EUROPEAN LANGUAGE GRID

META-NET
META-FORUM 2022

EUROPEAN LANGUAGE EQUALITY

Session 2: ELG platform
Final Release of the ELG Platform: Functional Overview, Demo and Tutorial

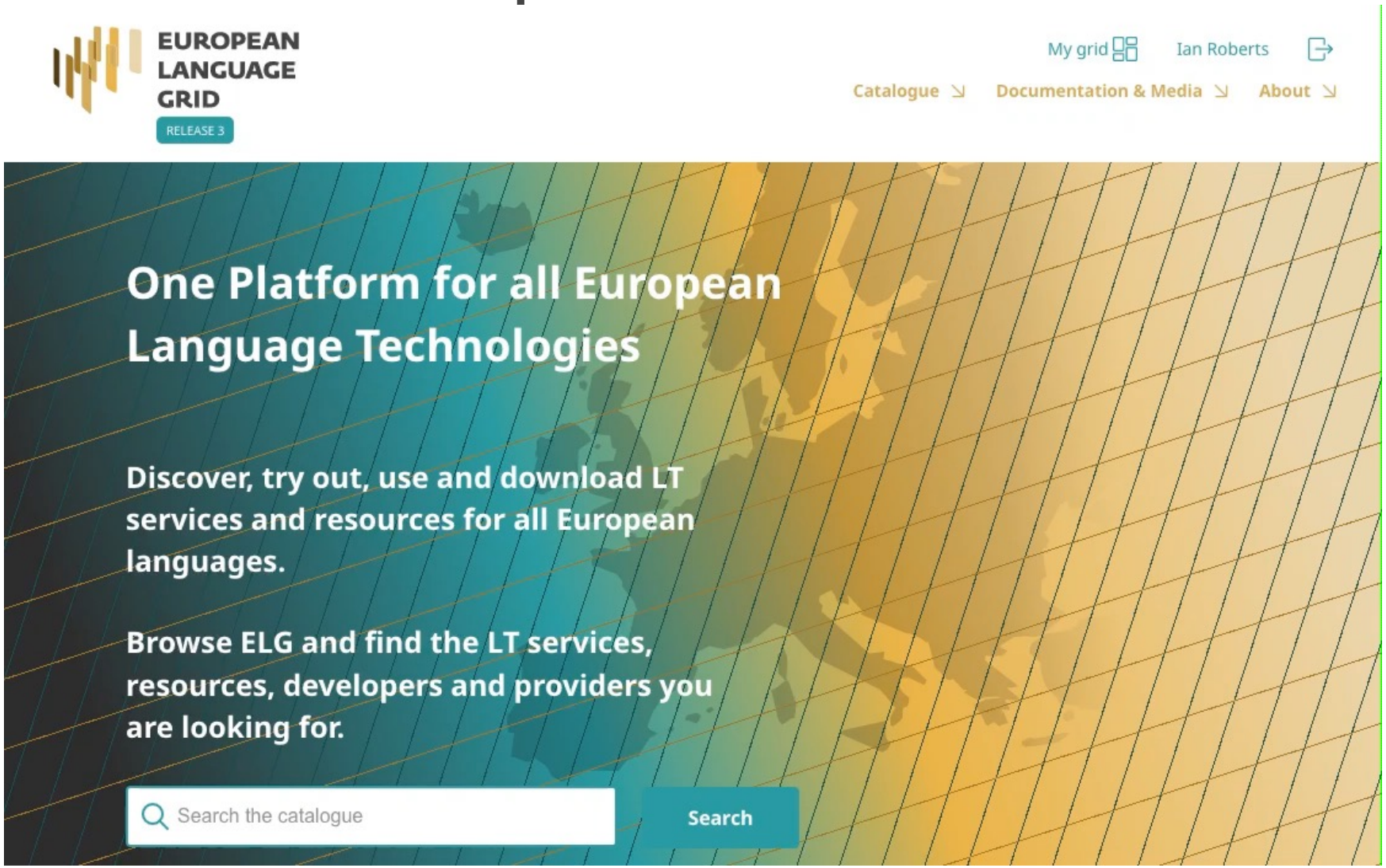
Penny Labropoulo - Ian Roberts (University of Sheffield) - Rémi Calizzano (DFKI)
penny@athenarc.gr - i.roberts@sheffield.ac.uk - remi.calizzano@dfki.de

08/09-06-2022 META-FORUM 2022 – Joining the European Language Grid (hybrid conference)
http://www.european-language-grid.eu – http://european-language-equality.eu

Tools and services in the ELG platform

- Additional metadata elements specific to tools and services include:
  - Service function, e.g. speech recognition, part of speech tagging, …
  - Software distribution details
    - Always “Docker image” for integrated services, other options include links to the source code repository
  - Hardware requirements (RAM, CPU, GPU, …)
  - Input and output formats, (human) languages, annotation types, sample input data
  - Any additional parameters

Tools and services in the ELG platform



Publishing your service on the ELG platform

1. Implement the appropriate API
   - ELG specifies flexible APIs for LT services
     - Input types: text, audio, images
     - Output formats: text, audio, “annotations”, classification
   - Helper tools available for Python & Java
2. Package your code as a container image
   - If you use helpers, you get this for free
   - Otherwise there are many existing open-source services to use as a template
3. Describe your service with ELG metadata
   - Including technical metadata giving image location, port number, required RAM/CPU, etc.
4. Submit for validation

1. Implement the appropriate API

- HTTP APIs based on simple JSON message structures
  - Binary data – audio & image – handled like web file upload forms
- Mix-and-match input and output types as required for your tool
  - Text in / text out – translation, summarization, …
  - Text in / annotations out – named entity recognition, semantic annotation, …
  - Audio in / text out – speech recognition
  - Audio in / annotations out – speaker diarisation, keyword detection, …
  - Image in / text out – OCR, image labelling
  - Text in / audio out – Text-to-speech
  - etc. etc.

1. Implement the appropriate API

- ELG provides helper tools for popular languages (Python, Java)
  - See the platform documentation
- These handle the details of HTTP server, JSON parsing and serialization, error handling
- You can concentrate on your business logic
- If your tool already has an API you can implement a lightweight “adapter”

2. Package your code as a container image

- Standard tools for this, most notably Docker
- Example dockerfiles available in ELG documentation
- Python & Java helpers generate the boilerplate for you
- Many existing ELG tools are open source with permissive licences, feel free to use them for inspiration

3. Describe your service with ELG metadata

- Interactive editor guides you through the process
- Key items:
  - Name & description, contact details for provider
  - Classification metadata – function, keywords, …
  - Documentation links
  - Input & output formats and languages
  - Technical metadata
    - Image location, endpoint address within the image
    - Unusual hardware requirements (lots of RAM or CPU)
  - Legal metadata
    - Licence / terms of use

4. Submit for validation

- All ELG submissions go through manual validation to assess licence and metadata coherence
- Services also go through “technical validation”
  - ELG team deploys your service to our cluster, assigns “try out” UI, tests the service with your example data
  - Iterative process – validator may have questions or suggested changes
- We want to help! Validator will work with you to address any issues
- Once all sides are happy, service is published
  - ELG-hosted services will be assigned a DOI

The ELG Python client https://gitlab.com/european-language-grid/platform/python-client

- Easy installation using pip:

```
> pip install elg
```

- Allow the usage of the ELG APIs through Python
  - Search the ELG
  - Download hosted corpora
  - Call compatible services

- Provide tools to facilitate the interactions with the ELG
  - Create an ELG compatible service from a Python tool
  - Deploy ELG services locally

DEMO 1: Basic function of the Python client

localhost:8889/notebooks/QuickStart.ipynb#

jupyter QuickStart (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3 (ipykernel)

Run Markdown

Quickstart

The SDK contains a class for each main ELG functionality.

The `Catalog` class is for browsing the catalogue, the `Entity` class is for representing an ELG entity (i.e., an ELG resource), the `Service` class is for using the ELG services, etc...

These classes can be imported directly from the `elg` package as follows:

In [ ]:
```
from elg import Catalog, Entity, Service, Authentication, Corpus
```

Browsing the catalogue

First you have to init a `Catalog` object.

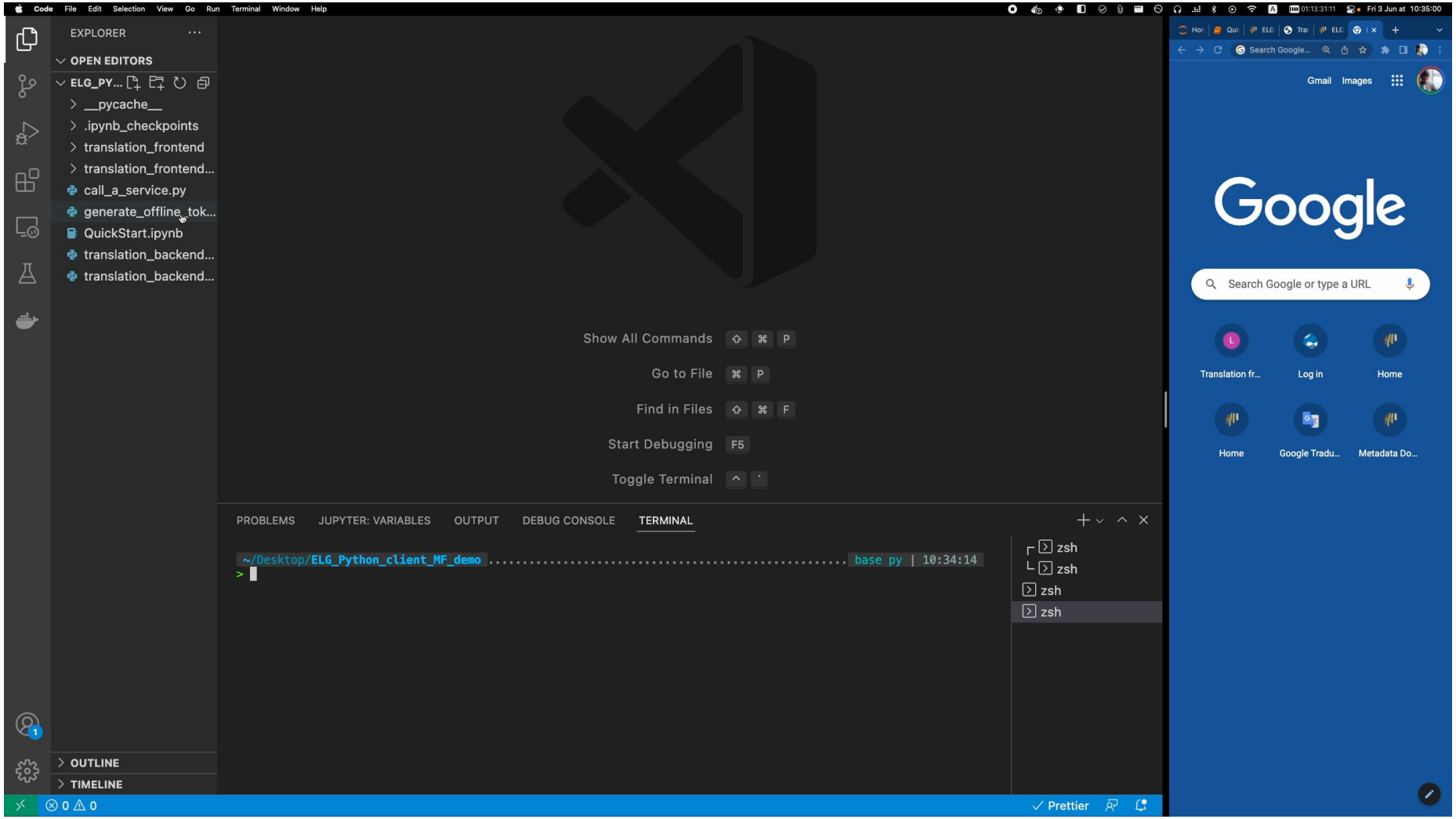
Then you can use the search method to search for resources. This method returns a list of `Entity` which can be displayed individually.

For example, we can search for a Machine Translation service for English and French.

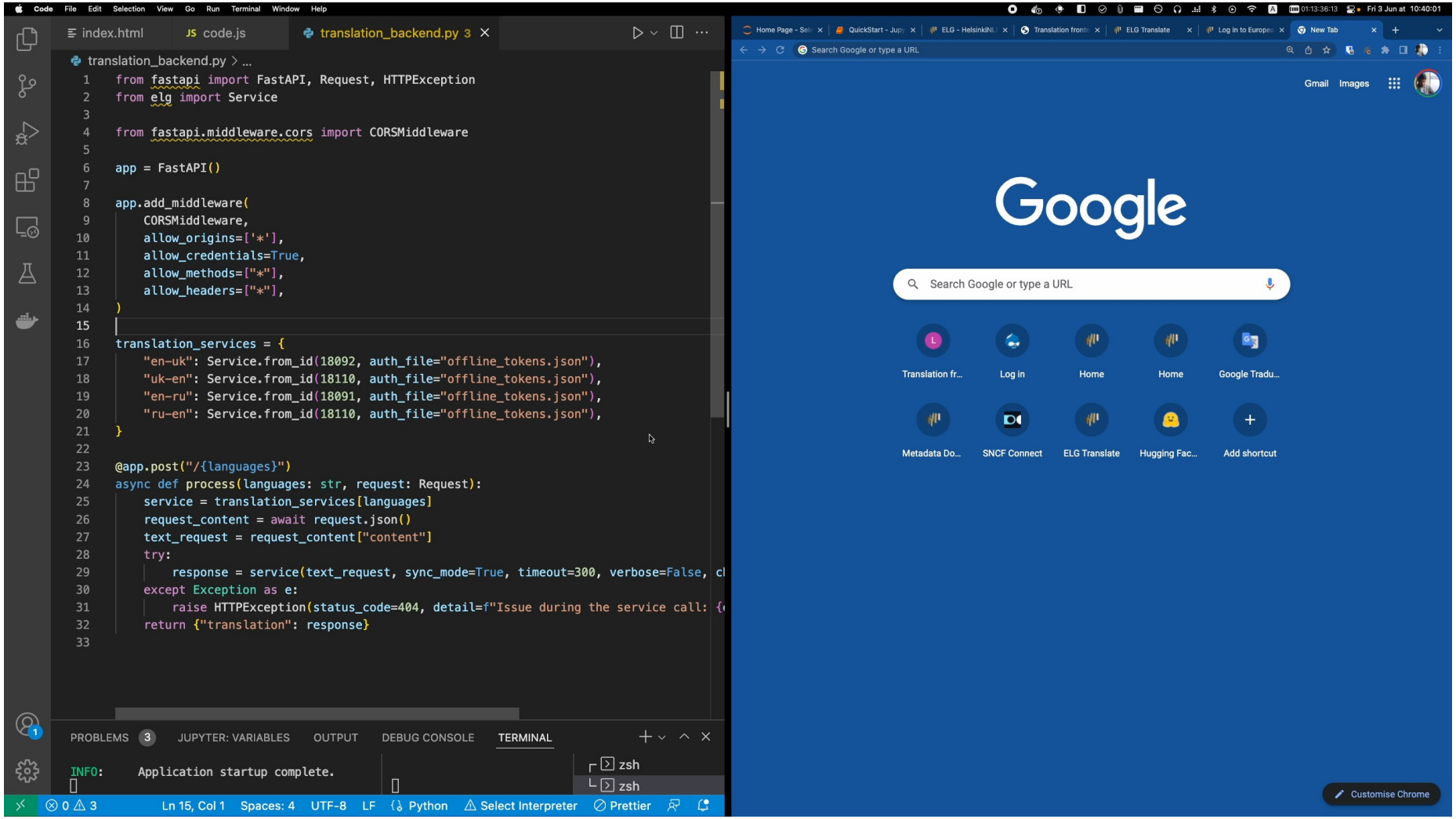
In [ ]:
```
catalog = Catalog()

# Search and get the result as a list of Entity
results = catalog.search(
    resource = "Tool/Service", # "Corpus", "Lexical/Conceptual resource" or "Language description"
    function = "Machine Translation", # function should be pass only if resource is set to "Tool/Service"
    languages = ["en", "fr"], # string or list if multiple languages
    limit = 100,
)
print(f"Machine Translation service for English and French:\n{list(results)[0]}")
```

DEMO 2: Build a translation system using ELG services and Python (1)



DEMO 2: Build a translation system using ELG services and Python (2)



DEMO 2: Build a translation system using ELG services and Python (3)

translation_backend_2.py > process

```python
from fastapi import FastAPI, Request, HTTPException
from elg import Service

from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=['*'],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

translation_services = {
    "en-uk": Service.from_id(18092, auth_file="offline_tokens.json"),
    "uk-en": Service.from_id(18110, auth_file="offline_tokens.json"),
    "en-ru": Service.from_id(18091, auth_file="offline_tokens.json"),
    "ru-en": Service.from_id(18110, auth_file="offline_tokens.json"),
    "t2s_en": Service.from_id(4837, auth_file="offline_tokens.json"),
}


@app.post("/{languages}")
async def process(languages: str, request: Request):
    service = translation_services[languages]
    request_content = await request.json()
    text_request = request_content["content"]
    if languages == "t2s_en":
        try:
            response = service(text_request, sync_mode=True, timeout=300, verbose=Fals
            path = f"{response.content[:20].replace('/', '_')}.wav"
            response.to_file(f"translation_frontend_2/audios/{path}")
        except Exception as e:
            raise HTTPException(status_code=404, detail=f"Issue during the service ca
        return {"path": path}
    try:
        response = service(text_request, sync_mode=True, timeout=300, verbose=False,
    except Exception as e:
        raise HTTPException(status_code=404, detail=f"Issue during the service call:
    return {"translation": response}
```

```
INFO:     127.0.0.1:63653 - "POST /t2
s_en HTTP/1.1" 200 OK
```

localhost:3000

Translation frontend

Translate from English to Ukrainian

translate

Translate from Ukrainian to English

translate

listen

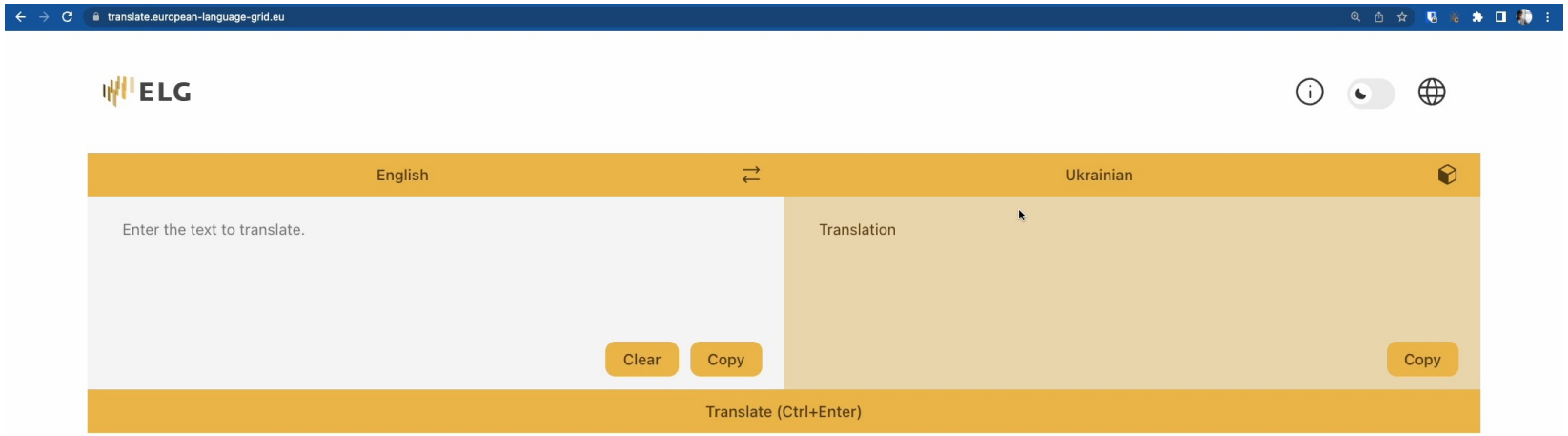
Translate from English to Russian

translate

Translate from Russian to English

translate

DEMO 3: translate.european-language-grid.eu



The European Language Grid has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 825627 (ELG)

ELG services, some use cases in real world applications

- Detect hate speech in comments to filter them
- Translate conversations between users
- Add speech synthesis to a chatbot
- Extract information from documents
- Many many more …

→ Easy integration with ELG APIs and the ELG Python client

European Language Grid

European Language Equality

Thank you!

The European Language Grid has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement № 825627 (ELG).
The European Language Equality project has received funding from the European Union under grant agreement № LC-01641480 – 101018166 (ELE),

Penny Labropoulo - Ian Roberts (University of Sheffield) - Rémi Calizzano (DFKI)
penny@athenarc.gr - i.roberts@sheffield.ac.uk - remi.calizzano@dfki.de

08/09-06-2022 META-FORUM 2022 – Joining the European Language Grid (hybrid conference)
https://www.european-language-grid.eu – https://european-language-equality.eu