2.3.2 Provider's grid and functionalities

Providers have access to all the metadata records they have created, through the "my items" page; they have also access to the functionalities that will allow them to create new items, i.e. the upload of metadata records, and the interactive editor.

6.2.3.2.1 Accessing the interactive editor

Users can create a metadata record by accessing the main editor page through the provider's grid and selecting the type of the item they are interested in (Figure 34). Depending on their selection, they will get forwarded to the corresponding editor page where they can fill the metadata information for the item they want to register.

[10] For more information, see Deliverable D2.6.


ADD ITEMS

Service or Tool
Services that run in the cloud, downloadable tools, source code, etc., that perform language processing and/or any Language Technology related operation, such as Machine Translation, Information extraction, linguistic annotation, automatic speech recognition, etc.

Corpus
Structured collections of pieces of data, such as collections of text documents, audio transcripts, audio and video recordings, parallel corpora, linguistically annotated corpora, treebanks, etc.

Model
Machine learning models, embeddings, language models, n-gram models

Grammar
Computational grammars, rule-based descriptions

Lexical/Conceptual resource
Computational lexica, terminological lexica, glossaries, thesauri, ontologies, gazeteers, word or phrase lists, etc.

Uncategorized language description
Language descriptions that are not models or grammars (e.g. typological databases)

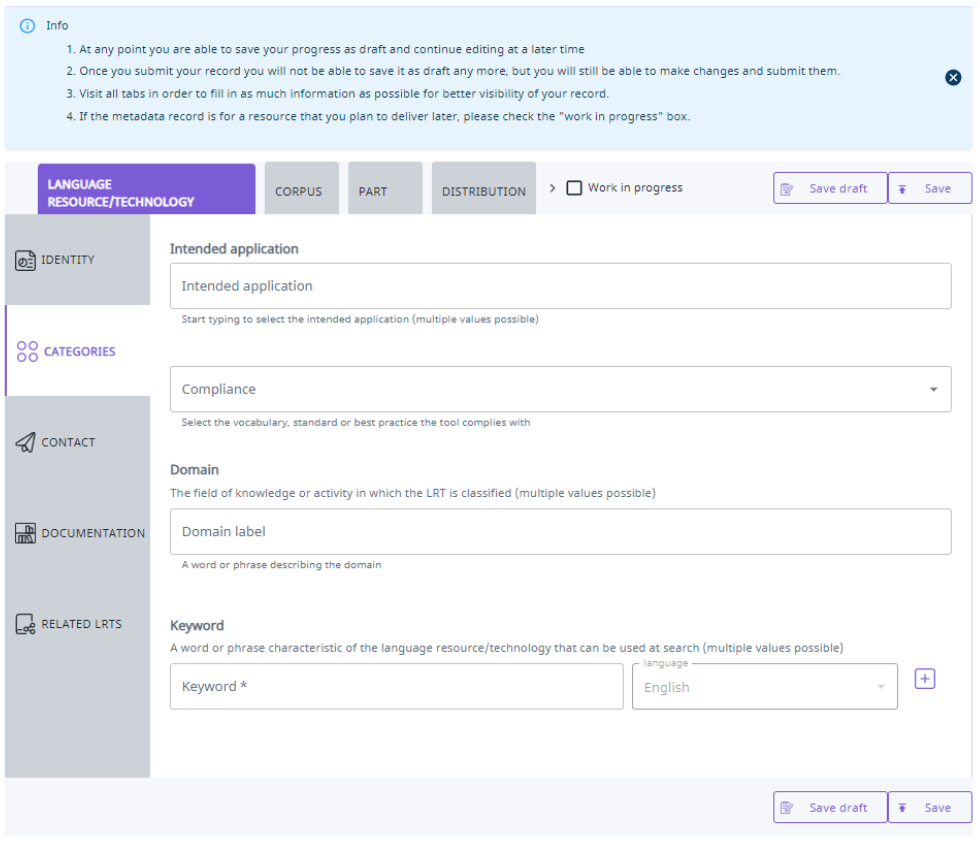
Project
Projects that have funded the development or used language resources and technologies.

Organization
Companies, research organizations, academic organizations, etc. active in Language Technology in Europe.

Figure 34: Selection of item type for the editor

6.2.3.2.2 Interactive editor

For the ELG metadata editor, we have designed and implemented forms that enable users to describe their items in compliance with the ELG metadata schema. Since the schema is rather complex, we attempt to meet the needs of our users by hiding this complexity and present them with a full-fledged GUI. As a result, we refrain from using long forms that could overwhelm users; instead, we have adopted a more interactive design based on the organization of the elements along a natural workflow split across multiple steps. The steps are not linear, meaning that users can proceed in any step, revisit, or revise previous steps at any time. For example, Figure 35 shows the form for creating the metadata record for a corpus.


Info

1. At any point you are able to save your progress as draft and continue editing at a later time
2. Once you submit your record you will not be able to save it as draft any more, but you will still be able to make changes and submit them.
3. Visit all tabs in order to fill in as much information as possible for better visibility of your record.
4. If the metadata record is for a resource that you plan to deliver later, please check the "work in progress" box.

Intended application

Intended application

Start typing to select the intended application (multiple values possible)

Compliance

Select the vocabulary, standard or best practice the tool complies with

Domain

The field of knowledge or activity in which the LRT is classified (multiple values possible)

Domain label

A word or phrase describing the domain

Keyword

A word or phrase characteristic of the language resource/technology that can be used at search (multiple values possible)

Keyword *

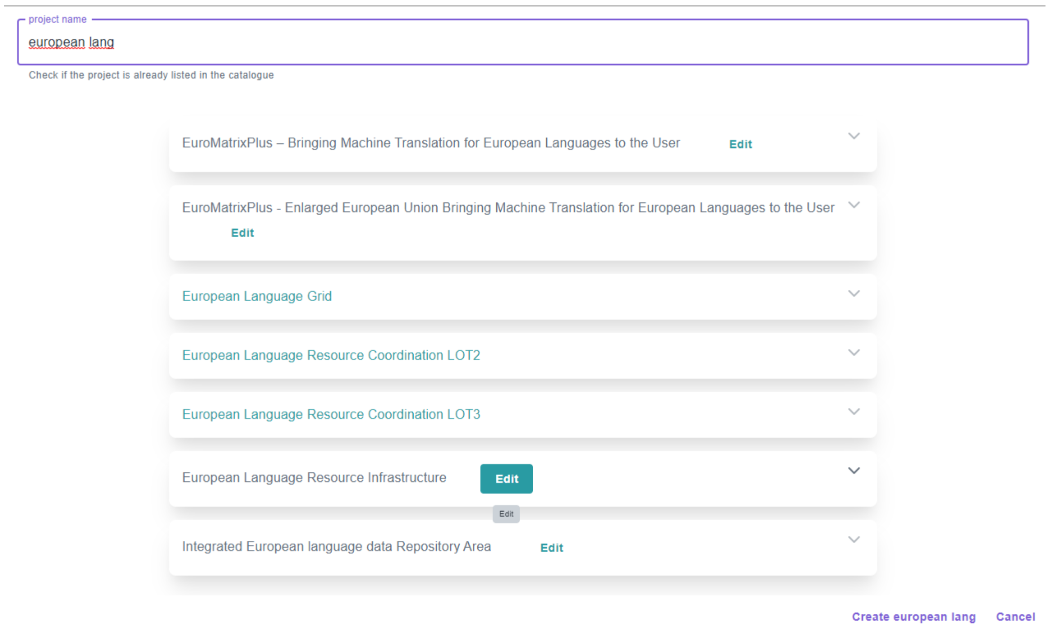
language: English

Figure 35: Editor form for corpus

By grouping and organizing the elements, we achieve the following goals:

- **Focus.** By breaking up the workflow into multiple steps, users can focus on one topic at a time. They are thus potentially less distracted and overwhelmed by other screen elements competing for their attention. We obviously modularized the application in ways that make sense to users and minimize the need to refer to information contained in other areas.
- **In-context help.** As users have fewer items on the screen, we can clearly explain each one. For this, we supplement buttons and menus with text and examples that illustrate their proper use (utilizing placeholders in form elements, tooltips, help examples).
- **Avoiding scrunched screen elements.** A second advantage of having fewer items on each screen is that our layout is less burdened by space constraints. For example, we avoid menu drop downs and long scrolling lists and instead, we present lists of selectable autocomplete items (where only related items are visible simultaneously). This makes selection faster and less error prone.
- **Flexible workflow.** In our application, we let users work on different parts of the metadata schema in a sequence that makes sense to them. We also let them refer to supporting data sources in this process. We achieve this by letting users temporarily save their progress (as a "Draft") so they can come back at any time to continue the editing workflow.

Each time a user creates a new item as a first step, they are required to fill in its title. To save users the time and effort required to find out if the item is already created by another user, the frontend application performs a lookup (using the appropriate backend endpoints) and returns matching items (Figure 36).


project name
european lang
Check if the project is already listed in the catalogue

EuroMatrixPlus – Bringing Machine Translation for European Languages to the User Edit

EuroMatrixPlus - Enlarged European Union Bringing Machine Translation for European Languages to the User Edit

European Language Grid

European Language Resource Coordination LOT2

European Language Resource Coordination LOT3

European Language Resource Infrastructure Edit

Integrated European language data Repository Area Edit

Create european lang Cancel

Figure 36: Search for existing items

This initial step serves the following purposes:

- Users are informed of existing matching items and their status (i.e., whether it's published or in the process of being edited)
- For items that are published or in process of being edited by other users, the record is locked (non editable)
- If the matching item is one that the user has initiated at a previous visit, the "Edit" button is visible and the user can proceed to editing it.

After the initial lookup step, the user will be presented with the form for the respective item type. The editor interface is meant to be used by experts, as well as users with less technical knowledge, so our goal is to keep it as simple as possible with straightforward steps so that we can guide each user to filling out the input elements. In order to accomplish the aforementioned goals, we make use of several backend endpoints.

More precisely all the required information such as required fields, help texts and labels are sent to the frontend from the catalogue backend as json responses. All the required fields are displayed with a star near their label. The placeholders are displayed when the user focuses on the elements and the help texts are always visible at the bottom or at the top of each field (Figure 37).

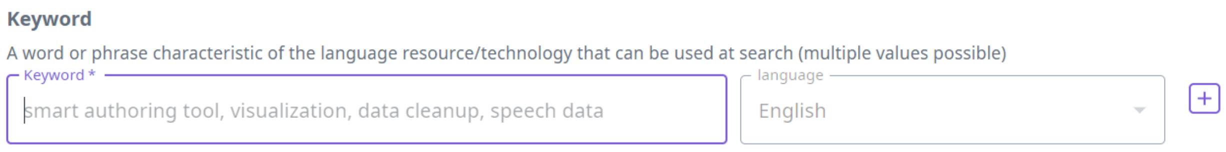

Keyword
A word or phrase characteristic of the language resource/technology that can be used at search (multiple values possible)
Keyword *
smart authoring tool, visualization, data cleanup, speech data
language
English

Figure 37: View of a multilingual element with help tips and examples

Depending on the data type of the elements, different features are used.

A rich editor is included for the "description" metadata element, which allows users to enter paragraphs, links, bullets, and other text formatting techniques in order to make long texts more readable in the final published metadata record.

For the multilingual elements users are forced to fill in the value for English and then they can add the same element for another language by pressing a + button next to the field and selecting the language from the dropdown list (Figure 38).

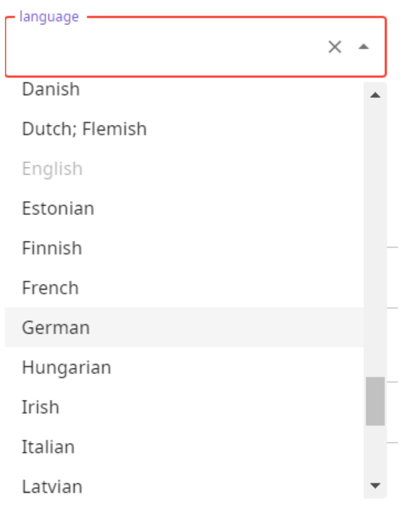


Figure 38: Selecting language for a multilingual element

Other complex elements remain hidden (Figure 39) until the user decides to fill them.

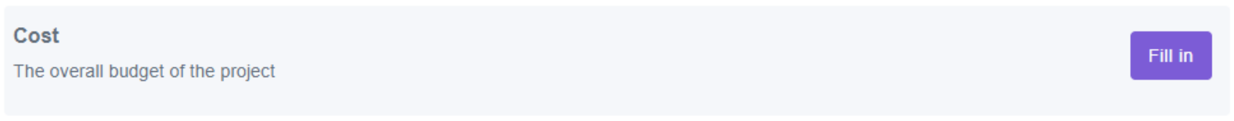


Figure 39: Hidden elements

After a user presses the "Fill in" button, she can add the appropriate values (Figure 40). By clicking "remove", the elements of this group will be emptied.

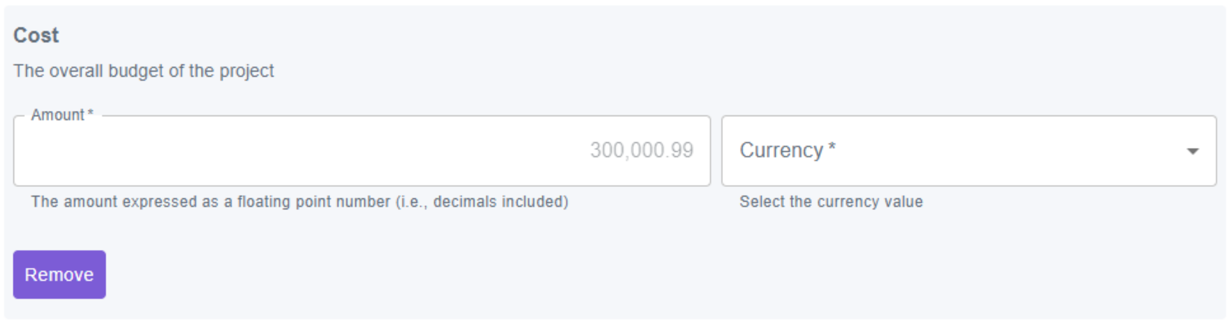

Cost
The overall budget of the project
Amount *
300,000.99
The amount expressed as a floating point number (i.e., decimals included)
Currency *
Select the currency value
Remove

Figure 40: Opening a hidden element in order to add metadata

Moreover, complex editor elements make use of backend lookup endpoints. These endpoints respond back with a list of possible suggestions that the users can choose from (Figure 41). Following this approach, the users have the option of selecting an existing value instead of filling in all the required fields from scratch making the whole process easier and faster.

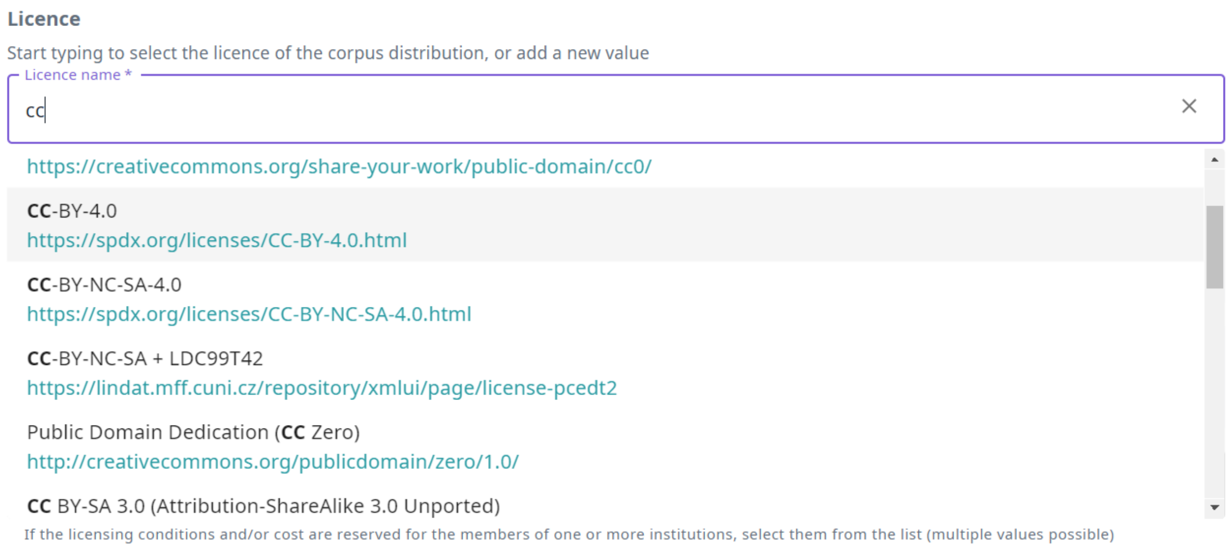
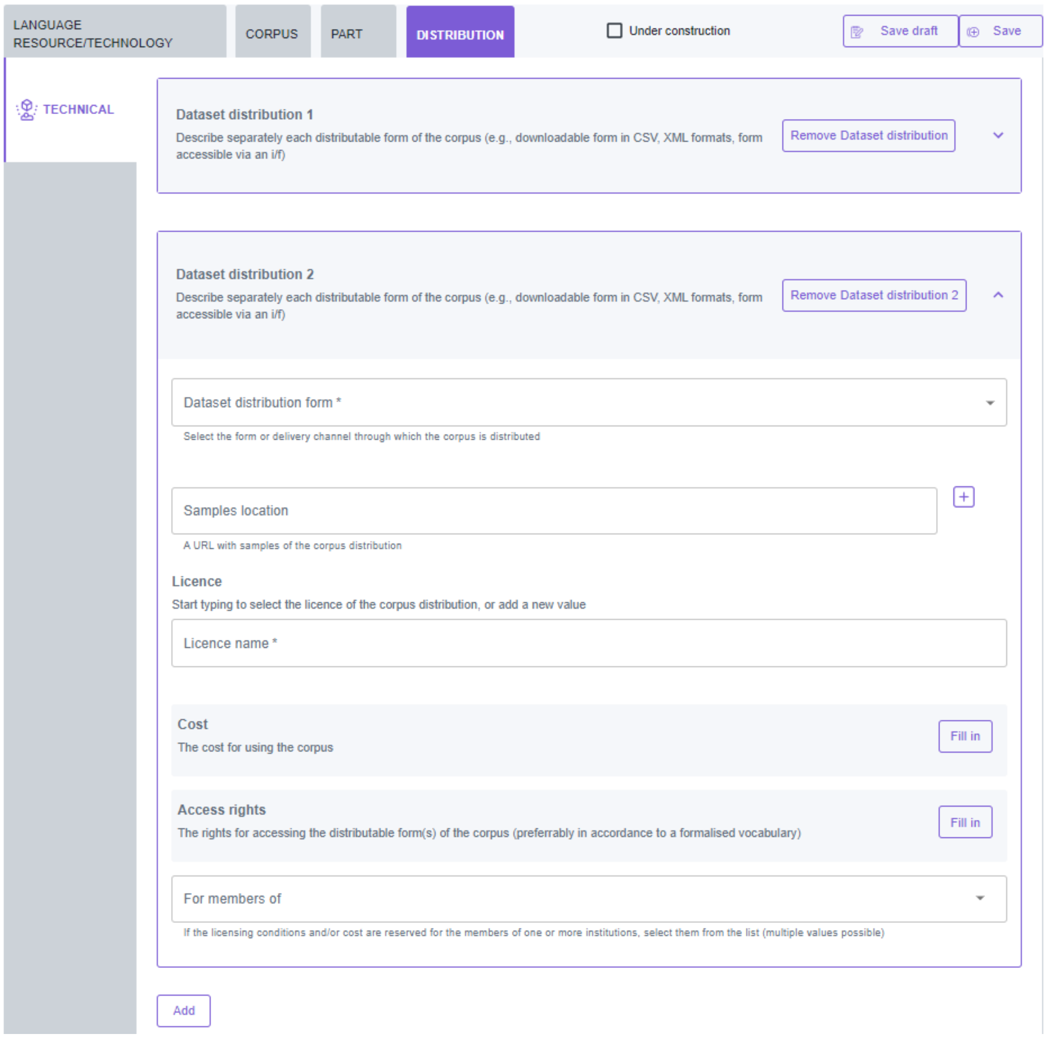

Licence
Start typing to select the licence of the corpus distribution, or add a new value
Licence name *
cc
https://creativecommons.org/share-your-work/public-domain/cc0/
CC-BY-4.0
https://spdx.org/licenses/CC-BY-4.0.html
CC-BY-NC-SA-4.0
https://spdx.org/licenses/CC-BY-NC-SA-4.0.html
CC-BY-NC-SA + LDC99T42
https://lindat.mff.cuni.cz/repository/xmlui/page/license-pcedt2
Public Domain Dedication (CC Zero)
http://creativecommons.org/publicdomain/zero/1.0/
CC BY-SA 3.0 (Attribution-ShareAlike 3.0 Unported)
If the licensing conditions and/or cost are reserved for the members of one or more institutions, select them from the list (multiple values possible)

Figure 41: Lookup field

Finally, when a group of elements gets rather long, we deploy accordions (Figure 42), not only to prevent scrolling as much as possible, but to support the creation flow and allow lightweight editing of an element. For example, for each Corpus users can describe multiple distributions. At any time, users can click the "Add" button and a new Distribution group will be created in a new area. Of course, the individual groups can also be removed.



LANGUAGE RESOURCE/TECHNOLOGY | CORPUS | PART | DISTRIBUTION

☐ Under construction

Save draft | Save

TECHNICAL

Dataset distribution 1
Describe separately each distributable form of the corpus (e.g., downloadable form in CSV, XML formats, form accessible via an i/f)
Remove Dataset distribution

Dataset distribution 2
Describe separately each distributable form of the corpus (e.g., downloadable form in CSV, XML formats, form accessible via an i/f)
Remove Dataset distribution 2

Dataset distribution form *
Select the form or delivery channel through which the corpus is distributed

Samples location
A URL with samples of the corpus distribution

Licence
Start typing to select the licence of the corpus distribution, or add a new value

Licence name *

Cost
The cost for using the corpus
Fill in

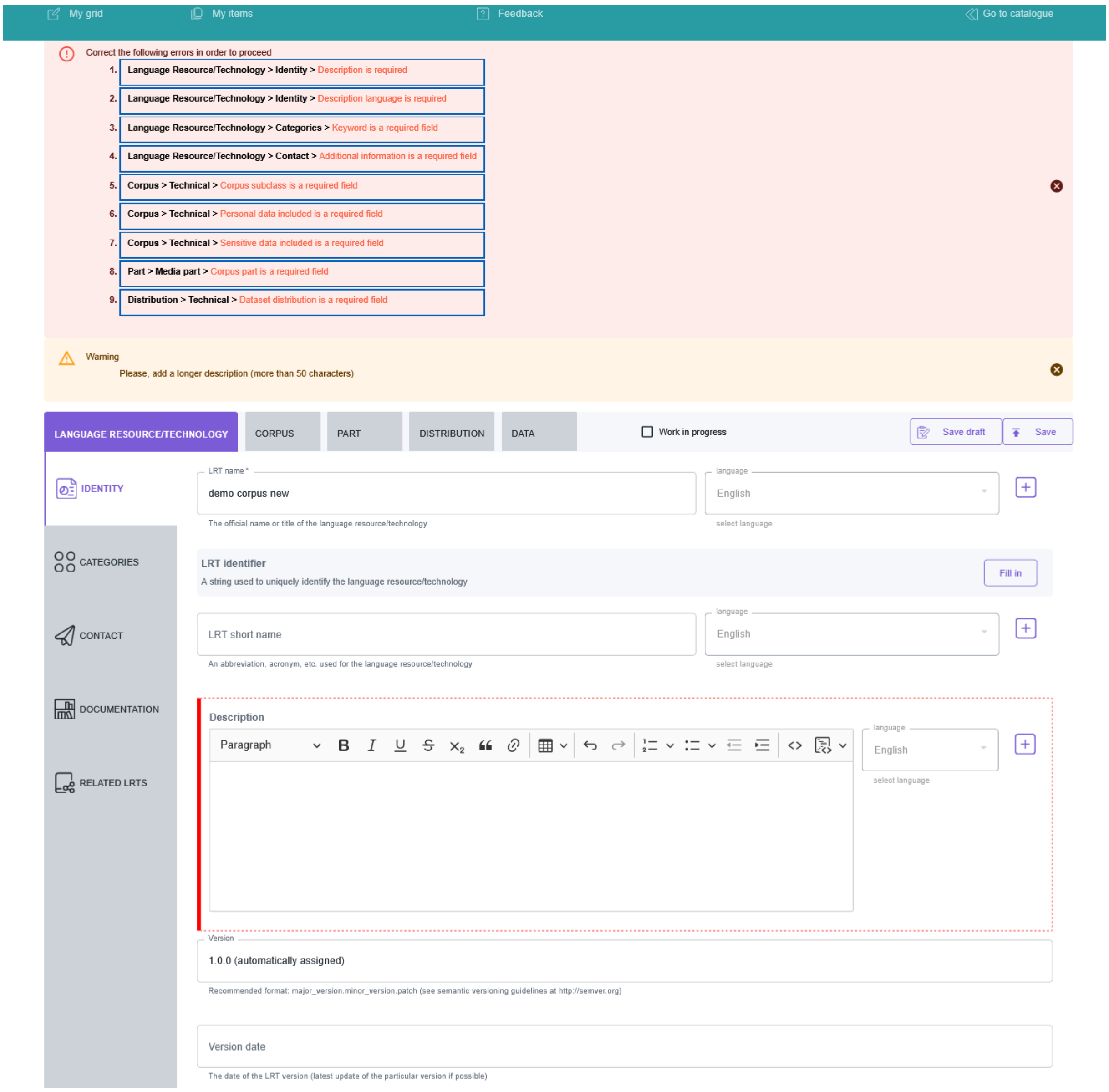
Access rights
The rights for accessing the distributable form(s) of the corpus (preferrably in accordance to a formalised vocabulary)
Fill in

For members of
If the licensing conditions and/or cost are reserved for the members of one or more institutions, select them from the list (multiple values possible)

Add

Figure 42: Use of accordion for long elements

Before the form is submitted to the backend, the client makes a throughout validation using the yup validation library (https://github.com/jquense/yup). By doing this, we make sure that all the required fields are filled in and that the users' input meets the constraints imposed by the schema. In case of errors, appropriate messages are displayed describing the error and the location where this error occurred (Figure 43). By clicking on the validation error, users are redirected to the source location and they can correct it.



Correct the following errors in order to proceed

1. Language Resource/Technology > Identity > Description is required
2. Language Resource/Technology > Identity > Description language is required
3. Language Resource/Technology > Categories > Keyword is a required field
4. Language Resource/Technology > Contact > Additional information is a required field
5. Corpus > Technical > Corpus subclass is a required field
6. Corpus > Technical > Personal data included is a required field
7. Corpus > Technical > Sensitive data included is a required field
8. Part > Media part > Corpus part is a required field
9. Distribution > Technical > Dataset distribution is a required field

Warning
Please, add a longer description (more than 50 characters)

LRT name * : demo corpus new
The official name or title of the language resource/technology
language: English
select language

LRT identifier
A string used to uniquely identify the language resource/technology

LRT short name
An abbreviation, acronym, etc. used for the language resource/technology
language: English
select language

Description
language: English
select language

Version: 1.0.0 (automatically assigned)
Recommended format: major_version.minor_version.patch (see semantic versioning guidelines at http://semver.org)

Version date
The date of the LRT version (latest update of the particular version if possible)

Figure 43: Validation error message

6.2.3.2.3 Uploading metadata records

As an alternative to using the interactive editor, providers can create metadata records in XML format compliant with the ELG metadata schema and upload them to the ELG platform. Before uploading XML files, users can validate them against an XML validator. Both functionalities can be accessed through the grid, as depicted in Figure 44.

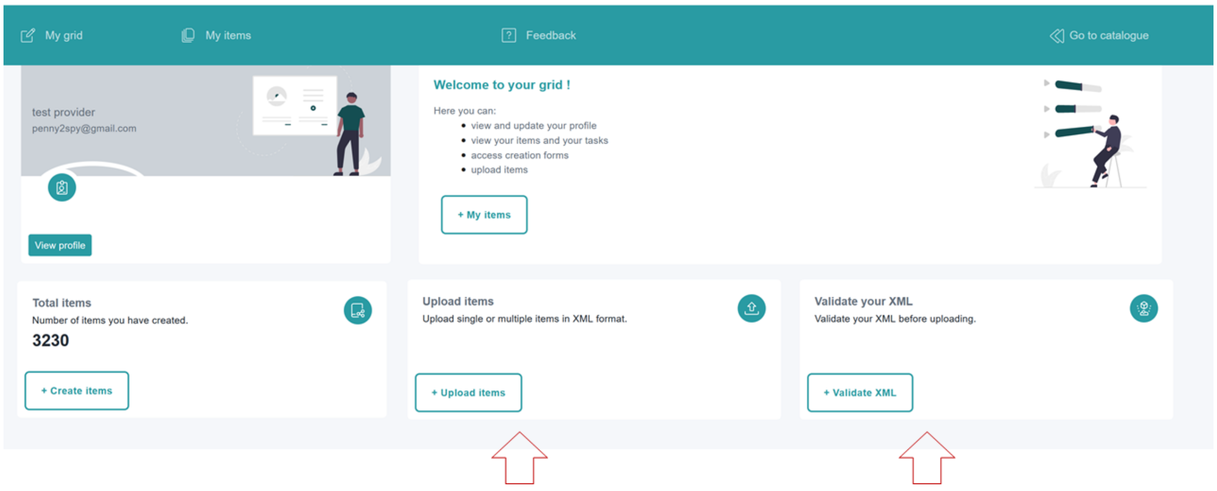
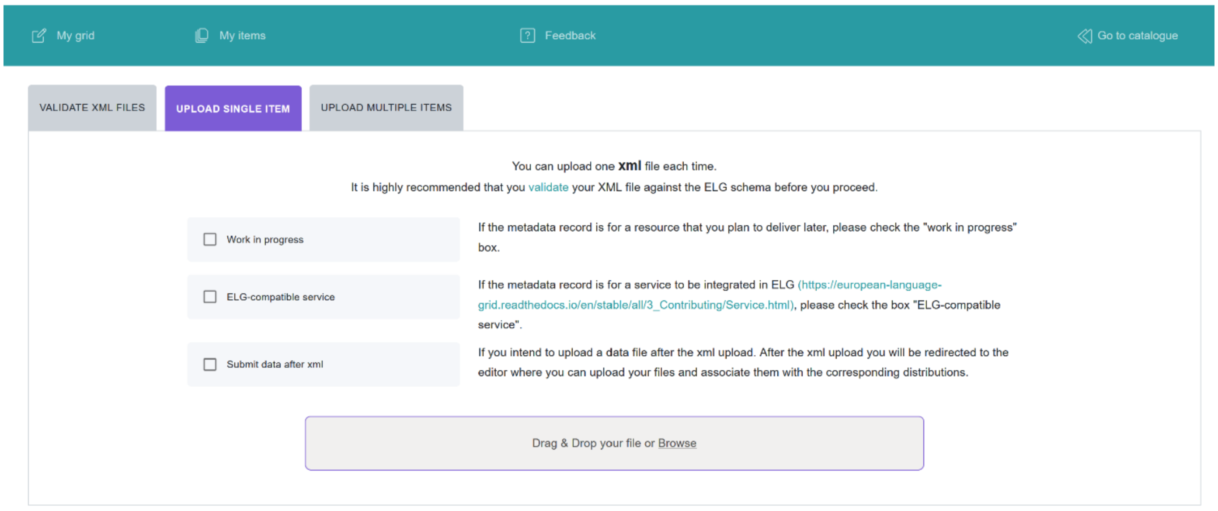


Figure 44: Upload and Validate functionalities

Through interactive processes, users can upload single or batch metadata files directly from their computer (Figure 45) and receive success or failure messages.



You can upload one **xml** file each time.
It is highly recommended that you validate your XML file against the ELG schema before you proceed.

If the metadata record is for a resource that you plan to deliver later, please check the "work in progress" box.

If the metadata record is for a service to be integrated in ELG (https://european-language-grid.readthedocs.io/en/stable/all/3_Contributing/Service.html), please check the box "ELG-compatible service".

If you intend to upload a data file after the xml upload. After the xml upload you will be redirected to the editor where you can upload your files and associate them with the corresponding distributions.

Figure 45: Upload metadata records

When the metadata files have been uploaded, users can view and further edit the records with the interactive editor, and manage them through the "my items" page described in the following subsection.

6.2.3.2.4 My pages and actions for providers

Users can manage the metadata records they have created through a dedicated page, and, in accordance with their user rights, perform the following actions:

- edit or update a metadata record

- submit it for publication
- delete a metadata record
- upload content files to a metadata record, and
- replace the content files that have been uploaded.

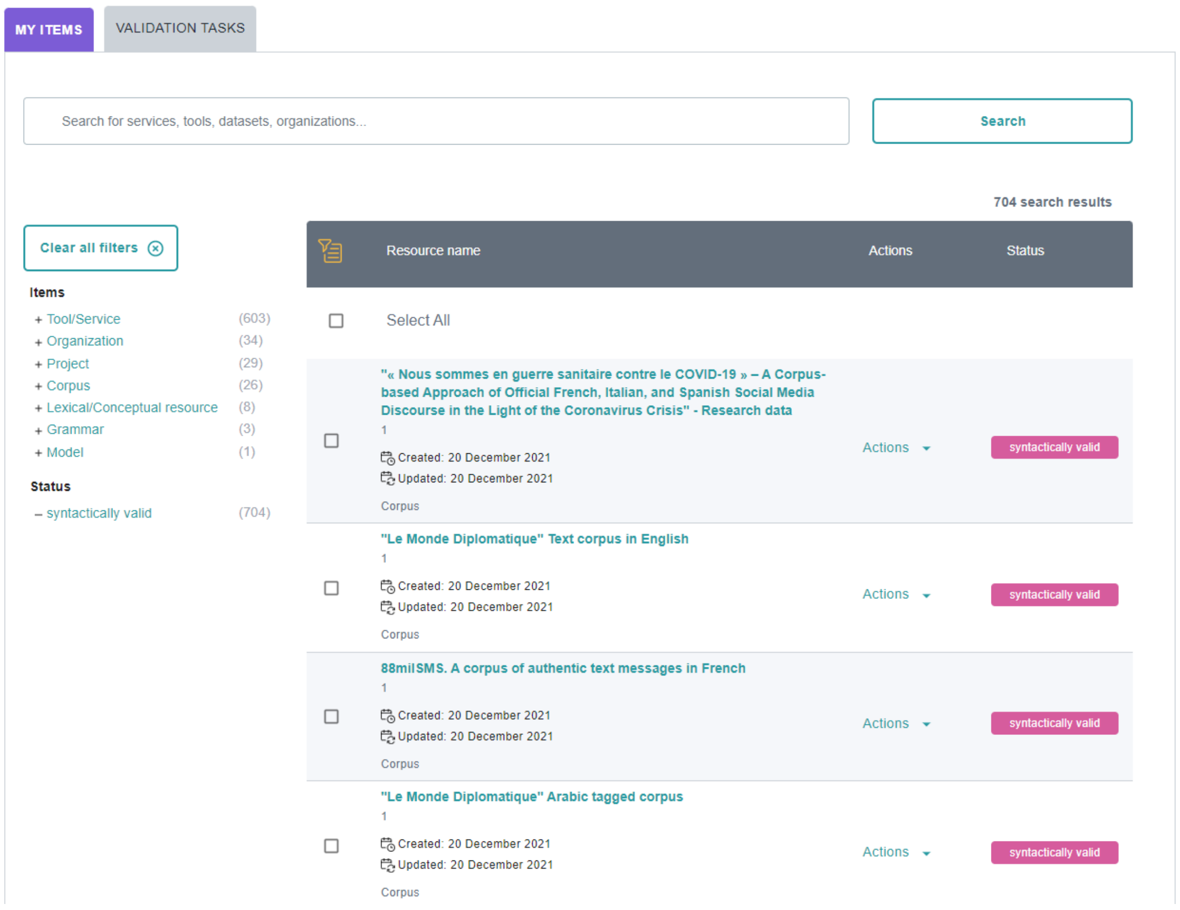
The allowable actions depend also on which state of the ELG publication lifecycle the metadata record is in. The following table shows the actions which can be performed on an item in the various phases of the publication lifecycle.

Status	**Edit**	**Submit for publication**	**Delete**	**Upload content**	**Replace content**
Draft	Y	N	Y	Y	Y
Syntactically valid	Y	Y	Y	Y	Y
Submitted for publication	N	n/a	N	N	N
Published	N	n/a	N	N	N

Table 1: Actions allowed on metadata records by publication status

There are currently two access modes where users can manage items: the "My items" page, from where they can select one or multiple items, and the landing page of each item.

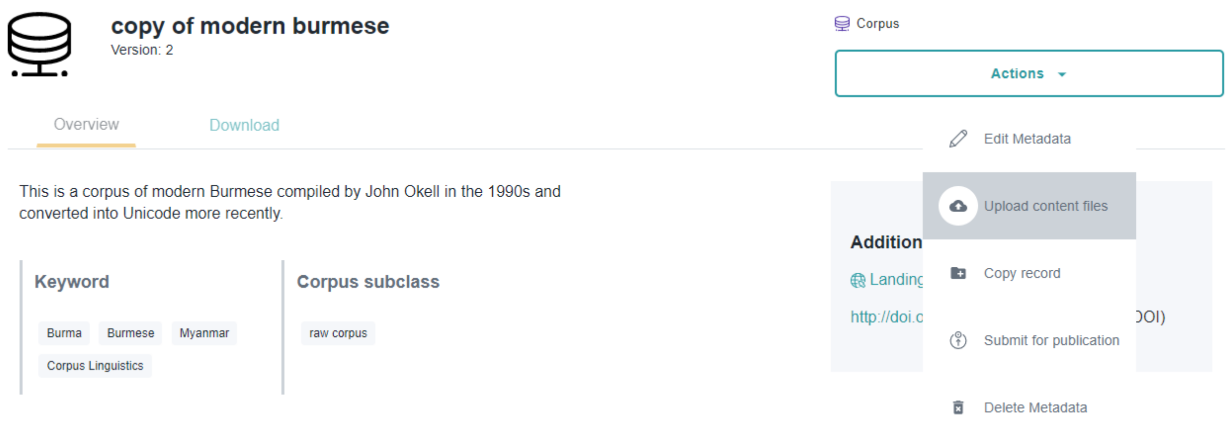
Since the application is frequently used, we provide users with an area for performing common actions. For example, a user has created a Draft metadata record, and would like to continue editing it. Or a user would like to submit an item for publication. Each user is provided with one-click access to her personal "My items" page (Figure 46) where she can perform single or bulk actions on the items she has created in the past.



Resource name	Actions	Status
"« Nous sommes en guerre sanitaire contre le COVID-19 » – A Corpus-based Approach of Official French, Italian, and Spanish Social Media Discourse in the Light of the Coronavirus Crisis" - Research data 1 Created: 20 December 2021 Updated: 20 December 2021 Corpus	Actions	syntactically valid
"Le Monde Diplomatique" Text corpus in English 1 Created: 20 December 2021 Updated: 20 December 2021 Corpus	Actions	syntactically valid
88milSMS. A corpus of authentic text messages in French 1 Created: 20 December 2021 Updated: 20 December 2021 Corpus	Actions	syntactically valid
"Le Monde Diplomatique" Arabic tagged corpus 1 Created: 20 December 2021 Updated: 20 December 2021 Corpus	Actions	syntactically valid

Figure 46: My items page

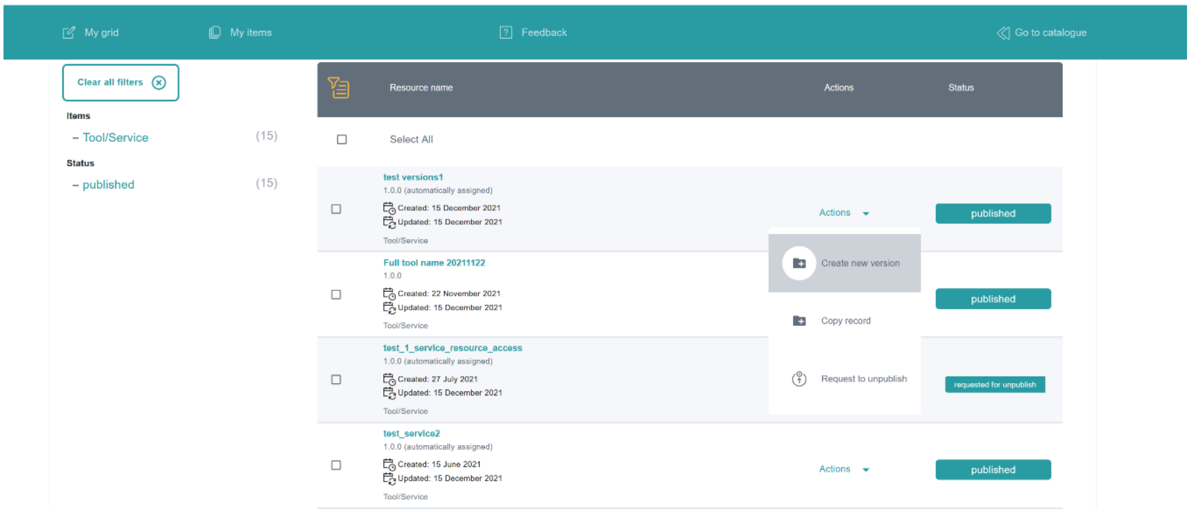
At the "My items" page, providers can perform actions on a single item or select multiple items and perform bulk actions, as described in Section 6.2.3. The same Actions can also be performed on the landing page of an item (Figure 47).



This is a corpus of modern Burmese compiled by John Okell in the 1990s and converted into Unicode more recently.

Figure 47: Actions on the landing page of a record

R3 has introduced a new functionality to facilitate the addition of versions for existing items, given that they share similar metadata. Users can create new versions of items they have already registered in the ELG catalogue through an action on the existing record (Figure 48). This new action will create a copy of the existing record with all the metadata that contained in the old version, allowing the user at the same time to edit the information required for the new version and automatically create a relation with the previous version.



test versions1
1.0.0 (automatically assigned)
Created: 15 December 2021
Updated: 15 December 2021
Tool/Service

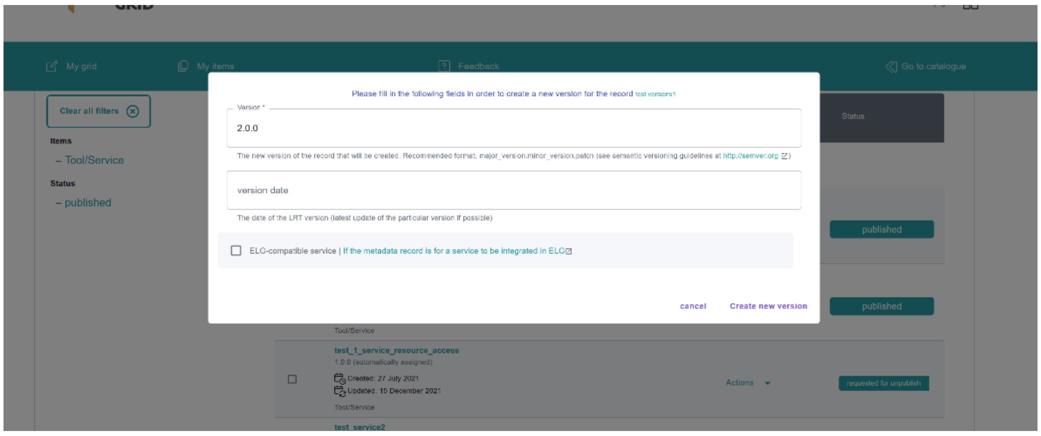
Full tool name 20211122
1.0.0
Created: 22 November 2021
Updated: 15 December 2021
Tool/Service

test_1_service_resource_access
1.0.0 (automatically assigned)
Created: 27 July 2021
Updated: 15 December 2021
Tool/Service

test_service2
1.0.0 (automatically assigned)
Created: 15 June 2021
Updated: 15 December 2021
Tool/Service

Figure 48: Action for creating new versions

After the user selects to create a new version, she is asked for the version number in the field that appears and, optionally, a version date (Figure 49). For organizations and projects, she will be prompted to add a new name. In addition, for tools/services, she will be asked whether it's an ELG-compatible service.



Please fill in the following fields in order to create a new version for the record test versions1

Version *
2.0.0

The new version of the record that will be created. Recommended format: major_version.minor_version.patch (see semantic versioning guidelines at http://semver.org)

version date

The date of the LRT version (latest update of the particular version if possible)

☐ ELG-compatible service | If the metadata record is for a service to be integrated in ELG

Tool/Service

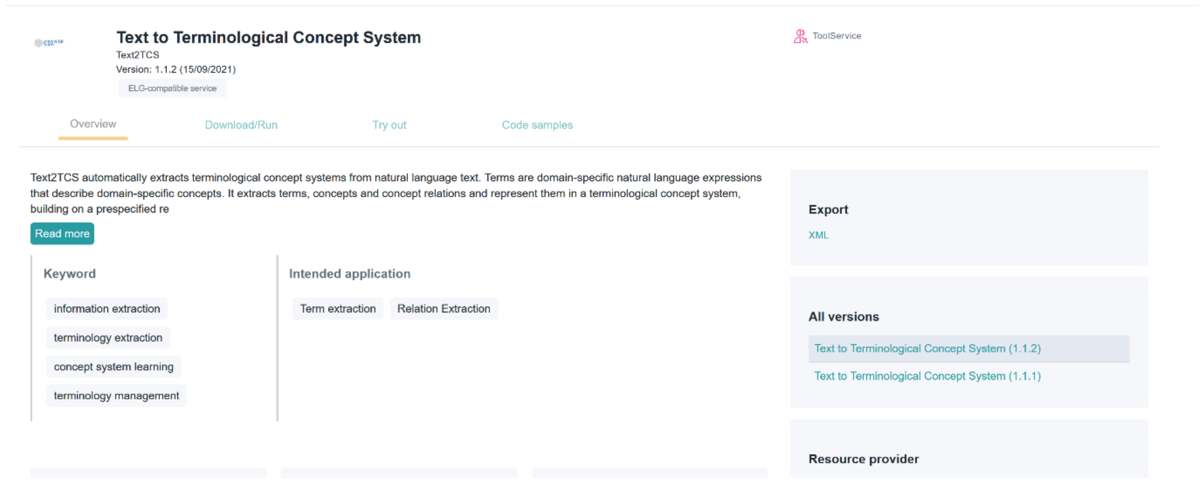
test_1_service_resource_access
1.0.0 (automatically assigned)
Created: 27 July 2021
Updated: 15 December 2021
Tool/Service

test_service2

Figure 49: Dialogue for the creation of a new version

When the new item is created, the user is redirected to its landing page, and can then proceed to edit it and submit for publication, following the publication lifecycle for all LRTs. If the user had uploaded content files for the previous version, these are not copied and the user can upload new files.

Finally, on the landing page of each version, links to previous/new versions are also clearly marked (Figure 50).



Text to Terminological Concept System
Text2TCS
Version: 1.1.2 (15/09/2021)
ELG-compatible service

ToolService

Overview | Download/Run | Try out | Code samples

Text2TCS automatically extracts terminological concept systems from natural language text. Terms are domain-specific natural language expressions that describe domain-specific concepts. It extracts terms, concepts and concept relations and represent them in a terminological concept system, building on a prespecified re

Read more

Keyword
- information extraction
- terminology extraction
- concept system learning
- terminology management

Intended application
- Term extraction
- Relation Extraction

Export
XML

All versions
- Text to Terminological Concept System (1.1.2)
- Text to Terminological Concept System (1.1.1)

Resource provider

Figure 50: Landing page for a resource with multiple versions

6.2.3.3 Validator's grid and functionalities

Validators have access to all the items they have been assigned to them through a dedicated catalogue page called "My validation tasks" (or, in short, "my validations"). Through the landing page of each item, they gain access to dedicated forms where they can perform the metadata, technical and legal validation of records as well as, for ELG-compatible services, the service registration.

The "My validations" page (Figure 51) is accessible only to technical, metadata and legal validators. It implements browse and search functionalities like the main catalogue page, making it easier for validators to access their tasks, have a quick overview of the records being validated, while overall the purpose is to save valuable time from the validators and help them perform their tasks efficiently.

The following table shows the validation operations foreseen for each item type / source of metadata.

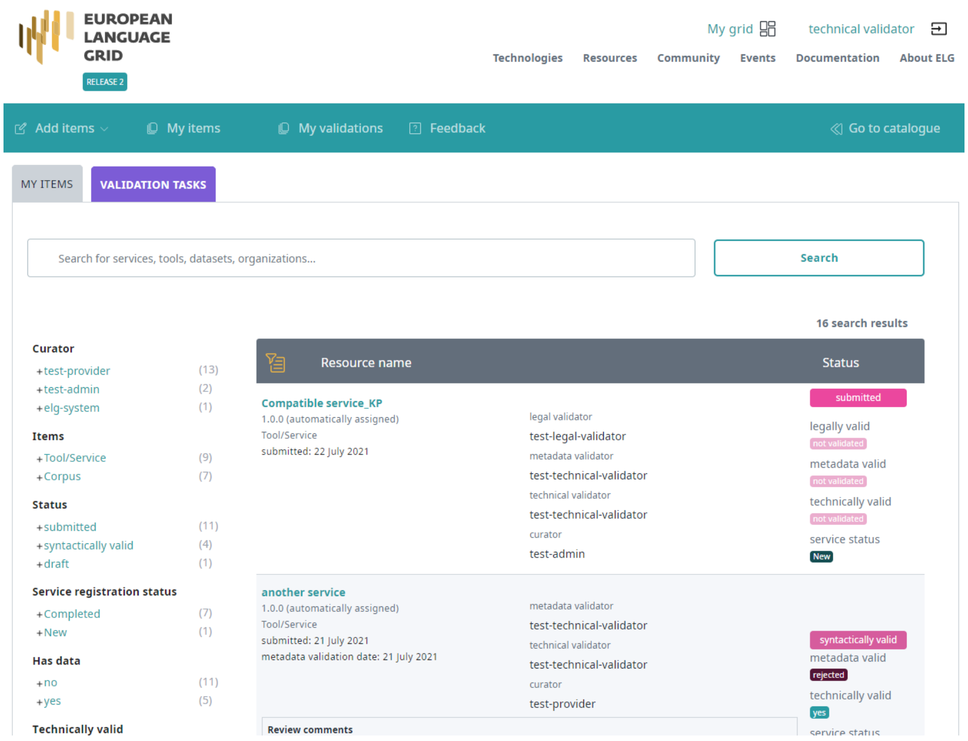
Type of items	**Validation types**		
	Metadata	Technical	Legal
Harvested metadata	N/A	N/A	N/A
Metadata only records	Yes	N/A	N/A
ELG compatible services	Yes	Yes	Yes
LRTs uploaded (hosted) in ELG	Yes	Yes	Yes

Table 2: Validation types by item type

As seen in Figure 51, each item occupies a row separated in three columns:

- the first one provides some basic information on the item (name, version, item type, submission date),
- the second column presents the names of the curator and validators, and
- the third column shows its status in the publication lifecycle and the validation status (i.e. whether it has been validated and approved or rejected).

In addition, if the item has been validated, there is a box with the validator notes (for internal purposes only) and, if rejected, the review comments.


EUROPEAN LANGUAGE GRID
RELEASE 2

My grid | technical validator

Technologies | Resources | Community | Events | Documentation | About ELG

Add items | My items | My validations | Feedback | Go to catalogue

MY ITEMS | VALIDATION TASKS

Search for services, tools, datasets, organizations... | Search

16 search results

Curator
+test-provider (13)
+test-admin (2)
+elg-system (1)

Items
+Tool/Service (9)
+Corpus (7)

Status
+submitted (11)
+syntactically valid (4)
+draft (1)

Service registration status
+Completed (7)
+New (1)

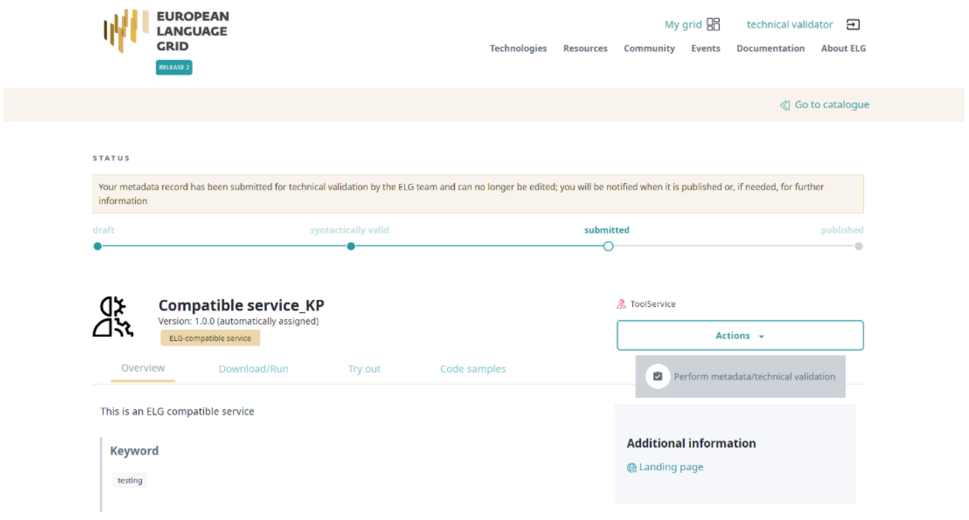
Has data
+no (11)
+yes (5)

Technically valid

Resource name		Status
Compatible service_KP 1.0.0 (automatically assigned) Tool/Service submitted: 22 July 2021	legal validator test-legal-validator metadata validator test-technical-validator technical validator test-technical-validator curator test-admin	submitted legally valid not validated metadata valid not validated technically valid not validated service status New
another service 1.0.0 (automatically assigned) Tool/Service submitted: 21 July 2021 metadata validation date: 21 July 2021 Review comments	metadata validator test-technical-validator technical validator test-technical-validator curator test-provider	syntactically valid metadata valid rejected technically valid yes service status

Figure 51: My validation tasks

When a validator clicks on an item, she is redirected to its landing page where she is able to perform the validation tasks through the appropriate form (Figure 52). The allowed actions depend on each validator's role.


EUROPEAN LANGUAGE GRID
RELEASE 2

My grid | technical validator

Technologies | Resources | Community | Events | Documentation | About ELG

Go to catalogue

STATUS

Your metadata record has been submitted for technical validation by the ELG team and can no longer be edited; you will be notified when it is published or, if needed, for further information

draft | syntactically valid | submitted | published

Compatible service_KP
Version: 1.0.0 (automatically assigned)
ELG-compatible service

ToolService

Actions
Perform metadata/technical validation

Overview | Download/Run | Try out | Code samples

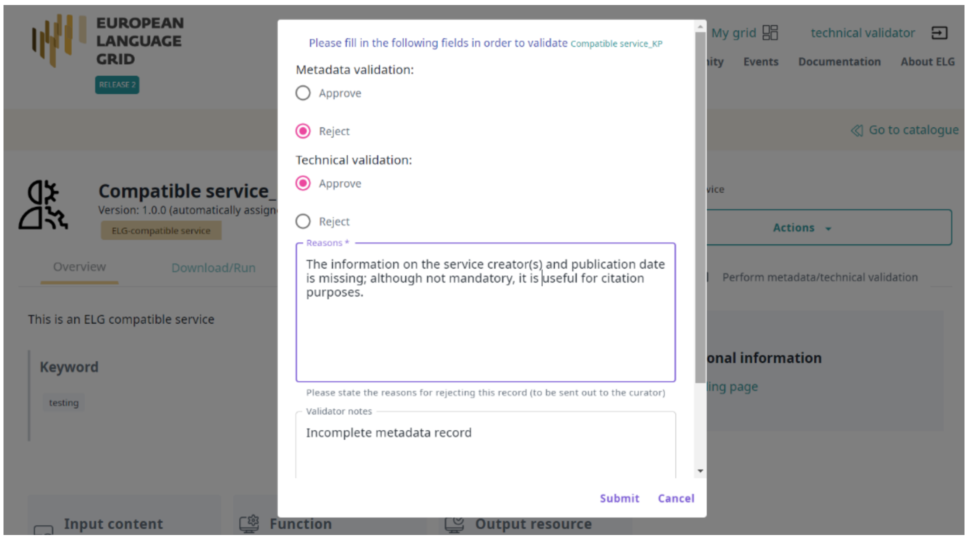
This is an ELG compatible service

Keyword
testing

Additional information
Landing page

Figure 52: Actions for metadata validation

For the metadata validation, a validator is asked to check whether the values of various elements are included for the item and whether their values match the description of the service.



Please fill in the following fields in order to validate Compatible service_KP

Metadata validation:
Approve [✓ Reject]

Technical validation:
[✓ Approve] Reject

Reasons *
The information on the service creator(s) and publication date is missing; although not mandatory, it is useful for citation purposes.
Please state the reasons for rejecting this record (to be sent out to the curator)

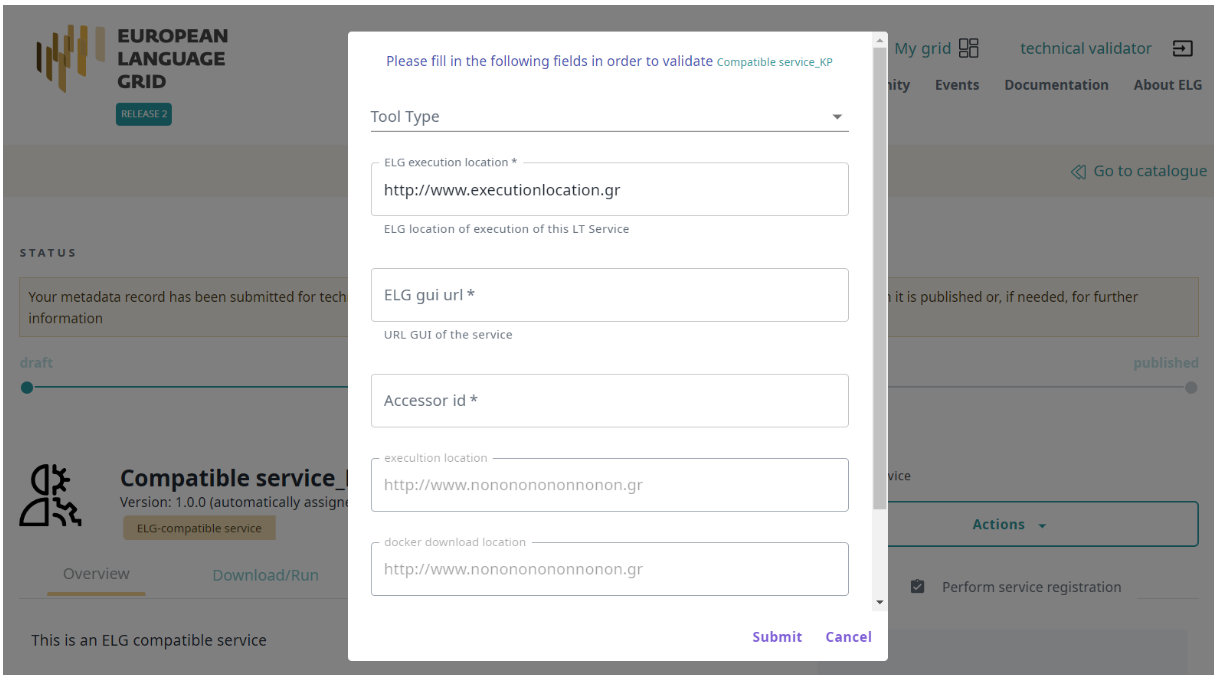
Validator notes
Incomplete metadata record

Submit Cancel

Figure 53: Validation form for technical validators

For ELG-compatible services and ELG-hosted data, technical and metadata validation is performed by the same individual using the same form (see Figure 53). If the validators are satisfied, approval of both types of validation is made and the Submit button should be pressed. If not, then the option to Reject the Technical validation and/or the Metadata validation (depending on the source of the issue) is allowed. This will generate a new field where the validator can write the recommendations they would like to share with the metadata creator. They can also add comments in the "Validator notes" field which will be visible only to her and other validators.

For ELG-compatible services, the technical/metadata validator performs also the service registration through a dedicated form (Figure 54) which is similar to the validation forms. The service registration procedure is described in detail in D2.6.



Please fill in the following fields in order to validate Compatible service_KP

Tool Type

ELG execution location *
http://www.executionlocation.gr
ELG location of execution of this LT Service

ELG gui url *
URL GUI of the service

Accessor id *

execultion location
http://www.nonononononnonon.gr

docker download location
http://www.nonononononnonon.gr

Submit Cancel

Figure 54: Service registration form
Authentication Solution

6.3 Technical Description

Keycloak implementation have not changed much since the initial release. See the technical description in Deliverable 3.2.

Since Release 1 the following main extensions have been made:

1. Custom themes were added for login, account and emails;
2. We developed a custom module for synchronization with the Catalogue backend. This is described in Deliverable D2.5;
3. Global roles were introduced, migrating from application-specific ones.

6.4 Authentication GUI customization

In order to customize Keycloak end user interface elements, the Keycloak provides custom theme creation. As recommended in the Keycloak documentation, we extended the built-in theme, by adding a new themes and setting this built-in theme as a base. We added 3 new themes:

1. Login theme – theme for different login screens, password recovery, account linking, account registration, accepting terms, offline login and similar.
2. Account theme – theme for administering existing users' accounts, password change, session lists.
3. Email theme – for adjusting email texts and email designs sent by the Keycloak system.

Custom themes are built and packed into a Docker image and published. When Keycloak initializes, it downloads the image with custom themes and maps to the Keycloak filesystem. Custom themes are set in the top ELG realm level, so all design changes apply to all components of the system – central portal, catalogue, python clients, etc.

Customized login screen (Figure 55): the ELG platform still allows registration with locally administered accounts only, however, if needed, logins can also be performed with external identity providers e.g. Google. In the case that the user is not yet registered, they can go to the registration page or reset the password if they have forgotten it.

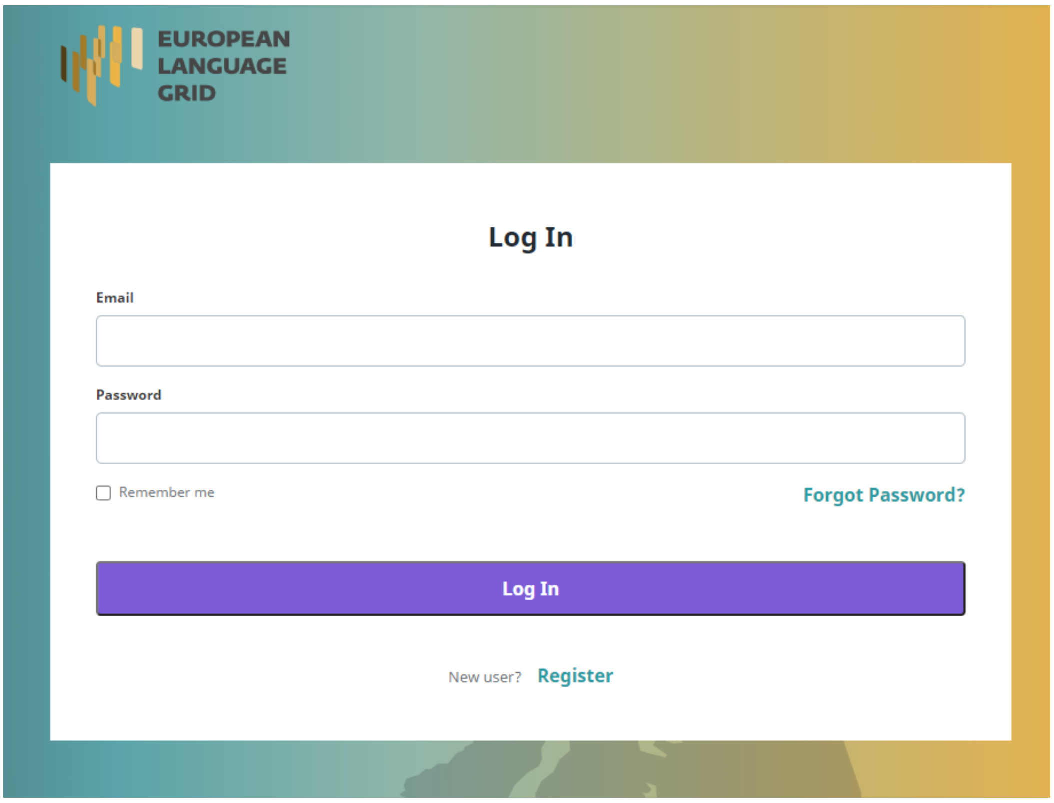


Figure 55: Login form

Registration form (Figure 56): a self-registration form is now available in the live environment. The default user registration data model was extended and new fields added. During registration, not all fields are visible to the end-user; some fields will be visible later in the account editing form.

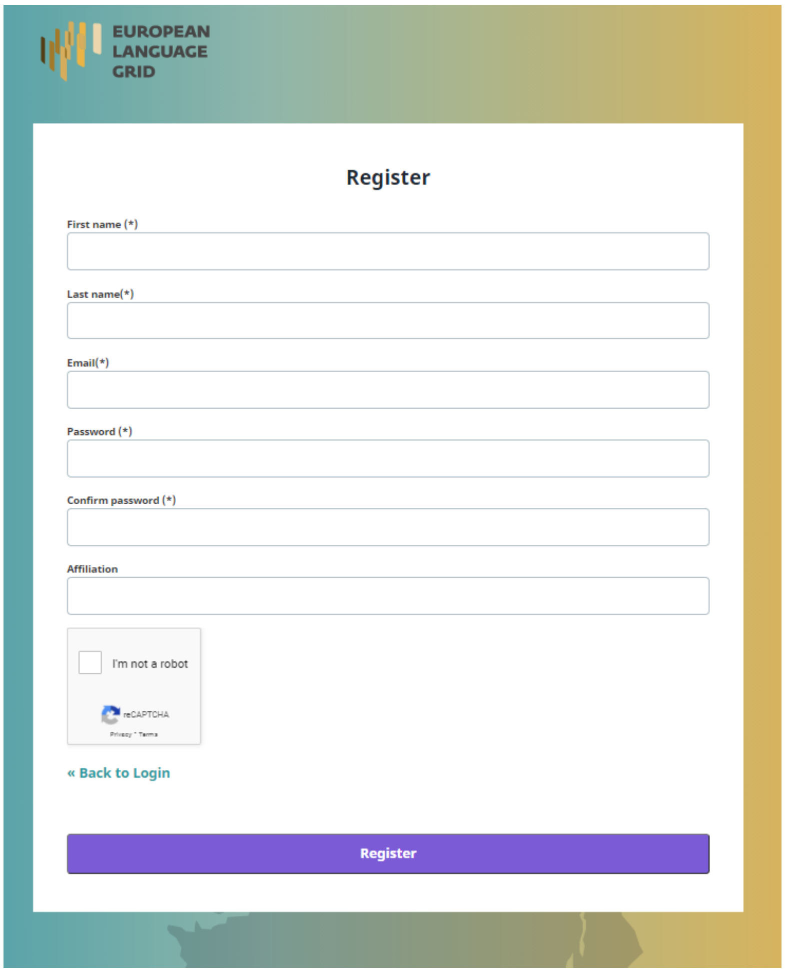
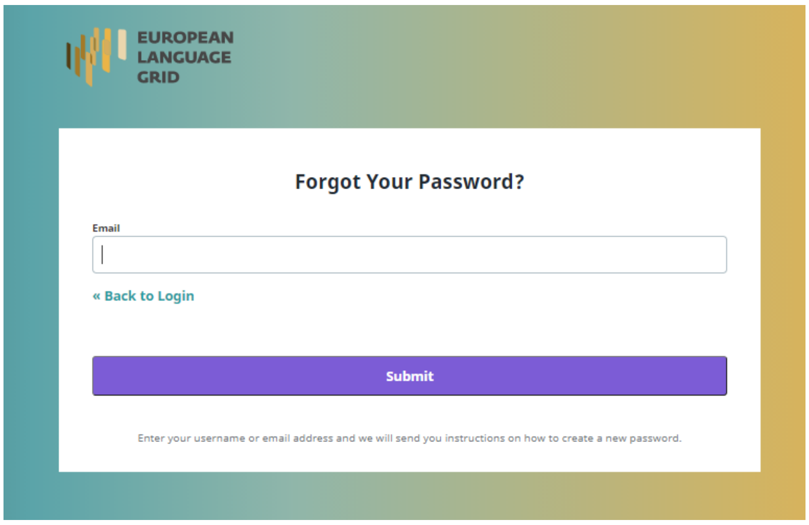


Figure 56: New user registration form

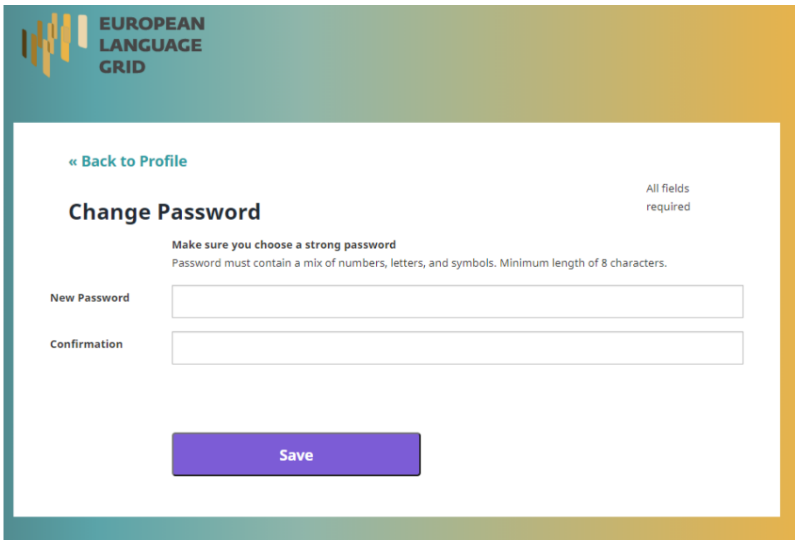
Password recovery form (Figure 57): in the case that the user forgets their credentials, they can restore their password by providing the e-mail address that was registered during sign-up. Then, the user will receive an e-mail from Keycloak to reset the password. This email message is also adjusted to fit the ELG design guidelines.



Enter your username or email address and we will send you instructions on how to create a new password.

Figure 57: Password recovery form

Password change form was adjusted according to the design guidelines and more text messages were added.


EUROPEAN LANGUAGE GRID

« Back to Profile

All fields required

Change Password

Make sure you choose a strong password
Password must contain a mix of numbers, letters, and symbols. Minimum length of 8 characters.

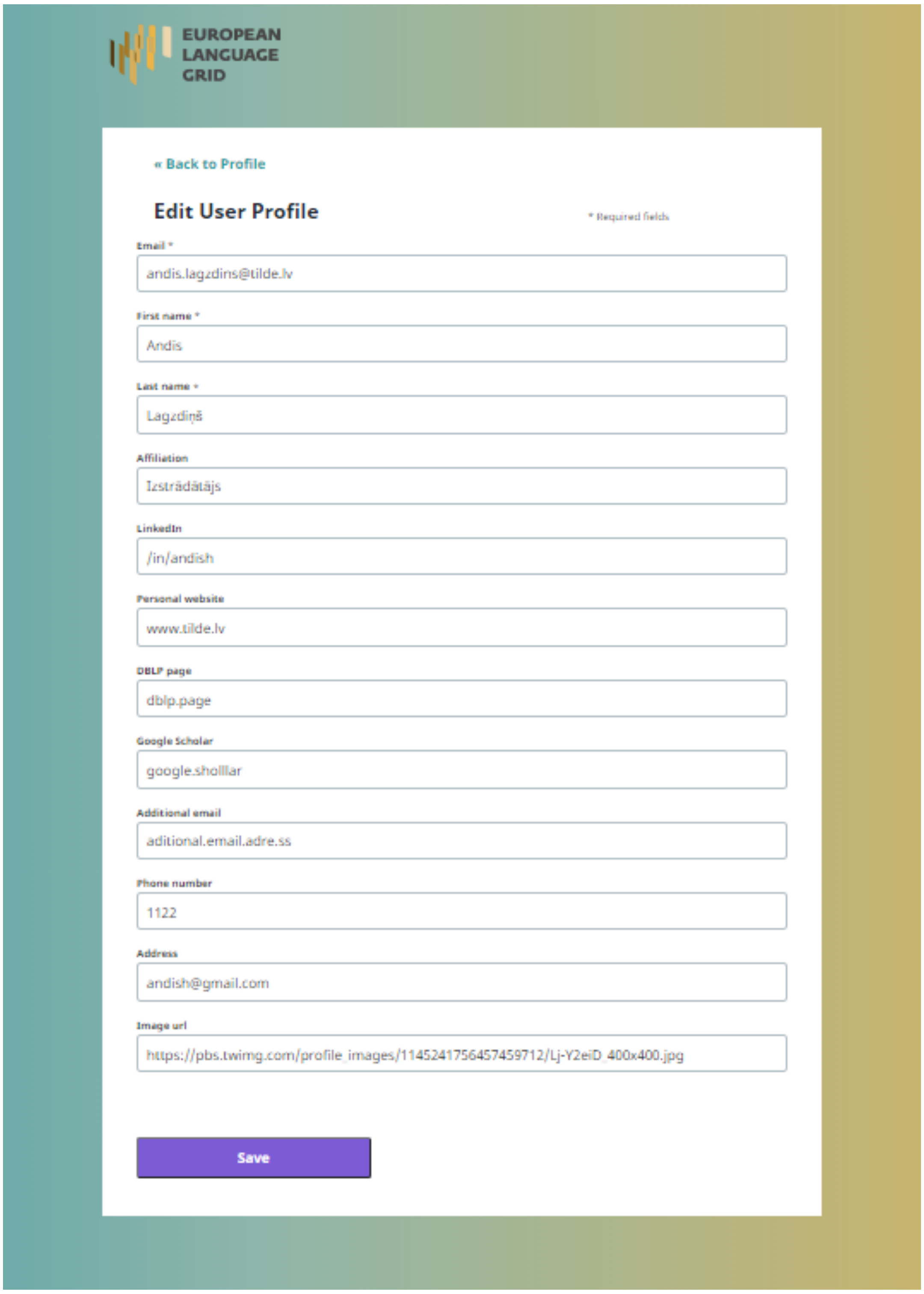
New Password

Confirmation

Save

Figure 58: Styled form for password change

In the R3 version of the ELG platform, the user account editing form is enabled. Here, the user can edit all data categories defined by the extended user data model, the user can also set his profile image.



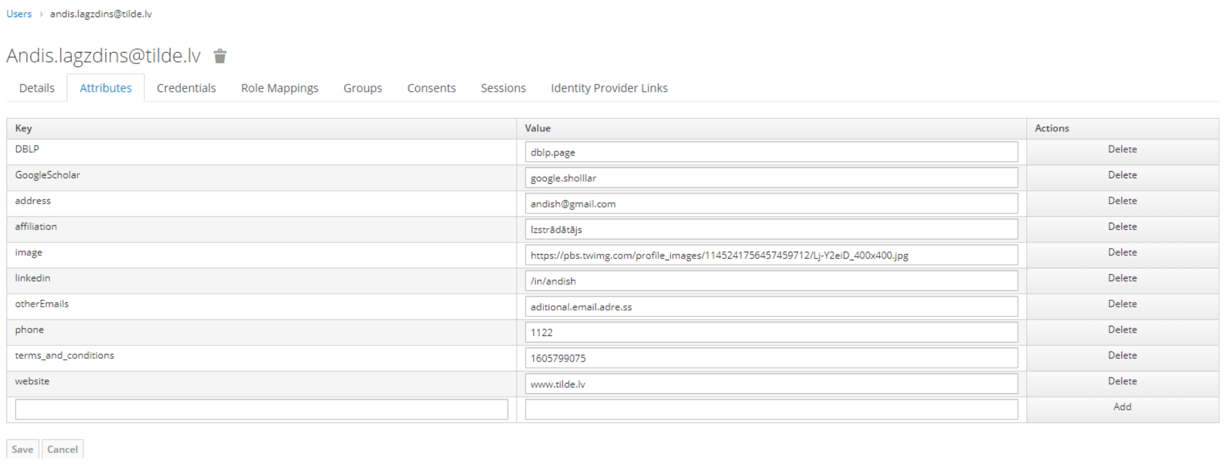
Edit User Profile

* Required fields

Email *	andis.lagzdins@tilde.lv
First name *	Andis
Last name *	Lagzdiņš
Affiliation	Izstrādātājs
LinkedIn	/in/andish
Personal website	www.tilde.lv
DBLP page	dblp.page
Google Scholar	google.sholllar
Additional email	aditional.email.adre.ss
Phone number	1122
Address	andish@gmail.com
Image url	https://pbs.twimg.com/profile_images/1145241756457459712/Lj-Y2eiD_400x400.jpg

Figure 59: User account edit form

Once a user has submitted their data, it is saved in the Keycloak database as attributes. Later, these attributes are mapped to data categories in the authorization token, and thus data included there are to be used in every platform's component in a decentralized manner. The Keycloak database is hosted by ELG infrastructure.


Users › andis.lagzdins@tilde.lv

Andis.lagzdins@tilde.lv

Key	Value	Actions
DBLP	dblp.page	Delete
GoogleScholar	google.sholllar	Delete
address	andish@gmail.com	Delete
affiliation	Izstrādātājs	Delete
image	https://pbs.twimg.com/profile_images/1145241756457459712/Lj-Y2eiD_400x400.jpg	Delete
linkedin	/in/andish	Delete
otherEmails	aditional.email.adre.ss	Delete
phone	1122	Delete
terms_and_conditions	1605799075	Delete
website	www.tilde.lv	Delete
		Add

Figure 60: User submitted attributes in Keycloak Management Console

7 Billing Solution

Service and data commercialization is very important for every sustainable solution thus requirements must be analysed from many different aspects – service and data providers, service and data consumers, EU and national laws, platform and existing workflow restrictions. As well, to ensure the most up-to-date user experience with purchase and automatic subscription process, it involves many implementation details such as pricing and conformance with legislation. The following options were considered:

- Create the functionality from scratch.
- Take an open-source solution, adjust and run it on ELG infrastructure.
- Choose an existing online solution to integrate with ELG.

After investigating the pros and cons, an existing online solution was chosen to proceed with. Creating functionality from scratch needed too much effort. After trying out some open-source solutions and estimating the amount of the work needed to create a user-friendly solution that follows EU and national laws, we decided to go with an existing online billing solution.

Following the decision to select an online solution we tried out selected online solutions and finally choose Chargebee[11] for integration with ELG. After the negotiations with Chargebee, we migrated our account to the EU data centre and got two environments – a test environment and the live environment. When the platform will start to generate revenue, we will need to move to a subscription plan and real payment providers like Paypal, credit-cards and debit payments.

7.1 Requirement analysis and market research

From the commercialization perspective we listed the following products:

- One time purchase without ELG computing (execution in ELG infrastructure)
- Recurring without ELG Computing

[11] https://www.chargebee.com

- One time purchase prepaid with ELG computing
- Recurring with ELG computing, prepaid

It is important for ELG to support all billing features that conform to applicable laws and regulations, to ensure that the paying customer is happy and to provide all back-office operations. The following is a list of key features of the billing software that should be included in the ELG billing module:

- Pricing
- Reporting & Analytics
- Billing and Invoicing
- Usage-based-Billing
- Trial Analytics
- Subscription Management
- Fraud Prevention
- Customer Data management
- Prepaid/Post-paid Billing
- Payment Processing
- Costumer Self-service
- Recurring, metered (price according the amount used), with ELG computing, after-paid

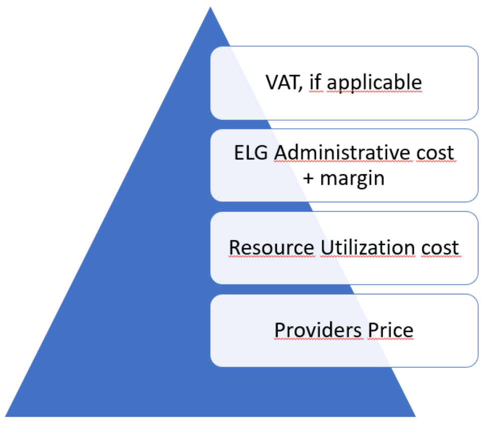


Figure 61: Price Calculation Components

Pricing for different services and products is quite complex and is dependent on various parameters. At first, the resource or service provider can have their expectation of the value the Provider wants to receive for its services or data. In case the service is hosted in the ELG environment, hosting and running expenses for that service in ELG cloud must be covered. Some services have only proxies in the ELG environment and the actual service load is transferred outside the ELG infrastructure while other services fully exploit the ELG environment. The final price must also include ELG's administrative costs and some margin, if planned. Finally, VAT must be applied in certain situations according to the regulations and national laws. VAT rates differ by the purchaser's home country.

The pricing must be quite flexible to support a very wide range of ELG products, price calculation options that must be taken in account are:

- Several price calculation parameters:
- Number of users, number of clients, number of employees
- By features – languages, datasets, methods etc
- Amount-based:
- Time based – hours of processing
- Text based – words/symbols/tokens etc.
- Session-based – e.g., count of chatbot session
- Time-base subscriptions – unlimited amount
- Hybrid – amount-based plus time-based subscription, one-time price*amount based, etc.
- One-time purchases – for example, ad-hock service, consultation, integration
- Bundling – per-product-bundle, service-pack pay, workflows, language-packs, service + data model, etc.

The billing module also requires to provide great user experience and flexibility with an extensive range of functions:

- Usability and user-friendly product management
- GUI customization
- Upselling triggers – have done this, you might want to buy this?
- Pending changes in plans (i.e., plans that change with next billing cycle)
- Multi-lingual support (customer portal, e-mail templates, other customer facing GUIs)
- Mobile friendly user interface
- Reporting and Business Intelligence

Compliance with different legal and tax jurisdictions is mandatory for the billing module. It includes verification of clients (VAT number must be verified in EU official VAT databases) reverse charge, several VAT rates must be applied, different client types must be supported (company vs. private person). The module must be compliant with the GDPR and other EU regulations, must ensure payment cancellation (refunds), handle pending cancelations and bad debit invoice cancellation.

After researching the current status in the billing platform market, we realized that there are hundreds of commercial platforms, less variety for open source, but still quite many. Usually, these were complicated systems as different payment and billing options complicates them. Based on the features they provide and existing reviews from other users, we created a shortlist of platforms to try out. We investigated the following solutions:

- Commercial Platforms – Salesforce[12], Chargebee, Zuora[13], Zoho[14], The Younium Subscription Stack[15]

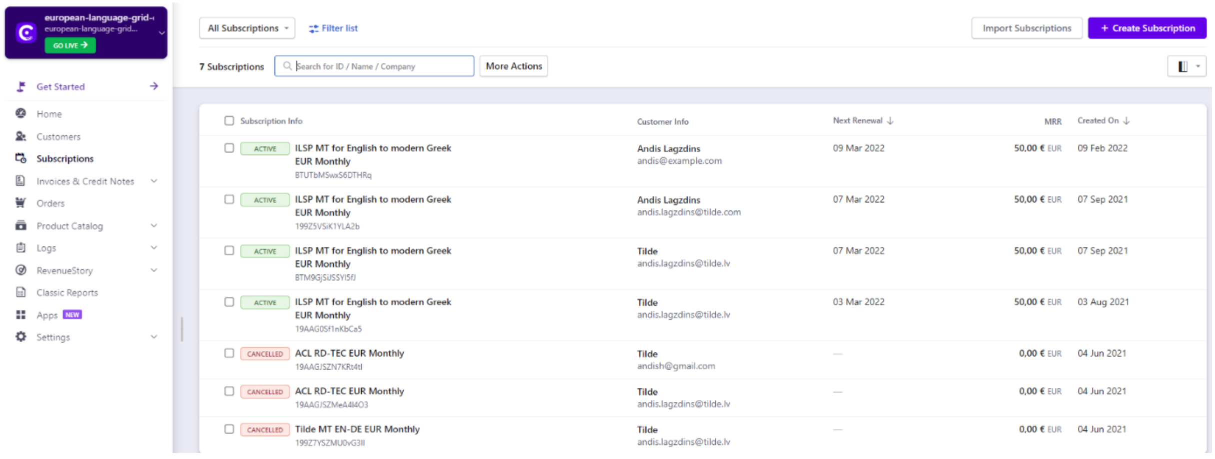
[12] https://www.salesforce.com
[13] https://www.zuora.com
[14] https://www.zoho.com
[15] https://www.younium.com

- Open source – Kill bill[16], Invoice Plane[17], Box billing[18], SolidInvoice[19], Invoice Neko[20], Bamboo Invoice[21], jBilling[22], Logic Invoice[23]

7.2 Billing Functionality Overview

Chargebee has great functionality and it is available at the Chargebee live documentation site[24]. The following functionality is essential for the ELG and was a driving factor for selecting Chargebee as the billing solution:

- Product Families and Plans – possibility to group different plans by product category is very useful as ELG have many services and datasets.
- Pricing – monthly price and annual price, one time price, amount-based price, package-based price.
- Checkout pages and customer self-service portal provides already built GUI and functionality to pay for service and later manage subscriptions, consents, do additional purchases.
- Emails – email functionality for payment dunning, onboarding, reminding and informing about actions done is very essential functionality and Charge provides easy and robust way of handling various email sending mechanisms.
- Invoices – after successful payment, client receives email with invoice that complies to all EU and national laws.
- Online back-office functionality – user friendly and rich functionality GUIs for subscription, customer management
- Reporting – provides various report generation engines and visually representation of them, as well provides data export functionality
- Payment gateways – Chargebee has prebuilt integration with most popular payment providers, for example, Stripe, Paypal can be attached to this billing platform very easy.



7 Subscriptions

Subscription Info	Customer Info	Next Renewal	MRR	Created On
ACTIVE ILSP MT for English to modern Greek EUR Monthly BTUTbMSwxS6DTHRq	Andis Lagzdins andis@example.com	09 Mar 2022	50,00 € EUR	09 Feb 2022
ACTIVE ILSP MT for English to modern Greek EUR Monthly 199Z5VSiK1YLA2b	Andis Lagzdins andis.lagzdins@tilde.com	07 Mar 2022	50,00 € EUR	07 Sep 2021
ACTIVE ILSP MT for English to modern Greek EUR Monthly BTM9GjSiJSSYI5fJ	Tilde andis.lagzdins@tilde.lv	07 Mar 2022	50,00 € EUR	07 Sep 2021
ACTIVE ILSP MT for English to modern Greek EUR Monthly 19AAG0Sf1nKbCa5	Tilde andis.lagzdins@tilde.lv	03 Mar 2022	50,00 € EUR	03 Aug 2021
CANCELLED ACL RD-TEC EUR Monthly 19AAGJSZN7KRt4tl	Tilde andish@gmail.com	—	0,00 € EUR	04 Jun 2021
CANCELLED ACL RD-TEC EUR Monthly 19AAGJSZMeA4I4O3	Tilde andis.lagzdins@tilde.lv	—	0,00 € EUR	04 Jun 2021
CANCELLED Tilde MT EN-DE EUR Monthly 199Z7YSZMU0vG3II	Tilde andis.lagzdins@tilde.lv	—	0,00 € EUR	04 Jun 2021

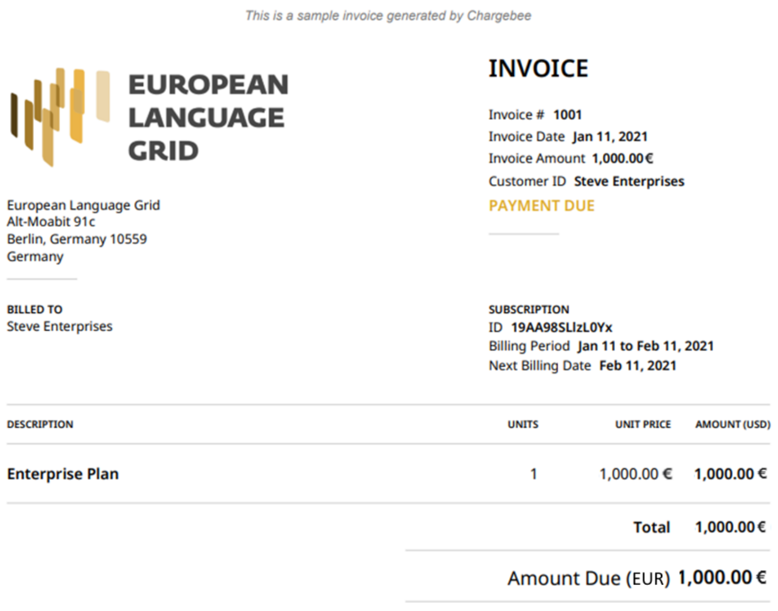
Figure 62: Chargebee subscription management window

[16] https://killbill.io
[17] https://www.invoiceplane.com
[18] https://www.boxbilling.com
[19] https://solidinvoice.co
[20] https://invoiceneko.com
[21] https://www.bambooinvoice.net
[22] https://www.jbilling.com
[23] https://www.logicinvoice.com
[24] https://www.chargebee.com/docs/2.0/index.html

7.3 Billing Configuration

The following configuration was necessary to set up the Chargebee tenant ready for ELG platform integration:

- Product families and plans, prices – created some product families (ASR services, TTS services, MT services, Dataset) and configured initial product for demo purposes – one service and one data set.
- Branding – colours, logos, layout were specified to ensure ELG common style look and feel. Branding was done for Email, Invoices, checkout screens and self-service portal.
- Business profile – about organization
- Product catalogue settings
- Billing configuration
- Subscription settings
- Payment gateways – currently added test payment gateway
- Chargeback settings
- Currencies
- Taxes – added configuration for EU Region
- Consent management – settings for obtaining customer's consents before processing data
- Personal data – settings for handling the client's personal data
- Languages – currently English was enabled, but it is possible to enable more languages from the list or translate to any.
- Email notifications – Chargebee supports more than 70 various events when to send email to customer. We have enabled and configured 20 various email templates.
- Invoicing options
- Configuration of check-out and self-service portals functionality
- Dunning settings for online payments
- Webhooks for integration with ELG


This is a sample invoice generated by Chargebee

EUROPEAN LANGUAGE GRID

European Language Grid
Alt-Moabit 91c
Berlin, Germany 10559
Germany

INVOICE

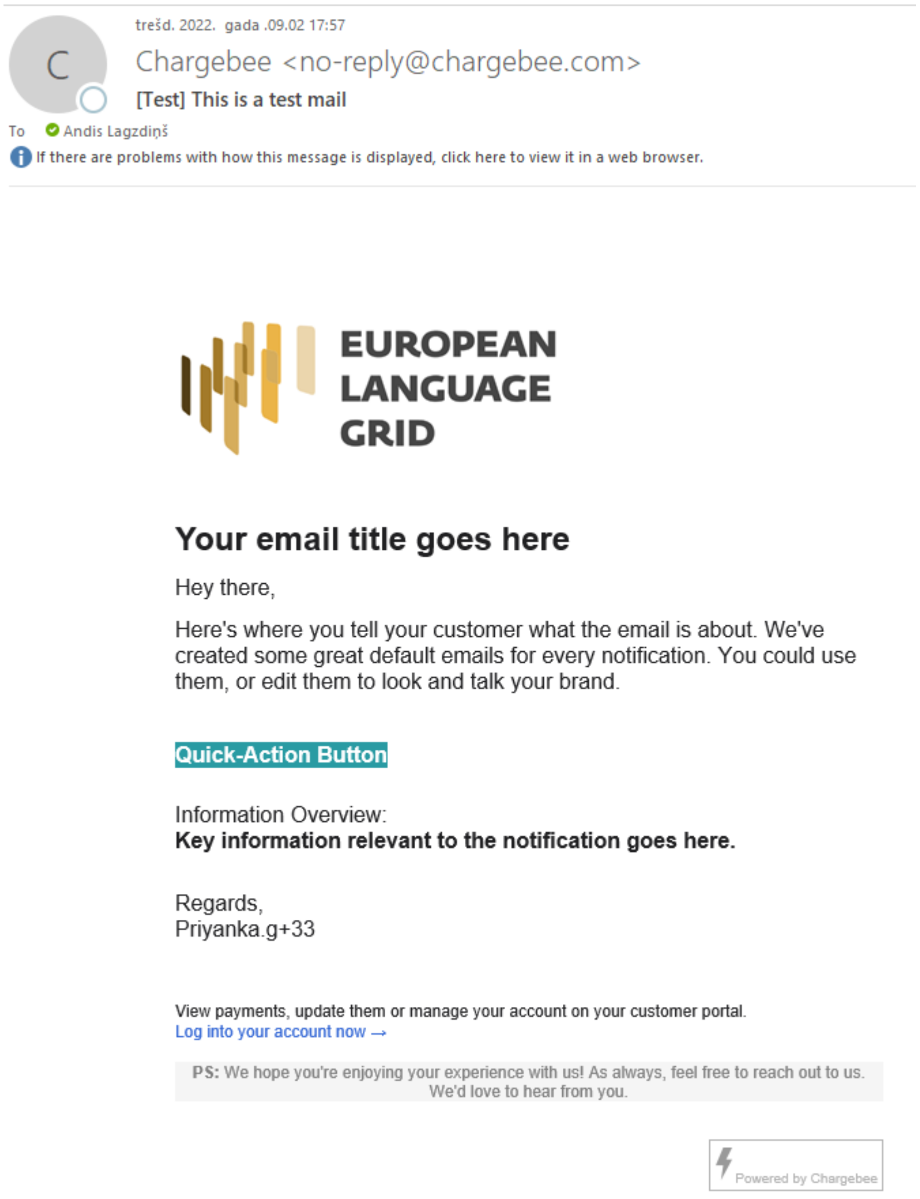
Invoice # **1001**
Invoice Date **Jan 11, 2021**
Invoice Amount **1,000.00€**
Customer ID **Steve Enterprises**
PAYMENT DUE

BILLED TO
Steve Enterprises

SUBSCRIPTION
ID **19AA98SLIzL0Yx**
Billing Period **Jan 11 to Feb 11, 2021**
Next Billing Date **Feb 11, 2021**

DESCRIPTION	UNITS	UNIT PRICE	AMOUNT (USD)
Enterprise Plan	1	1,000.00 €	**1,000.00 €**
		Total	**1,000.00€**
		Amount Due (EUR)	**1,000.00 €**

Figure 63: Sample ELG branded invoice generated by Chargebee


trešd. 2022. gada .09.02 17:57
Chargebee <no-reply@chargebee.com>
[Test] This is a test mail
To Andis Lagzdiņš
If there are problems with how this message is displayed, click here to view it in a web browser.

EUROPEAN LANGUAGE GRID

Your email title goes here

Hey there,

Here's where you tell your customer what the email is about. We've created some great default emails for every notification. You could use them, or edit them to look and talk your brand.

Quick-Action Button

Information Overview:
Key information relevant to the notification goes here.

Regards,
Priyanka.g+33

View payments, update them or manage your account on your customer portal.
Log into your account now →

PS: We hope you're enjoying your experience with us! As always, feel free to reach out to us. We'd love to hear from you.

Powered by Chargebee

Figure 64: Branded sample test email sent by Chargebee

7.4 Implemented Billing Solution

In ELG release 3 we have developed an example scenario for a service billing cycle for selected services and datasets. The Chargebee site is fully configured thus it can go live at any time. For demo scenario we created specific user role, and for users having this specific role a new function appears at service/dataset details window. After clicking the “subscribe” link, a new modal window appears that loads the GUI from the Chargebee site. That is the checkout page, where user can go and fill their information step-by-step.

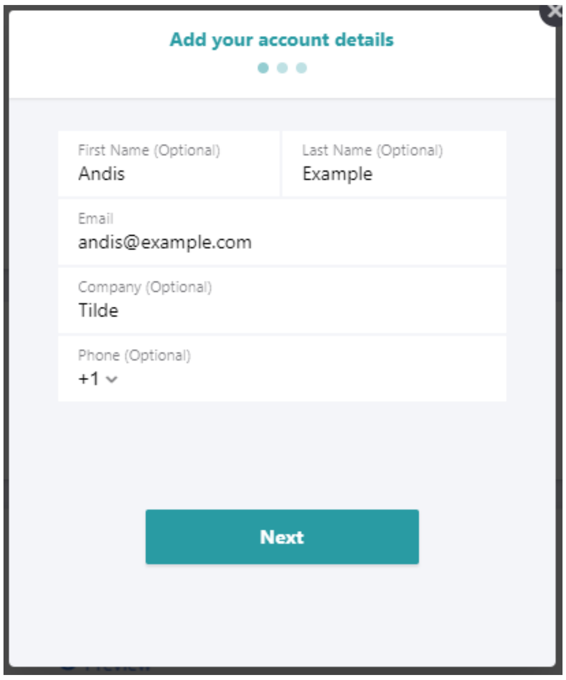


Figure 65: Account information at Billing checkout scenario

The User must fill in information about themselves, later they will be able to directly sign-in into the Chargebee self-service portal using this one-time password sent to the email address submitted here.

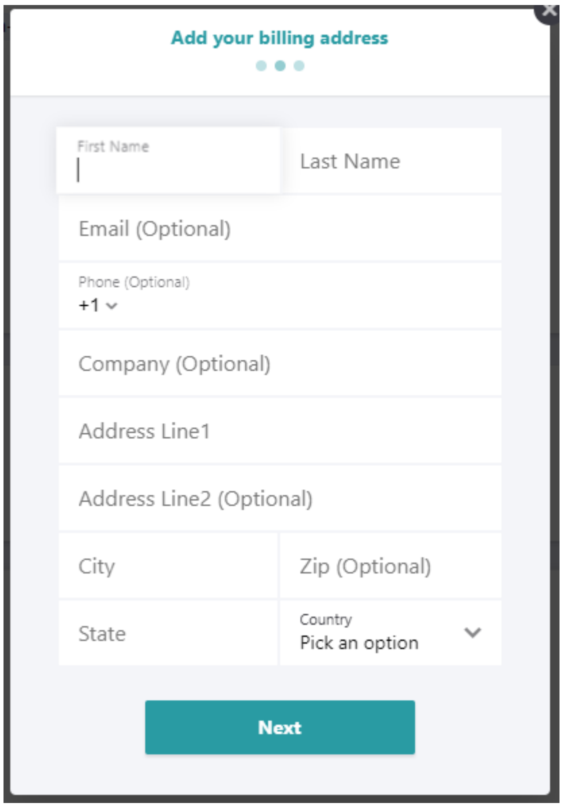
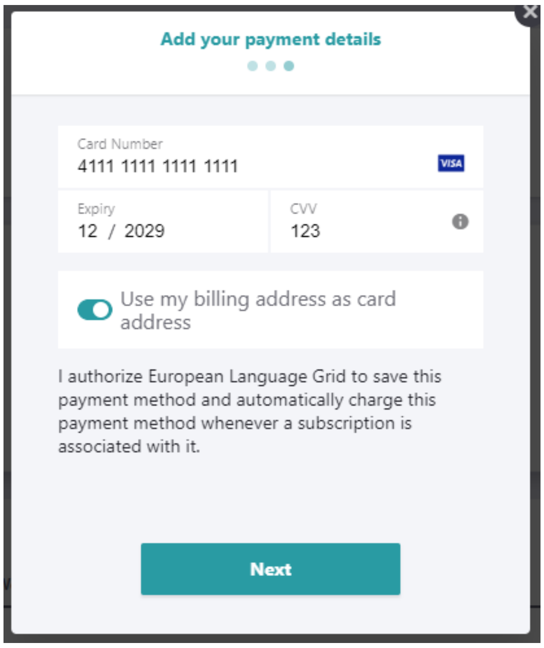


Figure 66: Billing address at Billing checkout scenario

The billing address must be filled in, this is an important step because this information will affect the VAT rate.



Card Number
4111 1111 1111 1111

Expiry
12 / 2029

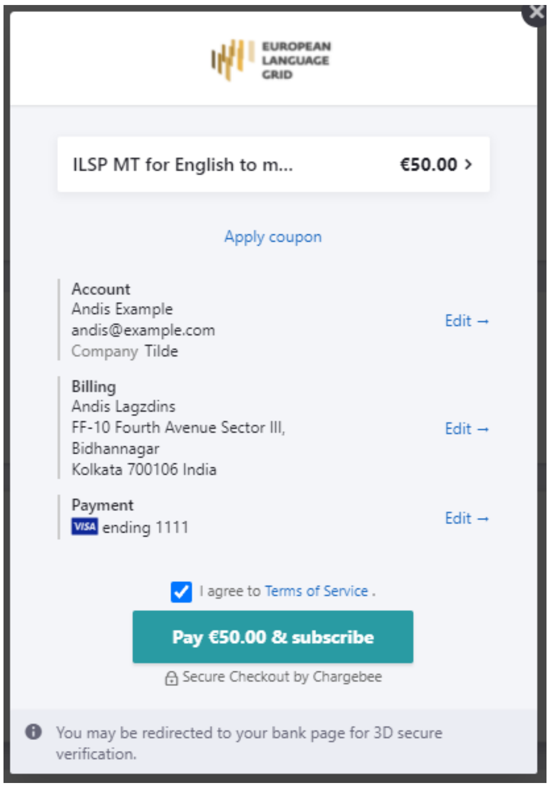
CVV
123

Use my billing address as card address

I authorize European Language Grid to save this payment method and automatically charge this payment method whenever a subscription is associated with it.

Figure 67: Payment details at Billing checkout scenario

Payment details must be entered in the step visible. For test purposes we configured the test payment gateway that lets one select to prefill a form date with fake credit card data during the checkout.



ILSP MT for English to m... €50.00 ›

Account
Andis Example
andis@example.com
Company Tilde

Billing
Andis Lagzdins
FF-10 Fourth Avenue Sector III,
Bidhannagar
Kolkata 700106 India

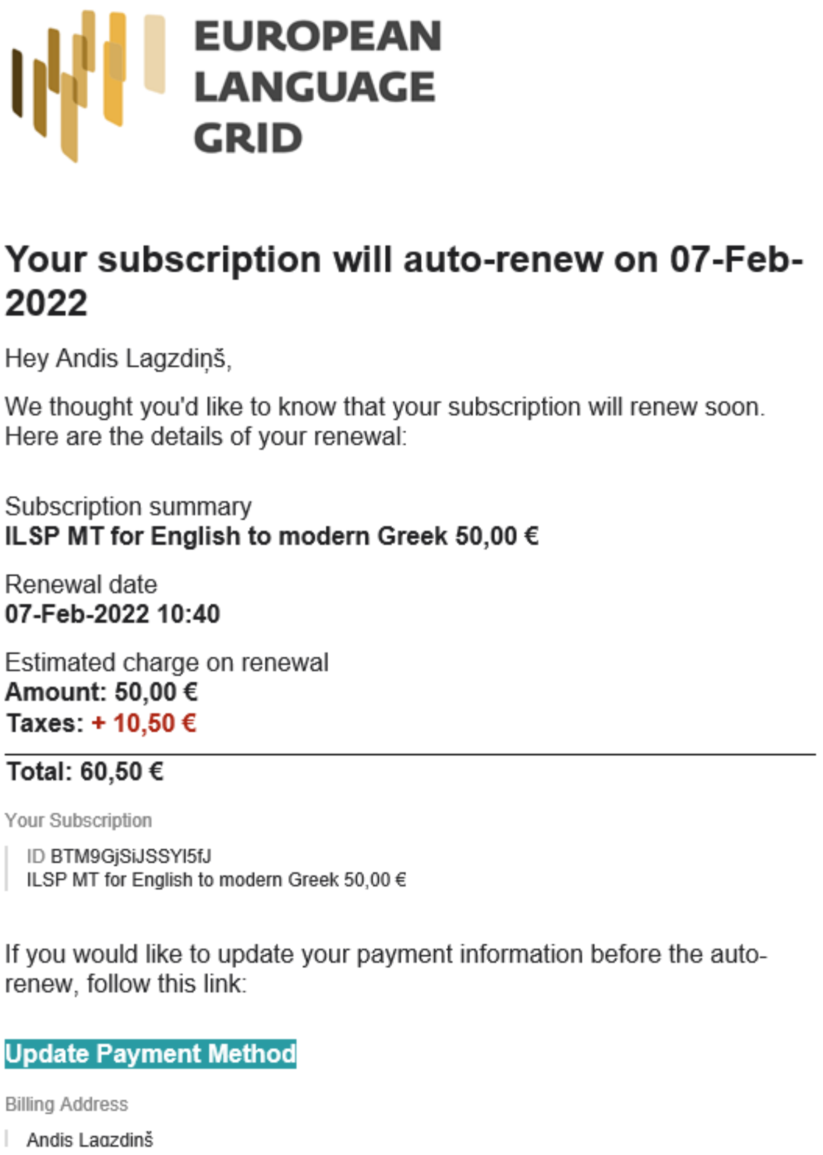
Payment
VISA ending 1111

[✓] I agree to Terms of Service .

You may be redirected to your bank page for 3D secure verification.

Figure 68: Summary at Billing checkout scenario

The very last step is the overview, where the end user has a last opportunity to change or edit any entered information before getting charged. In case there is a trial period set for this plan, then the actual charge will happen immediately after the trial period. Prior to each recurring payment the user is warned by email, that they will be charged with a link to the self-service portal that provides options to change or cancel the subscription.



EUROPEAN LANGUAGE GRID

Your subscription will auto-renew on 07-Feb-2022

Hey Andis Lagzdiņš,

We thought you'd like to know that your subscription will renew soon. Here are the details of your renewal:

Subscription summary
ILSP MT for English to modern Greek 50,00 €

Renewal date
07-Feb-2022 10:40

Estimated charge on renewal
Amount: 50,00 €
Taxes: + 10,50 €

Total: 60,50 €

Your Subscription

ID BTM9GjSiJSSYI5fJ
ILSP MT for English to modern Greek 50,00 €

If you would like to update your payment information before the auto-renew, follow this link:

Billing Address

Andis Lagzdinš

Figure 69: Email message warning about the next payment

8 Conclusion

In the deliverable D3.4 we report the progress of development of the ELG platform GUI. We present an overview of the overall frontend architecture and provide a technical description and implementation details for the solutions designed for the central portal, CMS, the catalogue UI, the implementation of UX and the authentication solution, and the billing module.