



BIODT
biodiversitydigitaltwin

School & Hackaton

March 3rd – March 7th
Officine Cantelmo, Lecce IT

LEXIS Platform

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava



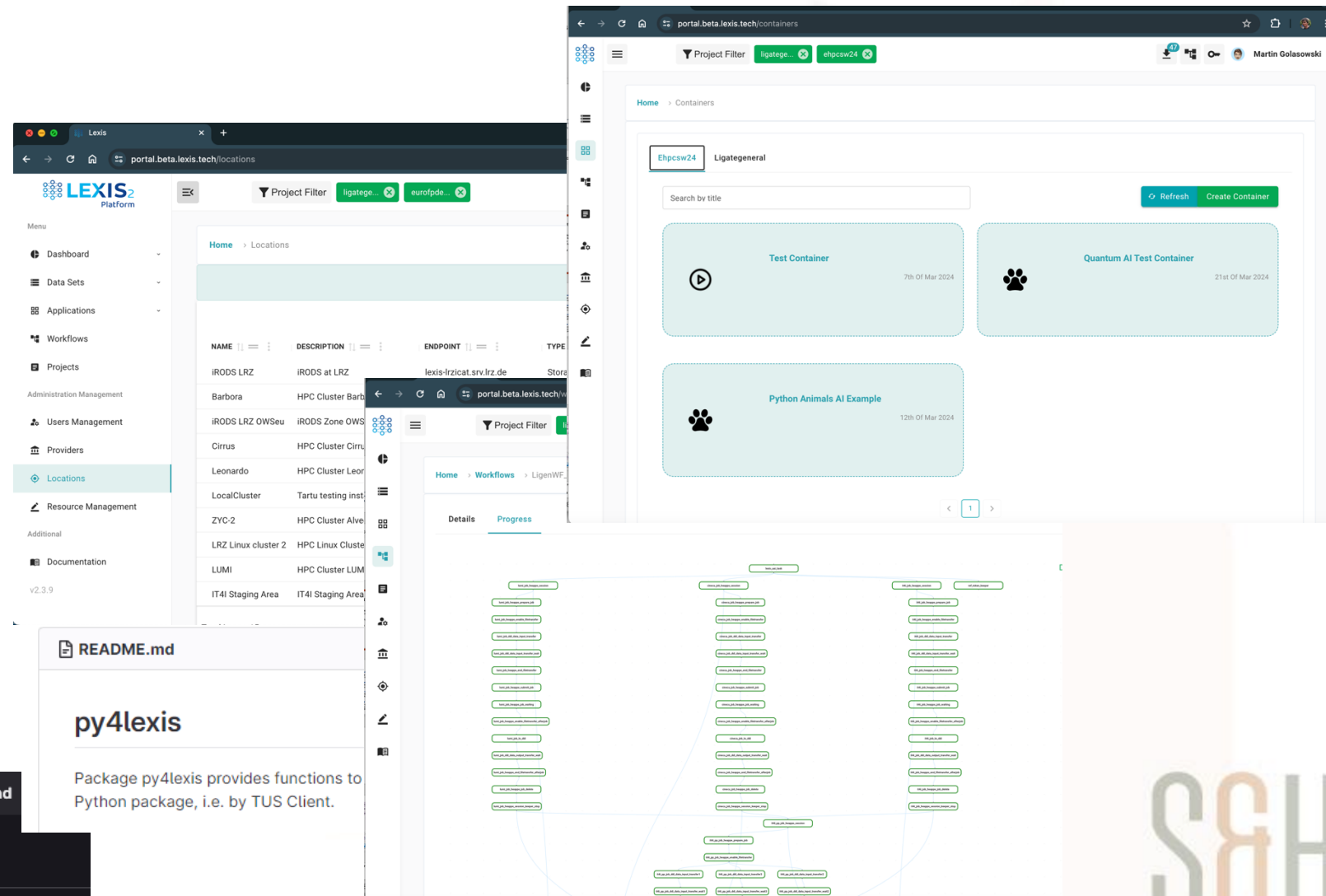
Funded by
the European Union

- 🔥 Pick a supercomputer
- 🔥 Fill out a request for allocation
- 🔥 Get it approved
- 🔥 Set up an account
- 🔥 Set up a SSH key
- 🔥 Know how to use Linux terminal
- 🔥 Login in to the Supercomputer
- 🔥 Learn how to launch jobs in SLURM
- 🔥 Learn about modules
- 🔥 Learn about storages and data transfer
- 🔥 ... finally compute something



- 🔥 Nice web interface
- 🔥 Execution through API
- 🔥 Allocations at one place
- 🔥 Across multiple clusters
- 🔥 Get logs
- 🔥 Manage data
- 🔥 Use common web login
- 🔥 ... and much more

Visit for more:
docs.lexis.tech
opencode.it4i.eu/lexis-platform

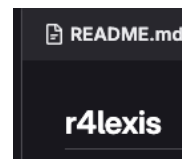
The image displays three overlapping screenshots of the LEXIS Platform web interface. The top-left screenshot shows the 'Locations' page with a table of resources:

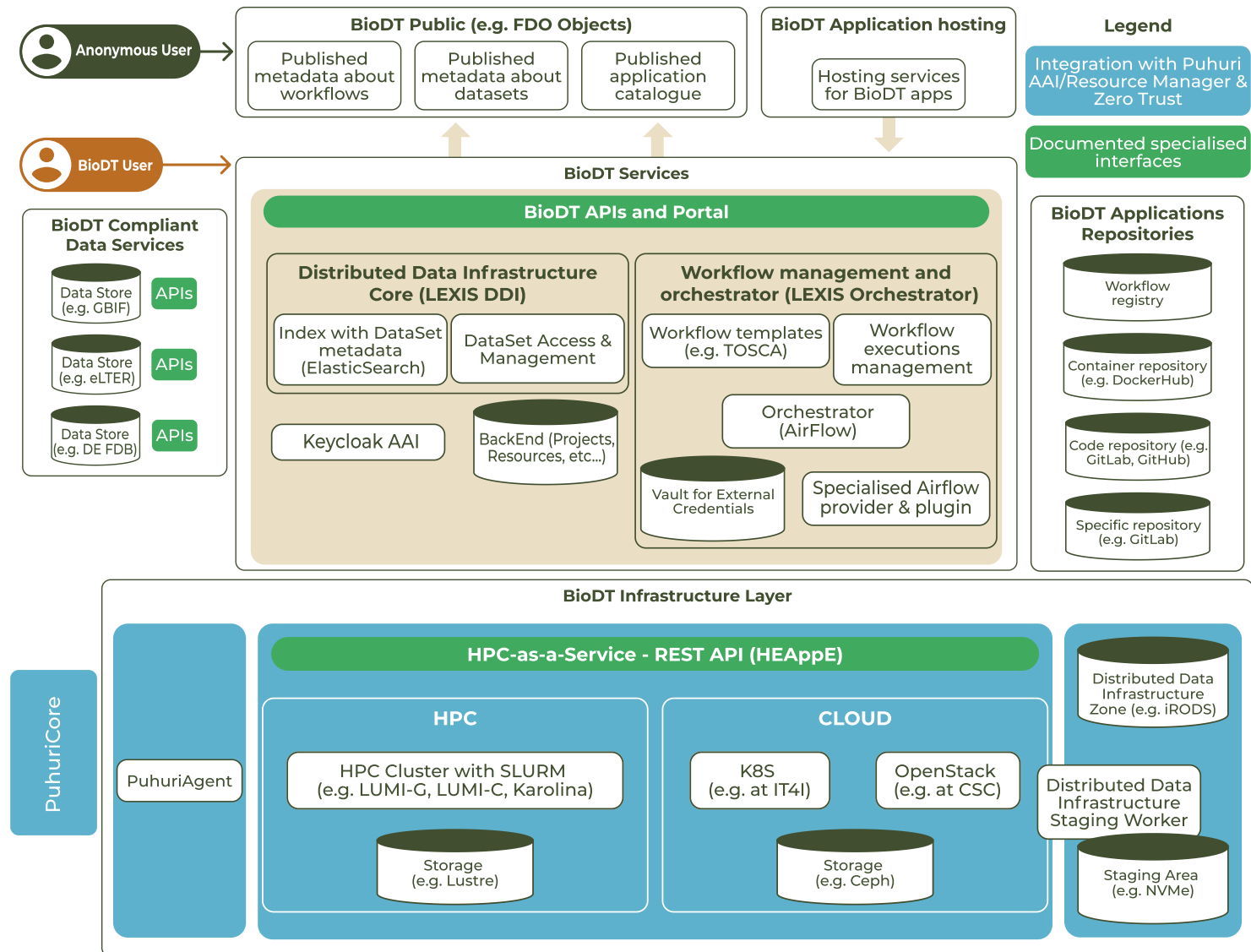
NAME	DESCRIPTION	ENDPOINT	TYPE
iRODS LRZ	iRODS at LRZ	lexis-lrzicat.srv.lrz.de	Storage
Barbora	HPC Cluster Barb...		
iRODS LRZ OWSeu	iRODS Zone OWS...		
Cirrus	HPC Cluster Cirr...		
Leonardo	HPC Cluster Leon...		
LocalCluster	Tartu testing inst...		
ZYC-2	HPC Cluster Alve...		
LRZ Linux cluster 2	HPC Linux Cluste...		
LUMI	HPC Cluster LUMI...		
IT4I Staging Area	IT4I Staging Area		

The top-right screenshot shows the 'Containers' page with a list of containers:

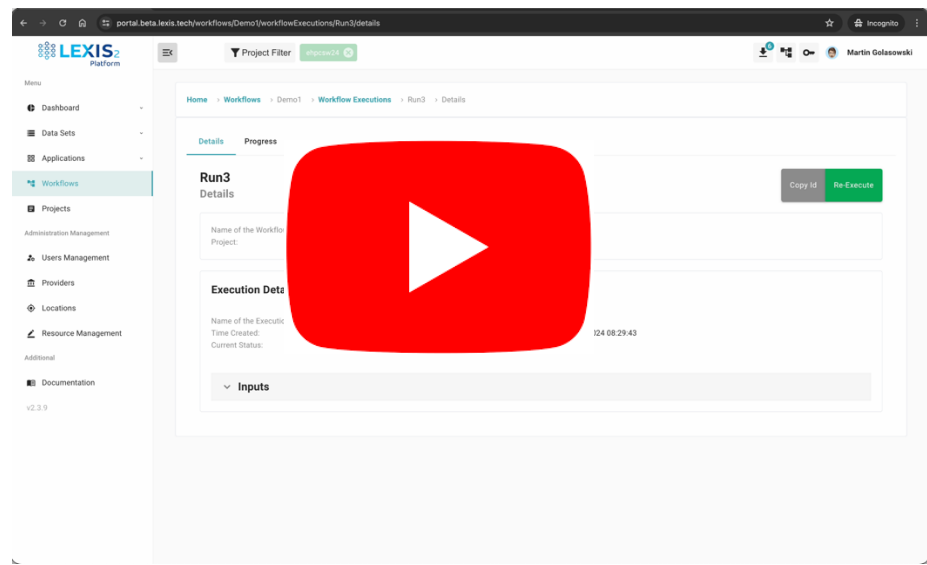
- Ehpcsw24** (Ligategeneral): Search by title, Refresh, Create Container
- Test Container**: 7th Of Mar 2024
- Quantum AI Test Container**: 21st Of Mar 2024
- Python Animals AI Example**: 12th Of Mar 2024

The bottom screenshot shows a 'Workflows' page with a complex flowchart diagram illustrating a data processing pipeline.



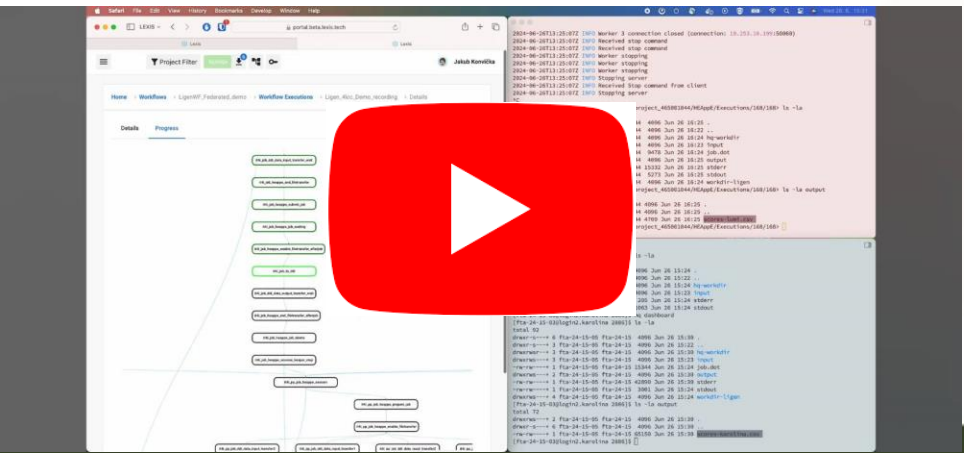


AI workflow demo with Python in Apptainer

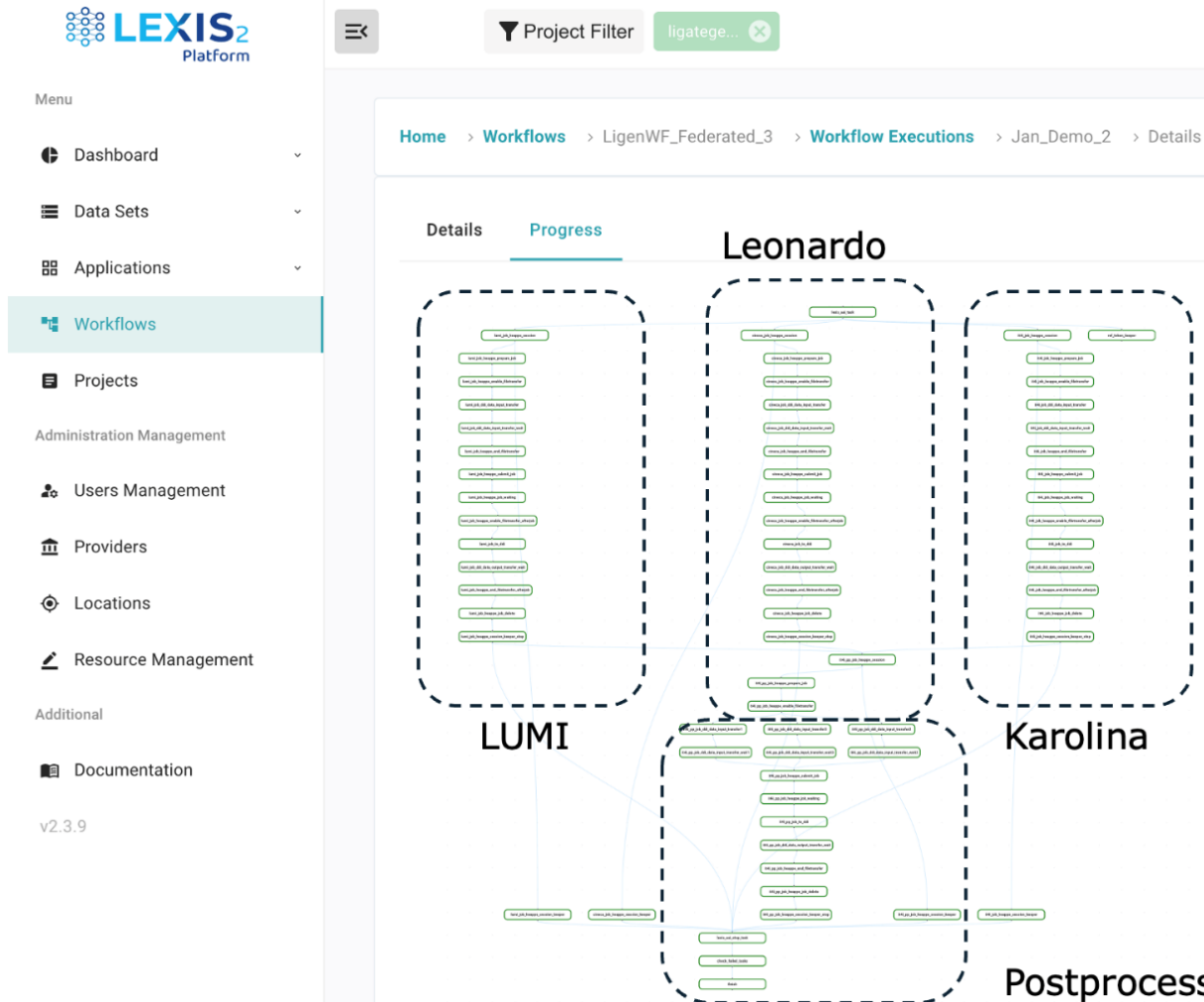


[Check it out!](#)

Urgent computing federated workflow execution



[Check it out!](#)



Executed on 3 locations

LUMI

Leonardo (CINECA)

Karolina (IT4I)

A screenshot of a web browser displaying the README.md file for the py4lexis package. The browser's address bar shows the file path. The README content includes the package name 'py4lexis', a description of its functionality for managing datasets within the LEXIS platform, and an 'Install' section. The 'Install' section recommends using a virtual environment and provides a code block with the command 'python -m venv ./venv'. Below this, there is a prompt for the type of virtual environment and another code block showing the command '-m pip install py4lexis'. The browser's interface, including the address bar and some UI elements, is visible.

```
Python  
library  
SDK  
  
minimal  
ed  
face  
  
preprocessing_start_date": "2022100300",  
preprocessing_dataset_path_geographical_data_path": "project/proj934a1763631a81e  
FlowExecutionName": "Run3",  
FlowExecutionConsent": True  
  
def prepare_data():  
    prepared = WRF_EXEC_PARAMS  
    prepared["inputParameters"]["preprocessing_start_date"] = datetime.datetime.now().str  
    prepared["inputParameters"]["preprocessing_start_date"] = '2022120400'  
    return prepared  
  
def submit_wf_execution(token):  
    res = requests.post(LEXIS_API + WORKFLOW_ID + '/execution',  
        json = prepare_data(),  
        headers = {'Authorization': 'Bearer ' + token})  
  
    if res.status_code != 201:  
        logger.error(res.content)  
        return  
  
    return res.json()
```

(venv) mgolas@Martins-MacBook-Pro ~/I/L/L/Py4Lexis (develop)> python examples/upload.py
Progress: | ██████████ | 30.0% Uploaded

- ✦ LEXIS Platform was chosen in BioDT because it enables following:
 - ✦ Keeping provenance of executed workflows
 - ✦ Execution of computational workflows on the HPC centres/Cloud through API
 - ✦ We leverage this in the GUI development
 - ✦ It allows others to execute pDTs by code and create their own tools atop of them without need for setup of the model execution.
 - ✦ Some use cases leverage LEXIS Distributed Data Infrastructure for sensitive data storage

README.md

py4lexis

Package py4lexis provides functions to manage Lexis workflows.
 Python package, i.e. by TUS Client.

```

    """
    Execute manually an existing workflow (DAG) which is selected by its workflow ID (dag_id).

    Parameters
    -----
    workflow_id: str
        Workflow ID (dag_id) of the existing workflow.
    workflow_parameters: dict
        Parameters of the existing workflow (DAG) as dictionary.
    workflow_run_id: str | None
        Workflow run id (dag_run_id). If None, will be set automatically.

    Returns
    -----
    dict[str, str | int]
        Response as dictionary.
    """

    if not self._suppress_print:
        print(f"Executing existing workflow (DAG) by its ID...")

    url: str = self.__session.api_air + "/dags/" + workflow_id + "/dagRuns"

    if workflow_run_id is None:
        workflow_run_id: str = "py4lexis_exec_" + datetime.now().isoformat() + "_" + str(round(random() * 100))

    workflow_parameters["access_token"] = self.__session.get_access_token()
    workflow_input: dict = {
        "conf": workflow_parameters,
        "dag_run_id": workflow_run_id
    }

    content: list[dict] = self.__session.make_request(method="post",
                                                    url=url,
                                                    headers=self.__session.get_air_headers(),
                                                    json_data=workflow_input,
                                                    verify=False,
                                                    to_json=True)

```

README.md

r4lexis

```

140
141 #' Execute LEXIS workflow
142 #'
143 #' @param lexis_session LexisSession R6Class
144 #' @param workflow_id workflow ID
145 #' @param workflow_parameters workflow parameters
146 #' @param workflow_run_id workflow run (execution) ID
147 #' @param verbosity verbosity
148 #'
149 #' @importFrom r4lexis lexis_request
150 #'
151 #' @return Modified http2 response containing the response content in resp$content
152 #' @export
153 execute_workflow <- function(lexis_session,
154                             workflow_id,
155                             workflow_parameters,
156                             workflow_run_id,
157                             verbosity = 0) {
158   is_lexis_session(lexis_session)
159   is_character(workflow_id, "workflow_id")
160   is_character(workflow_run_id, "workflow_run_id")
161   is_numeric(verbosity, "verbosity")
162
163   workflow_input <- list()
164   workflow_input$conf <- workflow_parameters
165   workflow_input$conf$access_token <- lexis_oauth_token$access_token
166
167   workflow_input$dag_run_id <- workflow_run_id
168
169   resp <- lexis_session$make_lexis_request(
170     request_api_path = paste0("dags/", workflow_id, "/dagRuns"),
171     req_method = "POST",
172     req_body_json = workflow_input,
173     resp_body_type = "json",
174     is_airflow = TRUE,
175     verbosity = verbosity
176   )
177
178   return(resp)
179 }
180

```



BIODT

biodiversitydigitaltwin

School & Hackaton



@BiodiversityDT



BioDT



Funded by
the European Union