API use cases

In addition to providing general public access to the database, the REST API can also be used to integrate data and services with external organisations in a coordinated, structured and systematic way. Two current examples of this are integration with terminology portals and speech technologies, both of which use a mix of public (read-only) and restricted (read-write) routes of the API.

Povejmo

One of the goals of the project is to enable a large language model to learn Slovene grammar. The training dataset will be largely based on data obtained from the Digital Dictionary Database, and the goal is to extract and incorporate as much data as possible, so that the model can also learn specific nuances.

To that end, the API's various `/export` and `/retrieve-batch` routes are particularly relevant, as they enable first finding all the relevant (single-word or multi-word) unit IDs and then different kinds of data for all the words. For example:

- Find the IDs of the first chunk of single-word lexical units in the database, using /export/lexical-units/ with `"type"="single_lexeme_unit"`.
- Accumulate all the IDs by repeatedly following the `"next"` links until next=null.
- Break the lexical unit IDs into batches that can be used in /retrieve-batch calls.
- For each batch of lexical unit IDs, retrieve associated data using:
  - /retrieve-batch/lexical-unit-lexemes/ with `"extra-data"=["forms-orthography"]` to get each lexical unit's lexeme along with all of its morphological forms.
  - /retrieve-batch/lexical-unit-definitions/ to get all definitions of different kinds for the lexical units' senses.
  - /retrieve-batch/lexical-unit-sense-relations/ with `"type"="synonym"` to get all the synonyms for the lexical units' senses.
  - /retrieve-batch/lexical-unit-collocations/ to get collocations.
- To get corpus examples, use the separate /export API calls (and following pagination, as above) rather than `/retrieve-batch/`, for performance reasons:
  - /export/lexical-unit-sense-examples/ with `"type"="single_lexeme_unit"` for the single-word lexical units.
  - /export/lexical-unit-sense-examples/ with `"type"="collocation"` for the collocations.
  -

Terminology Portal

One of the main parts of the Development of Slovene in a Digital Environment is a terminology portal that will feature various terminological resources and offer an openly accessible tool for term extraction from specialized corpora, as well as the server infrastructure needed to create new terminological resources. The main components of the portal include a search engine for all integrated resources and a terminology resource editor, and the resources are designed to be easily integrated with other language tools and services, including the Digital Dictionary Database.

As such, the portal uses API routes to register its dictionaries in the database, search and create terms, attach/detach them to/from the dictionaries, and fetch their forms and statuses. The API supports this as follows (see the route links for full examples):

- Register the dictionary as a resource in the database using /get-or-create/resource/, providing a code name for the dictionary (e.g., `"slm"`) as input. The API will return the resource's ID (e.g., `87`), first creating it if it does not yet exist.
- Get IDs of terms in the database, creating them if necessary, using /get-or-create/lexical-unit/. Input can be either the term's raw string (e.g., `"okrogla miza"`), or its pre-analysed sequence of tokens, with each token represented with corpus-style data (e.g., `[{"lemma":"okrogel", "msd":"Ppnzei", "form":"okrogel"}, {"lemma":"miza", "msd":"Sozei", "form":"miza"}]`). If a raw string is provided, the API uses a standard tool to get a sequence of tokens itself. Either way, it then checks if a matching lexical unit exists in the database, creates one if necessary, and returns its basic data, including its ID (e.g., `54321`). If many terms need processing, this can be done by using the /get-or-create-batch/lexical-unit/ call instead and providing a list of inputs.
- Get the word parts and their forms with statuses for a specific term, by using /retrieve/lexical-unit-lexemes/. Input would be the term's ID (e.g., `54321`) and specifying that form statuses and all form types are also requested (e.g., `"extra-data":["status-form-types", "forms-orthography", "forms-accentuation", "forms-pronunciation"]`). The output then a list, with each element is one of the term's word constituents, represented with both its basic data (such as id, lemma, part of speech, basic word-level features) and the extra requested data (the list of all the forms of all the types for the word and aggregated statuses for each type).
- Search for term candidates in the datases, using /search/lexical-unit/. This has similarities to /get-or-create/lexical-unit but differs in a few key ways. First, it does not create a lexical unit if no match exists, and thus does not require authentification, so less central components of the portal can also make use of it. Second, it does not require complete data in the input (e.g., perhaps only one or two of lemma/msd/form are specified for one or more of the term's components), making the search more flexible and potentially returning multiple matches (e.g., {"lemma":"klop"}).
- Attach the lexical unit to a resource, using /attach/lexical-unit/. The input would be the IDs of the term as a lexical unit (e.g., `54321`) and dictionary as a resource (e.g., `87`). This

would then connect the two in the database.
- Detach the lexical unit from a resource, using /detach/lexical-unit/(https://blisk.ijs.si/api/redoc/#tag/detach). The input would be the IDs of the term as a lexical unit (e.g., `54321`) and dictionary as a resource (e.g., `87`). This would then remove the term from the resource in the database, without deleting the term from the database in general.

Speech technologies

The project Tolmač (Eng. Interpreter) is focused on developing of a system for automatically translating lectures from Slovene to other languages, coordinated at the Faculty of Computer and Information Science at the University of Ljubljana, in close collaboration with the Centre for Language Resources and Technologies. The results of the project will be important for a wide range of people: real-time translations will make it easier for foreign students to follow lectures in Slovene, automatic subtitles will help people will hearing loss, and lecture excerpts and recordings will be accessible at a dedicated website. The speech technologies underlying the system rely on search and retrieval of both orthographic and pronunciation word forms of Slovene words.

To that end, the system can use API routes to preprocess text, search for different kinds of word forms, retrieve the forms of word and create new words along with their forms. The API supports this as follows:

- Parse a piece of Slovene text to get a sequence of tokens using /process-string-to-tokens.

  The API runs the standard CLASSLA parser with default parameters and returns a list of tokens in CoNLL-U format, with each token including a lemma (e.g., `"miza"`), MSD (`"Sozmm"`) and form (e.g., `"mizah"`). Thus this API call does not interact with the database, but serves as a handy wrapper for CLASSLA, so the user does not need to install it themselves.
- Search for a word form in the database using /search/form/, by providing a type (e.g., `"orthography"`) and a string (e.g., `"mizah"`). The output is a list of matching forms, along with basic associated data such as lexeme ID (e.g., `123`), lemma (e.g., `"miza"`) and JOS-system MSD (e.g., `"Sozdm"`). Associated pronunciations for all matching forms are included if requested (e.g., `"extra-data":["forms-pronunciation"]`).
- Get all the forms of a given lexeme using /retrieve/lexeme/, using the lexeme's ID as input. To get all the orthography and pronunciation forms of the lexeme, specify in the input (e.g., `"extra-data":["forms-orthography", "forms-pronunciation"]`).
- Create (if it does not yet exist) a lexeme in the database using /get-or-create/lexeme/. Input consists of a lemma (e.g., `"miza"`) and MSD (e.g., `"Sozmm"`). The MSD can be any appropriate MSD for the lexeme, not necessarily the MSD of the lemma itself, since only the lexeme-level parts of the lemma (e.g., `"Soz"`) will be considered. The API calls the Inflector tool, which generates full paradigms of different kinds of forms (orthography, accentuation, pronunciation), and then saves the new lexeme with its forms in the database. However, if a lexeme already exists in the database which matches the

database, no duplicate lexeme (along with forms) is created and the existing lexeme is returned.

Revision #8
Created 2022-12-02 14:29:19 UTC by Cyprian Laskowski
Updated 2025-04-04 08:17:16 UTC by Cyprian Laskowski