D1.6

GUI deployment

Matthias Büschelberger, Kathrin Frei, Valerio Lunardelli, Claudio Rosati,
Francesco Benedetto, Enrico Piccinini, Andrea Padovani, and Arrigo Calzolari

Document information

Project acronym:	INTERSECT
Project full title:	Interoperable Material-to-Device simulation box for disruptive electronics
Research Action Project type:	Accelerating the uptake of materials modeling software (IA)
EC Grant agreement no.:	814487
Project starting / end date:	1st January 2019 (M1) / 30th April 2022 (M40)
Website:	www.intersect-project.eu
Final version:	31/01/2022
Deliverable No.:	D1.6
Responsible participant:	Fraunhofer (participant number 5)
Contributing Consortium members:	AMAT, CNR
Due date of deliverable:	31/01/2022
Actual submission date:	31/01/2022
Dissemination level:	PU - Public

Authors: Matthias Büschelberger, Kathrin Frei, Valerio Lunardelli, Claudio Rosati, Francesco Benedetto, Enrico Piccinini, Andrea Padovani, Arrigo Calzolari

To be cited as: M. Büschelberger, K. Frei, V. Lunardelli C. Rosati, F. Benedetto, E. Piccinini, A. Padovani, and A. Calzolari (2022): GUI deployment (final version as of 31/01/2022). EC grant agreement no: 814487, CNR, Modena, Italy

Disclaimer:

This document's contents are not intended to replace consultation of any applicable legal sources or the necessary advice of a legal expert, where appropriate. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user, therefore, uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the authors' view.

Versioning and Contribution History

Version	Date	Modified by	Modification reason
v.01	13/01/2022	Matthias Büschelberger	First version
v.02	14/01/2022	Valerio Lunardelli	Update “Deviation of Planned Work in DoA ” and “Conclusions and Outlook”
v.03	18/01/2022	Matthias Büschelberger & Kathrin Frei	Reorganize chapters
v.04	26/01/2022	Arrigo Calzolari	Final version

Contents

[1] Acronyms are marked in purple in the text and defined at the end of the document.

1. Executive Summary

The content of this deliverable is related to *Task 1.4 – Graphical User Interface and integration on MARKETPLACE* and is based on the work of D1.5 – *GUI design and setup* as well as *D2.4 Materials-to-device and device-to-materials syntactic interconnections*[2] and D2.5 - *Semantic interoperability of the automated workflows*[3]. In this document, the solutions for the deployment and installation of the developed app components on the MarketPlace [1] platform are reported, whereas the description of their implementation are the outcome of the previous deliverables mentioned above.

The graphical user interface (GUI) is hierarchically the final component of the IM2D application, through which the software end user interacts with the resources and workflows provided by the REST APIs of SimPhoNy and AiiDA [2]. Hence, *Task 1.4* foresees the initial design and requirements towards the GUI as well as the subsequent deployment of the front- and back-end on the MarketPlace.

In the following sections, we will point to our Description of the work done (Sec. 2) and Deviation from Planned work in the DoA (Sec. 3). After presenting the legal terms and IP right of the GUI (Sec. 4), we will explain our multi-staged solution for the MarketPlace integration (Sec. 5) in detail. Finally, the achieved outcomes in the scope of this report are summarized and an outlook is given (Sec. 6).

2. Description of the work done

In order to meet the architectural requirements from the MarketPlace hardware infrastructure, the components of the back-end and the workflow runners of SimPhoNy and AiiDA (see D2.4[4] and D2.5) needed to be virtualized in Docker. The REST APIs developed (see D2.4 and D2.5) for the IM2D web-app were incorporated into the MarketPlace own API via standardized OpenAPI 3.0.0 specifications.

Due to the IP rights of the IM2D front-end towards Applied Materials (AMAT, proprietary of the Ginestra® code), the developed prototype GUI is presently available upon request by contacting the INTERSECT project coordinator or the AMAT customer service. Once purchased and installed, the IM2D GUI can easily be connected to the public IM2D API on the MarketPlace and, via Docker, on any local machine or optionally on any private cluster architecture.

For supporting the latter two configurations, an easily adaptable setup-file was implemented. It offers the opportunity to modify the network structure on, e.g., a cluster, any individual host/port mapping, as well as any compatible database and triplestore connection.

[2] D1.5: https://drive.google.com/file/d/1yDGZ68JW-a7w7iTYD4eWwoQeuZx9wcZX/view
[3] D2.5: https://drive.google.com/file/d/1X1ukclCS0YNQqOWTfrxrhg1maOSLvBPv/view
[4] D2.4: https://drive.google.com/file/d/1DN0pATpFloQ3Br0rWugdNn3fDBILBDri/view

3. Deviation from Planned work in the DoA

Originally, Ginestra® was a proprietary code of the MDLab company (initially part of the consortium) and we wanted to re-use and upscale the existing Ginestra® GUI as the IM2D GUI. Shortly after the beginning of the project, MDLab was acquired by the Applied Materials company (AMAT), which is now a partner of the project in place of MDLab (same team, same budget, etc). The new company decided, as an internal commercial strategy and independently from the INTERSECT project, to invest and improve the existing Ginestra® GUI, which is - as the rest of the Ginestra® code - their own proprietary software. This implied a few changes in the initial plan: as described in D1.5, an integration of the Java Ginestra® GUI with the MarketPlace via SimPhoNy-Remote [3] was explored and technically feasible. Nonetheless, the AMAT legal team did not want to upscale the new Ginestra® GUI as the IM2D platform GUI, since the Ginestra® GUI is a commercial product owned by AMAT, is covered by copyright and developed outside the project field. Therefore, the use of the Ginestra® GUI as a common part of the entire IM2D infrastructure would cause delicate legal issues about IP and license management. This is particularly critical for the proposed integration of IM2D into the MarketPlace public hub by using the SimPhoNy-remote framework, since MarketPlace is not conceived to host copyrighted software.

For these reasons, we defined an alternative strategy and we implemented all the technical steps necessary for the deployment of IM2D on MarketPlace (see Sec. 4), while -for the time being- the access is not public, but available on demand (see. Sec. 5.2).

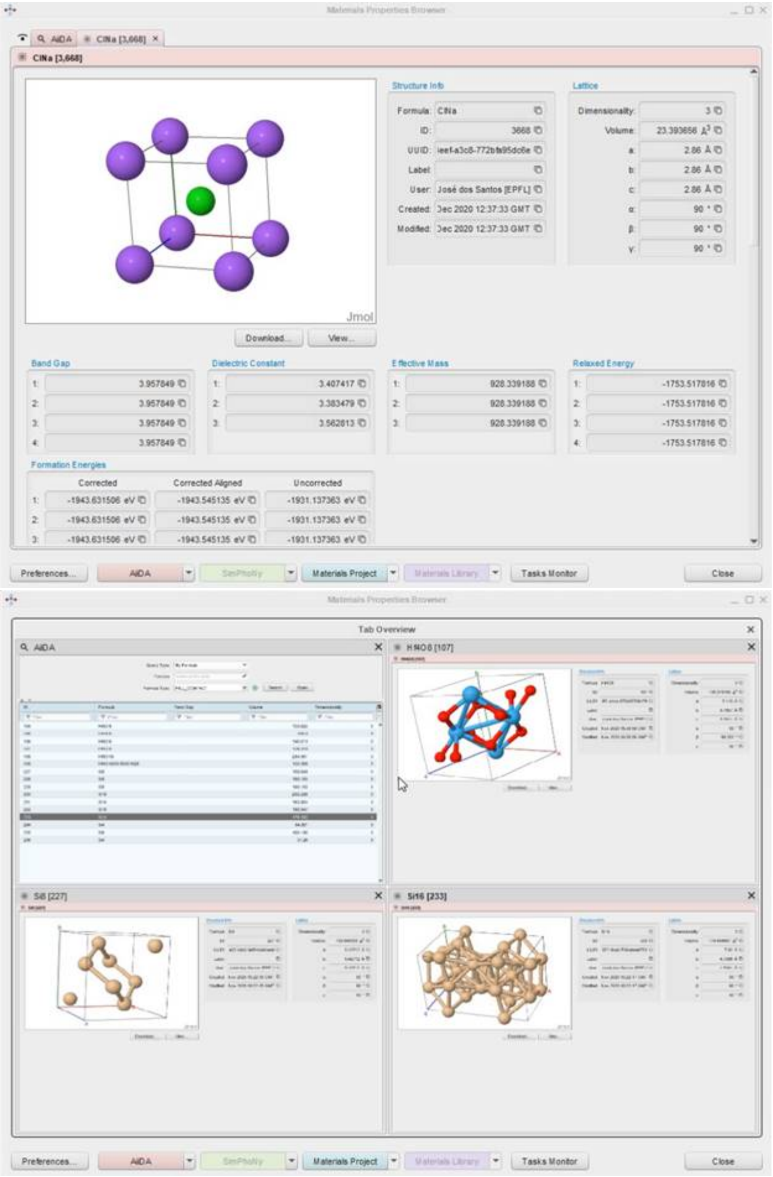
4. IM2D Graphical User Interface

In order to overcome the legal challenges addressed in the previous chapter, we adopted an alternative solution for the design and the implementation of the IM2D GUI. Instead of upscaling the native Ginestra® GUI, AMAT created a dedicated user-friendly front-end prototype for the IM2D platform, by developing and empowering the Ginestra®-AiiDA plugin. This is a flexible and robust Java app that can be easily integrated in MarketPlace exploiting the SimPhoNy-Remote framework (as originally planned in DoA). Indeed, being the plugin a genuine outcome of the INTERSECT project, it is redeemed by commercial licensing restrictions and can be distributed with the IM2D framework.

The front end of the Ginestra®-AiiDA plugin allows the semantic definition of different *persona* profiles (D2.5); it provides a seamless integration of AiiDA, Ginestra®, and SimPhoNy, as well as the access to the materials database through the OPTIMADE API (D2.6[5]). Even though its realization required to redesign the hierarchy of a few interoperability rules, i.e., some extra implementation work (reported in D2.5, D2.6), this change allowed us to overcome the legal/commercial issues and to complete the implementation of the overall IM2D platform. The legal limitations concerning the IP protection from the side of the MarketPlace will be discussed further in section 5.2.

[5] D2.6: https://drive.google.com/file/d/1EoKa1JxkqfCyM0jsIL1cq_6YHOqOqKEn/view

The graphical front end has a simple, intuitive, and user-friendly structure, based on an operation menu, buttons for the connection with the other part of the infrastructure (e.g., AiiDA), and windows for the visualization of 3D structures and material properties/parameters. Examples of the present status of this GUI are given in Figure 1. For further information about the features and functionalities of the IM2D front-end please see D2.4 - *Materials-to-device and device-to-materials syntactic interconnections*.



ClNa [3,668]

Structure Info

Formula:	ClNa
ID:	3668
UUID:	ieef-a3c8-772bfa95dc6e
Label:	
User:	José dos Santos [EPFL]
Created:	Dec 2020 12:37:33 GMT
Modified:	Dec 2020 12:37:33 GMT

Lattice

Dimensionality:	3
Volume:	23.393656 Å^3
a:	2.86 Å
b:	2.86 Å
c:	2.86 Å
α:	90 °
β:	90 °
γ:	90 °

Band Gap

1:	3.957849
2:	3.957849
3:	3.957849
4:	3.957849

Dielectric Constant

1:	3.407417
2:	3.383479
3:	3.562613

Effective Mass

1:	928.339188
2:	928.339188
3:	928.339188

Relaxed Energy

1:	-1753.517816
2:	-1753.517816
3:	-1753.517816
4:	-1753.517816

Formation Energies

	Corrected	Corrected Aligned	Uncorrected
1:	-1943.631506 eV	-1943.545135 eV	-1931.137363 eV
2:	-1943.631506 eV	-1943.545135 eV	-1931.137363 eV
3:	-1943.631506 eV	-1943.545135 eV	-1931.137363 eV

Figure 1. Top: Screenshot of the new IM2D GUI. This example shows the graphical visualization of the atomic structure and the corresponding materials properties of a crystalline compound (NaCl). Bottom: Screenshot of the visualization page of the material properties browser for the query of accessible material databases.

5. Back-end Services and their integration into the MarketPlace

5.1 Introduction to the MarketPlace

The MarketPlace project is an EU project (independent from INTERSECT), whose consortium consists of prominent EMMC [4] partners like SINTEF, Bosch, EPFL, ANSYS, Fraunhofer, Goldbeck Consulting Ltd., etc. The project's goals are the installation and organization of a decentralized and ontology-based digital marketplace for the materials modelling community. Applications can be provided by external REST-services and easily registered by any authenticated user in the MarketPlace app-store by evaluating the application's capabilities through the OpenAPI specification. The APIs of the external services are queryable through the MarketPlace app-store, and thus interoperable with other ontology-based applications available on the platform. Since the MarketPlace follows OpenAPI-standardizations, the IM2D-web-app can be incorporated into any related semantic software services, which are currently under development within common projects of the EMMC umbrella (e.g., OntoTrans [5], DOME 4.0 [6], SimDOME [7], OpenModel [8], etc.). This is boosted significantly by the semantic interoperability worked out in D2.5.

5.2 MarketPlace Integration Strategy

As stated previously, the D1.5 original strategy (to directly deploy the Ginestra® GUI together with AiiDA and SimPhoNy on the MarketPlace platform through SimPhoNy-Remote) had to be changed due to legal/licensing constraints. An integration into the MarketPlace infrastructure would require the precondition that the deposited and cloud-registered product could only be only available for a subset of users, i.e., the official holders of a Ginestra® license or the INTERSECT consortium members. Currently, the MarketPlace infrastructure does not allow this (status January 2022). As a result, the straightforward integration of IM2D including the front-end into the app store of MarketPlace is not possible by the state of the art. It has to be considered that the MarketPlace project was initially intended to host and distribute only *open source* software and did not focus on commercially-licensed frameworks. For this, there are no currently concrete business models elaborated for the integration of proprietary apps and their components in MarketPlace. However, since the EMMC community supports the inclusion of copyrighted software on the related digital marketplaces on materials modeling, the copyright management of the IM2D GUI could represent an ideal testbed to elaborate the terms of use regulation for the MarketPlace project. This would satisfy the potential interest to supply commercial purchase and distribution of proprietary 3rd-party software in favor of the code originators.

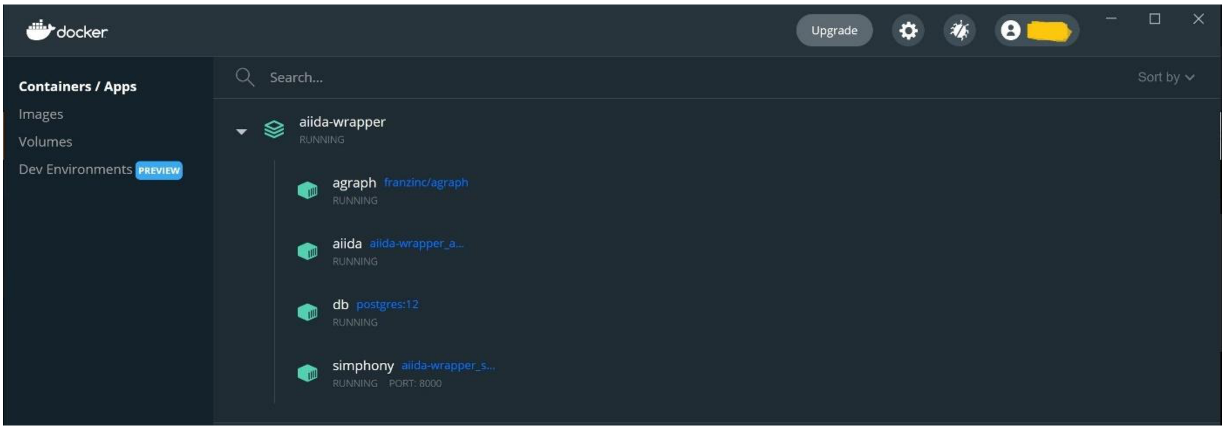
5.3 Dockerization of the app and deployment of the back-end services

As a consequence of the multiple scenarios described above, the MarketPlace integration action has been focused on the deployment of the back-end services of the IM2D app (AiiDA and SimPhoNy) into the Fraunhofer IWM-server infrastructure, the resident core of the MarketPlace project. The back-end services can be launched without authentication under https://app3.materials-data.space. Accordingly, the whole generated data produced by any user on this website is fully public and accessible without login. This data includes the full data provenance of the workflows which are executed and can be run, as well as the full semantic interpretation through SimPhoNy-CUDS based on EMMO.

Furthermore, the MarketPlace app store shares the information that a compatible GUI for the IM2D web-app can be requested by contacting the INTERSECT coordinator and the AMAT customer service. Since IM2D GUI is interoperable with Ginestra® and offers the opportunity to connect to any instance of the IM2D engine (no matter if local or remote), any IM2D GUI user can connect to SimPhoNy and AiiDA either through the MarketPlace, through any company-internal infrastructure, or even through localhost on any individual laptop (see Figures 2 and 5). This is majorly achieved through the containerization of the IM2D back-ends and engines in a Docker-compose network for the scope of this deliverable. The files and the manual for setting up this Docker-compose network are available on the Fraunhofer GitLab of the semantic interface of SimPhoNy [9]. This repository will be released on public GitHub and described in *D1.7 - Internal check tests and IM2D documentation* (M40, April 2022). The Docker-images can be easily built by command line. Additionally, these images will be pushed and hosted on the official Dockerhub [10], so that even IM2D users without programming experience can easily pull it to any local machine without the need of technical knowledge about Docker. The dockerized version of the whole IM2D framework is easily exchangeable, maintainable, and deployable on any host machine. This containerization was particularly needed for the integration on the MarketPlace-production servers under https://app3.materials-data.space.

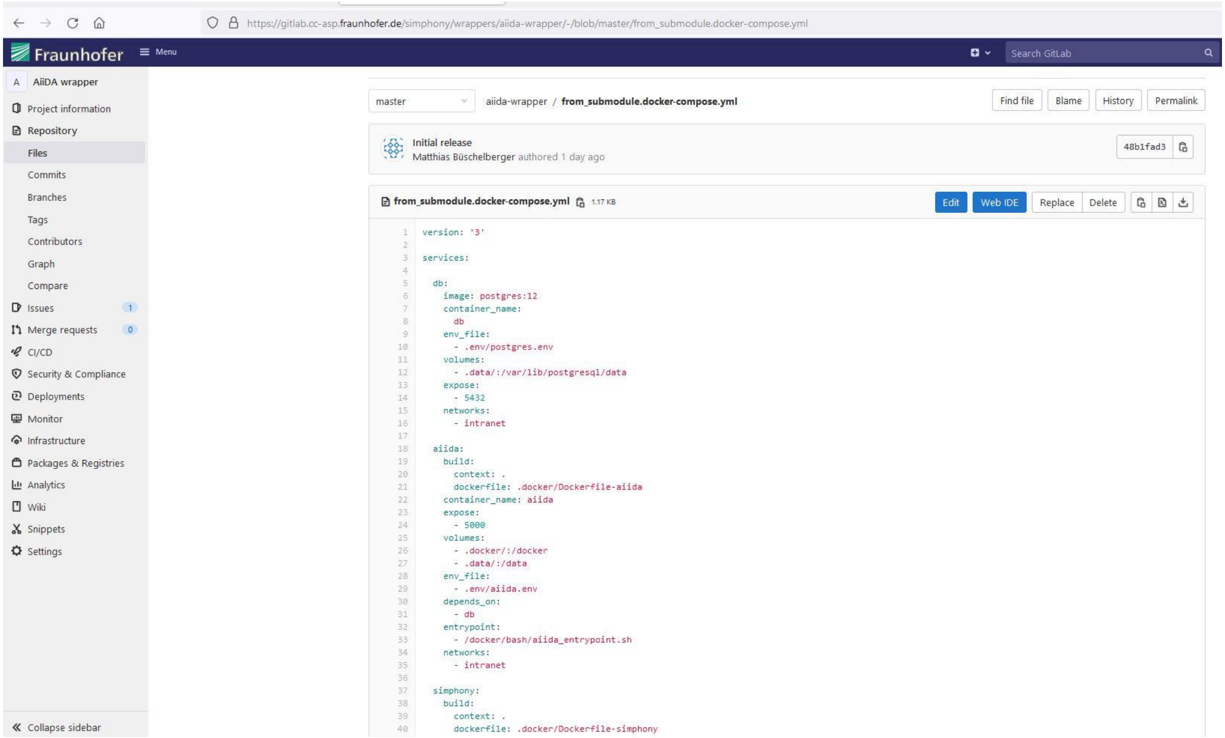
5.4 Benefits of the Dockerization

The containerization has the big advantage that the services of the AllegroGraph-triplestore for SimPhoNy, the Postgres-database (DB) for AiiDA, the AiiDA-engine, and the SimPhoNy-engine can run separately and communicate with each other through http-protocols (see Figures 2 and 3). This is particularly beneficial when the app is to be installed on a cluster and the individual services need to be distributed over various cluster-nodes.



aiida-wrapper RUNNING
agraph franzinc/agraph RUNNING
aiida aiida-wrapper_a... RUNNING
db postgres:12 RUNNING
simphony aiida-wrapper_s... RUNNING PORT: 8000

Figure 2: Screenshot of the IM2D-containers in a Docker compose running locally on windows. The IM2D REST API is exposed through the SimPhoNy-container on port 8000 and can be queried under the localhost.



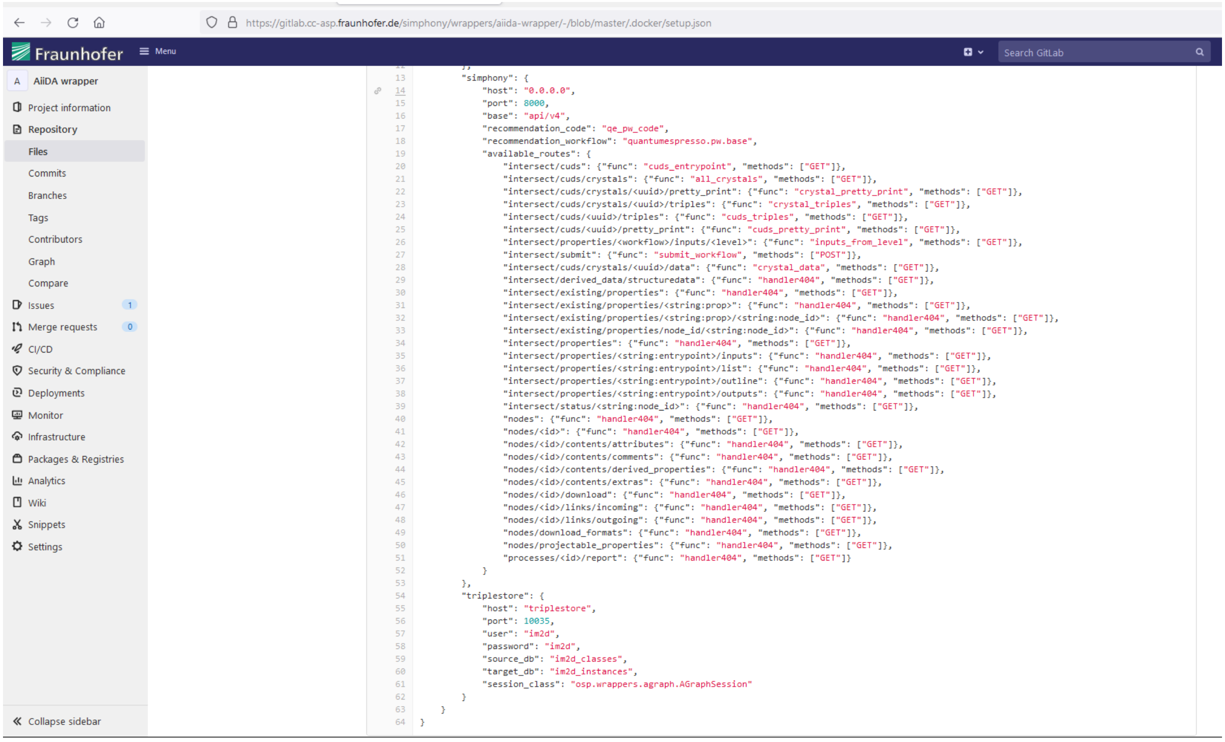
Initial release
Matthias Büschelberger authored 1 day ago

```yaml
version: '3'

services:

  db:
    image: postgres:12
    container_name:
      db
    env_file:
      - .env/postgres.env
    volumes:
      - .data/:/var/lib/postgresql/data
    expose:
      - 5432
    networks:
      - intranet

  aiida:
    build:
      context: .
      dockerfile: .docker/Dockerfile-aiida
    container_name: aiida
    expose:
      - 5000
    volumes:
      - .docker/:/docker
      - .data/:/data
    env_file:
      - .env/aiida.env
    depends_on:
      - db
    entrypoint:
      - /docker/bash/aiida_entrypoint.sh
    networks:
      - intranet

  simphony:
    build:
      context: .
      dockerfile: .docker/Dockerfile-simphony
```

Figure 3: Screenshot of the Docker-compose file on the Fraunhofer GitLab for building the images and containers for the IM2D-app from scratch.

Additionally, the semantic and syntactic data of AiiDA and SimPhoNy in the Postgres-DB and the AllegroGraph-triplestore can be preserved, while both engines in their own containers can be maintained and updated without any loss of data. The entry points of the containers are designed to automatically reconnect to the configured existing databases when the containers are restarted, rebuilt, or exchanged.

Furthermore, due to the modular structure of SimPhoNy, IM2D can connect to other triplestores (e.g., OntoKB from OntoTrans, StarDog, etc.) and AiiDA-instances (e.g., Quantum Mobile). This setup is easily configured through a JSON-file that is parsed when the containers are started or restarted (see Figure 4). At the same time, SimPhoNy updates the t-box of the ontology within the database of the triplestore, in the case a new ontology-version has been assigned in the setup-file. The individuals of the CUDS-objects are stored separately from the ontology classes, so that the existing data structures of individual computations are kept even though the class-hierarchy is refreshed.



```
    },
    "simphony": {
        "host": "0.0.0.0",
        "port": 8000,
        "base": "api/v4",
        "recommendation_code": "qe_pw_code",
        "recommendation_workflow": "quantumespresso.pw.base",
        "available_routes": {
            "intersect/cuds": {"func": "cuds_entrypoint", "methods": ["GET"]},
            "intersect/cuds/crystals": {"func": "all_crystals", "methods": ["GET"]},
            "intersect/cuds/crystals/<uuid>/pretty_print": {"func": "crystal_pretty_print", "methods": ["GET"]},
            "intersect/cuds/crystals/<uuid>/triples": {"func": "crystal_triples", "methods": ["GET"]},
            "intersect/cuds/<uuid>/triples": {"func": "cuds_triples", "methods": ["GET"]},
            "intersect/cuds/<uuid>/pretty_print": {"func": "cuds_pretty_print", "methods": ["GET"]},
            "intersect/properties/<workflow>/inputs/<level>": {"func": "inputs_from_level", "methods": ["GET"]},
            "intersect/submit": {"func": "submit_workflow", "methods": ["POST"]},
            "intersect/cuds/crystals/<uuid>/data": {"func": "crystal_data", "methods": ["GET"]},
            "intersect/derived_data/structuredata": {"func": "handler404", "methods": ["GET"]},
            "intersect/existing/properties": {"func": "handler404", "methods": ["GET"]},
            "intersect/existing/properties/<string:prop>": {"func": "handler404", "methods": ["GET"]},
            "intersect/existing/properties/<string:prop>/<string:node_id>": {"func": "handler404", "methods": ["GET"]},
            "intersect/existing/properties/node_id/<string:node_id>": {"func": "handler404", "methods": ["GET"]},
            "intersect/properties": {"func": "handler404", "methods": ["GET"]},
            "intersect/properties/<string:entrypoint>/inputs": {"func": "handler404", "methods": ["GET"]},
            "intersect/properties/<string:entrypoint>/list": {"func": "handler404", "methods": ["GET"]},
            "intersect/properties/<string:entrypoint>/outline": {"func": "handler404", "methods": ["GET"]},
            "intersect/properties/<string:entrypoint>/outputs": {"func": "handler404", "methods": ["GET"]},
            "intersect/status/<string:node_id>": {"func": "handler404", "methods": ["GET"]},
            "nodes": {"func": "handler404", "methods": ["GET"]},
            "nodes/<id>": {"func": "handler404", "methods": ["GET"]},
            "nodes/<id>/contents/attributes": {"func": "handler404", "methods": ["GET"]},
            "nodes/<id>/contents/comments": {"func": "handler404", "methods": ["GET"]},
            "nodes/<id>/contents/derived_properties": {"func": "handler404", "methods": ["GET"]},
            "nodes/<id>/contents/extras": {"func": "handler404", "methods": ["GET"]},
            "nodes/<id>/download": {"func": "handler404", "methods": ["GET"]},
            "nodes/<id>/links/incoming": {"func": "handler404", "methods": ["GET"]},
            "nodes/<id>/links/outgoing": {"func": "handler404", "methods": ["GET"]},
            "nodes/download_formats": {"func": "handler404", "methods": ["GET"]},
            "nodes/projectable_properties": {"func": "handler404", "methods": ["GET"]},
            "processes/<id>/report": {"func": "handler404", "methods": ["GET"]}
        }
    },
    "triplestore": {
        "host": "triplestore",
        "port": 10035,
        "user": "im2d",
        "password": "im2d",
        "source_db": "im2d_classes",
        "target_db": "im2d_instances",
        "session_class": "osp.wrappers.agraph.AGraphSession"
    }
  }
}
```

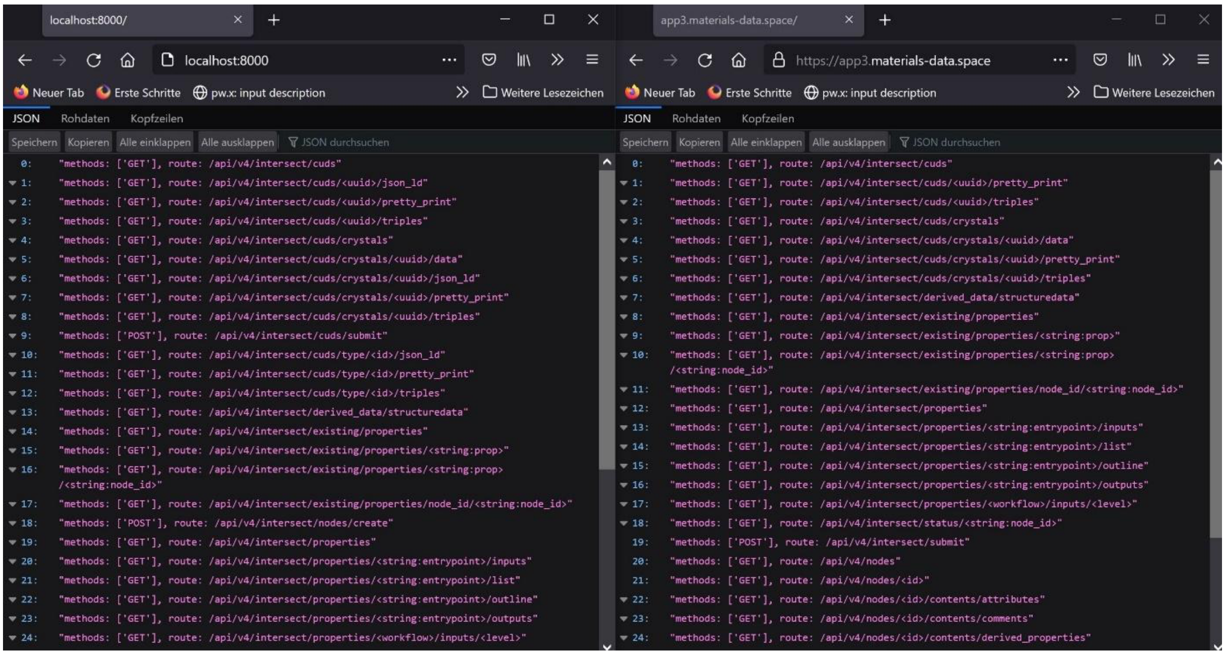
Figure 4: JSON-file for the configuration of the connection details for the Docker-compose network. While each service can run under different ports, SimPhoNy can expose different http-entry points of the AiiDA-REST API, extend it by other http-routes, and connect to other triplestores by defining other OSP-wrappers as data-source and data-sink.

5.5 Registration on the MarketPlace web-platform

Since the REST API on the MarketPlace is publicly accessible, any other GUI - apart from the one developed in INTERSECT - can connect and interact with the back-end service in order to push and pull data for and from the quantum mechanical simulations. Among other EU projects involved in the development of graphical front-ends for OSPs, there are SimDOME and OntoTrans with which we collaborate. Hence, this deployment also provides a running solution for the interaction of linked open data, semantic web application, and digital marketplaces for materials modeling.

The production deployment of the IM2D app will be continuously updated through the server administrators of the Fraunhofer IWM. This will especially occur more frequently until the due date of *D1.7 - Internal check tests and IM2D documentation* at M40.

Apart from the deployment of the REST API, the IM2D-back-end was registered under the web platform of the MarketPlace [11], which is the main app store of the decentralized infrastructure.



```
0:    "methods: ['GET'], route: /api/v4/intersect/cuds"
1:    "methods: ['GET'], route: /api/v4/intersect/cuds/<uuid>/json_ld"
2:    "methods: ['GET'], route: /api/v4/intersect/cuds/<uuid>/pretty_print"
3:    "methods: ['GET'], route: /api/v4/intersect/cuds/<uuid>/triples"
4:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals"
5:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals/<uuid>/data"
6:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals/<uuid>/json_ld"
7:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals/<uuid>/pretty_print"
8:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals/<uuid>/triples"
9:    "methods: ['POST'], route: /api/v4/intersect/cuds/submit"
10:   "methods: ['GET'], route: /api/v4/intersect/cuds/type/<id>/json_ld"
11:   "methods: ['GET'], route: /api/v4/intersect/cuds/type/<id>/pretty_print"
12:   "methods: ['GET'], route: /api/v4/intersect/cuds/type/<id>/triples"
13:   "methods: ['GET'], route: /api/v4/intersect/derived_data/structuredata"
14:   "methods: ['GET'], route: /api/v4/intersect/existing/properties"
15:   "methods: ['GET'], route: /api/v4/intersect/existing/properties/<string:prop>"
16:   "methods: ['GET'], route: /api/v4/intersect/existing/properties/<string:prop>
      /<string:node_id>"
17:   "methods: ['GET'], route: /api/v4/intersect/existing/properties/node_id/<string:node_id>"
18:   "methods: ['POST'], route: /api/v4/intersect/nodes/create"
19:   "methods: ['GET'], route: /api/v4/intersect/properties"
20:   "methods: ['GET'], route: /api/v4/intersect/properties/<string:entrypoint>/inputs"
21:   "methods: ['GET'], route: /api/v4/intersect/properties/<string:entrypoint>/list"
22:   "methods: ['GET'], route: /api/v4/intersect/properties/<string:entrypoint>/outline"
23:   "methods: ['GET'], route: /api/v4/intersect/properties/<string:entrypoint>/outputs"
24:   "methods: ['GET'], route: /api/v4/intersect/properties/<workflow>/inputs/<level>"
```

```
0:    "methods: ['GET'], route: /api/v4/intersect/cuds"
1:    "methods: ['GET'], route: /api/v4/intersect/cuds/<uuid>/pretty_print"
2:    "methods: ['GET'], route: /api/v4/intersect/cuds/<uuid>/triples"
3:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals"
4:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals/<uuid>/data"
5:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals/<uuid>/pretty_print"
6:    "methods: ['GET'], route: /api/v4/intersect/cuds/crystals/<uuid>/triples"
7:    "methods: ['GET'], route: /api/v4/intersect/derived_data/structuredata"
8:    "methods: ['GET'], route: /api/v4/intersect/existing/properties"
9:    "methods: ['GET'], route: /api/v4/intersect/existing/properties/<string:prop>"
10:   "methods: ['GET'], route: /api/v4/intersect/existing/properties/<string:prop>
      /<string:node_id>"
11:   "methods: ['GET'], route: /api/v4/intersect/existing/properties/node_id/<string:node_id>"
12:   "methods: ['GET'], route: /api/v4/intersect/properties"
13:   "methods: ['GET'], route: /api/v4/intersect/properties/<string:entrypoint>/inputs"
14:   "methods: ['GET'], route: /api/v4/intersect/properties/<string:entrypoint>/list"
15:   "methods: ['GET'], route: /api/v4/intersect/properties/<string:entrypoint>/outline"
16:   "methods: ['GET'], route: /api/v4/intersect/properties/<string:entrypoint>/outputs"
17:   "methods: ['GET'], route: /api/v4/intersect/properties/<workflow>/inputs/<level>"
18:   "methods: ['GET'], route: /api/v4/intersect/status/<string:node_id>"
19:   "methods: ['POST'], route: /api/v4/intersect/submit"
20:   "methods: ['GET'], route: /api/v4/nodes"
21:   "methods: ['GET'], route: /api/v4/nodes/<id>"
22:   "methods: ['GET'], route: /api/v4/nodes/<id>/contents/attributes"
23:   "methods: ['GET'], route: /api/v4/nodes/<id>/contents/comments"
24:   "methods: ['GET'], route: /api/v4/nodes/<id>/contents/derived_properties"
```

Figure 5: Screenshot of the landing page of the IM2D-REST API through the web browser, left: on localhost through Docker; right: through the Docker-deployment on the MarketPlace production server. The IM2D front-end is able to connect to both instances.

The registration of the IM2D back-end on the MarketPlace increases the findability of the app and the generated data within the scope of related digital marketplace EU projects such as DOME 4.0, OpenModel, VIMMP, OntoTrans, MarketPlace, etc. This is additionally boosted by the standardization of the OpenAPI-3.0.0-specification, which leads to the accessibility of the IM2D REST API through the MarketPlace and hence the enrichment with other semantic materials data structures from available applications at the MarketPlace-store.

6. Conclusions and outlook

As described above and in *D2.4*, we realize an IM2D user-friendly graphical front-end for textual input, generation of workflow and dependency paths, able to access a material explorer and a device modeler, which manages the generation of input and output, and the execution of the linked model and access to material database, on the base of the *persona* profiles. The engines and data sources of SimPhoNy and AiiDA were successfully deployed on the MarketPlace production servers, so that the generated data can be queried and the available workflows can be run through the developed REST APIs without further user authentication.

This API is registered through the openAPI-3.0.0 specification of the MarketPlace web services and is connectable to any other OSP-front-end, which values the app and its public data as a substantial contribution to publicly available simulation toolboxes backboned by semantic interoperability.

At present, the legal constraints due to the copyrighted source code by AMAT and lack of regulation on hosting proprietary software on Marketplace hindered the public accessibility to the IM2D infrastructure. In view of the positive intentions of all actors to distribute IM2D on public hubs, these legal issues are expected to be solved in the next months. In the meantime, full access to IM2D from Marketplace is available on demand, simply contacting the INTERSECT project coordinator or the AMAT customer service. We finally remark that the changes to the initial plan (see above) only affected the technical implementation of IM2D deployment, but not the user side.

In the near future, the standalone SimPhoNy REST API based on the Python Flask-package [12] will be upscaled to a Swagger UI [13] through available Flask-extensions [14] and will provide a more user-friendly interaction with the IM2D engines and back-end through any common web browser. Since the REST API was already specified in the YAML-format during the MarketPlace registration, this change is going to be a minor one.

Furthermore, the IM2D web-app will change its domain url from https://app3.materials-data.space to https://im2d.materials-data.space in the near future. The according changes will not affect the hosted data and the MarketPlace registration will be updated accordingly.

References

[1] MarketPlace, Materials Modelling Marketplace for Increased Industrial Innovation. www.the-marketplace-project.eu

[2] AiiDA, Automated Interactive Infrastructure and Database for Computational Science. https://www.aiida.net

[3] SimPhoNy-Remote, https://github.com/simphony/simphony-remote

[4] European Materials Modelling Council, https://emmc.eu

[5] Ontology driven Open Translation Environment, https://ontotrans.eu

[6] Digital Open Marketplace Ecosystem 4.0, https://dome40.eu

[7] Simdome: https://simdome.eu

[8] OpenModel: Integrated Open Access Materials Modelling Innovation Platform for Europe, https://open-model.eu

[9] https://gitlab.cc-asp.fraunhofer.de/simphony/wrappers/aiida-wrapper

[10] Docker hub: https://hub.docker.com

[11] See product list under https://the-marketplace.eu/create/products-list

[12] https://flask.palletsprojects.com/en/2.0.x

[13] https://swagger.io/tools/swagger-ui

[14] https://flask-restplus.readthedocs.io/en/stable/swagger.html

Acronyms

API - Application Programming Interface

CUDS - Common Universal Data Structure

DB - Database

DoA - Description of Action

EMMC - European Materials Modelling Council

EMMO - Elementary Multiperspective Material Ontology

GUI - Graphical User Interface

IM2D - Interoperable Materials-to-Device

IP - Intellectual property

OSP - Open Simulation Platform

REST - Representational State Transfer