EUROPEAN
LANGUAGE
GRID

D2.4
ELG platform (first release)

Authors:	Penny Labropoulou, Dimitris Galanis, Miltos Deligiannis, Katerina Gkirtzou, Athanasia Kolovou, Dimitris Gkoumas, Stelios Piperidis (ILSP), Florian Kintzel, Nils Feldhus, Georg Rehm (DFKI), Ian Roberts, Kalina Bontcheva (USFD), Andis Lagzdiņš, Jūlija Meļņika (TILDE)
Dissemination Level:	Public
Date:	30-04-2020

About this document

Project	ELG – European Language Grid
Grant agreement no.	825627 – Horizon 2020, ICT 2018-2020 – Innovation Action
Coordinator	Dr. Georg Rehm (DFKI)
Start date, duration	01-01-2019, 36 months
Deliverable number	D2.4
Deliverable title	ELG platform (first release)
Type	Other (Report + Software)
Number of pages	53
Status and version	Final – Version 1.0
Dissemination level	Public
Date of delivery	Contractual: 30-04-2020 – Actual: 30-04-2020
WP number and title	WP2: Grid Platform – Language Grid
Task number and title	Task 2.3: Implementation, integration and initial population of the ELG platform & Task 2.4: Development and integration of APIs between technical components of the core ELG system
Authors	Penny Labropoulou, Dimitris Galanis, Miltos Deligiannis, Katerina Gkirtzou , Athanasia Kolovou, Dimitris Gkoumas, Stelios Piperidis (ILSP), Florian Kintzel, Nils Feldhus, Georg Rehm (DFKI), Ian Roberts, Kalina Bontcheva (USFD), Andis Lagzdiņš, Jūlija Meļņika (TILDE)
Reviewers	Andrés Garcia Silva, José M. Gómez Pérez (EXPSYS), 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
	Sail Labs Technology GmbH (SAIL), Austria
	Expert System Iberia SL (EXPSYS), Spain
	University of Edinburgh (UEDIN), United Kingdom
EC project officers	Philippe Gelin, Alexandru Ceausu
For copies of reports and other ELG-related information, please contact:	DFKI GmbH European Language Grid (ELG) Alt-Moabit 91c D-10559 Berlin Germany 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 © 2020 ELG Consortium

Table of Contents

List of Figures

List of Tables

List of Terms

Functional content	Language processing tools and services that can be executed either locally or in a cloud infrastructure.
Language Data Resource, Language Data	A Language Resource composed of data, as opposed to a Language Technology tool or service.
Language Resource	A resource composed of linguistic material used in the construction, improvement or evaluation of language processing applications, but also, in a broader sense, in language and language-mediated research studies and applications; examples include data sets of various types, such as textual, multimodal or multimedia corpora, lexical data, grammars, language models, etc. in machine readable form. The term is often used in the bibliography and related initiatives with a broader meaning, encompassing also the (a) tools and services used for the processing and management of data sets, and (b) standards, guidelines and similar documents that support the research, development and evaluation of LT (cf. http://languagelog.ldc.upenn.edu/myl/ldc/LR_background.html). In this document, we use the term as first defined in the META-SHARE metadata model, i.e., including both data resources and Language Technology tools/services to avoid confusion with previous metadata descriptions. The alternative term "Language Resource/Technology" is also used.
Language Technology tool or service	A tool, service, or piece of software that performs language processing or any Language Technology related operation.
Non-functional content	Any non-executable type of resource (data resource or source code of tools/services) that can be included in the European Language Grid.

List of Abbreviations and Acronyms

API	Application Programming Interface
ASR	Automatic Speech Recognition
CMS	Content Management System
CRUD	Create, Read, Update and Delete
DoA	Description of Action
DMP	Data Management Plan
ELG	European Language Grid
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IE	Information Extraction
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation – Linked Data
K8s	Kubernetes
LR	Language Resource
LRT	Language Resource and Technology
LT	Language Technology
MT	Machine Translation
MVP	Minimum Viable Product
NCC	National Competence Centre
NLP	Natural Language Processing
OWL	Web Ontology Language
R1	Release 1
REST	Representational State Transfer
SPA	Single Page Application
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSD	XML Schema Definition

Abstract

This document introduces the first release of the ELG platform. We present the platform's architecture, its main building blocks, components and functionalities as well as the software and application frameworks used. The document includes an overview of the base infrastructure, the platform backend (with a detailed presentation of the catalogue application and the execution component for the LT processing services), and a short account of the catalogue frontend. The current contents of the catalogue and the ELG metadata model are also briefly described. Finally, the first release ("Release 1") includes a user manual with instructions for ELG platform users (consumers and providers of Language Resources and Technologies). The manual is an online document, allowing for a continuous update throughout the platform evolution. ELG Release 1 is available at https://live.european-language-grid.eu.

1 Introduction

This document, Deliverable 2.4, consists of two parts:

- the **software** of the first release (Release 1, R1) of the ELG platform[1], i.e., the modules implementing the main functionalities foreseen for this release, and the current platform contents, namely a set of functional ready-to-deploy Language Technology (LT) services and Language Resources (LRs) together with their metadata descriptions, as well as metadata entries for LT-related actors and activities;
- this **report**, which provides an overview of the current state of the platform, its architectural design and main components, the software and application frameworks used for their implementation, the application programming interfaces (APIs), as well as the operations and supported user interactions.

ELG R1 includes the main building blocks required for the operation of the platform, albeit with a limited set of functionalities, as detailed in the following sections:

- the **user management** component with a basic set of user roles,
- the components that support the **uploading, storing and downloading of resources** with their documentation (metadata records),
- the components that enable the **browse and search** features **of the catalogue** and expose the metadata records to users,
- the components that are responsible for the **execution of the LT services**,
- and the **APIs** required for interacting with other layers (front-end, execution of functional services).

This report is organized as follows. Section 2 provides an overview of the ELG platform, its architecture and the development roadmap. Section 3 presents the base infrastructure on which the platform is deployed. Section 4 describes the ELG platform backend, including (a) all the components that support the catalogue operations, such as the user management module, the database and indexing mechanisms, the metadata management

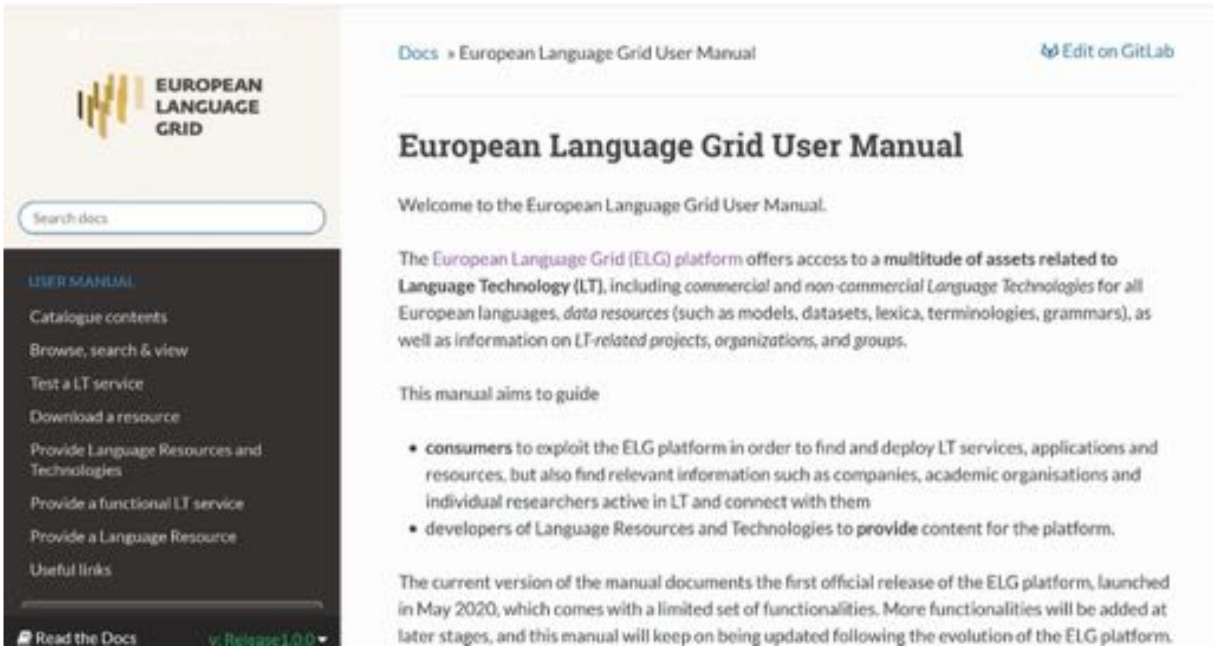
[1] The platform code is maintained at https://gitlab.com/european-language-grid/platform, as it is under constant development throughout the project duration. We currently discuss the appropriate software license to apply to the code base, so that it can be shared with an open permissive license enabling contributions from third parties.

modules, etc., the metadata model, and the catalogue population procedures, and (b) the components required for the execution of LT services. Section 5 presents the user interface (UI) of the catalogue.

The ELG manual (Figure 1), with instructions for all users (content consumers and providers) of the ELG platform, can also be considered part of this deliverable. At the time of writing it includes the following contents:

- introduction and overview of the catalogue contents
- instructions for consumers (search and view of the catalogue and catalogue entries)
- instructions for providers, with subsections for functional LT services (LT tools and services in an ELG-compliant form ready to be deployed in the platform) and data resources (i.e., corpora, models, lexica, terminologies, etc.); step-by-step instructions, examples and links to additional material are included in this section. A section with useful links to more detailed information material (e.g., the specifications of the ELG API for LT service execution, instructions on dockerization, etc.) is also included.

Given that the implementation of the ELG platform is ongoing, the emerging manual is available online at https://european-language-grid.readthedocs.io/en/release1.0.0/, as a living document that we continue to update following the evolution of the platform.



European Language Grid User Manual

Welcome to the European Language Grid User Manual.

The European Language Grid (ELG) platform offers access to a **multitude of assets related to Language Technology (LT)**, including *commercial* and *non-commercial Language Technologies* for all European languages, *data resources* (such as models, datasets, lexica, terminologies, grammars), as well as information on *LT-related projects, organizations, and groups.*

This manual aims to guide

- **consumers** to exploit the ELG platform in order to find and deploy LT services, applications and resources, but also find relevant information such as companies, academic organisations and individual researchers active in LT and connect with them
- developers of Language Resources and Technologies to **provide** content for the platform.

The current version of the manual documents the first official release of the ELG platform, launched in May 2020, which comes with a limited set of functionalities. More functionalities will be added at later stages, and this manual will keep on being updated following the evolution of the ELG platform.

Figure 1: ELG user manual

2 The ELG Platform

2.1 Overview

The European Language Grid (ELG) platform [Rehm et al. 2020] aims to become the primary platform for Language Technology (LT) in Europe; it is developed to be a scalable cloud infrastructure that provides, in an easy-to-integrate way, access to hundreds of commercial and non-commercial LTs for all European languages, including both functional (LT tools/services deployed within ELG) and non-functional (data sets, lexica, models, etc.) resources. Once the platform is fully operational, the European LT community will be able to upload their technologies and data sets into the ELG in an easy and efficient way, to deploy them through the Grid, and to connect them with other resources. In addition, the ELG platform will offer information for and about the LT domain and activities, such as information on projects, stakeholders, events, application and service types, etc.

The ELG project and platform targets a wide range of users with different requirements and expectations (cf. Deliverables D2.1, D2.3 and D7.1 for more information):

- **content providers** and **developers** and **integrators**, i.e., providers and consumers of LT resources,
- **information providers** and **information seekers**, i.e., providers and consumers of LT-related (meta)-information,
- **citizens**, i.e., individuals that are interested in LT (essentially a subset of information consumers),
- **ELG platform and content administrators**, i.e., the ELG technical team that maintains and monitors the day-to-day operation and performance of the platform.

Users may assume the identity of one or more of the above categories when interacting with the system, but with different needs each time. In the rest of the document, we refer to them as *consumers*, *providers* and *platform administrators*. For ELG R1, the focus has been on content consumers, leading to the development of the main functionalities that allow them to view and search for resources, to try out functional services and to download resources stored in the platform. Components required for the population of the platform, and deployment of services have also been implemented, with a limited set of functionalities for the current release, as detailed in the following sections. Administration modules have been implemented in order to support the required operations, though not available through a GUI. For R2 and R3, the platform will be improved with more functionalities. The existing ones will be further tested and enhanced.

2.2 ELG Platform Architecture

An overview of the ELG platform architecture is shown in Figure 2. The platform consists of three main layers: the **base infrastructure**, the **platform backend** and the **platform frontend** (user interface).

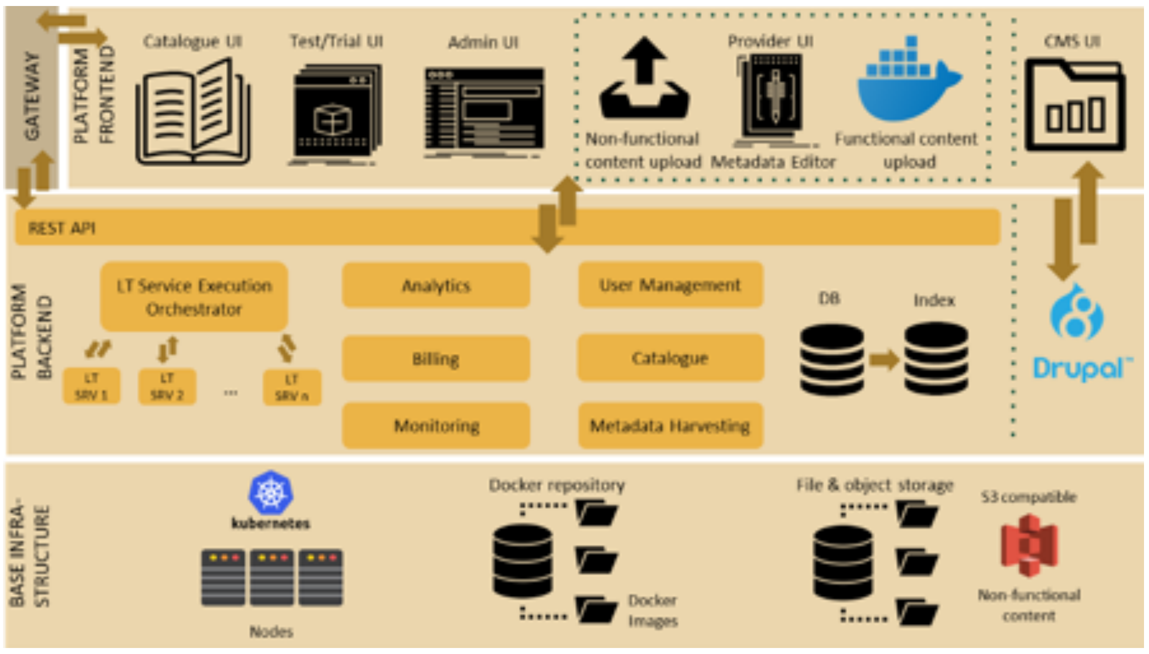


Figure 2: ELG architecture

The **base infrastructure** (Section 3) is the layer on which all ELG software components are deployed and run; it includes the supporting tools that facilitate development and management of the ELG platform software.

The **platform backend** layer (Section 4) consists of all the components that empower ELG, for example, core components such as the database, the index, the user management component as well as the LT services.

The **platform frontend** layer (Section 5) consists of (a) the static pages maintained in the Content Management System (CMS) and which aim to provide information on the project and the LT domain and activities, and (b) the catalogue UI, i.e., the browse/search page with all the metadata records, the view pages of the metadata records, pages for registering resources, etc. The catalogue UI consumes REST services exposed by the ELG platform backend (e.g., catalogue application, LT Service execution server).

This report focuses mainly on the catalogue, the LT service execution layer and the supporting components and functionalities. The CMS and its contents at the current stage are described in Deliverables D3.1 and D3.2.

2.3 Development Roadmap

As specified in the ELG Description of Action (DoA), the platform is developed and delivered in three major releases in April 2020 (M16), February 2021 (M26) and October 2021 (M34) (cf. Table 1).

ELG platform Release 1 (R1) (D2.4; due M16)	• Backend components required for the operation of the catalogue: ○ simple user management component, ○ components supporting documentation/uploading, storing/downloading of all resource types (tools and services, data sets, etc.), ○ APIs required for interacting with other layers • First version of the guidelines on its use and provision of resources, instructions for containerization and invoking remotely accessible web services • Limited sets of tools and services and LRs
ELG platform Release 2 (R2) (D2.5; due M26)	• Updated version of the platform including the components and APIs required for running language processing services (containerized services stored in the ELG and web services via REST APIs) directly in ELG • Updated version of the guidelines on its use and provision of resources, instructions for containerization and invoking remotely accessible web services • Updated catalogue with resources from ELG partners
ELG platform Release 3 (R3) (D2.6; due M34)	• Updated version of the platform including management and maintenance of the platform: monitoring of the platform, monitoring of remotely offered services, platform usage analytics, prototype version of user billing and payment services • Updated catalogue with resources from ELG pilots and collaborating initiatives

Table 1: ELG platform release plan

Two additional pre-releases have been made available so far: (1) a demonstration version (Minimum Viable Product version, MVP), developed at an early stage and presented in October 2019 (M10 of the project) at META-FORUM 2019, i.e., the first Annual ELG Conference (see D7.5); (2) an alpha release made available on request to interested users – especially those who are preparing proposals for the ELG Open Call – in February 2020 (M14). It has been used for testing the platform as well as for the registration of LT services and data LRs; this version, which, through incremental updates has given rise to the R1, runs in the production cluster.

3 ELG Base Infrastructure

The base infrastructure is the layer on which the ELG platform (backend and UI) runs. It also includes tools for software development and consists of the following parts: two **kubernetes clusters** (one for development and a second one for production), **source code/docker repositories** and a **file and object storage module**.

3.1 Kubernetes Cluster

The ELG platform software can be divided into two parts:

- the core components, e.g., the catalogue database, the metadata index, the catalogue application, etc., which are developed and maintained exclusively by the ELG technical team, and
- the set of LT tools/services that are integrated into ELG, i.e., made available through the ELG platform, and that have been developed by consortium partners or external providers.

All aforementioned tools and services run as Docker containers in a Kubernetes (in short k8s) cluster. This facilitates management and deployment (see D1.3 and D2.2). The components that run in the cluster are grouped in k8s namespaces based on their functionalities. A k8s namespace is a virtual sub-cluster, which can be used to restrict access to the respective containers that run within it. For instance, we have an “elg-core” namespace for core components such as Keycloak, which manages user authentication and nginx, the web server that acts as gateway/proxy. In the “elg-backend” namespace, we deployed the PostgreSQL database, the indexer and the REST-based catalogue application. LT services are deployed by default in the “elg-srv” namespace; however, separate dedicated namespaces are allocated for LT services for which restricted access is required.

Currently, ELG uses two clusters, the first cluster hosts the production instance[2] of the platform, while the second one[3] (which is deployed on more limited hardware resources) is used for development and testing. Both clusters run on VMs/hardware of SysEleven, a Berlin-based cloud service provider. SysEleven is ELG’s subcontractor and was chosen in a selection process at the beginning of the project (see D1.1).

Deployment of an application in a Kubernetes cluster requires a set of configuration files (in YAML format) that specify the required information such as the Docker image location, the number of replicas that will be deployed, the ports that will be exposed to the rest of the cluster, and the deployed service’s name. The latter can be used for access by other k8s applications/services. In ELG we have to manage several software components and we also need different deployments of the platform, such as a single-node deployment for development/testing, deployments in SysEleven, etc. For these we use Helm[4], a k8s package manager that automates deployments. Helm uses (generic) templates for describing k8s resources. For each different installation/deployment, the templates are filled with the respective configuration data by running a script. The generated configuration files are then submitted to the cluster via the respective k8s API. The required scripts, helm templates and configuration files that instantiate a specific ELG deployment are kept in a separate branch in ELG’s GitLab repository, i.e., we have separate branches for production and development. For automating deployments, a continuous integration/deployment pipeline was created. When a source file in a branch is changed

[2] https://live.european-language-grid.eu/catalogue/#/
[3] The ELG development cluster is accessible to consortium members only by using IP whitelisting.
[4] https://helm.sh

the CI/CD pipeline is triggered. It downloads the latest helm templates and scripts from GitLab and runs the scripts that inject the configuration. Finally it deploys the final resource files to the respective k8s cluster.

All ELG services (e.g., CMS, LT service execution, catalogue backend and UI) are exposed to the public internet via an Nginx web server (an ingress controller). For each service the appropriate code is inserted to the respective k8s config files that specifies the mapping between a publicly accessible URL/endpoint to the respective internal k8s backend service name (and port). Nginx is also deployed as a Docker container.

3.2 Management of Source Code and Docker Images

The platform's source code is hosted in an "organization" account on GitLab (https://gitlab.com/european-language-grid), a code repository and development platform (for Git projects), similar to GitHub and BitBucket. GitLab was chosen because it provides numerous features for free, such as an unlimited number of private and public repositories, a built-in Docker registry for storing images and built-in continuous integration (CI) and continuous development (CD) functionalities; these CI/CD pipelines are different from the ones described in Section 3.1, which are used for deploying to our k8s clusters. The GitLab CI/CD is used, for example, in order to create the images that host core components of ELG as well as images for LT services. In addition to GitLab's Docker registry, we make use of other registries, such as DockerHub or Azure Container Registry.

3.3 Storage

For the storage of physical files, e.g., data language resources and software delivered as source code, etc., we use an S3-compatible Object Storage solution that is hosted by SysEleven with, currently, 10TB of disk quota. S3-compatible Object Storage provides functionalities that facilitate access to, and management of the data, organized in "buckets" with different access policies, thus ensuring access control and security. Access to an object is restricted to the owner of the S3 account, who also controls the creation and management of storage buckets in terms of data uploads, deletions and access policies, and allows temporary access to requested objects if and when needed. The management of the S3 Object Storage is handled through the available S3 API using dedicated clients, designed to work with S3-compatible Object Storage, such as Minio and S3cmd.

4 ELG Platform Backend

The ELG platform backend consists of the following components.

- **Catalogue application**: a set of REST services for managing the metadata descriptions of ELG resources. It is powered by a PostgreSQL database and an Elasticsearch index (Section 4.1.4).
- **LT Service Execution Backend:** includes all the LT services and the LT Service Execution Server that orchestrates the processing operations (Section 4.2).
- **User management and authentication** (Section 4.1.3).

To put the features of these components in context, we first give a brief overview of the metadata model[5] that is used for describing the ELG entities (Section 4.1.1), then describe the publication lifecycle of metadata records (Section 4.1.2) and the user types foreseen for R1 (Section 4.1.3).

[5] The terms "metadata schema" and "metadata model" are used with the same meaning in this report.

4.1 ELG Catalogue

4.1.1 Metadata Model

The **ELG metadata model** (or ELG-SHARE[6]) is used for the description of all entities of interest to the ELG target users (for a detailed description, see D2.3 and [Labropoulou et al. 2020]). It constitutes the backbone of the ELG catalogue, which brings together language processing services and tools, LRs (data sets of different types and media, models, lexica, terminologies, etc.) as well as actors and activities related to LT (Figure 3).

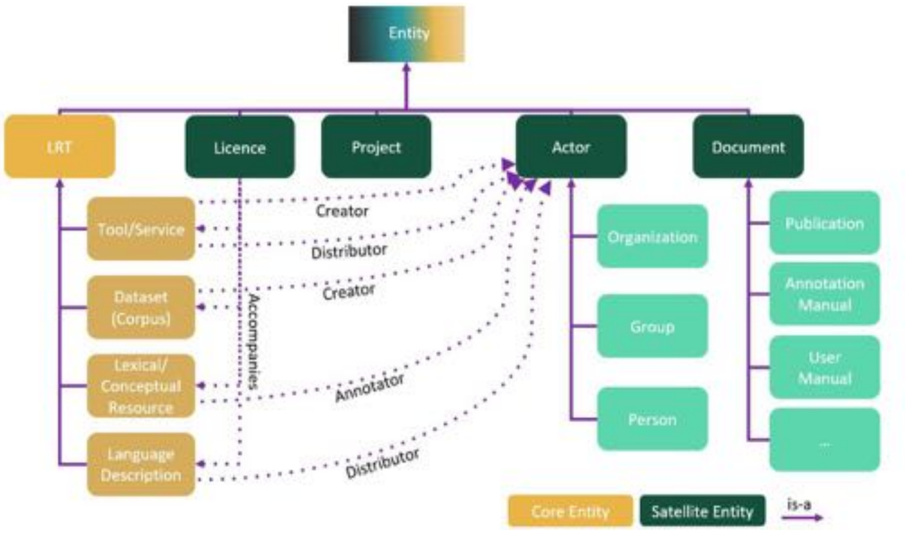


Figure 3: Overview of the ELG-SHARE entities

The model caters for the description of ELG core entities, i.e.,

- *LT tools/services*, covering all software that performs language processing and/or any LT-related operation (e.g., basic processing tools, applications, web services etc. that perform annotation, Machine Translation systems, speech recognizers, etc.).
- *Corpora* (data sets), defined for our purposes as structured collections of pieces of data (textual, audio, video, multimodal/multimedia, etc.), typically of considerable size and selected according to criteria external to the data (e.g., size, type of language, type of producer or expected audience, etc.) to represent as comprehensively as possible the object of study.
- *Lexical/conceptual resources*, i.e., resources (such as terminological glossaries, word lists, semantic lexica, ontologies, gazetteers, etc.) organized on the basis of lexical or conceptual units (lexical items, terms, concepts, phrases, etc.) with their supplementary information (e.g., grammatical, semantic, statistical information, etc.).
- *Language descriptions*, i.e., resources aiming to describe a language or some aspect(s) of a language via a systematic documentation of linguistic structures (e.g., computational grammars, statistical and machine learning-computed language models).

[6] The ELG metadata model builds upon, extends and updates META-SHARE and its application profiles. Modifications have been made (a) in the contents (e.g., addition of elements for GDPR, improvement of the description of LT services and models), in response to the project requirements and more recent developments in the metadata area at large, (b) in the implementation, which now combines the XSD approach used for META-SHARE profiles with the deployment of two ontologies, namely META-SHARE (https://w3id.org/meta-share/meta-share/) and OMTD-SHARE (https://w3id.org/meta-share/omtd-share/), for the definition of the elements and values. For a detailed description, see Deliverable D2.3 and [Labropoulou et al. 2020].

The ELG model also provides for entities involved in the production and usage of LTs/LRs and, in general, LT activities, i.e., actors (*organizations*, *groups*, *persons*), *documents* (e.g., user manuals, publications, etc.), *projects* and *licences/terms of use*. The model includes a large number of metadata elements grouped along three key concepts: *resource type*, *media type* and *distribution*. The *resource type* element distinguishes LRTs in the four classes presented above. *Media type* refers to the form/physical medium of a data resource (or of its parts, in the case of multimodal resources), i.e., text, audio, image, video and numerical text (used for biometrical, geospatial and other numerical data). Finally, *distribution*, following the DCAT vocabulary[7], refers to the physical form of the resource that can be distributed and deployed by consumers; for instance, software resources may be distributed as web services, executable files or source code files, while data resources as PDF, CSV or plain text files or through a user interface. Administrative and descriptive metadata are mostly common to all LRTs, while technical metadata differ across resource and media types as well as distributions. Figure 4 provides an example with part of the metadata model for tools/services.

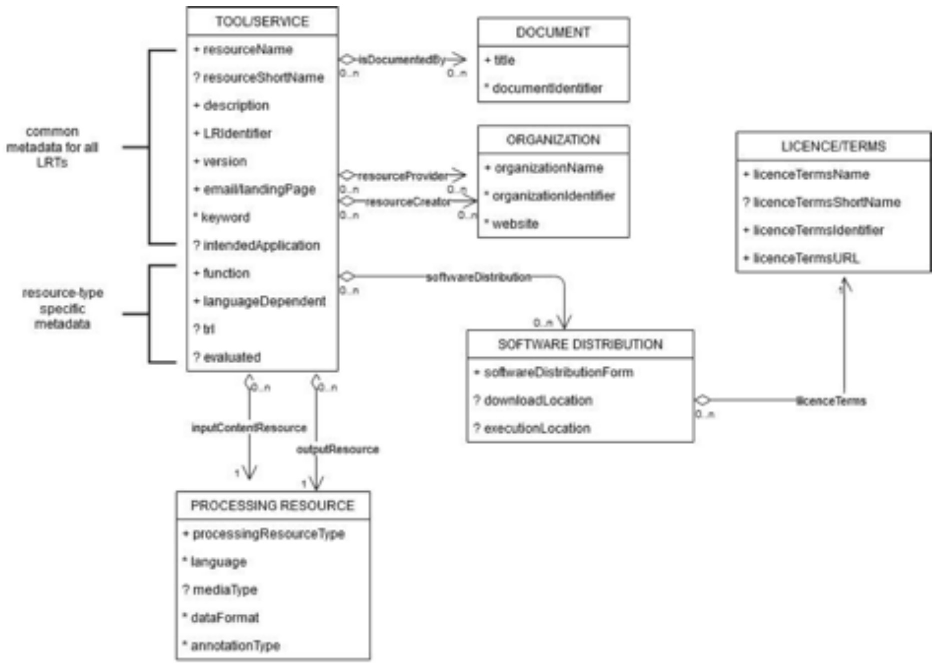


Figure 4: Excerpt of the ELG metadata model (focusing upon tools/services)

The abundance of information in the schema makes the task of creating metadata records quite tedious. To ensure flexibility and uptake, we distinguish *mandatory*, *recommended* and *optional* metadata elements. Minimal metadata records with only the mandatory and strictly recommended elements are, thus, possible. The criteria used in ELG to determine the optionality status of elements include: required for discovery, especially features considered of high interest to ELG consumers (D2.1, D3.1); considered indispensable for accessing the resources and, in the case of functional services, ensuring proper deployment through the platform; supporting usage of resources; deemed valuable for experiments and projects and essential for achieving interoperability with existing metadata schemas used in the wider LT and neighbouring communities.

[7] https://www.w3.org/TR/vocab-dcat-2/

The design and implementation of the model have been completed (with the exception of the billing module). Nevertheless, we anticipate updates and improvements for upcoming releases of the platform, taking into account user feedback and technical requirements of the platform as its development progresses.

The model is implemented in the form of an XML Schema Definition (XSD).[8] Its elements are linked to entities from two ontologies, namely the META-SHARE[9] ontology, which includes the majority of elements and controlled vocabularies, and the OMTD-SHARE ontology[10], reserved for the controlled vocabularies of LT categories (also referred to as LT taxonomy), data formats and methods. Each metadata element and value has an identifier which contains the IRI of the corresponding entity. This approach contributes to the FAIRness[11] of the metadata model, facilitates linking to metadata records in other catalogues, and supports import/export in the JSON-LD serialization format, which increases the visibility of ELG metadata records overall. The use of XSD enables us to transform the metadata schema easily into an entity-relationship model[12], thus facilitating its documentation and conversion into the relational database used in the ELG catalogue backend. Ontology entities were automatically converted into XML elements, as used in the XSD. This approach enables an easy update of the schema alongside the ontology update. All relations, labels and definitions are copied into the XML elements with the same script, feed the display labels in the landing pages of the metadata records, and will be utilized for the metadata editor in Release 2 of the platform.

For R1, we use a dual approach with regard to the metadata requirements for the population of the catalogue:

- For entries provided by individuals we require adherence only to the minimal schema, since the creation of XML records is not trivial. The metadata editor in R2 and R3 will support the provision of full records, and allow for providers of existing metadata records to enrich them, should they wish to.
- For entries converted from metadata records from other catalogues (see D5.3), we convert as much metadata as possible into the ELG schema, taking full advantage of the full version of the model.

4.1.2 Publication Lifecycle of Metadata Records and Publication Process

The publication lifecycle of a metadata record follows the principles of META-SHARE, and is divided into the following states:

- **internal:** used as the initial state for all metadata records;
- **ingested:** used as an intermediate step before making the record publicly available, which can be used for checking the uploaded metadata, modifying and correcting the descriptions, as well as for testing of functional services;
- **published:** for finalized metadata records; published records are available in the public catalogue.

[8] The ELG metadata schema is available at https://gitlab.com/european-language-grid/platform/ELG-SHARE-schema, accompanied with documentation, and metadata record templates for all resource and media types.
[9] https://w3id.org/meta-share/meta-share/
[10] http://w3id.org/meta-share/omtd-share/
[11] The FAIR Guiding Principles for scientific data management and stewardship aim to improve the findability, accessibility, interoperability, and reuse of digital assets. The principles emphasise machine-actionability (i.e., the capacity of computational systems to find, access, interoperate, and reuse data with none or minimal human intervention) because humans increasingly rely on computational support to deal with data as a result of the increase in volume, complexity, and creation speed of data, see https://www.go-fair.org/fair-principles/.
[12] https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model

The publication procedure for resources and the user management model of the ELG platform determine which users can access metadata records in each of the three states. This has been devised as a measure of precaution and monitoring of registered resources.

Metadata records can be submitted to the ELG platform only by users who have registered in the platform and given the appropriate authorization; these can be individuals who wish to share their own LRTs or are acting as representatives of an organization to which they are affiliated and uploading LRTs developed by that organization. The registration procedure for users includes consent to the ELG terms of use, whereby they commit to provide resources for which they have the legal right to share them and that they are technically "safe". The provision of resources by registered users who are logged in allows for giving credit to the specific providers, but also entrusting them with all responsibility for any issues that may rise with regard to them.

Authenticated and authorized users must provide at least a metadata description of their resource according to the ELG schema. They can also upload the resource itself, in which case it will be stored and preserved according to the Data Management Plan (D5.1) and, in the upcoming releases, may be selected and processed by LT services also integrated in the platform.

Further requirements apply specifically to LT services in order to be integrated as ready-to-deploy in the ELG platform (see Section 4.3.1). For their publication, they go through a review process that aims to ensure technical validity of the service itself. The review process is performed by the ELG technical team on ingested metadata records, and includes checking the technical metadata required for the import and deployment of the service, and the addition of ELG-specific information that will enable the execution of the service in the ELG platform. Before publishing a tool/service, the reviewer checks that (a) it follows the ELG technical specifications; (b) its technical metadata are as required; (c) it can be executed in the ELG platform.

The ELG catalogue can be populated with metadata records harvested from other sources, following specific agreements (see D5.1). In this case, the resources may remain in their source repository or be transferred to the ELG system, depending on the negotiations with each source. In any case, ELG administrators can check the metadata records before or after their import in the catalogue and decide to publish them.

Finally, information (metadata records) for LT actors and activities (e.g., LT companies, research organizations active in the LT areas, projects, etc.) is collected through their relations from the core entities populating the catalogue as well as through other processes (e.g., collection from NCCs, surveys, etc.). These records can then be displayed in the public catalogue and individuals can claim them for enrichment, curation and maintenance.

4.1.3 User Management, Authentication and Authorization

For the first release of the ELG platform, the user management model comprises a set of broad user categories and roles with access policies defined in response to the functionalities of the current release:

- **Unregistered users**: Users who are not registered with the platform and who are, hence, not logged in, can browse the catalogue with published metadata records, search for resources and view their descriptions; they can download resources with open licences; however, they are not allowed to register a resource, or run a published LT service; they do not have access to internal/ingested resources.
- **Registered users**: These are the users with accounts in the ELG user database. Each registered user can be assigned one or more of the following roles:

- **Consumer:** In addition to the access rights of unregistered users, they can also try out LT services with the GUIs provided in the platform, or in command mode.
- **Provider:** Providers have all the access rights of a consumer and are also able to register resources. They have access to the resources they have registered even when still ingested.
- **Reviewer:** These are users who validate resources and confirm their publication. They have the same rights as a consumer and advanced rights for the resources they are reviewing.
- **Administrator:** the team of ELG colleagues who develop and maintain the platform; administrators have full access to all resources and functionalities of the platform.

The user management and authentication module is based on Keycloak, an open-source identity and access management solution. Its adoption relieves us from the need to design and implement login and registration forms, to create and maintain mechanisms for issuing and distributing access tokens etc.

Many catalogue functionalities do not require authentication (e.g., search, view resource); however, some functionalities such as LT service execution are provided only for registered users. In these cases, users are redirected to a Keycloak page in order to log in to their account. Once logged in, they are redirected back to the catalogue frontend along with an access token (in JSON Web Token, JWT, format) that keeps them logged in.

A key concept in Keycloak is the realm, i.e., a dedicated domain that consists of a specific set of users, clients, security policies, access roles etc. Using Keycloak's administration console we created a realm dedicated to ELG, and registered the required clients under it. A client is the only way for an application (e.g., the catalogue frontend) to use and access Keycloak. We created a "catalogue frontend" client and defined the four roles described above, i.e., those of Consumer, Provider, Reviewer, Administrator.

Currently, granting advanced roles for registered users of the platform is a procedure supervised by the administrators. For this reason, the user registration form is currently deactivated. Instead, we manually add users using the Keycloak administration console; by default, all new users get the "consumer" role. Individuals that are interested to receive ELG login credentials can send an email to contact@european-language-grid.eu.

User authorization and assignment of the appropriate access level to a resource or functionality is managed through an ELG-specific set of modules, which are a part of the catalogue backend application.

4.1.4 Catalogue Application

The catalogue backend application is built with Django (version 2.2.8) and the Django REST Framework[13]. It mainly acts as a RESTful backend providing all necessary functionalities for database management and user access control to the various aspects of the ELG platform.

The backend defines a set of REST API endpoints used for handling requests from the catalogue UI and other platform modules, such as the LT service execution server. The REST endpoints are controlled by the user authorization policies, in order to distinguish between those that are accessible by everyone (e.g., the retrieval of a published metadata record) and those that require user authentication and a specific user role (e.g., users can create metadata records if they are authenticated and have been assigned the "Provider" role).

[13] https://www.django-rest-framework.org

A key functionality of the catalogue backend is the management of metadata records through CRUD, i.e., create, read, update and delete operations. Currently, the defined REST API endpoints handle requests for

- creating metadata records, by uploading XML files adhering to the ELG-SHARE schema;
- retrieving a metadata record in JSON format, to be rendered by the frontend (catalogue UI);
- downloading a resource in accordance to its licensing terms (for R1, restricted to open licences);
- authorizing service execution (see Section 4.2.1);
- collecting usage information (see Section 4.4).

For ELG R1, metadata record updates and non-functional content uploads are managed internally by the platform administrators. Users who want to update a description or upload a resource must contact the administrators. Users will be able to manage their metadata records through the online editor from ELG R2 onwards.

4.1.5 Database

A PostgreSQL database is used for the persistent storage of metadata records, which are inserted with "internal" status; their owner is the user who uploaded the file. The backend checks whether the uploaded metadata description adheres to the ELG model and, if not, rejects it and informs the user of the XML validation errors.

4.1.6 Indexing and Search

Publicly available metadata records can be searched through the catalogue UI, on a subset of the metadata information indexed by Elasticsearch, either by search queries or using faceted search. This subset includes

- for all entities (LRTs, projects and organizations in R1): name or title, description, keywords;
- in addition, for LRTs: resource type (i.e., Tool/Service, Corpus, Lexical/Conceptual Resource, Language Description), languages, licences, intended application, and resource publication status;
- and for LT tools/services, also the service function.

The backend returns results in JSON format, when queried by the catalogue UI component.

4.1.7 Upload/Download and Storage of non-functional Content

Non-functional content uploaded to the platform (together with a metadata record) is stored in a private S3 compatible storage bucket. Access is restricted and all downloads are managed by the catalogue backend. Upon receiving a download request from the catalogue UI, the backend checks the user's access permissions, the licensing terms of the resource, as well as the user's consent to the licence(s) attached to the metadata record and decides whether this request is eligible; if it is, it returns a temporary pre-signed download URL to the catalogue UI; otherwise, a "permission denied" response is issued.

The catalogue backend currently accepts compressed files in .zip, .tar and.tar.gz formats and stores information about the size, date of upload and the hash of the object that reflects changes to its contents (ETag). Every uploaded resource, related to a metadata record, is stored with a unique key, consisting of the metadata record's storage object identifier (a uuid assigned to each metadata record upon creation) and a filename.

4.2 LT Service Execution Backend

One of the main goals of ELG is to provide a substantial number of LT services through the platform. To accomplish full integration and deployment of the services, we rely on containerisation and the specification of generic LT processing APIs. For accessing the LT services from the ELG catalogue, from the command line or by using any programming language, a common public REST API has been made available (Section 4.2.1). The LT service execution is powered by Knative, a platform installed on top of K8s that provides scale-down-to-zero and autoscaling (Section 4.2.2).

4.2.1 Internal LT APIs

The LT tools we currently work with and integrate into ELG broadly fall into one of the following categories: Information Extraction (IE), Text Classification (TC), Machine Translation (MT), Automatic Speech Recognition (ASR) and Text to Speech Generation (TTS). For each category a specific API was defined (Task 2.5) with the aim of standardizing the invocation of these services and their integration into the ELG platform. This API specification and the adoption of Docker images for packaging the LT tools solves several interoperability issues and facilitates their deployment in the ELG platform. The ELG requirements for integrating an LT service are the following:

Expose an ELG compatible endpoint: An application that exposes an HTTP endpoint for the provided LT tool should be created. The application should consume (via the HTTP endpoint) requests that follow the ELG JSON format, call the LT tool and produce responses, again in the ELG JSON format. For instance, for an IE tool, its HTTP endpoint should accept POST requests in the JSON-based format presented in Figure 5.

```
{
  "type":"text",
  "params":{...}, /* optional */
  "content":"The content, as a string inline",
  // mimeType optional - this is the default if omitted
  "mimeType":"text/plain",
  "features":{ /* arbitrary JSON metadata about this content, optional */ },
  "annotations":{ /* optional */
    "<annotation type>":[
      {
        "start":number,
        "end":number,
        "features":{ /* arbitrary JSON */ }
      }
    ]
  }
}
```

Figure 5: JSON input

Most parts of the request are optional, only “type” and “content” are required. The “features” and “annotations” fields may be useful for services that can build on partially-annotated output from other services. The “start” and “end” of each annotation specify the position of the annotation within the text. The JSON response for an IE tool should be in the format presented in Figure 6.

The response contains sets of annotations and, for each annotation, the start and end offsets are included together with any other available features. Detailed documentation and examples for all the different cases of requests and responses is available through ELG GitLab[14] and in D4.1.

[14] https://GitLab.com/european-language-grid/platform/elg-apis/-/blob/master/doc/T2.5-api-design.md

Dockerization: The application should be Dockerized and the respective image should be uploaded into a Docker Registry, such as the GitLab, DockerHub or Azure container registry. Three different options for the provision of LT tools are supported:

- **LT tools packaged in one image:** One docker image is created that contains the application that exposes the ELG-compatible endpoint.
- **LT tools running remotely outside the ELG infrastructure:** For these tools one proxy image is created that exposes one (or more) ELG-compatible endpoints; the container communicates with the actual LT service that runs externally, i.e., outside the ELG infrastructure.
- **LT tools requiring an adapter:** For tools that already offer an application that exposes a non-ELG compatible endpoint (HTTP-based or other), a second adapter image should be created that exposes an ELG-compatible endpoint and that acts as proxy to the container that hosts the actual LT tool.

```
{
  "response":{
    "type":"annotations",
    "warnings":[...], /* optional */
    "features":{...}, /* optional */
    "annotations":{
      "<annotation type>":[
        {
          "start":number,
          "end":number,
          "features":{ /* arbitrary JSON */ }
        }
      ]
    }
  }
}
```

Figure 6: JSON output

4.2.2 Kubernetes and Knative

All LT services are packaged as Docker images that follow the ELG specifications. The LT containers as well as the core components of the platform (e.g., the catalogue backend) run in a k8s cluster within predefined namespaces (e.g., for services the “elg-srv” namespace). For security and isolation reasons, an LT provider can request a dedicated namespace with restricted access for its services. For instance, LT services provided by ILSP and Expert System run in “elg-srv-ilsp” and “elg-srv-expsys” namespaces respectively. When an LT service is deployed using an adapter, the LT service and the adapter containers are deployed in the same k8s pod[15].

On top of k8s, we installed Knative[16] to efficiently manage the workload of the various LT services and to avoid draining the available hardware resources. A substantial number of LT services are planned to be integrated into the ELG platform, many of which require a significant amount of hardware resources (CPU and/or memory). It is simply impossible to keep all of them up and running all the time. Knative addresses this issue by offering a scale-to-zero functionality; i.e., the number of running containers (replicas) of a certain LT service is scaled down to zero for the time period that the service is not used by anyone. Also, Knative offers autoscaling

[15] A pod is a group of containers deployed together on the same host.
[16] https://knative.dev

functionalities, i.e., when a user initiates a processing request for a service, knative receives this request and starts the corresponding container (if replicas = 0) and forwards the request to it. Depending on the number of requests for this service and based on the configuration, it spawns additional containers. An example of such a configuration, for a MT service (French-to-English) created by Charles University, is given in Figure 7.

```
image: "registry.GitLab.com/european-language-grid/cuni/srv-translation-t2t/fr-en"
noIngress: true
containerport: "8080"

scalability: "dynamic"
minScale: "0"
maxScale: "2"
concurrency: "500"

requests_memory: "2048Mi"
requests_cpu: "5m"
limits_memory: "2560Mi"
limits_cpu: "1000m"
```

Figure 7: Knative configuration example

The “image” field specifies the location of the LT image, “containerport” defines the port where the ELG-compliant REST service hosted in the container runs. The parameters “minScale” and “maxScale” define the minimum and maximum number of replicas that can run for this service. “Concurrency” specifies a target number of concurrent requests to be served by each replica of the LT container; a sustained load above this level will cause additional instances of the container to be spawned, subject to the configured “maxScale” limit. The last four fields specify hardware requirements/limitations for the containers that run for this LT service[17].

4.2.3 LT Service Execution Server and External LT API

The LT Service Execution Server is responsible for orchestrating LT processing. This server is different from the catalogue backend application. It implements all functionalities relevant to LT processing. In this way, we achieve modularity and a clean separation of platform functionalities. The LT Service Execution Server provides:

- a client/connector for sending HTTP processing requests to and receiving HTTP responses from the backend LT services;
- a mechanism for retrieving the k8s REST endpoint of an LT service; this endpoint is configured during the registration of the service and is kept alongside other information in the respective database table;
- a mechanism for limiting access to LT services, i.e., for each registered ELG user[18], we keep a daily usage record containing the number of requests/calls that have been submitted to the LT service execution server as well the total size of these requests (in bytes); if the user exceeds the predefined limits/quotas, execution is not allowed anymore;
- a common REST API used for calling an integrated LT service from the catalogue frontend or from the command line. Table 2 shows the current endpoints of the API[19]. {Domain} is the domain name of the deployed ELG platform (e.g., the production ELG domain name is live.european-language-grid.eu);

[17] https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/
[18] When registering to the ELG platform, users must read and consent to the ELG Terms of Use, which inform them of their personal data collected when interacting with the platform, and the broader policies of ELG with regard to GDPR (General Data Protection Regulation).
[19] The REST API also includes two more endpoints for sending or creating structured requests (e.g., text split into paragraphs).

{ltServiceID} is the id of the service that is being invoked and it is assigned during the registration process and stored in the same database table as the k8s REST endpoint for the specific LT service.
- simple error handling, i.e., if something goes wrong during processing, an appropriate message is returned to the client that initiated the process request.

ID	Endpoint	Type	Consumes	Produces
A	https://{domain}/execution/processText/{ltServiceID}	POST	application/json	application/json
B	https://{domain}/execution/processText/{ltServiceID}	POST	text/plain or text/html	application/json
C	https://{domain}/execution/processAudio/{ltServiceID}	POST	audio/x-wav or audio/wav	application/json
D	https://{domain}/execution/processAudio/{ltServiceID}	POST	audio/mpeg	application/json

Table 2: Example endpoints of LT processing services

According to a typical execution scenario, a user visits the ELG platform, and finds a tool (e.g., the ILSP Named Entity Recognizer for English); the user visits the respective resource landing page[20]. The landing page for services includes, besides the "Information" tab with a description of the tool, technical and administrative information, a "Try out" tab with a specially designed GUI that supports testing for a limited set of calls. Users can put sample input into the GUI and get the result which is rendered in a task-specific viewer. This functionality is reserved for users who are logged in. When the user clicks the "Process" button, a POST call to endpoint B (Table 2) is initiated containing the input data as payload. The LT execution server communicates with the catalogue backend (via an appropriate endpoint) and checks whether this user is allowed to call this service (based on the respective daily usage/quotas). If access is granted, the catalogue returns the HTTP k8s endpoint of this service. Then, it uses the respective client/connector to generate a request to the endpoint; the request is received from knative and forwarded to the appropriate container. If there are no available containers up and running for the specified LT service, knative spawns a new one before forwarding the request (see Section 4.2.2). The response with the results from the LT service container is returned back to the LT execution server which returns it to the client that initiated the call.

Only authenticated users can access the LT service execution server; this is achieved by using gatekeeper[21], a proxy for the Keycloak Identity and Access Management server. The gatekeeper container runs[22] in front of the LT execution server container and allows only requests that contain a valid JWT user to pass through.

The LT service execution orchestrator is implemented in Java using Spring Boot. Spring Boot was selected because it (a) facilitates application configuration, (b) allows the creation of a consolidated standalone application which can be easily Dockerized and deployed, (c) provides an easy way to create REST services, and (d) provides easy integration with a large number of tools and libraries (e.g., databases or messaging middleware).

4.3 Catalogue Population

4.3.1 Registration of Language Resources and Technologies

Individuals who want to make their LRTs available through ELG first have to register and ask for "provider" status by sending an email to the ELG team. As soon as their request is approved, they will receive the appropriate

[20] https://live.european-language-grid.eu/catalogue/#/resource/service/tool/480
[21] https://github.com/keycloak/keycloak-gatekeeper
[22] A separate keycloak client was created for gatekeeper, see Section 3.2 for more information.

credentials to log in. Organizations have to follow the same procedure and nominate a dedicated contact person. Once logged in, the user can click the "Upload" button to initiate the registration process. They are first prompted to upload an XML file with the description of their resource; the XML file must comply with at least the minimal version of the ELG schema. Upcoming releases will also provide an online metadata editor. For this release, in order to facilitate the registration process, pre-filled metadata templates can be used[23].

The XML file is then validated against the ELG XSD and also against a number of rules implemented as part of the import procedure, which check syntactic and partial semantic integrity of the description. Validation errors are reported to the user for correction. If the file is valid, it is imported and assigned the status "internal".

In the next releases, at a second step, the provider will be prompted to upload also an archive (ZIP, GZIP, TAR) with the actual contents of the resource. For R1, this step is performed manually by the ELG administrators. Thus, users who wish to upload their resources to ELG, must send an email, with the details of these resources (e.g., size and format) and they will be notified on the procedure to follow.

As the final step, the ELG administrators review and publish the metadata records.

4.3.2 Registration and Publication of ELG-compliant LT Services

For R1, the process of registering an LT service to ELG is as follows:

1. The LT provider creates a Docker image that contains an LT tool exposed via an ELG-compliant REST service; the Docker image has to be uploaded to a Docker registry (e.g., GitLab, DockerHub).
2. The provider creates a metadata record in XML format compatible with the ELG metadata schema; pre-filled metadata record templates for the LT service categories are available online (see footnote 23).
3. The provider logs into the ELG platform and uploads the XML file.
4. The metadata record is validated at import against the ELG-SHARE XSD and additional rules that check for syntactic and partial semantic integrity.
5. Once uploaded to the catalogue, the metadata record has "internal" status. The administrators assign the metadata record to a reviewer; the record is now "ingested" and visible to reviewers, too.
6. The LT service is deployed to the k8s cluster by creating a configuration yaml file and uploading it to the respective GitLab repository[24]. The CI/CD pipeline responsible for k8s deployments (Section 3) will automatically deploy the new service. If requested (by the LT provider), before creating the yaml file a separate dedicated namespace is created for the LT service[25].
7. A user with the "reviewer" role assigns to the LT service:
   a. The k8s REST endpoint that will be used for invoking it.[26]
   b. An id for the service that will be used to call the LT Service.
8. The reviewer can use the try out UIs or the command line to test the service; integration issues are identified and solved in collaboration with the LT provider. This iterative process is continued until the tool/service is correctly integrated. The procedure requires access to the k8s cluster for the reviewer.

[23] https://gitlab.com/european-language-grid/platform/ELG-SHARE-schema
[24] https://gitlab.com/european-language-grid/platform/infra.git
[25] A separate namespace is usually allocated for a set of services.
[26] The endpoint follows this template http://{k8s service name for the registered LT service}.{k8s namespace for the registered LT service}.svc.cluster.local{the path where the REST service is running at}.

9. When the LT service works as expected, the administrators will publish it to the catalogue at which point it will be available to all ELG users.

4.3.3 Registration of other Entities

The population of the ELG catalogue with metadata records for the LRTs generates additional metadata records for related entities, i.e., entities that are mentioned in these metadata records, such as organizations and individuals that have developed the resource, projects that have funded them, licenses with which they are distributed, etc. These records contain minimal information, i.e., only what is required by the ELG schema to describe a related entity, i.e., a name/title, and, optionally, an identifier and information that could uniquely describe it (e.g., email for persons, website for organizations, a URL with the text for licences, etc.). At a later stage, these records can be enriched by interested individuals once they have successfully claimed the respective record.

This process often leads to the generation of duplicate entries for related entities. To tackle this, the ELG catalogue application offers a retrieval mechanism that matches records of entities already imported in the database. The criteria for the retrieval are based, in priority order, on (a) the ELG internal identifiers, and (b) on the metadata elements that have been declared in the ELG model as unique identifiers for each entity type (e.g., for organizations their website URL functions as a unique identifier and for persons their email address).

Furthermore, we have exploited existing sources with openly available metadata records in order to pre-populate the ELG catalogue with entries for the related entity types in order to (a) facilitate the registration process of the LRTs, and (b) provide additional information without requesting this from the metadata providers. For R1, we have used information from two sources:

- the CORDIS data set of EU projects funded under the H2020 framework[27], from which we manually extracted a subset of LT-related projects, and converted their metadata into the ELG schema for projects;
- the GitHub repository of the SPDX Licence List[28], an aggregated list of commonly found free and open source licenses from the Linux Foundation; the conversion of their metadata from the SPDX JSON format to the ELG internal JSON format was performed with an internal script.

4.3.4 Current Contents

The ELG catalogue of R1 includes the following set of public metadata records:

- 173 tools/services: these are all ready-to-deploy services provided by the ELG consortium partners (see D4.1 for a detailed report)
- 280 data resources, which consist of 249 data sets (corpora), 21 lexical/conceptual resources, 10 language descriptions (grammars and models) (see D5.1 for more information)
- 10 projects, with information derived as described in Section 4.3.3
- 9 organizations (i.e., the ELG project consortium partners).

All metadata records contain at least the information required by the minimal version of the schema and a large set of them, especially those harvested from catalogues, a richer set of information converted from the original metadata records. These contents help show the full functionality potential of the ELG platform.

[27] https://data.europa.eu/euodp/en/data/dataset/cordisH2020projects
[28] https://github.com/spdx/license-list-data

4.4 Monitoring and Analytics

The catalogue backend implements a statistics/analytics module for collecting usage information. Currently, the information collected includes the overall number of views for a given metadata record and number of downloads/times used for a resource. User-specific actions are also tracked; these pertain to the downloads and service executions performed by a user. The statistics/analytics module will be extended in future releases.

For user downloads the information collected includes:

- username;
- date/time of download;
- relevant metadata record;

Service execution statistics include

- username;
- service name;
- execution start date/time;
- execution end date/time;
- number of bytes processed;
- execution status (succeeded/failed);
- calling client type.

4.5 Licensing and Billing

Once fully operational, the ELG platform will also provide access to resources distributed with commercial licences. This requires a billing strategy for the ELG platform and for the LRTs it makes available, as well as metadata records that indicate their licensing terms. Appropriate technical safeguards must be implemented to ensure that access to LRTs is granted in accordance with the licensing terms, where possible; for instance, access to LRTs distributed with restricted licences must be made available only to users that fulfil the criteria specified in the licences. The ELG licensing and billing module is currently under design.

For this reason, in R1, the emphasis has been on LRTs and functionalities that can be provided without restrictions. This has led to the following decisions:

- for LT services, only the trial functionality is available and only for registered users (with quotas),
- for data resources, the first round of collection concentrated on resources that use open licenses (e.g., Creative Commons) and that are available free of charge.

5 The ELG Frontend

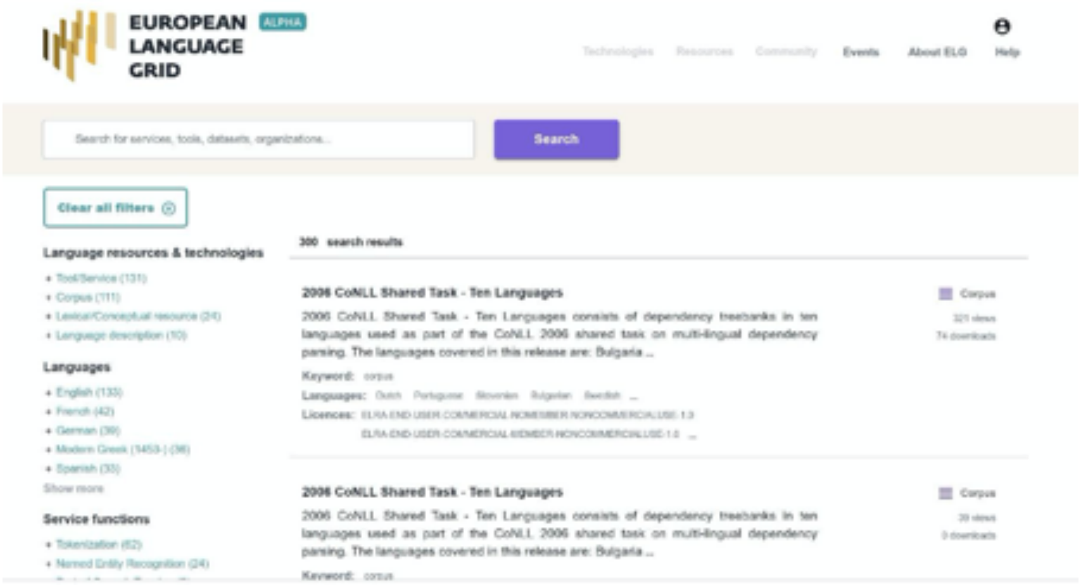
The ELG frontend consists of (a) the “website” pages, maintained in the Drupal CMS and providing information on the LT domain and activities, the ELG project, etc., and (b) the catalogue UI that enables users to interact with the catalogue contents (e.g., search, view metadata, upload). The website part is described in detail in Deliverables D3.1 and D3.2. This section focuses on the catalogue UI.

5.1 Catalogue UI

The catalogue UI is implemented as a Single Page Application (SPA), which provides fast page updates instead of unnecessary full page reloads. All functionalities are built using React, a JavaScript library for the development of user interfaces; the implementation details are presented in D3.2.

In ELG R1 the following functionalities are offered via the catalogue UI:

- **Browse**: The browse/search page (Figure 8) provides a list of all published entries of the ELG catalogue (e.g., LT tools/services, corpora, language descriptions, organizations, etc.) accompanied with a snippet of its description, a set of tags that have been identified as useful for consumers (languages, keywords and licences) and popularity indicators (number of views and downloads/usage).



EUROPEAN LANGUAGE GRID ALPHA

Technologies Resources Community Events About ELG Help

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

Search

Clear all filters

Language resources & technologies
- Tool/Service (131)
- Corpus (111)
- Lexical/Conceptual resource (24)
- Language description (10)

Languages
- English (133)
- French (42)
- German (39)
- Modern Greek (1453-) (36)
- Spanish (33)

Show more

Service functions
- Tokenization (62)
- Named Entity Recognition (24)

300 search results

2006 CoNLL Shared Task - Ten Languages

Corpus
321 views
74 downloads

2006 CoNLL Shared Task - Ten Languages consists of dependency treebanks in ten languages used as part of the CoNLL 2006 shared task on multi-lingual dependency parsing. The languages covered in this release are: Bulgaria ...

Keyword: corpus

Languages: Dutch Portuguese Slovenian Bulgarian Swedish ...

Licences: ELRA-END-USER-COMMERCIAL-NOMEMBER-NONCOMMERCIALUSE-1.0
ELRA-END-USER-COMMERCIAL-MEMBER-NONCOMMERCIALUSE-1.0 ...

2006 CoNLL Shared Task - Ten Languages

Corpus
39 views
0 downloads

2006 CoNLL Shared Task - Ten Languages consists of dependency treebanks in ten languages used as part of the CoNLL 2006 shared task on multi-lingual dependency parsing. The languages covered in this release are: Bulgaria ...

Keyword: corpus

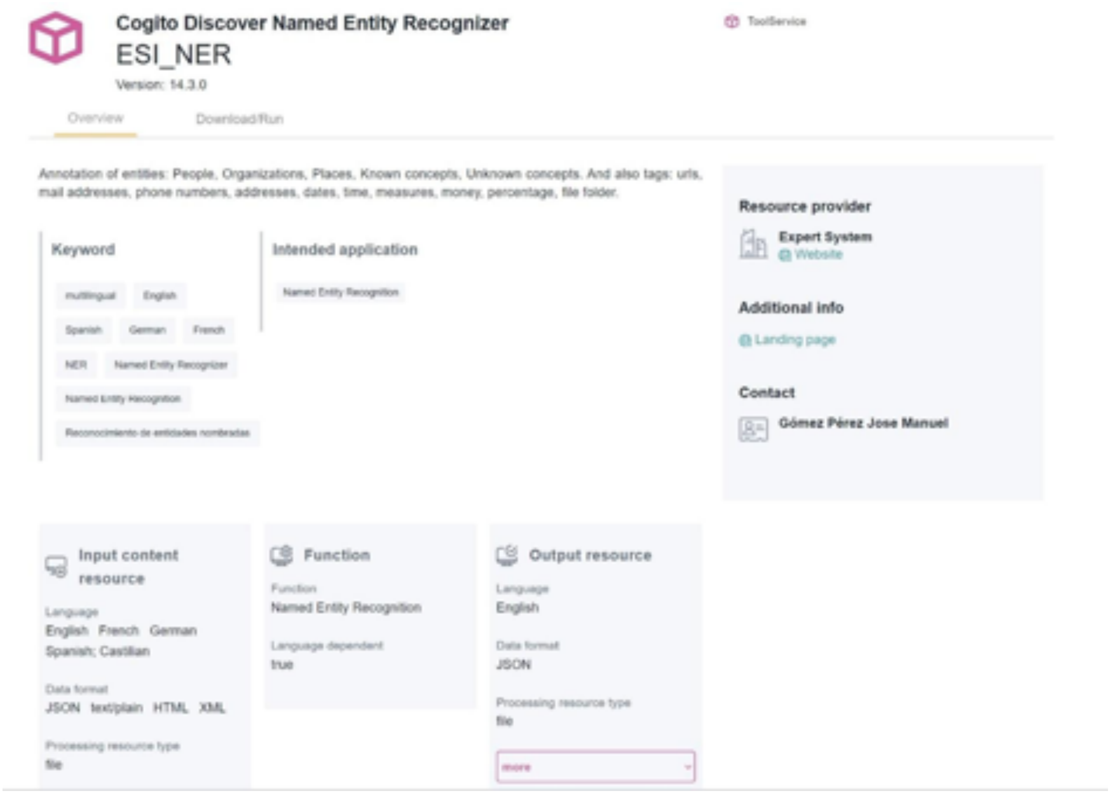
Figure 8: Browse/search page of the ELG catalogue

- **Search**: A user enters a keyword or phrase (Figure 9) in a search box and the application narrows down the results dynamically; the minimum number of characters a user has to type to refine the search results is set to four. The search is performed on the indexed metadata elements (Section 4.1.6).

Figure 9: Keyword search

- **Faceted search**: Users can filter the full catalogue contents or previous search results by selecting values from the facets on the left side of the browse/search page (Figure 8). The facets have been selected based on user preferences (D3.2). Facets and free text search can be combined in order to refine search queries and support users in easily finding the resources they desire. Faceted search provides a long list of sub-sections (facets) that can be explored gradually.
- **View published metadata entries**: By clicking the title of an entry, users can view their full description; Figure 10 provides the landing page of a tool/service. For R1, landing pages have been implemented

for all LRT types as well as for Organizations and Projects. The design of the landing pages takes into consideration user preferences (see D3.1), design and accessibility considerations and the ELG metadata schema. As described in more detail in D3.2, the information is grouped in semantically similar sets and organized in tabs and specific areas of the page layout.


Cogito Discover Named Entity Recognizer

ESI_NER

Version: 14.3.0

ToolService

Overview | Download/Run

Annotation of entities: People, Organizations, Places, Known concepts, Unknown concepts. And also tags: urls, mail addresses, phone numbers, addresses, dates, time, measures, money, percentage, file folder.

Keyword

multilingual · English · Spanish · German · French · NER · Named Entity Recognizer · Named Entity Recognition · Reconocimiento de entidades nombradas

Intended application

Named Entity Recognition

Resource provider

Expert System · Website

Additional info

Landing page

Contact

Gómez Pérez Jose Manuel

Input content resource

Language: English French German Spanish; Castilian

Data format: JSON text/plain HTML XML

Processing resource type: file

Function

Function: Named Entity Recognition

Language dependent: true

Output resource

Language: English

Data format: JSON

Processing resource type: file

more

Figure 10: Landing page of a tool/service

- **Download resources**: All users are able to download resources directly from the ELG storage system in accordance with their licensing terms. Downloading is offered via the Download tab, which displays all distributions of a resource and information on the licensing and billing conditions for each of them. When clicking the "download" button, users are prompted to view and accept the licence text before downloading the resource.
- **Try out LT services**: An important and rather unique feature that the ELG platform provides to the users is that they can test the ELG-compliant LT services via pages called "Try out" UIs. For R1, this feature is reserved only to registered users. The "Try out" UIs are separate HTML pages that use JavaScript code and are embedded as iframes[29] in the catalogue UI. They are responsible for sending and receiving data to and from the LT execution server and for visualizing the processing execution results[30]; Figure 11 shows an example. Of course, the "Try out" tab is provided only for those services that are integrated into the ELG platform; this information is provided by the ELG backend. The ELG backend also provides the LT service execution endpoint while the authentication token that is required is retrieved from the user's active keycloak session.

[29] https://www.w3schools.com/tags/tag_iframe.asp
[30] The "Try out" UIs run in containers that are deployed in the same cluster as all the other ELG components.

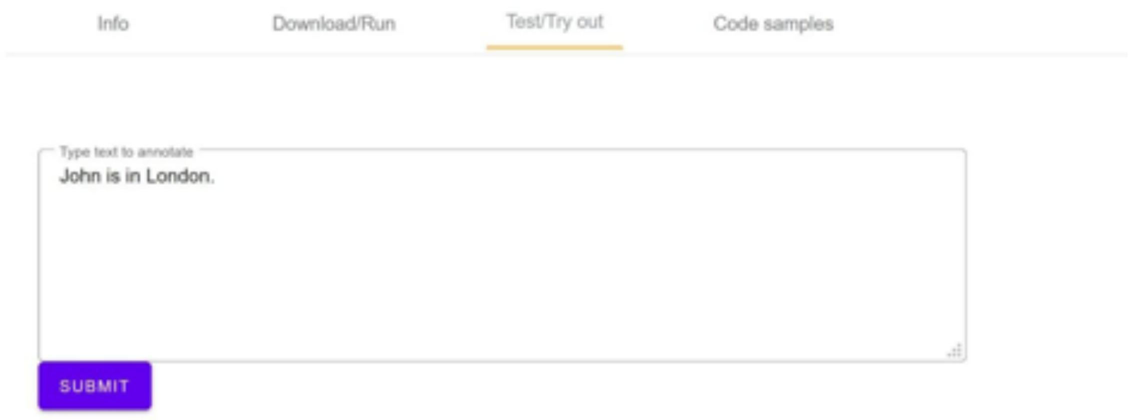


Figure 11: Try out GUI for Machine Translation services

- **Upload metadata descriptions**: Users with the "provider" role can upload metadata descriptions of their resources. For the current release, they can upload XML files compliant with the ELG schema; if the metadata file (in XML format) is valid, the user receives a "success" message; if it's invalid, the validation errors are aggregated and returned in a single message. As already mentioned, for R1, resource uploading functionalities are available only to platform administrators.
- **Administration pages**: Access to the administration pages, implemented with Django as a separate application, is available only for authorized users (the platform administrators) via a top-level menu.

5.2 Integration with the Website

The current entry point for all users of the ELG platform is https://live.european-language-grid.eu. This page brings together the website part and the catalogue. Access to the catalogue is available through the "Search/Explore Catalogue" button.

The catalogue UI and the ELG website are implemented as two different applications serving different purposes; however, the integration of the two systems is required for some functionalities. For instance, when a user enters a search query in the website's search field and presses the "Search/Explore catalogue" button, they are redirected to the catalogue UI where the results of the query are displayed.

In addition, a common "look and feel" should be adopted by both applications. The catalogue UI follows the overall design concept of the ELG website and corporate identity to achieve this. Consequently, they both share a similar page layout, with common header and footer sections, and menu items. To integrate these sections, they both consume a REST API provided by the CMS. This API provides the information about which links should be displayed and which links should be enabled and disabled. Finally, both applications make use of common UI frameworks (Google Material design), common icons, fonts and colors.

6 Conclusions

In this deliverable we present Release 1 of the ELG platform. This release comes with an important set of functionalities and mechanisms, demonstrating the potential capacities of ELG, once fully operational. Users can browse through the catalogue, search for specific LT services and resources, view them, test LT services and download resources with open licences. Providers can describe and upload their resources and integrate LT services into the platform. The catalogue has already been populated with a substantial number of LT services and resources, and seeded with a small set of descriptions of stakeholders and projects. We have designed and implemented APIs for the execution of LT services, and the interaction with the platform. Upcoming releases will enhance the platform with improvements of the existing functionalities and the addition of new ones.

7 References

Penny Labropoulou, Katerina Gkirtzou, Maria Gavriilidou, Miltos Deligiannis, Dimitris Galanis, Stelios Piperidis, Georg Rehm, Maria Berger, Valérie Mapelli, Michael Rigault, Victoria Arranz, Khalid Choukri, Gerhard Backfried, José Manuel Gómez Pérez, and Andres Garcia-Silva. "Making Metadata Fit for Next Generation Language Technology Platforms: The Metadata Schema of the European Language Grid". In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Christopher Cieri, Khalid Choukri, Thierry Declerck, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*, Marseille, France, 2020. European Language Resources Association (ELRA). Accepted for publication.

Georg Rehm, Maria Berger, Ela Elsholz, Stefanie Hegele, Florian Kintzel, Katrin Marheinecke, Stelios Piperidis, Miltos Deligiannis, Dimitris Galanis, Katerina Gkirtzou, Penny Labropoulou, Kalina Bontcheva, David Jones, Ian Roberts, Jan Hajic, Jana Hamrlová, Lukáš Kačena, Khalid Choukri, Victoria Arranz, Andrejs Vasiļjevs, Orians Anvari, Andis Lagzdiņš, Jūlija Meļņika, Gerhard Backfried, Erinç Dikici, Miroslav Janosik, Katja Prinz, Christoph Prinz, Severin Stampler, Dorothea Thomas-Aniola, José Manuel Gómez Pérez, Andres Garcia Silva, Christian Berrío, Ulrich Germann, Steve Renals, and Ondrej Klejch. "European Language Grid: An Overview". In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Christopher Cieri, Khalid Choukri, Thierry Declerck, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*, Marseille, France, 2020. European Language Resources Association (ELRA). Accepted for publication.

A. LREC 2020 Papers on the ELG Platform and the ELG Metadata Schema

Back in September/October 2020, the ELG project prepared three scientific articles, which were submitted to the conference LREC 2020. All paper submissions were accepted by the LREC 2020 Programme Committee. While one paper is referenced and included in Deliverable D7.8, the other two are referenced in this deliverable (see above, "References"). The papers are fully included in the appendix (see below).