PIACERE

Deliverable D5.9

IOP prototype – v3

Editor(s):	Eneko Osaba, Iñaki Etxaniz, Gorka Benguria
Responsible Partner:	TECNALIA
Status-Version:	Final-v1.0
Date:	31.05.2023
Distribution level (CO, PU):	PU

Project Number:	101000162
Project Title:	PIACERE

Title of Deliverable:	IOP prototype – v3
Due Date of Delivery to the EC	31.05.2023

Workpackage responsible for the Deliverable:	WP5 - Package, release and configure Infrastructure as Code
Editor(s):	Eneko Osaba (TECNALIA)
Contributor(s):	Gorka Benguria (TECNALIA), Iñaki Etxaniz (TECNALIA)
Reviewer(s):	Elisabetta Di Nitto
Approved by:	All Partners
Recommended/mandatory readers:	WP3, WP5, WP6

Abstract:	This deliverable describes Key Result 9 – IaC Optimization Platform, which is the main outcome of T5.3 from M1 to M30. The deliverable includes a Technical Specification Report and a software prototype of the tool called Infrastructure as Code Optimization Platform (IOP) [KR9]. It includes also an explanation of the optimization algorithms implemented in the tool. This is the third (and final) version after D5.6 and D5.7 and reports the final innovations, lessons learnt and outlook to the future.
Keyword List:	IOP, Catalogue of Infrastructural Elements, Optimization, Multi-Objective Algorithm.
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	20.04.2023	First TOC and sections assignment	TECNALIA
v0.2	05.05.2023	IOP new version included	TECNALIA
v0.3	15.05.2023	IEC new version. General edition	TECNALIA
v0.4	18.05.2023	Internally reviewed version	POLIMI
v0.5	25.05.2023	Reviewer comments addressed	TECNALIA
v1.0	29.05.2023	Final quality check. Ready for submission	TECNALIA

Table of contents

List of tables

List of figures

List of Lists

Terms and abbreviations

CSP	Cloud Service Provider
DevOps	Development and Operation
DoA	Description of Action
EC	European Commission
GA	Grant Agreement to the project
IaC	Infrastructure as Code
IEP	IaC execution platform
IOP	IaC Optimization Platform
KPI	Key Performance Indicator
SW	Software
IEC	Infrastructural Elements Catalogue
KR	Key Result
SOA	Service Oriented Architecture
REST	Representational State Transfer
OAS	Open API Specification
API	Application Programming Interface
JWT	Java Web Token
SD	Self Description

Executive Summary

This manuscript includes the technical description of the third version of the *Infrastructure as Code* (IaC) *Optimization Platform* (IOP) [KR9][1] implemented in PIACERE project. The main role of the IOP in the whole project is to find the most optimized combination of infrastructural and resource configuration for properly deploying a service. In order to do that, the IOP counts on an Infrastructural Elements Catalogue (IEC), which contains a set of resource descriptions, each one characterized by its own features, such as the cost, performance, availability… The results provided by the IOP tools depend on the elements included in the IEC, and also on the objectives to optimize and non-functional requirement introduced by the user, which are introduced in the IOP through a DOML file.

The IOP architecture and its design are described in this manuscript, allowing the comparison of the different tool variants and their evolution to the last version of the IOP. Finally, requirements and functionalities are described in this deliverable, as well as the coverage provided by this last version of the tool.

In addition to the technical and functional description of the IOP and IEC, this document also contains the installation and usage information of this last version of the IOP and the IEC. Furthermore, this deliverable describes what should be done for properly testing the software implemented.

This document, D5.9, is the last of a deliverable series on the IOP which has included also D5.7 [1] and D5.8 [2]. It is based on these previous versions to make the document self-contained, and hence a major part of the previous content has been preserved in the appendixes and updated to reflect the changes. The principal sections are devoted on the final results, including the latest work and new sections for experimentation, lessons learnt and outlook to the future.As a summary, the work developed between M24 and M30 has been focused on two main topics. The first one has been related with making the IOP more use-case-oriented, with the main intention of providing the best service to the PIACERE use cases and developing a tool that answers real-world situations. The second objective has been to conduct a scientific-oriented experimentation with the IOP, in order to clearly determine which are the main algorithms that compose the tool. Thanks to the work carried out in the last six months the new version of the IOP has been correctly integrated with the PIACERE IDE, it has been tested on the project use cases and we have also deepened on the correct choosing of the solving technique.

[1] The IOP was defined since the DoA as the Key Result 9 (KR9) component in PIACERE framework. A sub-component of the IOP, that is necessary for the optimization process, is the Infrastructural Elements Catalogue (IEC). Hence, we include also the IEC description in this document. However, due to the very different nature of the Catalogue part and the Optimization part, and to that we have not a proper name for the "optimization part of the IOP", in this document we will use "IOP" to refer only to the Optimization sub-module, in the same way as we use "IEC" to refer the Catalogue sub-module.

1 Introduction

1.1 About this deliverable

This manuscript represents the third and last release of the IOP Prototype deliverable, and it is based on the work carried out in WP5 - *Package, release and configure Infrastructure as code*, and more specifically, in the task *T5.3 - Best configuration deployments based on optimization algorithms*. For this reason, the objective is to outline the principal advancements and evolutions of IOP and IEC compared to the versions released at M24. These advancements regard the transformation of the IOP into a more use-case-oriented tool. Also, the activity carried out in this last period contemplates a scientific-oriented experimentation for properly deciding the algorithms that compose the IOP Optimizer. Thus, this document details different concepts around the IOP and IEC such as technical description, implementation, and installation.

1.2 Document structure

This manuscript is composed of 6 sections. Section 2 is devoted to providing a general description of KR9, which is the Key Result that regards the IOP and IEC. In this section, we also provide a clear explanation of the advances achieved between M24 and M30. Also, in this section we outline the requirements covered by the IOP and IEC and we highlight the main innovations of the tools. In Section 3 we delve into the experimentations conducted for testing the effectiveness of the tools. The following Section 4 is devoted to deepening on the lessons learnt, spotlighting also the future work planned for the tool. Finally, Section 5 presents the conclusions, while in Section 6 we summarize the references.

Also, it is necessary to point out that this deliverable is constituted by several appendixes, highlighting the implementation, delivery, and usage details, which in previous releases where part of the core document. Even if these aspects present little modifications - due to the short period of time that has elapsed between D5.8 and this D5.9, we have included them as appendixes for the sake of completeness.

2 KR9-IOP overview

KR9 – IaC Optimized Platform (IOP) is related with the work conducted in Task T5.3 - *Best configuration deployments based on optimization algorithm.* This task is devoted to the implementation of an intelligent optimization engine able to select and deploy the most optimized IaC configuration, subject to a set of objectives and requirements. To reach these goals, the IOP also contemplates an infrastructural elements catalogue (IEC) in which the features of all elements available are contained.

Firstly, the IEC is a persistence component whose principal function is to contain all the infrastructural elements data required by different PIACERE components. As a persistence tool, two crucial aspects are covered on the development of the IEC, how the information is added and how it is fetched. The IEC is pre-loaded with a list of cloud services from different providers. Regarding the information flow, three principal interactions are contemplated between IEC and other PIACERE tools: i) the Eclipse-based GUI/IDE, which is able to extract the data and present it to the user; ii) the Monitoring components and the Runtime Controller, which will feed the IEC with the information about deployed instances; and iii) the Optimizer, which consumes the list of services looking for the best alternatives.

Secondly, the optimization problem designed for giving an answer to the need fixed in the project involves a group of services to be deployed, and the objective of searching an appropriate IaC deployment configuration the best meets the preestablished constraints and restrictions. In this context, the IOP needs to find these optimized configurations after analysing the input data received. In this regard, this input information is offered in DOML format, and it encompass the objectives to optimize the non-functional requirements to meet. After retrieving all the input information, the optimizer executes a multi/many-objective optimization technique against the IEC content, aiming to find the most appropriate elements for building the deployment configuration.

Finally, two crucial aspects must be considered regarding the IOP Optimizer. On the one hand, the problem to be solved could be composed by several conflicting objectives, so that it requires the development of a multi- or even a many-objective approach. On the other hand, the IOP will be needed in two different phases of the whole PIACERE system: the initial deployment (in the design-time phase) and the redeployment of a service (in runtime phase, and after the *Self-Healing* component requires it).

As can be read in the DoA, the IOP is considered successful if it is able to propose the most optimized deployment configuration of the infrastructural code, taking into consideration the constraints predefined. To this end, several deployment configurations will be presented to the user and ranked.

2.1 Changes in v3

Main changes affect the technical advancement of the tools, covering a wider spectrum of needs and deeming additional functionalities. Thus, the sections less affected by changes are those related to the implementation, delivery, technical specification and user manual (sections that have been moved to Appendix A).

Regarding the implementation of the Catalogue, some changes have been introduced with respect to the previous version, D5.8. As the workflow of PIACERE framework involving the Catalogue was established quite a while ago and is stable, no interactions with new components are contemplated. The past version of the Catalogue was already integrated with the IOP, the IDE and the PRC. However, in light of the advances made in the knowledge of the

problem domain in this period, some adaptations have been made to the Catalogue, which are detailed in the following.

On the other hand, and related to the IOP Optimizer, in the first version on this deliverable (D5.7) the first laboratory prototype was introduced, which included some multi-objectives algorithms codified using two different well-known optimization frameworks: jMetal [1] and MOEA[2]. This first version was not DOML compliant, and it was not integrated with the IDE. Also, it was not tested by the use case owners.

Besides that, an advanced version of the IOP was presented in the second version of this deliverable (D5.8), which was more oriented to give an answer to the use case needs. For this reason, this version of the IOP was DOML compliant, it was embedded in the IDE and it was preliminary tested by use case owners. To develop this version of the tool, the jMetal framework has been chosen as the basis. Furthermore, the NSGA-II has been contemplated for single and two-objective multi-objective problems, and the NSGA-III for three-objective multi-objective problems. This first tests were crucial for the evolution conducted from M24 to M30.

With all this, the main innovation of the work proposed in this last version of this document gravitates on three different axes: *i)* advances made in the problem formulation, *ii)* works conducted regarding the integration with the IDE and *iii)* tests for determining the evolutionary multi-objectives algorithms included in the tool.

While the progresses performed regarding the third of these aspects are deepen in the upcoming Section *3 Overview of preliminary experiments*, we detail in this section the developments carried out on the first two pivotal points.

2.1.1 Changes regarding the IEC

2.1.1.1 New elements in the Catalogue

2.1.1.1.1 New providers

As one of the Use Cases expected to use the VMware[3] vSphere technology to set up its infrastructure, the content of the Catalogue has been enlarged with the inclusion of some services by the VMware provider. A set of Virtual Machines have been defined, with a granularity that offers to the users a basic range of options.

The list of virtual machines included in this version is reflected in the following table. The list can be expanded in the future if it is considered necessary for the Use Case.

Table 1: VMware virtual machines defined

Name	Provider	vCPUs	RAM	HD (GB)
Nano	VMware	1	0,5	10
Micro	VMware	1	1	20
Small	VMware	1	2	40
Medium	VMware	2	4	40
Large	VMware	2	8	80
2large	VMware	4	8	80
Xlarge	VMware	4	16	160

[2] http://moeaframework.org/

[3] https://www.vmware.com/

x2large	VMware	8	16	160
X3large	VMware	8	32	320
X4large	VMware	8	64	320

In the same line, a set of virtual machines of the IONOS[4] provider is foreseen to be included in the Catalogue too. At the time of writing, the details of the selected IaaS services are still not decided.

2.1.1.1.2 Images

As a result of the discussions in WP3, it was detected the convenience to store somewhere information related with images that are then used during the design workflow. The obvious place to store them was the Catalogue. The "image" word here can refer to two different types of images: an image of a virtual machine or an image of a docker container.

A simple data structure was defined to maintain these images, that are defined by the user early in the workflow.

- *Name (string)*
- *Type: (string) [Docker | VM]*
- *Description (string)*
- *Provider (string)*

A simple API, *image-resource*[5], has been provided to manage addition/deletion of the images in the Catalogue:

```
GET    /api/images          getAllImages
GET    /api/images/count    countImages
POST   /api/images          createImage
GET    /api/images/{id}     getImage
PUT    /api/images/{id}     updateImage
DELETE /api/images/{id}     deleteImage
```

Figure 1: API to manage images in the Catalogue

As an example, a call to the *getAllImages* end point will return a json with an array of the images:


```
[
  {
    "id": 1,
    "imageType": "Docker",
    "imageName": null,
    "imageUrl": "docker.hub.io/Ericsson/tia:1.0",
    "imageDescription": null,
    "imageProvider": "Provider A"
  },
  {
    "id": 2,
    "imageType": "VM",
```


[4] https://cloud.ionos.com/

[5] https://iec.ci.piacere.digital.tecnalia.dev/swagger-ui/index.html?urls.primaryName=iecbackend#/image-resource

```
    "imageName": "Ubuntu_6.4.2",
    "imageUrl": null,
    "imageDescription": "VM file for Ubuntu 6.4.2",
    "imageProvider": "Provider B"
  },
  {
    "id": 3,
    "imageType": "VM",
    "imageName": "Ubuntu_6.4.1",
    "imageUrl": "",
    "imageDescription": " VM file for Ubuntu 6.4.1",
    "imageProvider": "Ubuntu"
  }
]
```


Figure 2: Response to the getAllImages API call.

2.1.1.1.3 Pre-existing resources

Another data object that has been detected that is needed for some Use Cases is what we call the "pre-existing-resource". This refers to elements of the infrastructure that are defined outside the PIACERE workflow, but that are needed when the IaC is being defined for a specific environment. Existing network elements in the company, or templates of already used virtual machines are examples of these type of elements.

This kind of data is needed, for example, when the Infrastructural Code Generator (ICG) is constructing a Terraform script that defines the IaC deployment.

The data model defined to maintain these pre-existing-data is composed by:

- *Data Name (string)*
- *Data type (string) [dc | compute_cluster | datastore | pool | template | network]*
- *Name (string)*
- *Datacenter id (int)*
- *Provider (string)*
- *User (string)*

The *Provider* and the *User* fields are added in order to be able to filter the data and for authorization purposes. Six types of pre-existing-data have been defined, by the moment: dc, compute cluster, datastore, pool, template and network.

As in the case of the "images", a separate API[6] has been defined for the pre-existing resources, that is shown in the next figure:

[6] https://iec.ci.piacere.digital.tecnalia.dev/swagger-ui/index.html?urls.primaryName=iecbackend#/existing-resource-resource

```
GET    /api/existing-resources getAllExistingResources
POST   /api/existing-resources createExistingResource
GET    /api/existing-resources/count countExistingResources
GET    /api/existing-resources/{id} getExistingResource
PUT    /api/existing-resources/{id} updateExistingResource
DELETE /api/existing-resources/{id} deleteExistingResource
```

Figure 3: API to manage pre-existing resources in the Catalogue

As an example, a call to the *getAllExistingResources* end point will return a json with an array of the resources (in this moment there are six resources defined):


```
[
  {
    "id": 1,
    "dataName": "vsphere_datacenter",
    "dataType": "dc",
    "name": "MB",
    "datacenterId": "",
    "user": "User A",
    "provider": "Provider A"
  },
  {
    "id": 2,
    "dataName": "vsphere_compute_cluster",
    "dataType": "compute_cluster",
    "name": "MB-PIAC-NIC-1",
    "datacenterId": "${data.vsphere_datacenter.dc.id}",
    "user": "User X",
    "provider": "Provider X"
  },
  {
    "id": 3,
    "dataName": "vsphere_datastore",
    "dataType": "datastore",
    "name": "VNX01-0200-NIC-TA-PIAC-DRO-VMW-P",
    "datacenterId": "${data.vsphere_datacenter.dc.id}",
    "user": "User X",
    "provider": "Provider Y"
  },
  {
    "id": 4,
    "dataName": "vsphere_resource_pool",
    "dataType": "pool",
    "name": "PIAC",
    "datacenterId": "${data.vsphere_datacenter.dc.id}",
    "user": "User Y",
    "provider": "Provider X"
  },
  {
    "id": 5,
    "dataName": "vsphere_virtual_machine",
    "dataType": "template",
    "name": "Centos7_PIAC",
    "datacenterId": "${data.vsphere_datacenter.dc.id}",
    "user": "User Y",
    "provider": "Provider Y"
  },
  {
    "id": 6,
    "dataName": "vsphere_network",
```

```
    "dataType": "network",
    "name": "DRO-MB-P-BG001-2098",
    "datacenterId": "${data.vsphere_datacenter.dc.id}",
    "user": "User B",
    "provider": "Provider B"
  }
]
```


Figure 4: Response to the getAllExistingResources API call.

2.1.1.2 Gaia-X compatibility

In the previous version of the deliverable, an introduction to Gaia-X conceptual model was included, along with the intention to provide the Catalogue as a Gaia-X compliant service. Four steps were defined there.

The first two, related to the creation and validation of the Self-Description (SD) of Tecnalia as Participant, were achieved and presented. Since then, we have worked on the next two steps, related with the creation of the self-description for the PIACERE Catalogue and its validation, the results are presented in next paragraphs.

An aspect worth mentioning here is that the Gaia-X specification is an evolving one, that is not definitive, nor yet totally completed. For example, some of the important entities defined in the Gaia-X Trust Framework [2] are the Participant (although their roles — as *provider*, *consumer*, and *federator* — are to be defined in future releases of the Framework) and the ServiceOffering. A **Participant** (see Figure 5) is a Legal Person or Natural Person, which is identified, onboarded and has a Gaia-X Self-Description. Instances of a Participant neither being a legal nor a natural person are prohibited. The roles a Participant can have within the Gaia-X Ecosystem are Provider, Consumer, and Federator. These are not yet part of Trust Framework and are to be defined in future releases.

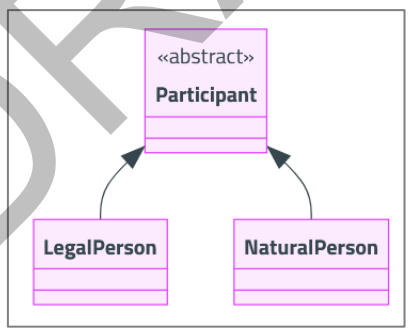

Figure 5: Participant in Gaia-X.

The **ServiceOffering**, in turn, is the aggregation of resources. These resources can be physical or virtual, being the latter a SoftwareResource or a DataResource, either one of them can be instantiated (see diagram in Figure 6).

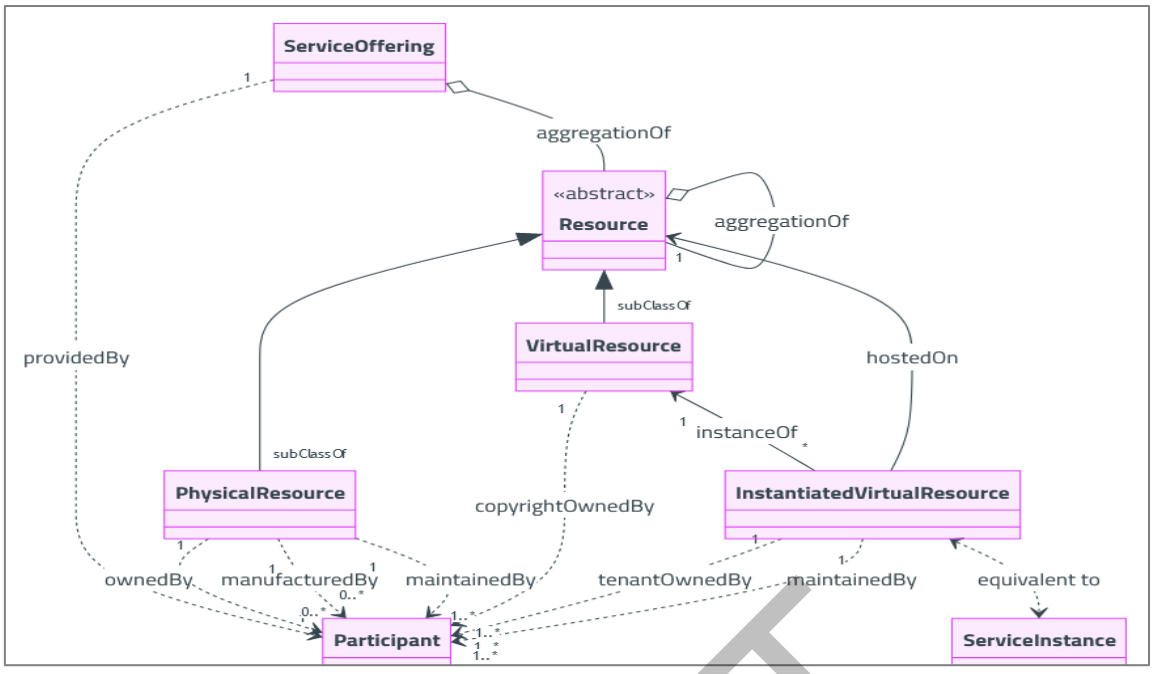

Figure 6: Service Offering in Gaia-X.

Gaia-X Self-Descriptions (SD) describe Entities from the Gaia-X Conceptual Model in a machine interpretable format. This includes Self-Descriptions for the Participants, as well as the Resources and Service Offerings from the Providers. The Participants are responsible for the creation of their Self-Descriptions. In addition to self-declared information by Participants about themselves or their offerings, a Self-Description may comprise statements issued and signed by trusted parties.

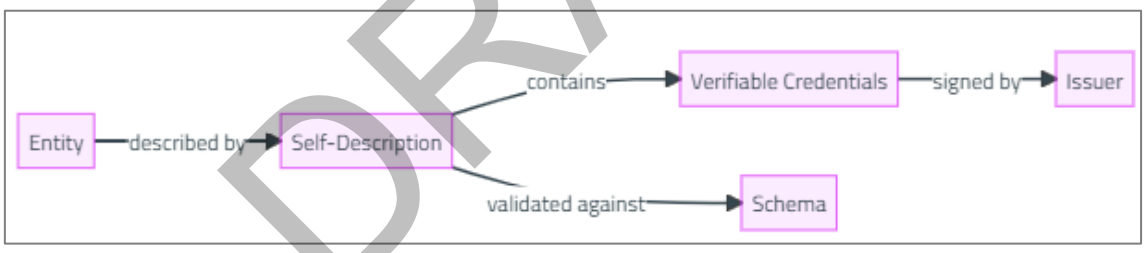

Figure 7: Service Offering in Gaia-X.

Self-Descriptions are W3C Verifiable Presentations[7] in the JSON-LD[8] format. Self-Descriptions comprise one or more Verifiable Credentials. Verifiable Credentials themselves contain a set of Claims: assertions about Entities expressed in the RDF data model. Both Verifiable Credentials and Verifiable Presentations come with cryptographic signatures to increase the level of trust.

Self-Descriptions contain verifiable credentials about the attributes of Entities and relations to other Entities based on subject-predicate-object triples (cf. the RDF data model). Cross-referencing between Self-Descriptions is enabled by unique Identifiers for the Entities. Identifiers in Gaia-X are URIs and follow the specification of RFC 3986

[7] https://www.w3.org/TR/vc-data-model/#presentations

[8] https://www.w3.org/TR/json-ld11/

With all this in mind, we detail in the following the self-description files prepared for the Catalogue. The documents have been signed using the Gaia-X wizard[9], and the result has been validated against the online Gaia-X compliance tool[10]. At the moment of writing, only the *LegalParticipant* and *ServiceOffering* categories can be validated in this tool –as can be checked in the shapes-implemented endpoint[11]– but we have also developed the files for the *SoftwareResoure*, *InstantiatedResource* and *DataCenter* categories.

2.1.1.2.1 Tecnalia Self-description

The definition of the Participant SD in the has suffered changes since the previous version of this deliverable, due to the evolution of the Gaia-X Architecture Document from V22.06 *[3]* to the actual version, v22.10. The former version is presented again here (see List 1), while the actual version of the participant Tecnalia, takes the form shown in List 2.


```
{
- selfDescriptionCredential: {
   – @context: [
        – "https://www.w3.org/2018/credentials/v1",
        – "https://registry.gaia-x.eu/v2206/api/shape"
                ],
   – type: [
        – "VerifiableCredential",
        – "LegalPerson"
                ],
   – id: "https://hodeix.digital.tecnalia.dev/.well-
     known/participantTecnalia.json",
   – issuer: "did:web:hodeix.digital.tecnalia.dev",
   – issuanceDate: "2022-09-27T09:36:23.235Z",
   – credentialSubject: {
        – id: "did:web:hodeix.digital.tecnalia.dev",
        – gx-participant:legalName: "Fundacion Tecnalia Research &
          Innovation",
        – gx-participant:website: "https://www.tecnalia.com/",
        – gx-participant:registrationNumber: {
             gx-participant:registrationNumberType: "vatID",
             gx-participant:registrationNumberNumber: "ESG48975767"
          },
        – gx-participant:headquarterAddress: {
             gx-participant:addressCountryCode: "ES",
             gx-participant:addressCode: "ES-PV",
             gx-participant:street-address: "Parque Cientifico y
             Tecnologico de Bizkaia, Edificio 700",
             gx-participant:postal-code: "48160",
             gx-participant:locality: "Derio"
          },
        – gx-participant:legalAddress: {
             gx-participant:addressCountryCode: "ES",
             gx-participant:addressCode: "ES-PV",
             gx-participant:street-address: "Parque Cientifico y
             Tecnologico de Gipuzkoa, Mikeletegi Pasalekua 2",
             gx-participant:postal-code: "20009",
             gx-participant:locality: "San Sebastian"
          },
        – gx-participant:termsAndConditions:
          "70c1d713215f95191a11d38fe2341faed27d19e083917bc8732ca4fea497670
```


[9] https://wizard.lab.gaia-x.eu/v1

[10] https://compliance.lab.gaia-x.eu/development/docs/#/credential-offer/CommonController_issueVC

[11] https://registry.lab.gaia-x.eu/v1/api/trusted-shape-registry/v1/shapes/implemented

```
                0"
                        },
    -   proof: {
            type: "JsonWebSignature2020",
            created: "2022-10-18T06:51:48.649Z",
            proofPurpose: "assertionMethod",
            verificationMethod: "did:web:hodeix.digital.tecnalia.dev",
            jws:
            "eyJhbGciOiJQUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..kbjXzBzvS
            E7fPDKHoPIs0SceNjC7nYAR9tk6_... "
                }
        },
-   complianceCredential: {
      -   @context: [
                "https://www.w3.org/2018/credentials/v1"
          ],
      -   type: [
                "VerifiableCredential",
                "ParticipantCredential"
          ],
      -   id: "https://catalogue.gaia-
          x.eu/credentials/ParticipantCredential/1666075916527",
      -   issuer: "did:web:compliance.gaia-x.eu",
      -   issuanceDate: "2022-10-18T06:51:56.527Z",
      -   credentialSubject: {
            id: "did:web:hodeix.digital.tecnalia.dev",
            hash:
            "cd4671eb4fa46e9ae80284dae339e6171f7738d481e1bc72db88bfa632452917"
          },
      -   proof: {
            type: "JsonWebSignature2020",
            created: "2022-10-18T06:51:56.527Z",
            proofPurpose: "assertionMethod",
            jws:
            "eyJhbGciOiJQUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..qDGT7bU1W
            9tG3hl9VpsxCIv67HeCNsc_nd3lYleseQ...",
            verificationMethod: "did:web:compliance.gaia-x.eu"
          }
}
```


List 1: Self-description of the TECNALIA participant (V22.06).

The main difference with the previous version —apart of the minimalist definition of Tecnalia adopted to avoid mislead the structure with much information— is the simplification it has suffered, with the removal of all the *selfDescriptionCredential* header, maintaining only the *credentialSubject* part.


```
{
  "id": "did:web:hodeix.digital.tecnalia.dev",
  "type": "gx:LegalParticipant",
  "gx:legalName": "Fundacion Tecnalia Research & Innovation",
  "gx:webAddress": "https://www.tecnalia.com",
  "gx:legalRegistrationNumber": {
    "gx:vatID": "ESG48975767"
  },
  "gx:headquarterAddress": {
    "gx:countrySubdivisionCode": "ES-PV"
  },
  "gx:legalAddress": {
    "gx:countrySubdivisionCode": "ES-PV"
  },
  "gx:termsAndConditions":
```

```
"70c1d713215f95191a11d38fe2341faed27d19e083917bc8732ca4fea
   4976700"
}
```


List 2: Self-description of the TECNALIA participant (V22.10).

The self-description is then signed using the Gaia-X wizard[12], with the private key of the participant, which implies the verification method to be the Tecnalia DID, "did:web:hodeix.digital.tecnalia.dev". The result (see List 3) is to wrap the raw SD (marked in grey) with the *VerifiableCredential* type and context, and add in the end a proof of the signature, that allows in the future the validation that the content remains unchanged.


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://registry.lab.gaia-x.eu/development/api/trusted-shape-
registry/v1/shapes/jsonld/trustframework#"
  ],
  "type": [
    "VerifiableCredential"
  ],
  "id": "did:web:hodeix.digital.tecnalia.dev",
  "issuer": "did:web:hodeix.digital.tecnalia.dev",
  "issuanceDate": "2023-05-23T14:23:41.839Z",
  "credentialSubject": {
    "id": "did:web:hodeix.digital.tecnalia.dev",
    "type": "gx:LegalParticipant",
    "gx:legalName": "Fundacion Tecnalia Research & Innovation",
    "gx:webAddress": "https://www.tecnalia.com",
    "gx:legalRegistrationNumber": {
      "gx:vatID": "ESG48975767"
    },
    "gx:headquarterAddress": {
      "gx:countrySubdivisionCode": "ES-PV"
    },
    "gx:legalAddress": {
      "gx:countrySubdivisionCode": "ES-PV"
    },
    "gx:termsAndConditions":
"70c1d713215f95191a11d38fe2341faed27d19e083917bc8732ca4fea4976700"
  },
  "proof": {
    "type": "JsonWebSignature2020",
    "created": "2023-05-23T14:23:42.207Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:web:hodeix.digital.tecnalia.dev",
    "jws":"eyJhbGciOiJQUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..h9xezrJcc
rpuMngrh9yuyjY1q_RCZUjB6CzaGD2Hj2ycPCTwT1urChvI2OV4OXx0mqPk_xfUe49nVzMygCGvwFM
hQ4UIQ9NPfHFj_5rHlbLrIkff89Q1KkPBLMjTDTZGyRbSM3eymPu2Qe_vKIM89TQTL93Yd36vesdzf
KoIDb0gstFXpCgBpLC1VXLndIImwM3NdR3rSfavUX21CZ2wo7A6QYPSKMI8Am1ZJRWm3B1smBmGbZX
FwC3PgQKWhrMicXIT2jFV49g8FSv-UQ6B8F8f6dmxzDuuAf0kN6sFEbFszFSpwg-
_vJ2Z_tRWUp7xB9wCORnS5sUYjFxeWguBaw"
  }
}
```


List 3: Signed self-description of the TECNALIA participant (V22.10).

[12] https://wizard.lab.gaia-x.eu/

2.1.1.2.1 Service offering

The **ServiceOffering** is an abstraction of the offered service in the form of a *VerifiablePresentation* that contains two *VerifiableCredentials*, the first one as *LegalName* for the participant (Tecnalia) as the issuer of the service; and the second one as the proper *ServiceOffering*. The List 4 shows the Service Offering file.

```
{
  - @context: [
        "https://www.w3.org/2018/credentials/v1",
        "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/termsandconditions#",
        "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/participant#"
    ],
  - type: [
        "VerifiablePresentation"
    ],
  - verifiableCredential: [
      - {
          - @context: [
                "https://www.w3.org/2018/credentials/v1",
                "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/termsandconditions#",
                "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/participant#"
            ],
            type: "VerifiableCredential",
            id: "https://hodeix.digital.tecnalia.dev/.well-known/participantTecnalia_raw.json",
            issuer: "did:web:hodeix.digital.tecnalia.dev",
            issuanceDate: "2023-04-17T00:00:00.000Z",
          - credentialSubject: {
                id: "did:web:hodeix.digital.tecnalia.dev",
                type: "gx:LegalParticipant",
                gx:legalName: "Fundacion Tecnalia Research & Innovation",
                gx:webAddress: "https://www.tecnalia.com",
              - gx:legalRegistrationNumber: {
                    gx:vatID: "ESG48975767"
                },
              - gx:headquarterAddress: {
                    gx:countrySubdivisionCode: "ES-PV"
                },
              - gx:legalAddress: {
                    gx:countrySubdivisionCode: "ES-PV"
                },
                gx:termsAndConditions: "70c1d713215f95191a11d38fe2341faed27d19e083917bc8732ca4fea4976700"
            }
        },
      - {
          - @context: [
                "https://www.w3.org/2018/credentials/v1",
                "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#"
            ],
            type: "VerifiableCredential",
            id: "https://ci.piacere.digital.tecnalia.dev/.well-known/serviceOfferingPiacereCatalogue_2210.json",
            issuer: "did:web:hodeix.digital.tecnalia.dev",
            issuanceDate: "2023-04-17T00:00:00.000Z",
          - credentialSubject: {
                id: "https://ci.piacere.digital.tecnalia.dev/.well-known/PiacereCatalogue_2210.json",
                type: "gx:ServiceOffering",
              - gx:providedBy: {
                    id: "did:web:hodeix.digital.tecnalia.dev"
                },
              - gx:termsAndConditions: [
                  - {
                        gx:URL: "https://www.tecnalia.com/informacion-legal",
                        gx:hash: "70cf73cbb47217865004ae15949348f65272dda4da862e54c6b83615e8256809"
                    }
                ],
              - gx:policies: [
                    "default:allow"
                ],
              - gx:dataAccountExport: [
                  - {
                        gx:requestType: "email",
                        gx:accessType: "digital",
                        gx:formatType: "text/html"
                    },
                  - {
                        gx:requestType: "supportCenter",
                        gx:accessType: "physical",
                        gx:formatType: "text/html"
                    },
                  - {
                        gx:requestType: "webform",
                        gx:accessType: "digital",
                        gx:formatType: "text/html"
                    }
                ]
            }
        }
    ]
}
```


List 4: Service offering for the Catalogue.

2.1.1.2.2 Software resource

The **SoftwareResource** file contains basic information about the service, serving as an abstraction of the real resource behind:


```
{
  - @context: [
      "https://www.w3.org/2018/credentials/v1",
      "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#"
    ],
    type: "VerifiablePresentation",
  - verifiableCredential: [
    - {
      - @context: [
            "https://www.w3.org/2018/credentials/v1",
            "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#"
        ],
        type: "VerifiableCredential",
        id: "https://ci.piacere.digital.tecnalia.dev/.well-known/PiacereCatalogue_2210.json",
        issuer: "did:web:hodeix.digital.tecnalia.dev",
        issuanceDate: "2023-04-17T00:00:00.000Z",
      - credentialSubject: {
            id: "https://ci.piacere.digital.tecnalia.dev/.well-known/PiacereCatalogue_2210.json",
          - type: [
                "gx:ServiceOffering",
                "gx:SoftwareResource"
            ],
          - gx:providedBy: {
                id: "did:web:hodeix.digital.tecnalia.dev"
            },
          - gx:termsAndConditions: [
            - {
                  gx:URL: "https://www.tecnalia.com/informacion-legal",
                  gx:hash: "70cf73cbb47217865004ae15949348f65272dda4da862e54c6b83615e8256809"
              }
            ],
            gx:policy: "default:allow",
          - gx:dataAccountExport: [
            - {
                  gx:requestType: "email",
                  gx:accessType: "digital",
                  gx:formatType: "text/html"
              },
            - {
                  gx:requestType: "supportCenter",
                  gx:accessType: "physical",
                  gx:formatType: "text/html"
              }
            ],
            gx:name: "Piacere Catalogue",
            gx:description: "This software offers a Catalogue of cloud services that the user can deploy using PIACERE
            tool.",
            gx:copyrightOwnedBy: "2023 Tecnalia",
          - gx:license: [
                "https://www.apache.org/licenses/LICENSE-2.0",
                "http://spdx.org/licenses/Apache-2.0",
                "https://opensource.org/licenses/Apache-2.0"
            ]
        }
      }
    ]
}
```


List 5: Software resource file for the Catalogue.

2.1.1.2.3 Instantiated resource

The **InstantiatedResource** offers more information about the service being instantiated, the data center in which is located and where to obtain the service API (in this case, in the integration environment of PIACERE):

```
{
  - @context: [
        "https://www.w3.org/2018/credentials/v1",
        "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#"
    ],
  - type: [
        "VerifiablePresentation"
    ],
  - verifiableCredential: [
    - {
      - @context: [
            "https://www.w3.org/2018/credentials/v1",
            "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#"
        ],
        type: "VerifiableCredential",
        id: "https://ci.piacere.digital.tecnalia.dev/.well-known/instantiatedPiacereCatalogue_2210.json",
        issuer: "did:web:hodeix.digital.tecnalia.dev",
        issuanceDate: "2023-03-09T13:26:22.009Z",
      - credentialSubject: {
            id: "https://ci.piacere.digital.tecnalia.dev/.well-known/instantiatedPiacereCatalogue_2210.json",
          - type: [
                "gx:ServiceOffering",
                "gx:SoftwareResource",
                "gx:InstantiatedVirtualResource"
            ],
          - gx:providedBy: {
                id: "did:web:hodeix.digital.tecnalia.dev"
            },
            gx:policy: "default:allow",
          - gx:termsAndConditions: {
                gx:URL: "https://www.tecnalia.com/informacion-legal",
                gx:hash: "70c1d713215f95191a11d38fe2341faed27d19e083917bc8732ca4fea4976700"
            },
          - gx:dataAccountExport: {
                gx:requestType: "email",
                gx:accessType: "digital",
                gx:formatType: "text/html"
            },
            gx:name: "Piacere Catalogue",
            gx:description: "This software offers a Catalogue of cloud services that the user can deploy using PIACERE
            tool.",
          - gx:maintainedBy: {
                id: "did:web:hodeix.digital.tecnalia.dev"
            },
          - gx:tenantOwnedBy: {
                id: "did:web:hodeix.digital.tecnalia.dev"
            },
            gx:hostedOn: "https://ci.piacere.digital.tecnalia.dev/.well-known/datacenterTecnalia_2210.json",
            gx:instanceOf: "https://ci.piacere.digital.tecnalia.dev/.well-known/PiacereCatalogue_2210.json",
          - gx:serviceAccessPoint: {
                gx:name: "Piacere Catalogue",
                gx:protocol: "https",
                gx:version: "8.2.1",
                gx:port: " 8080",
                gx:openAPI: "https://iec.ci.piacere.digital.tecnalia.dev/v3/api-docs"
            }
        }
    }
  ]
}
```


List 6: Instantiated resource file for the Catalogue.

2.1.1.2.4 Data Center

Finally, we have defined the **DataCenter** file that supports the service instantiation. It defines the physical address where the data center is located:

```
{
  - @context: [
        "https://www.w3.org/2018/credentials/v1",
        "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#"
    ],
    type: "VerifiablePresentation",
  - verifiableCredential: [
      - {
        - @context: [
              "https://www.w3.org/2018/credentials/v1",
              "https://registry.lab.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#"
          ],
          type: "VerifiableCredential",
          id: "https://ci.piacere.digital.tecnalia.dev/.well-known/datacenterTecnalia_2210.json",
          issuer: "did:web:hodeix.digital.tecnalia.dev",
          issuanceDate: "2023-04-17T00:00:00.000Z",
        - credentialSubject: {
              id: "https://ci.piacere.digital.tecnalia.dev/.well-known/datacenterTecnalia_2210.json",
            - type: [
                  "gx:ServiceOffering",
                  "gx:PhysicalResource"
              ],
            - gx:providedBy: {
                  id: "did:web:hodeix.digital.tecnalia.dev"
              },
            - gx:termsAndConditions: [
                - {
                      gx:URL: "https://www.tecnalia.com/informacion-legal",
                      gx:hash: "70cf73cbb47217865004ae15949348f65272dda4da862e54c6b83615e8256809"
                  }
              ],
              gx:policy: "default:allow",
            - gx:dataAccountExport: [
                - {
                      gx:requestType: "email",
                      gx:accessType: "digital",
                      gx:formatType: "text/html"
                  },
                - {
                      gx:requestType: "supportCenter",
                      gx:accessType: "physical",
                      gx:formatType: "text/html"
                  }
              ],
              gx:name: "Datacenter Zamudio (Spain)",
              gx:description: "Tecnalia datacenter located at Zamudio, Bizkaia (Spain)",
            - gx:maintainedBy: {
                  id: "did:web:hodeix.digital.tecnalia.dev"
              },
            - gx:locationAddress: {
                  gx:countrySubdivisionCode: "ES-RI"
              },
            - gx:location: [
                - {
                      gx:gps: "43.295935, -2.871209"
                  }
              ],
              gx:environmentalImpactReport: "https://www.tecnalia.com/ambitos/movilidad-sostenible"
          }
      }
    ]
}
```


List 7: Self-description file of the Catalogue DataCenter.

2.1.2 Changes regarding the IOP Optimizer

This subsection is devoted to highlight the main changes done in Y3 regarding the IOP Optimizer.

2.1.2.1 Advances made in the optimization problem formulation

To adequately comprehend the optimization problem at hand, let us introduce an example in this section. First of all, as mentioned before, for properly conducting the optimization process, the IOP counts on the IEC to be full of resources that the solver can consider in order to select the most optimal deployment configuration.

For the purpose of this explanation, we introduce here a simplified IEC composed of three kinds of elements, [Storage (ST), Database (DB), Virtual Machine (VM)], having five options for each of these categories:

ST: [St_EU, St_Italy, AZ_USA, G_2, St_Germany, St_2]

DB: [mysq.dyn, db.sql, postgre.GR, r2.small, m5.large]

VM: [C1Italy, C2Spain, m3.small, t4.medium, DS15v3]

Additionally, all elements have some associated attributes, such as the expected performances, the overall availability, provider, or cost. Having said that, the main objective of the IOP is to find the best combination of these elements that comply with the user requirements.

In this regard, and for obtaining all the information about the user's needs, the IOP should obtain as input a file written in DOML. This file includes a specific section in which users introduce their optimization objective and requirements. At the same time, it is also interesting to spotlight that the DOML is also the output of the IOP Optimizer. More specifically, the output produced is the same DOML introduced as input, but with the information of the optimization introduced on it. Despite different DOML excerpts are depicted along the following subsections, interested reader can find complete files in this same document, in APPENDIX B. Also, we refer readers to [4] for further details on DOML.

For the sake of clarity, we depict in Figure 8 an excerpt of a DOML example. Analyzing this file, we can see how the user wants to find the most optimized configuration, composed of a single ST, one DB and three VMs (depicted in the part elements) optimizing three different objectives: cost, availability, and performance. Furthermore, the user deepens on its requirements, introducing three different ones: the cost of the whole deployment should be less than 100€, the overall expected availability must be higher than 98%, and all the chosen elements should be deployed in Europe. As mentioned in the last deliverable D5.8, the user has also the chance of introducing a minimum performance desired for the whole deployment.

```
optimization opt {
    objectives {
        "cost" => min
        "availability" => max
        "performance" => max
    }
    nonfunctional_requirements {
        req1 "Cost <= 100.0" max 100.0 => "cost";
        req2 "Availability >= 98.0%" min 98.0 => "availability";
        req3 "Region" values "00EU" => "region";
        req4 "elements" => "Storage, DB, VM, VM, VM";
    }
}
```

Figure 8: Excerpt of a possible input DOML

Considering these objectives and requirements, and resorting the reduced IEC presented above, the IOP codifies candidate solutions as a list of integer values, each one depicting the index of a specific element. Thus, a tentative solution looks like: [0, 1, 0, 2, 3]: [100, 98.5, 10]. On the one hand, and deeming that the combination that the user wants to deploy is [ST, DB, VM, VM, VM], the solution [0, 1, 0, 2, 3] is equal to [St_Eu, db.sql, C1Italy, m3.small, t4.medium]. On the other hand, [100, 98.5,

10] array represents the cost of the whole deployment (100), the expected availability (98.5) and the overall performance (10).

On the basis of this formulation, the following advances have been conducted in the last six months of the project.

2.1.2.1.1 Adding new requirements

Since March 2023, the PIACERE Working group organizes weekly session dedicated to test the different tools implied in the Design Time phase of the whole system. Being the IOP part of this phase, it has been tested in these sessions, in which the need for an additional requirement arised. This requirement regards the maximum memory that the elements that compose the deployment must have, and it is included as depicted in Figure 9.

```
nonfunctional_requirements {

    Req1 "Region" values "00EU" => "region";
    Req2 "Provider" values "AMAZ" => "provider";
    Req3 "elements" => "VM, VM";
    Req4 "max_VM_memory" => "1024,1024";

}
```

Figure 9: Example of the new requirement

More specifically, the new requirement is the one identified as `Req4`, which in combination with the rest of the requirements implies that the IOP should find a deployment composed of two different virtual machines, deployed in Europe, provided by Amazon and each one with a maximum of 1024MB. In Figure 10 we depict a possible solution found by the IOP, in which the memory of the selected virtual machines is 900 and 1024, respectively.

```
concrete_infrastructure opt_infra1{
    provider AMAZ {
        vm p2_8xlarge{
            properties {
                vm_flavor = "p2_8xlarge";
                vm_name = "p2_8xlarge";
                vm_Availability = 99.9;
                vm_Response_time_Virtual_Machine_Performance = 2;
                vm_Zone = "UKEU";
                vm_Memory = 900;
                vm_Frequency_per_Core = 1500;
                vm_Virtual_CPU_Cores = 32;
                vm_provider_OU = "AMAZ";
                vm_public_IP_type = "IPV4";
                vm_Cost_Currency = 5600;
                vm_Region = "00EU";
                vm_Instance_Storage = 250;
                vm_Optimized_for = "GPUI";
            }
            maps nginx_vm
        }
        vm m4_4xlarge{
            properties {
                vm_flavor = "m4_4xlarge";
                vm_name = "m4_4xlarge";
                vm_Availability = 99.8;
                vm_Response_time_Virtual_Machine_Performance = 2;
                vm_Zone = "SPEU";
                vm_Memory = 1024;
                vm_Frequency_per_Core = 1500;
                vm_Virtual_CPU_Cores = 16;
                vm_provider_OU = "AMAZ";
                vm_Cost_Currency = 640;
                vm_Region = "00EU";
                vm_Instance_Storage = 120;
                vm_Optimized_for = "GEPU";
            }
            maps mysql_vm
        }
        net opt_network{
            maps common_network
        }

    }
}
```


Figure 10: Solution compliant with the new requirement

2.1.2.1.2 Showing the best solutions

One of the main needs detected during the Design Time sessions regards the number of solutions provided by the IOP. When the optimization problem to solve is not much restrictive, the previous version of the tool provides an amount of solutions that could be considered as excessive for the user. In other words, the user will not find useful to receive, for example, 20 different deployment configurations.

For this reason and based on the ranking criterion followed by the IOP, the tool provides the five best solutions founds even if the problem to solve is not very restrictive (such as the one depicted in Figure 11, where no non-functional requirements are introduced). For the sake of completeness, the IOP ranks the outcomes based on the first of the objectives introduces. In this case, the five solutions with the less cost will be offered to the user.

```
optimization opt{

    objectives {

        "cost" => min
        "performance" => max
    }

    nonfunctional_requirements {

        Req1 "elements" => "VM, VM";

    }

}
```

Figure 11:Example of a non-restrictive non-functional requirements

Thus, in Figure 12 we represent the output DOML built by the IOP for the objectives and requirements shown in previous figure.

```
optimization opt{

    objectives {

        "cost" => min
        "performance" => max
    }

    nonfunctional_requirements {

        Req1 "elements" => "VM, VM";

    }

    solution sol1 {
        objectives {
            cost 14.530000000000001 euro
            performance 13.0 metric
        }
        decisions ["[m1.tiny, t2.nano]"]
    }
    solution sol2 {
        objectives {
            cost 25.0 euro
            performance 15.0 metric
        }
        decisions ["[C1_USA, m1.tiny]"]
    }
    solution sol3 {
        objectives {
            cost 190.53 euro
            performance 103.0 metric
        }
        decisions ["[t2.nano, VM_1_CS]"]
    }
    solution sol4 {
        objectives {
            cost 196.0 euro
            performance 110.0 metric
        }
        decisions ["[m1.tiny, VM_1_CS]"]
    }


}
```

Figure 12: Solution provided by the IOP for the DOML

2.1.2.1.3 Mapping elements

An additional improvement included in the new version of the IOP regards the mapping of the elements that compose the solutions provided by the IOP. In this sense, and in order the tool to know the mappings that should be considered, the IOP first analyses the input DOML in order to retrieve the information about the mapping elements. More concretely, the layer to check is the coined as `infrastructure` layer. In Figure 13 we represent an example DOML, in which two different VMs should be mapped: `nginx_vm` and `mysql_vm`. Furthermore, a network named as `common_network` must also be included in each of the concretizations found by the IOP.

```
infrastructure infra_example1 {

    vm nginx_vm {
        arch "x86-64"
        os "Ubuntu-22.04.2-LTS"
        mem_mb 1024.0
        sto "16GB"
        cpu_count 1
        size "small"
        loc {
            region "00EU"
        }
        iface nginx_iface {
            belongs_to nginx_subnet
        }
        credentials nginx_vm_credentials
    }

    vm mysql_vm {
        arch "x86-64"
        os "Ubuntu-22.04.2-LTS"
        mem_mb 1024.0
        sto "16GB"
        cpu_count 1
        size "small"
        loc {
            region "00EU"
        }

        iface mysql_iface {

            belongs_to mysql_subnet
        }

        credentials mysql_vm_credentials

    }

    net common_network {

        subnet nginx_subnet {

            protocol "TCP/IP"

            cidr "10.0.144.0/25"

        }
```

Figure 13: An excerpt of an example infrastructure layer, with two VMs and a network.

In Figure 8, an example of a solutions including this mapping is shown. In this figure, the mapping of the elements can be seen inside the concretization of the solution `opt_infra1`. It should be clarified that the criterion for assigning the mapping is based on the order in which they appear in the infrastructure layer. For example, the first VM appearing in the infrastructure layer is the first to be assigned in the concretization.

2.1.2.1.4 Modifying the active infrastructure

The last of the evolutions included in the new version of the IOP regarding the problem solving is related with the active infrastructure. In few words, each DOML should have one active infrastructure in order to be properly validated. In the new version of the IOP, the tool modifies the active infrastructure putting as the new one the best of the solutions found. In Figure 14 we represent an example including just one optimized solution, aiming to facilitate the visualization of this new feature. In this example, the new active infrastructure is the one found by the IOP: `opt_infra1`.

```
concretizations {
        concrete_infrastructure opt_infra1{
        provider AMAZ {
            vm p2_8xlarge{
                properties {
                    vm_flavor = "p2_8xlarge";
                    vm_name = "p2_8xlarge";
                    vm_Availability = 99.9;
                    vm_Response_time_Virtual_Machine_Performance = 2;
                    vm_Zone = "UKEU";
                    vm_Memory = 900;
                    vm_Frequency_per_Core = 1500;
                    vm_Virtual_CPU_Cores = 32;
                    vm_provider_OU = "AMAZ";
                    vm_public_IP_type = "IPV4";
                    vm_Cost_Currency = 5600;
                    vm_Region = "00EU";
                    vm_Instance_Storage = 250;
                    vm_Optimized_for = "GPUI";
                }
                maps nginx_vm
            }
            vm m4_4xlarge{
                properties {
                    vm_flavor = "m4_4xlarge";
                    vm_name = "m4_4xlarge";
                    vm_Availability = 99.8;
                    vm_Response_time_Virtual_Machine_Performance = 2;
                    vm_Zone = "SPEU";
                    vm_Memory = 1024;
                    vm_Frequency_per_Core = 1500;
                    vm_Virtual_CPU_Cores = 16;
                    vm_provider_OU = "AMAZ";
                    vm_Cost_Currency = 640;
                    vm_Region = "00EU";
                    vm_Instance_Storage = 120;
                    vm_Optimized_for = "GEPU";
                }
                maps mysql_vm
            }
            net opt_network{
                maps common_network
            }

        }
    }
    active opt_infra1
}
```

Figure 14: Example of how the IOP modified the active infrastructure.

2.1.2.2 Advances made regarding the IDE integration

In the period that goes from M12 to M24 we elaborated on the preliminary integration of the IOP in the Eclipse based IDE that is being developed in the context of PIACERE project. In this regard, the steps that a user should conduct in order to optimize a DOML are quite simple, and consist of just clicking the right mouse button on a DOML file, select *PIACERE* and then *Optimize DOML*. We represent in Figure 15 a graphical example of this process.

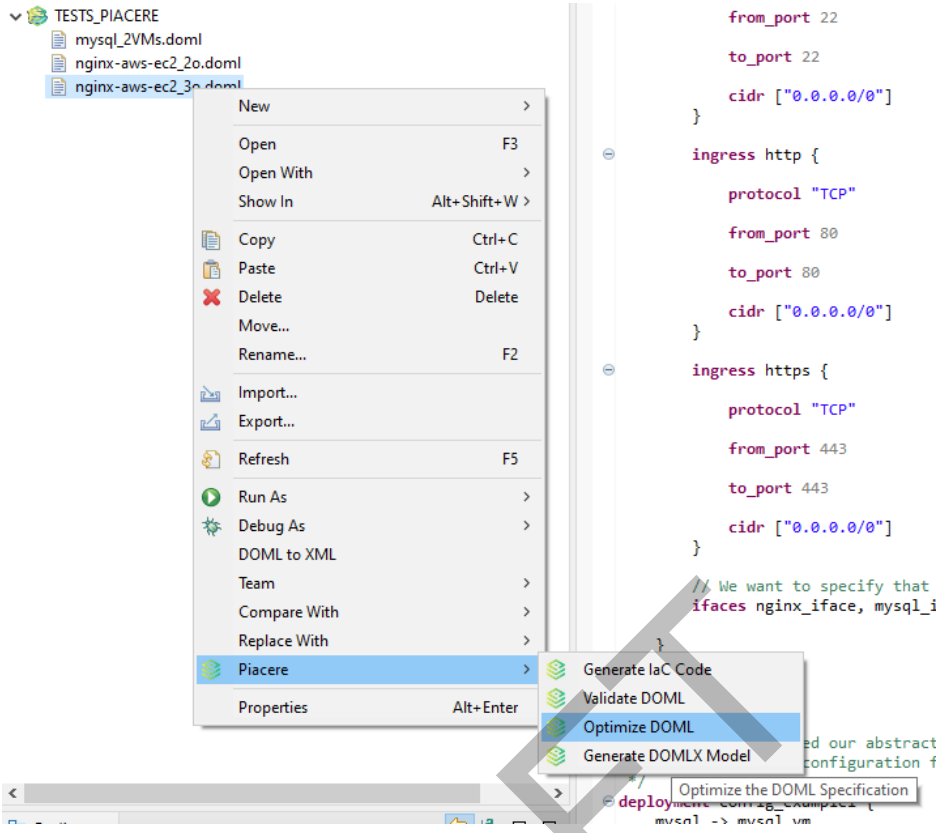

Figure 15: How to optimize a DOML in the IDE

After this first integration, use case owners started to conduct preliminary tests of the tool in the IDE. This first experiments arise a need that directly affect the operation of the IOP: the necessity of having a more descriptive error message. Up to M24, the IDE just shown a message saying "*The model cannot be optimized*" when the IOP chases or when it does not find any solution. For this reason, and in order to guide the user in the process of performing an optimization, use case owners propose the improvement of generating better message errors.

With this motivation in mind, six different alerts have been generated, giving an answer to six specific situations related to the IOP. We describe now these scenarios and how the IDE provides the error message.

2.1.2.2.1 Non-existence of Optimization section in the input DOML

The first of the errors regards the non-existence of the Optimization section in the input DOML. This section is the one containing the objectives to optimize and the requirements that the optimizer should meet. As can be seen in Figure 16, if this situation occurs the IDE triggers an alert saying "*The input DOML has no optimization part. It is compulsory for conducting the optimization process"*.

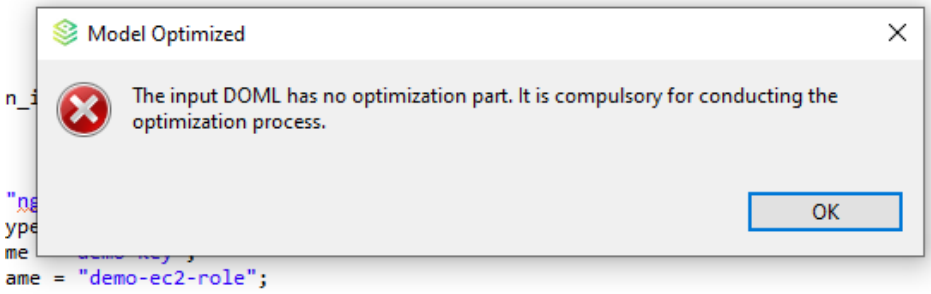

Figure 16: IDE error related to the non-existence of the optimization section.

2.1.2.2.2 Objectives non introduced

In order the IOP perform the optimization, it need a pull of objectives. As mentioned along these deliverables, the objectives can be any combination among cost, availability, and performance. If the user does not introduce any of it, such as depicted in the example in Figure 17, an alert saying *No objectives in the input DOML. Objectives should be "cost", "availability" or "performance" is arisen*. This error is shown in Figure 18.

```
optimization opt {

    nonfunctional_requirements {
        req1 "Cost <= 200" max 200.0 => "cost";
        req2 "Availability >= 96.0%" min 96.0 => "availability";
        req3 "elements" => "VM, VM, VM, Storage, Storage";
    }
}
```

Figure 17: Example of an input DOML with no objectives section.

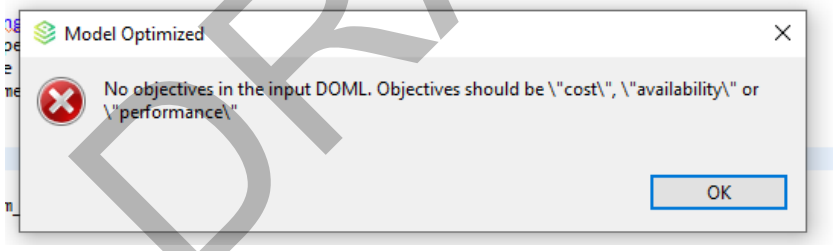

Figure 18: Error alert regarding the non-existence of optimization objectives.

2.1.2.2.3 Objectives badly introduced

As mentioned in the previous section, the IOP can build a problem optimizing any combination of cost, availability, and performance objectives. The IOP is a tool flexible enough for contemplating additional objectives to these three, but at this moment the tool just works with them. For this reason, if the user introduces another objective, as can be seen in Figure 19, the IDE shows *an There is an objective badly introduced in the input DOML. Objectives should be "cost", "availability", or "performance"* error. This error is represented in Figure 20.

```
optimization opt {
    objectives {
        "cost" => min
        "memory" => max
    }
    nonfunctional_requirements {
        req1 "Cost <= 200" max 200.0 => "cost";
        req2 "Availability >= 96.0%" min 96.0 => "availability";
        req3 "elements" => "VM, VM, VM, Storage, Storage";
    }
}
```

Figure 19: Example of an objective badly introduced.

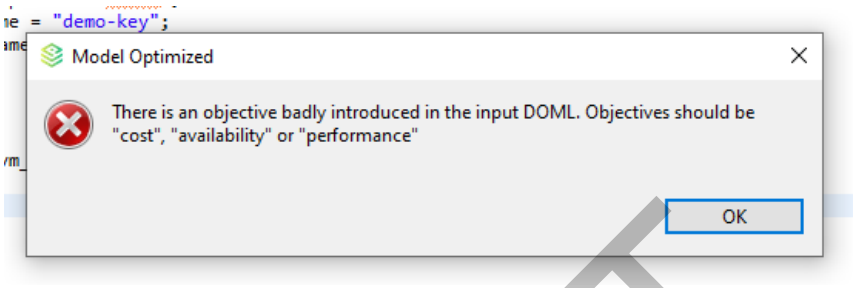

Figure 20: Error message regarding a badly introduced objective.

2.1.2.2.4 Non- considered requirement introduced

As in the case of the objectives, the IOP is a tool flexible enough for consider additional requirements needed by the user. In any case, at this time, seven requirements are considered, regarding the i) cost, ii) performance, iii) availability, iv) elements to deploy, v) providers, vi) region of the elements and vii) memory. For this reason, if the input DOML contemplates a requirement not considered by the IOP, as can be seen in Figure 21, so the system triggers an alert saying *There is a requirement badly introduced in the input DOML. Requirements should be cost, availability, performance, region, provider, or elements*. Figure 22 represents this situation.

```
optimization opt {
    objectives {
        "cost" => min
        "availability" => max
    }
    nonfunctional_requirements {
        req1 "Cost <= 200" max 200.0 => "cost";
        req2 "Cores = 3" min 3 = "Cores";
        req3 "elements" => "VM, VM, VM, Storage, Storage";
    }
}
```

Figure 21: Input DOML with a non-considered requirement.

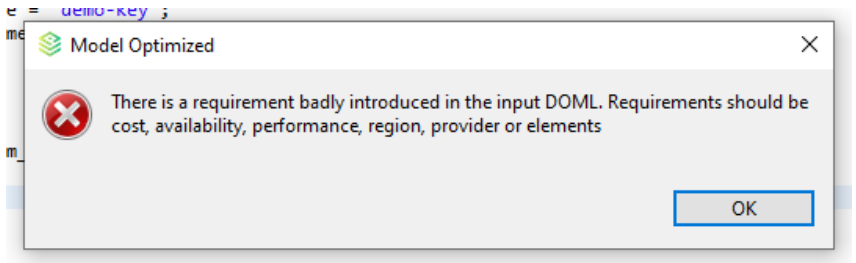

Figure 22: IOP error related with non-considered requirement.

2.1.2.2.5 Non-considered element introduced

At this moment, the IOP considers three different types of elements for building the deployment configuration: Virtual Machines, Storages and Data Bases. This is so because in the IEC these elements are deemed. For this reason, the IOP could work with further elements if in the future the IEC stores additional kind of elements.

With this situation, the IOP contemplates errors in which the user introduces infrastructural elements in the `elements` non-functional requirements part that are outside the scope of the current version of the tool. In Figure 23 we represent this situation with a DOML asking for a system element. In Figure 24 we show the corresponding error depicted by the IDE, which is *There is an element badly introduced in the input DOML. Elements should be VM, DB or Storage*.

```
optimization opt {
    objectives {
        "cost" => min
        "availability" => max
    }
    nonfunctional_requirements {
        req1 "Cost <= 200" max 200.0 => "cost";
        req2 "Persistence >= 96.0%" min 96.0 => "availability";
        req3 "elements" => "VM, VM, VM, System";
    }
}
```

Figure 23: A DOML with a non-considered element introduced.

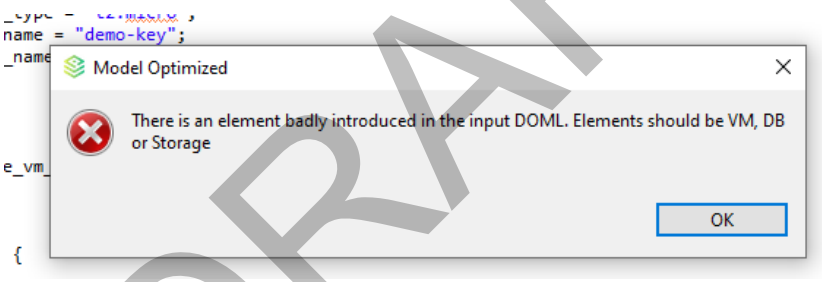

Figure 24: IOP error related with the introduction of a non-considered element

2.1.2.2.6 Very restrictive requirements

The IOP has the advantage of finding the most optimized deployment configurations based on the objectives predefined by the user. Furthermore, the tool provides the possibility of introducing some requirements for refining the search. For example, it could be possible that the user only wants elements given by the one specific provider, such as Azure or Amazon.

This functionality offered by the IOP arises a tentative situation: the potential of finding no solutions if the user requirements are very restrictive. This scenario should also be taken into account by the IOP in order to guide the practitioner to the best use of the system.

In Figure 25 we represent a DOML that fits with the error that we are describing in this subsection. In this case, the user needs a deployment composed of two virtual machines provided by Amazon, asking for an outstanding availability and a very low cost. Of course, this deployment does not exist, so the IOP cannot find a solution that meets the requirements employed as input. For this reason, as shown in Figure 26, the IDE returns a "*No possible solutions found. Maybe the restrictions are so strict to find a possible solution"* error message.

```
optimization opt {
    objectives {
        "cost" => min
        "availability" => max
    }
    nonfunctional_requirements {
        req1 "Cost <= 10" max 10.0 => "cost";
        req2 "Persistence >= 99.5%" min 99.5 => "availability";
        Req3 "Provider" values "AMAZ" => "provider";
        req4 "elements" => "VM, VM";
    }
}
```

Figure 25: Example of so restrictive DOML.

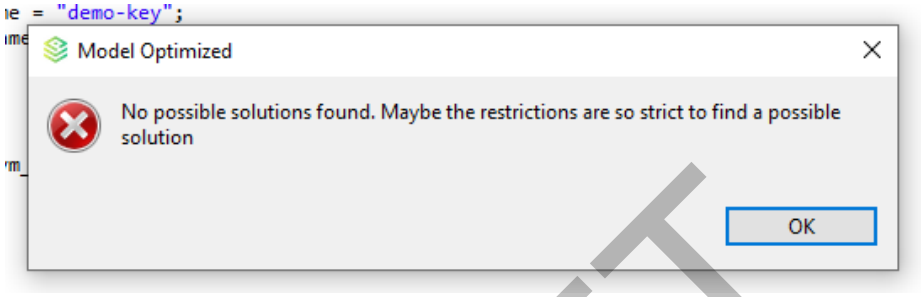

Figure 26: IOP alert regarding so restrictive demands.

2.2 Functional description and requirements coverage

The user requirements satisfied by this interim version are described in Table 2. All these requirements have been obtained from the *PIACERE WP2 Requirements* internal document.

Table 2: user requirements satisfied by the IOP.

Req ID	Description	Status	Requirement Coverage at M30
REQ03	IOP will include a catalogue of infrastructural elements (e.g. (edge, node) computation, networks, cloud services (e.g. IaaS, PaaS, SaaS)) classifiable by a set of constraints (e.g. memory, disk, …). This catalogue of infrastructural elements should be clearly defined, including possible restrictions and dynamic variations. These infrastructural elements will be transformed as optimization variables, and they will be intelligently treated by the optimization algorithm seeking to find the best configuration deployment.	Satisfied	The connection between the Optimizer and the IEC is already made, and it works properly. The optimization is performed using the elements that compose the IEC.
REQ04	Provide the means for the IOP to properly consume all the data related with the catalogue of infrastructural elements status, as well as their characteristics and possible variations. Special mention shall be done here to the values monitored by the self-	Partially satisfied	The Optimizer properly consumes the data of the IEC, and it properly takes all its characteristics for conducting the optimization. Next steps include the consideration of the measures coming from self-learning and monitoring for updating the characteristics of the IEC

	learning algorithm / monitoring component. This module shall provide real measures regarding the infrastructural elements in order to update their characteristics.		elements.
REQ12	The IEM shall allow redeployment and reconfiguration, both full and partial, as allowed by the used IaC technology.	**Discarded**	This requirement was finally discarded, since it corresponds to other KR.
REQ37	CSE to have a simulated mode limited to provisioning.	**Discarded**	This requirement was finally discarded, since it corresponds to other KR.
REQ46	The monitoring component shall gather metrics from the instances of the infrastructural elements at run time. These metrics need to be related to the NFR and accessible to the IOP (through the dynamic part of the infrastructural catalogue)	Partially satisfied	The metrics are gathered and stored in the monitoring system. The connection between the IEC and the monitoring system has not been done yet. When done, is expected that the IOP will be able to access the information modified by the monitoring component, since these modifications are directly done on the IEC data.
REQ98	The IOP components provide feedback on the DOML code, without doing automatic writes. The end user can choose to accept or not the feedback received. (ex REQ56&75)	Satisfied	The Optimizer receives a DOML as input, and it provides as solution the same DOML but with the results of the optimization inserted.

Furthermore, the internal requirements satisfied up to M30 are described in Table 3. These requirements are the continuation of those depicted in the previous version of this deliverable, and they have been polished and adapted as the project advances.

Table 3. internal requirements satisfied by the IOP.

Req ID	Description	Status	Requirement Coverage at M30
WP5.3-REQ1	Load/read information about the catalogue of infrastructural elements	Satisfied	The connection between the IEC and the Optimizer has been satisfactorily carried out.
WP5.3-REQ2	IOP shall consider both functional and non-functional requirements for searching the best solutions that fit the user demand.	Satisfied	This requirement was already satisfied in M12. In any case, in this new stage of the project, the consideration of this requirements is done thought DOML language. Furthermore, in this second year of the project additional requirements have been considered and implemented (such as the multi-element optimization or the choosing of a provider).
WP5.3-REQ3	IOP shall use optimization algorithms such as genetic algorithms and provide a set of	Satisfied	This requirement has also been satisfied in M12. In any case, up to M24 further advances have been

	potential combinations of elements that fulfill the established user requirements.		conducted on this aspect. These advances contemplate the adaptation of the algorithms to the use-case oriented problem and the ranking of the provided solutions.
WP5.3-REQ4	The IOP should be able to obtain the information provided by the user in DOML language	Satisfied	The IOP understands DOML language and can obtain the information related to the optimization objectives and requirements properly.

2.3 Main innovations

The IOP is a tool that counts with a group of innovations. These novelties are crucial both to respond to the needs identified by customers over the 30 months of the project, and to ensure both the replicability and scalability of the tool. We present now a summary of the innovations of the IOP:

- The IOP has been developed for solving the problem of finding the most optimized combination of infrastructural and resource configuration for properly deploying a service. Additionally, the IOP allows the user to configure the optimization problem, being able to choose different objectives to optimize and non-functional requirements to consider. This approach has been scarcely considered in the literature before [5]. As a result of this innovation, three different publications have been achieved with their focus put on the IOP [6] [7] [8].
- The IOP is a multi-method tool, which resorts on two different multi-objective algorithms depending on the needs of the user. This feature has been endowed to the IOP after the conduction of a comprehensive scientific oriented experimentation, which has been detailed in the upcoming Section 4.
- Despite the IOP works with two evolutionary computation multi-objective methods (how they work are described in the following Section 3), its structure allows the easy introduction of further multi-objective techniques, just for testing purposes or for exploring the improvement of its performance.
- The IOP is an open tool, flexible enough for considering further objectives and requirements, aiming to contemplate additional real-world situations. Furthermore, and because of its multi-method nature, the consideration of more objectives will not impact in the performance of the tool, having the NSGA-III method for dealing with many-objective problems.
- The Catalogue of the IOP offers a multi-provider service catalogue. This allows the user to search and select multiple services in a single tool, without having to worry about looking different providers offerings.
- The Catalogue offers also the possibility to define user-specific templates for its in-house services (for example, those provided by VMware vSphere). This way, the user can specify a set of his favourite configurations and use them in a recurrent manner in PIACERE.
- The compliance of the Catalogue of services with Gaia-X standards makes the Catalogue tool more trustable and interoperable. This compliance has been worked out and has been verified as far as the actual specification and tools of Gaia-X allow. In a further step, the inclusion in the future Gaia-X Catalogue of Services will allow more interaction with users and tools in the European cyberspace.

3 Overview of preliminary experiments

The experiments conducted with the IOP can be divided into two different branches. The first one, which can be coined as scientific experimentation, is the one with the goal of clearly determining which are the best optimizing algorithms to use in the IOP. The second tests have the objective of showing the applicability of the tool and how it can be adapted to solve the needs of the users.

3.1 Scientific experimentation

In that previous version of the IOP, which was described in D5.8, the tool resorts to NSGA-II and NSGA-III for carrying out the optimization. This selection was made based on the intuition and the experience of the developers involved in T5.3 of the project. In any case and being aware that it could be a good choice, we must take a step forward in the development of the IOP, undertaking a detailed experimentation to determine which is really the most suitable multi-objective method to deal with the problem of finding optimized IaC deployment configurations.

With this motivation, we have conducted a comprehensive experimentation, in which we have tested the behaviour of nine different multi-objective evolutionary computation solvers over 12 DOML instances. These use cases are heterogeneous enough to assure that the conclusions drawn are significant.

The interest on conducting this study is remarkably high since the IOP is expected to work on a real environment. For this reason, its performance should be as good as possible, and the selection of the optimization algorithms employed should be made after a thorough and rigorous study.

So, in order to perform a significant experimentation, we have measured the performance of the following nine different evolutionary computation based multi-objective techniques:

- Non-dominated Sorting Genetic Algorithm (NSGA-II, [9])
- Weighting Achievement Scalarizing Function Genetic Algorithm (WASFGA, [10]).
- Global Weighting Achievement Scalarizing Function Genetic Algorithm (GWASFGA, [11]).
- Many-Objective Metaheuristic Based on the R2 Indicator (MOMBI, [12]).
- Improved Many-Objective Metaheuristic Based on the R2 Indicator (MOMBI2, [13]).
- Multiobjective Cellular Genetic Algorithm (MoCell, [14]).
- S metric selection evolutionary multi-objective optimization algorithm (SMSEMOA, [15]).
- Strength Pareto evolutionary algorithm 2 (SPEA2, [16]).
- Non-dominated Sorting Genetic Algorithm III (NSGA-III, [17])

All these algorithms have been chosen for three main reasons: *i)* they are well-known, and they have been previously validated by the scientific community, *ii)* they can be easily adapted to the discrete problem dealt by the IOP, and *iii)* they can be implemented through the multi-objective framework embraced by the IOP: jMetal JAVA framework [1].

In order to properly assess the performance of all the nine multi-objective solvers, an in-depth experimentation has been carried out using 12 different input DOMLs. Each of these input files has been generated with the intention of emulating real situations that the IOP will face once it is deployed in a real environment. Thus, these DOML files present different objectives to optimize, as well as a variable number of elements to deploy. This directly impacts on the complexity of the problem, depending on the number of objectives and the size of the solution

to be found. We summarize in **¡Error! No se encuentra el origen de la referencia.** the main characteristics of the benchmark composed. As can be seen, each instance has been named as `DOML_A_x-y-z`, where `A` represents the number of objectives, and `x`, `y`, `z` the number of VMs, DBs and STs to deploy, respectively. For the sake of replicability, the generated benchmark is openly available under demand, or online at [18].

Furthermore, regarding the IEC that composes the problem, we have created an extended version of the current IEC, comprised of a total of 156 elements. Divided by type, the IEC contains 99 Virtual Machines, 24 Databases and 33 Storage elements. All these elements have realistic features, coming from providers such as Amazon, Google, Openstack or Azure. Arguably, the size is the employed catalogue is big enough for reaching rigorous conclusions.

Table 4. Main characteristics of the 12 input DOMLs generated. Regarding the optimizing objectives: c - cost; a - availability; p - performance. Instances optimizing three objectives consider c & p & a

Instance	objectives	Elements to deploy			Requirements
		VMs	DBs	STs	
DOML_2_0-4-4	2 (C & P)	0	4	4	
DOML_2_1-1-1	2 (C & A)	1	1	1	cost<200€
DOML_2_2-2-2	2 (C & A)	2	2	2	Availability>96%
DOML_2_4-3-3	2 (C & A)	4	3	3	
DOML_2_5-5-5	2 (C & P)	5	5	5	
DOML_2_6-0-0	2 (C & A)	6	0	0	
DOML_3_0-4-4	3	0	4	4	
DOML_3_1-1-1	3	1	1	1	cost<200€
DOML_3_2-2-2	3	2	2	2	Availability>96%
DOML_3_4-3-3	3	4	3	3	
DOML_3_5-5-5	3	5	5	5	
DOML_3_6-0-0	3	6	0	0	

Regarding the parameterization employed for each technique, as recommended in studies such as [19], similar parameters have been employed in all the methods developed. Thus, all methods count on a population composed of 50 individuals, while for the crossover and mutation functions, SBX [20] and Polynomial Mutation have been chosen, respectively. For the selection, Binary Tournament mechanism has been utilized. Finally, regarding the maximum number of evaluations, it has been fixed in 2500.

Having said that, we depict in Table 5 the main results obtained for each technique in all the generated instances. Every instance has been run 10 independent times, in order to obtain statistically relevant results. Thus, we represent the average hypervolume value obtained by each method.

Table 5. General overview of the experimentation carried out.

Instance	GWASFGA	MoCell	MOMBI	MOMBI2	SMSEMOA	SPEA2	WASFGA	NSGA-II	NSGA-III
DOML_2_0-4-4	0.725	0.891	0.976	0.817	0.966	0.858	0.632	**0.998**	0.907
DOML_2_1-1-1	0.790	0.964	**0.989**	0.597	0.976	0.816	0.523	0.966	0.784
DOML_2_2-2-2	0.796	**0.983**	0.973	0.923	0.922	0.982	0.925	**0.983**	0.973
DOML_2_4-3-3	0.937	0.964	0.976	0.937	0.966	**0.985**	0.973	0.964	0.913
DOML_2_5-5-5	0.546	0.928	0.890	0.824	0.961	0.939	0.670	**0.970**	0.950
DOML_2_6-0-0	0.886	0.980	0.924	0.901	0.966	0.983	0.805	0.995	**0.998**
DOML_3_0-4-4	0.626	0.977	0.969	0.749	0.969	0.953	0.497	0.987	**0.996**
DOML_3_1-1-1	0.718	0.554	0.660	0.648	0.745	0.693	0.610	0.727	**0.750**
DOML_3_2-2-2	0.543	0.727	0.711	0.592	0.734	0.709	**0.841**	0.625	0.756
DOML_3_4-3-3	0.549	0.858	0.963	0.674	0.970	0.965	0.590	0.957	**0.966**
DOML_3_5-5-5	0.513	0.971	0.790	0.822	0.944	0.927	0.517	0.946	**0.968**
DOML_3_6-0-0	0.458	0.940	0.975	0.526	**0.984**	0.972	0.610	0.964	0.944

At a first glimpse, we can conclude that the best performing methods along the benchmark are the NSGA-II and the NSGA-III. In any case, and following the guidelines provided in [21], the Friedman's non-parametric test for multiple comparisons has been conducted for confirming these preliminary conclusions. For performing this statistical test, the results average obtained by each method has been used. Thus, we depict in Table 6**¡Error! No se encuentra el origen de la referencia.** the results gathered after performing the test through KEEL platform [22].

Table 6. Average Friendman Rankings for each considered algorithm. The less the ranking, the better the performance.

Algorithm	Ranking
GWASFGA	7.9583
MoCell	4.5
MOMBI	4.1667
MOMBI2	7.2083
SMSEMOA	3.2917
SPEA2	4.1667
WASFGA	7.0833
NSGA-II	**3.1667**
NSGA-III	3.4583

Analysing the results provided by the Friedman's test, we can certify that the best performing technique along the whole benchmark is the NSGA-II, which is the one that has reached the best ranking. In any case, a deeper analysis of the results shown in Table 4 can lead us to much more interesting and valuable conclusions, taking into account that the IOP will work in a real environment in which its performance should be improved as much as possible.

Thus, if we analyse the performance of each technique according to the number of objectives it has to optimize, we reach to much more accurate insights. Thus, we have conducted two separate Friendman's non-parametric tests. On the one hand, the first test has been performed employing the outcomes obtained for instances `DOML_2_0-4-4`, `DOML_2_1-1-1`, `DOML_2_2-2-2`, `DOML_2_4-3-3`, `DOML_2_5-5-5` and `DOML_2_6-0-0`. On the other hand, results got in use cases `DOML_3_0-4-4`, `DOML_3_1-1-1`, `DOML_3_2-2-2`, `DOML_3_4-3-3`, `DOML_3_5-5-5` and `DOML_3_6-0-0` have been used for performing the second test. In Table 7, we represent the outcomes of these separated Friedman's non-parametric tests.

Table 7. Separated Friendman's test for instances optimizing two and three objectives. Rankings for each considered algorithm. The less the ranking, the better the performance

Two Objectives		Three Objectives	
Algorithm	Ranking	Algorithm	Ranking
GWASFGA	7.9167	GWASFGA	8
MoCell	4.1667	MoCell	4.8333
MOMBI	3.5833	MOMBI	4.75
MOMBI2	7.25	MOMBI2	7.1667
SMSEMOA	4	SMSEMOA	2.5833
SPEA2	3.6667	SPEA2	4.6667
WASFGA	7.3333	WASFGA	6.8333
NSGA-II	**2.3333**	NSGA-II	4
NSGA-III	4.75	NSGA-III	**2.1667**

The findings that can be drawn by examining the results shown in the table are even more valuable than those described above. Despite NSGA-II is, in overall, the method that performs better for the whole benchmark, if we undertake a separate analysis we see that NSGA-III behaves better for instances where the number of objectives to optimize is three.

As a final conclusion, and after carrying out the experimentation described in this study, we determine that the most appropriate way for implementing the IOP is to convert it into a multi-method system. Thus, the IOP should have a pool composed of two solving algorithms, using each of them according to the user needs: for problems with two optimizing objectives, NSGA-II should be used; while for situations with three objectives, the IOP should resort to NSGA-III.

3.2 Experimentation for showing the IOP applicability

For showing the applicability of the IOP in a use-case oriented scenario, we depict in this manuscript an example inspired on the use-case owned by Ericsson. On this regard, we should clarify that, because of the length of both input and output DOMLs, we have added these files as part of APPENDIX B of this same deliverable. As can be seen in the DOML input, in this preliminary version of the Ericsson's use case, the user asks the IOP to conduct an optimization under the following conditions:

```
optimization opt {
    objectives {
        "cost" => min
        "performance" => max
        "availability" => max
    }
    nonfunctional_requirements {
        req1 "cost <= 300" max 300.0 => "cost";
        req2 "performance >= 70%" min 70.0 => "performance";
        req3 "elements" => "VM, Storage";
    }
}
```

Figure 27: optimization section of the preliminary Ericsson use case

Which implies the optimization of the whole three considered objectives (cost, availability, and performance), and the consideration of three requirements: i) a maximum cost of 300, ii) a minimum performance of 70, and iii) the searching of a deployment including a Virtual Machine and a Storage.

Under these optimization conditions, the IOP returned five different solutions, considering all the features described in this same manuscript. As a summary, these are the solutions found.

```
solution sol1 {
    objectives {
        cost 10.530000000000001 euro
        performance 8.0 metric
        availability 98.75 %
    }
    decisions ["[Storage1_USA, t2.nano]"]
}
solution sol2 {
    objectives {
        cost 10.530000000000001 euro
        performance 8.0 metric
        availability 98.75 %
    }
    decisions ["[Storage1_Spain, t2.nano]"]
}
solution sol3 {
    objectives {
        cost 16.0 euro
        performance 15.0 metric
        availability 98.75 %
    }
    decisions ["[Storage1_USA, m1.tiny]"]
}
solution sol4 {
    objectives {
        cost 16.0 euro
        performance 15.0 metric
        availability 98.75 %
    }
    decisions ["[Storage1_Spain, m1.tiny]"]
}
solution sol5 {
    objectives {
        cost 21.0 euro
        performance 10.0 metric
        availability 99.5 %
    }
    decisions ["[Storage1_Spain, C1_USA]"]
}
```

Figure 28: the five solutions provided by the IOP to the preliminary Ericsson use case

As explained, the complete input and output DOMLs are available in the APPENDIX B of this manuscript.

4 Lessons learnt and outlook to the future

Regarding the IOP Optimizer, along these 30 months of work, several valuable lessons have been learnt. During the early stages of the project, and while the specifications where being consolidated, we designed and developed an academic-oriented IOP prototype, which had the objective to preliminarily solve the problem of searching the most appropriate IaC deployment configuration the best meets the preestablished objectives.

At this first step, the IOP did not even consider requirements, and it received and provided the information using JSON format. The main goal of the development team was to determine how to design the problem and decide which will be the JAVA framework used for implementing the tool. Thanks to this work, the first lessons was learnt:

- **Lesson learnt #1**: After analysing some well-known multi-objective optimization frameworks, such as MOEA[13], JCLEC-MO[14], Jenetic[15], ECJ[16] and jMetal[17], we decided to resort on the last of these frameworks for implementing the IOP. Several reasons encourage us to choose jMetal over the rest of alternatives: i) jMetal works better with single-objective problems, ii) jMetal offers more algorithms for working with many-objective problems, iii) Algorithms are more flexible to configure and modify in jMetal, and iv) the definition of the PIACERE problem has been demonstrated to be easier to conduct in jMetal in comparison to MOEA.

Once this first prototype of the IOP was built, the most challenging work was to make it usable for the use case owners of the project, and to really give an answer to the real-world oriented requirements that compose the problem. In this regard, several improvements were conducted, and a second lesson was learnt:

- **Lesson learnt #2**: in order the IOP to be utilizable in the context of PIACERE project, a more productive version was needed. The IOP was adapted to the DOML language, in order to be able to receive and provide the information in this format. Also, the IOP must resort on a single algorithm for conducting the optimization, and it should consider the existence of several non-functional requirements related with the cost, performance, availability, elements to deploy, region, provider, and memory. Furthermore, the IOP must provide the information in a way easy to understand to the user and fully compliant for the DOML checker.

Lastly, in the last phase of the project, a special effort has been conducted for accurately chose which should be the evolutionary computation based multi-objective algorithms used for the IOP. This last experimentation led us to another lesson learnt:

- **Lesson learnt #3:** after the conduction of a comprehensive experimentation, using 12 different DOML instances and 9 multi-objective algorithms, we have concluded that NSGA-II and NSGA-III should be the algorithms to be employed in the IOP Optimizer tool The first of the algorithm should be used for solving two-objective problems, while when the number of objectives is higher, the IOP should resort on the NSGA-III. In any

[13] moeaframework.org

[14] https://cs.gmu.edu/~eclab/projects/ecj/

[15] https://www.uco.es/kdis/research/software/jclec-mo/

[16] https://jenetics.io/

[17] jmetal.github.io/JMetal/

case and embracing the no-free-lunch theorem, it should be highlighted that these conclusions are based *on the experimentation conducted under the conditions explained in this manuscript.*

Regarding the IOP Catalogue, a number of lessons can be extracted from the work done in the project:

- **Lesson learnt #4** The Catalogue has suffered some adaptations towards the final phase of the project, to include data types that where not initially foreseen to be contained (images, pre-existing data). Although it has not involved big troubles, maybe an earlier definition of the workflow and data in the use cases would avoid re-engineering or repeated work.

In relation with the idea of having to change some things that were presupposed at the beginning, we can consider another lesson:

- **Lesson learnt #5:** The specification of the Catalogue has to be flexible enough to include different types of assets. In their original form, it included services as virtual machines, databases and storage. The requirements of the project have made necessary to introduce other types of resources, and it should be prepared to easily include even more until the end of the project.

Regarding the future work related with the IOP Optimizer, several research lines have been established to continue the interesting work developed in this project. On the one hand, we have planned to consider further objectives and non-functional requirements, in order the IOP to be able to provide efficient answers to further real-world scenarios. On the other hand, we have planned to conduct even a wider experimentation in order to test the efficiency of other evolutionary computation algorithms, searching to improve even more the quality of the IOP Optimizer.

As for the Catalogue, one of the main tasks foreseen is to extend the database of resources, offering more services for each provider and including more different providers. In this line, to maintain the services offered in the Catalogue synchronized with the real offer of the providers can be a tedious, day-to-day, thorough job, and to find some help for it or an automatic solution is desirable. Finally, an integration with the monitoring tools already provided by different cloud providers is a feature that will avoid the use of specific agent that have to be deployed along with the infrastructure.

5 Conclusions

This document has been devoted to detailing the third and last version of the IOP, which is comprised of the Infrastructural Elements Catalogue and the IOP Optimizer. First, the IEC in PIACERE has the goal of storing the characteristics of the elements available at service providers, and also the instances of every service being employed by the applications being deployed in the infrastructure.

Secondly, the IOP Optimizer resorts on multi-objective EC methods for finding the most optimized deployment configurations for the IaC. To do that, the optimizer obtains the features of the elements stored on the IEC. The correct choosing of the elements should be conducted following the need of the user, in terms of optimizing objectives and non-functional requirements. Thus, the IOP is considered as successful if the solutions offered are good enough for meeting the users' demands.

In the last version of this deliverable, we have put the focus on introducing the main improvements conducted in the months that go from M24 and M30. Also, we have highlighted the main innovations of the tools, and we have also put attention on discussing the lessons learnt and future work plan. Lastly, an important part of this deliverable has been dedicated to the description of the experimentation carried out for demonstrating the effectiveness of the tools developed.

6 References

[1] J. J. Durillo and A. J. Nebro, “jMetal: A Java framework for multi-objective optimization,” *Advances in Engineering Software,* vol. 42, no. 10, pp. 760-771, 2011.

[2] Gaia-X, “Gaia-X Trust Framework,” [Online]. Available: https://gaia-x.gitlab.io/policy-rules-committee/trust-framework/gaia-x_trust_framework/. [Accessed 15 05 2013].

[3] Gaia-X European Association for Data and Cloud AISBL, , "Gaia-X Architecture Document, 21.06 Release," 2021.

[4] PIACERE Consortium, “PIACERE Abstractions, DOML and DOML-E v3,” 2023.

[5] M. Ciavotta, G. Gibilisco, D. Ardagna, E. Di Nitto, M. Lattuada and M. da Silva, “Architectural design of cloud applications: A performance-aware cost minimization approach,” *IEEE Transactions on Cloud Computing,* vol. 10, no. 3, pp. 1571-1591, 2020.

[6] E. Osaba, J. Diaz De Arcaya, J. Alonso, J. L. Lobo, G. Benguria and I. Etxaniz, “Multiobjective Optimization Analysis for Finding Infrastructure-as-Code Deployment Configurations.,” in *ACM-11th International Conference on Computer and Communications Management (ICCCM 2023). Accepted for being presented in August 2023.*, 2023.

[7] E. Osaba, J. Diaz De Arcaya, J. L. Lobo, G. Benguria, I. Etxaniz and J. Alonso, “An Evolutionary Computation based Platform for Optimizing Infrastructure-as-Code Deployment Configurations.,” in *9th International Congress on Information and Communication Technology*, 2023.

[8] E. Osaba, J. Diaz de Arcaya, L. Orue-Echevarria, J. Alonso, J. L. Lobo, G. Benguria and I. Etxaniz, “PIACERE project: description and prototype for optimizing infrastructure as code deployment configurations.,” in *Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 71-72)*, 2022.

[9] K. Deb, A. Pratap, S. Agarwal and T. Metarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE transactions on evolutionary computation,* vol. 6, no. 2, pp. 182-197, 2002.

[10] A. B. Ruiz, R. Saborido and M. Luque, “A preference-based evolutionary algorithm for multiobjective optimization: the weighting achievement scalarizing function genetic algorithm,” *Journal of Global Optimization,* vol. 62, pp. 101-125, 2015.

[11] R. Saborido, Ruiz A. B. and M. Luque, “Global WASF-GA: An evolutionary algorithm in multiobjective optimization to approximate the whole Pareto optimal front,” *Evolutionary computation,* vol. 25, no. 2, pp. 309-349, 2017.

[12] R. H. Gómez and C. A. C. Coello, “MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator,” *IEEE congress on evolutionary computation,* pp. 2488-2495, 2013.

[13] R. Hernández Gómez and C. A. C. Coello, “Improved metaheuristic based on the R2 indicator for many-objective optimization,” *2015 annual conference on genetic and*

evolutionary computation, pp. 679-686, 2015.

[14] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro and E. Alba, "Mocell: A cellular genetic algorithm for multiobjective optimization," *International Journal of Intelligent Systems,* vol. 24, no. 7, pp. 726-746, 2009.

[15] N. Beume, B. Naujoks and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research,* vol. 181, no. 3, pp. 1653-1669, 2007.

[16] E. Zitzler, M. Laumanns and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK-Report,* vol. 103, 2001.

[17] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints," *IEEE transactions on evolutionary computation,* vol. 18, no. 4, pp. 577-607, 2013.

[18] E. Osaba, "DOML instances for optimizing Infrastructure-as-Code deployment configurations," Mendeley Data, 2023. [Online]. Available: https://data.mendeley.com/datasets/gs5kw9hmz8.

[19] E. Osaba, R. Carballedo, F. Diaz, E. Onieva, A. D. Masegosa and A. Perallos, "Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems," *Neurocomputing,* vol. 271, pp. 2-8, 2018.

[20] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex systems,* vol. 9, no. 2, pp. 115-148, 1995.

[21] J. Derrac, Garcia S, D. Molina and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation,* vol. 1, no. 1, pp. 3-18, 2011.

[22] J. Derrac, S. Garcia, L. Sanchez and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework.," *J. Mult. Valued Logic Soft Comput,* vol. 17, 2015.

[23] JHipster, "JHipster," [Online]. Available: http://www.jhipster.tech/. [Accessed November 2017].

[24] Eco - Association of the Internet Industry, "Core Catalogue Features," [Online]. Available: https://www.gxfs.de/federation-services/federated-catalogue/core-catalogue-features/.

[25] (OAI), The OpenAPI Initiative, "OpenAPI Specification v3.1.0," 15 February 2021. [Online]. Available: https://spec.openapis.org/oas/v3.1.0.html. [Accessed 08 11 2021].

[26] PIACERE Consortium, «D5.7 IOP Prototype v1,» 2021.

[27] PIACERE Consortium, «D5.8 IOP Prototype v1,» 2022.

[28] G.-X. E. A. f. D. a. C. AISBL, «Gaia-X Architecture Document, 21.06 Release,» 2021.

APPENDIX A: Implementation, delivery and usage

1 IEC Implementation

The main purpose of the catalogue is to manage the different infrastructure elements, using a data model that allows their classification based on properties. These characteristics, associated with the services registered in the catalogue, allow an optimal deployment configuration by the optimization algorithms.

1.1 Fitting into overall PIACERE Architecture

The Infrastructural Elements Catalogue (IEC) is one of the components of the PIACERE architecture. Although it is considered part of the IOP (KR9), in this section we detach them, to better understand its specificities

The Catalogue interacts with several tools in the PIACERE ecosystem, as can be seen in the following Workflow Diagrams (Figure 29 and Figure 30) and in the Sequence Diagram (Figure 31):

- The GUI/IDE, that is used by the user to include/edit element information in the Catalogue.
- The IOP, to whom it provides information about available elements and its related dynamic information.
- The Runtime Controller (PRC), from which receives information (that is originated in the IEM) about deployed instances or tore down instances.
- The Runtime Monitoring, that provides monitoring information of the running instances.

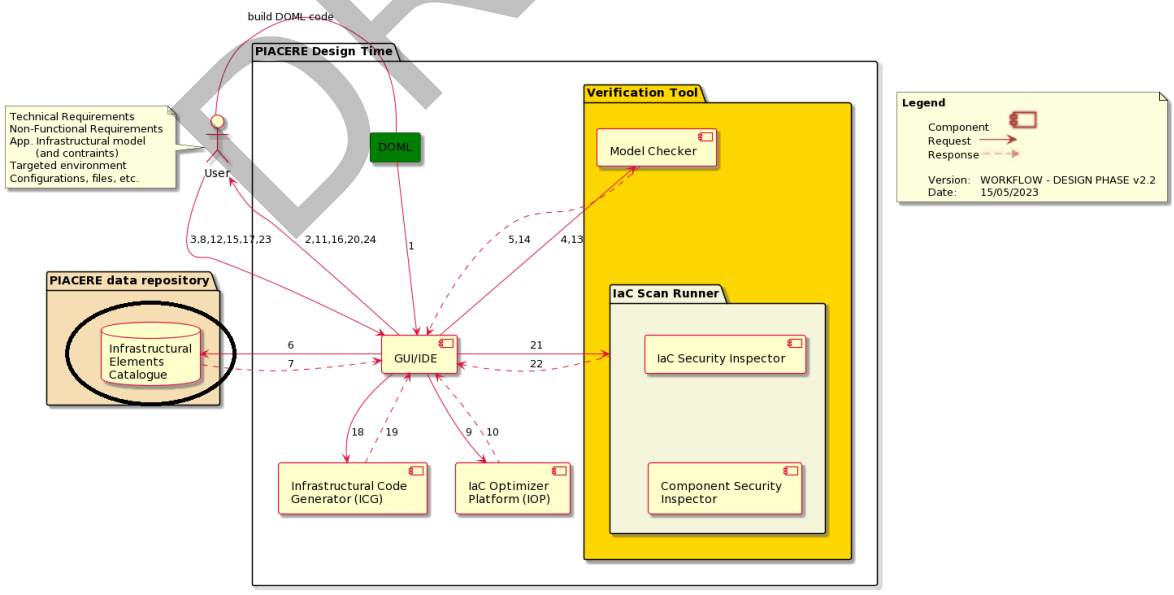

Figure 29: PIACERE Design Time Diagram (v2.2) with the Catalogue interactions.

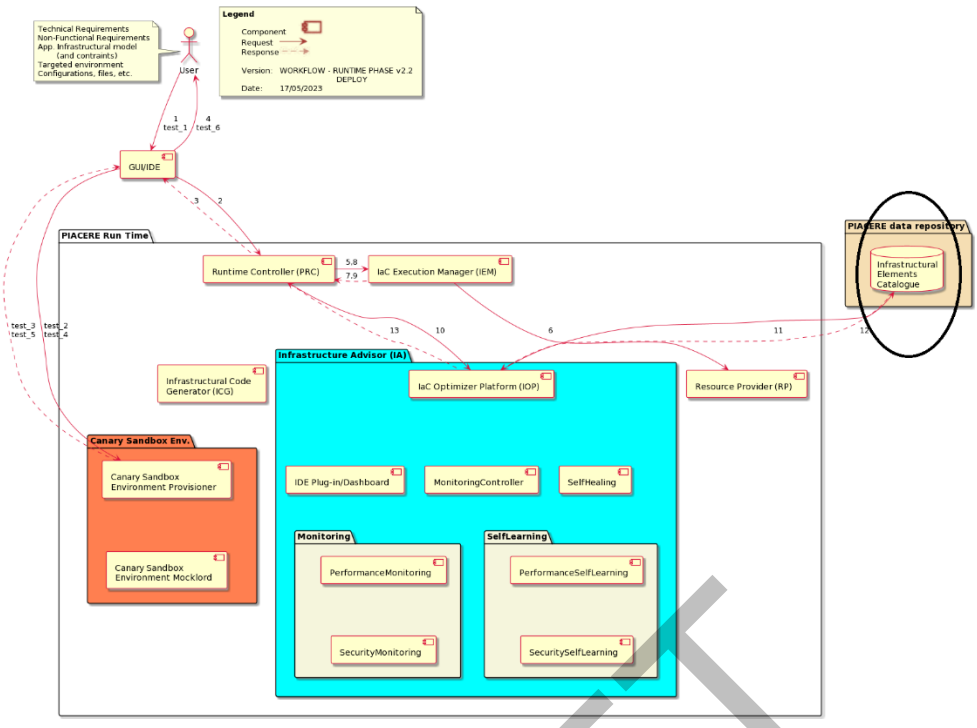

Figure 30: PIACERE Run Time Diagram (v2.2) with the Catalogue interactions.

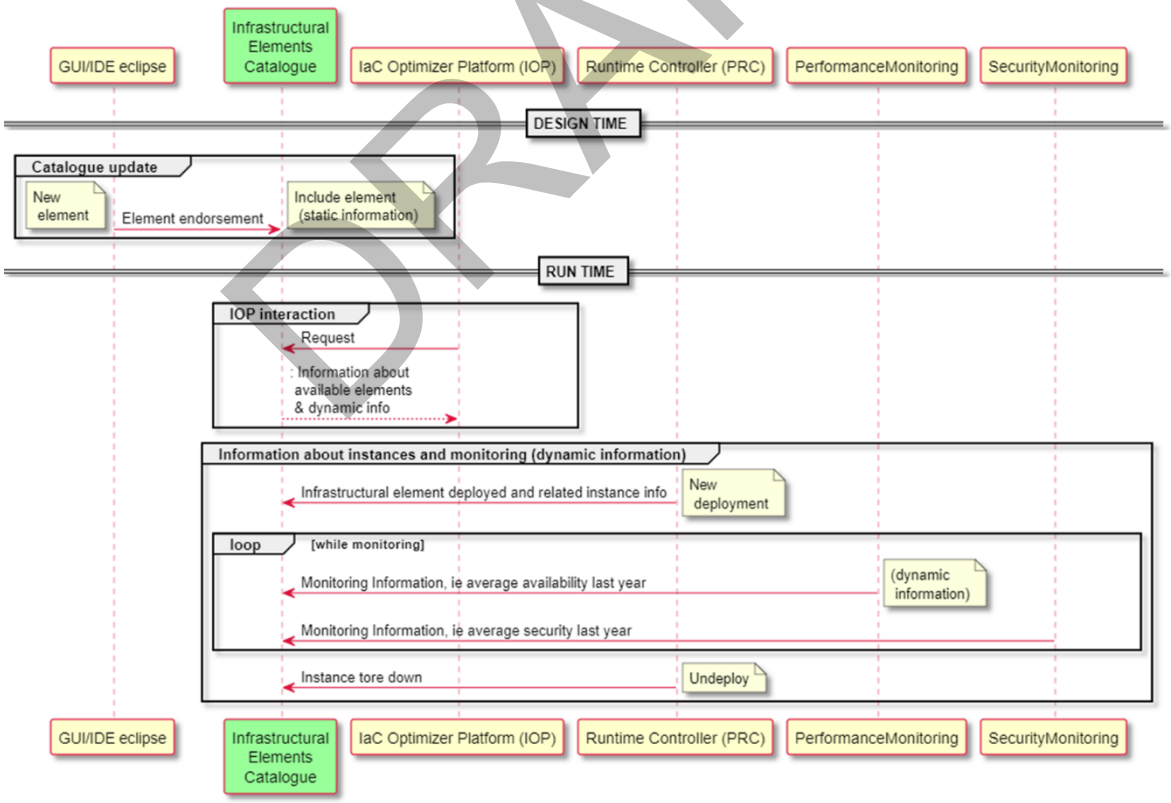

Figure 31: Sequence Diagram of the Infrastructural Elements Catalogue

1.2 Technical description

This subsection describes the technical specification of this third prototype. First, the main architecture of the prototype is shown and described, along with the description of all components. This subsection finishes with the technical specifications of the developed system.

1.2.1 Architecture and component's description

IEC architecture is based on a microservices style, hence it is divided in a front-end and a backend, so that it´s easier to scale and survive infrastructure issues. The components are described in the following figure, and main purposes are detailed below.

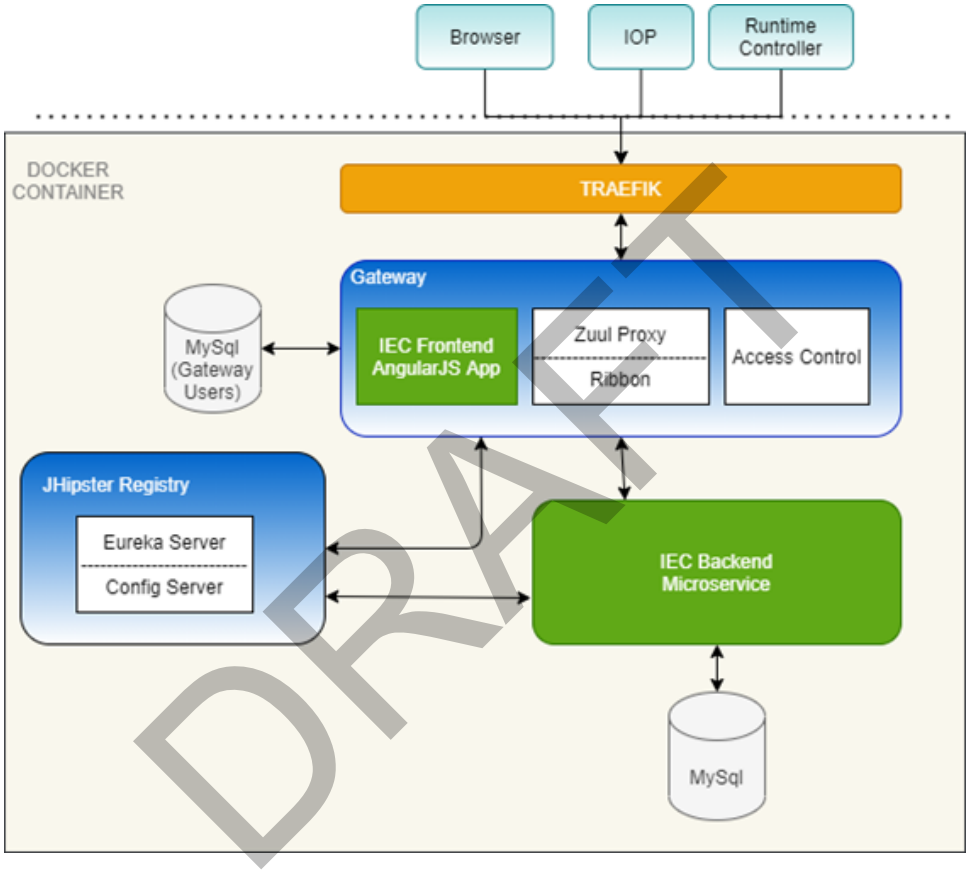

Figure 32: Main architecture of the Infrastructural Elements Catalogue

The IEC is composed by two principal components, which main purposes are briefly described as follows (for further details, see the section *Package Information* in *Appendix 1.3 Delivery and usage*).

- `IEC-Frontend`: this component is the entry point of the IEC system. All the external callings to the whole IEC are made through it. There are two main uses of this component, the frontend for managing the service Catalogue and the API exposed services to interact with other components.
- `IEC-Backend`: this component cannot be accessed externally. It manages all the logic of the IEC and exposes the rest API services to the other components.

1.2.2 Technical specifications

The technical specification of the prototype can be summarized in the following points:

- This prototype has been developed using **JHipster Framework** [20]. The framework provides all the needed for a modern web application and microservice architecture.
- It uses **Spring boot** for application configuration.
- In the client side, IEC-Frontend gateway uses **Yeoman**, **Webpack**, **Angular** and **Bootstrap** technologies.
- In the server side, IEC-Backend microservice, uses Maven, Spring, Spring MVC Rest, Spring Data JPA and Netflix OSS.
- For access control the JSON Web Token (**JWT**) mechanism is used. This is a stateless security mechanism which uses a secure token holding the user´s login name and authorities.
- Data persistence is implemented in a **MySQL** database.
- The **JHipster Registry** offers service discovery using Netflix Eureka.

1.3 Delivery and usage

1.3.1 IEC-Backend

The main structure of the prototype developed in this first stage of the project is composed by the packages shown in the following Figure 33.

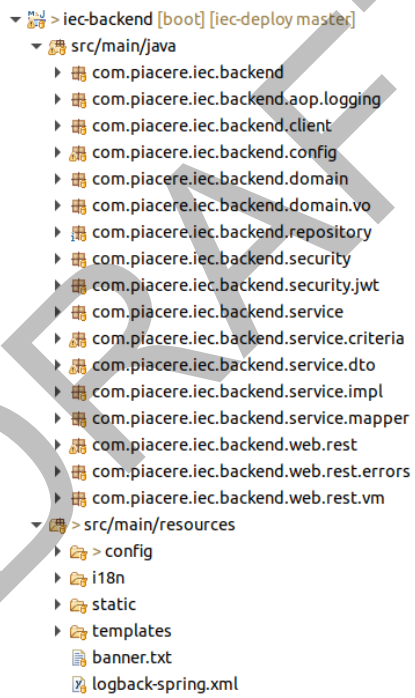

Figure 33: Structure of the catalogue

Each of these packages has its main objective and its context within the whole prototype. Furthermore, these packages are also comprised by several JAVA classes. With all this, the main purpose and composition of each component is as follows:

- `com.piacere.iec.backend.aop.logging`: this package is composed of `LoggingAspect.java` which define the aspect for logging execution of service and repository Spring components.
- `com.piacere.iec.backend.aop.client`: Composed by `UserFeignClientInterceptor.java` which implements `RequestInterceptor.java`. This class checks and add JWT token to the request header.

- `com.piacere.iec.backend.aop.config`: this package contains all classes related to configuration purposes.

Figure 34: structure of config package

- `com.piacere.iec.backend.aop.domain`: this package contains data model classes.

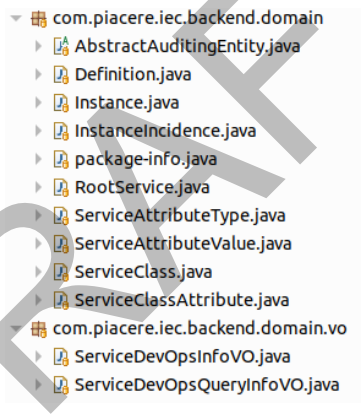

Figure 35: structure of domain package

- `com.piacere.iec.backend.repository`: this package contains Spring Data SQL repository classes.
- `com.piacere.iec.backend.security`: this package contains Spring Security related classes for security management.
- `com.piacere.iec.backend.service`: this package contains Iec-backend services for CRUD operations and other requirements needed.
- `com.piacere.iec.backend.web`: this package contains classes to expose Iec-backend rest end points.

1.3.2 IEC-Frontend

The main structure of the prototype developed in this first stage of the project is composed of JAVA classes related to Spring boot project and AngularJS files.

1.3.2.1 Spring boot package information

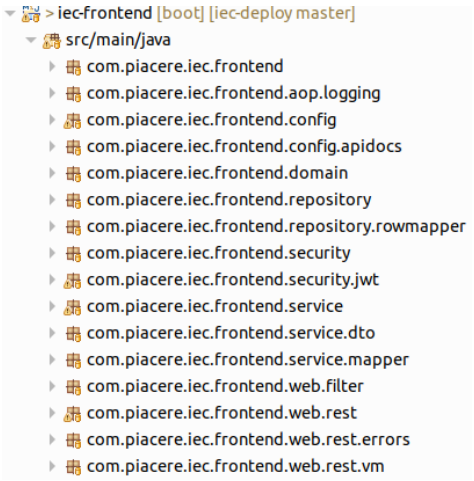

Figure 36: relation of packages

Each of these packages has its main objective and its context within the whole prototype. Furthermore, these packages are also comprised by several JAVA classes. With all this, the main purpose and composition of each component is as follows:

- `com.piacere.iec.frontend.aop.logging`: This package is composed by `LoggingAspect.java` which define the aspect for logging execution of service and repository Spring components.
- `com.piacere.iec.frontend.config`: This package contains all classes related to configuration purposes.
- `com.piacere.iec.frontend.domain`: This package contains user data model and Authority classes.
- `com.piacere.iec.frontend.repository`: This package contains Spring Data SQL repository classes for user and security management.
- `com.piacere.iec.frontend.security`: This package contains Spring Security related classes for security management.
- `com.piacere.iec.frontend.service`: This package contains Iec-frontend services for CRUD operations and other requirements needed for user and security management.
- `com.piacere.iec.frontend.web`: This package contains classes to expose Iec-frontend rest end points for user and security management.

1.3.2.2 AngularJS package information

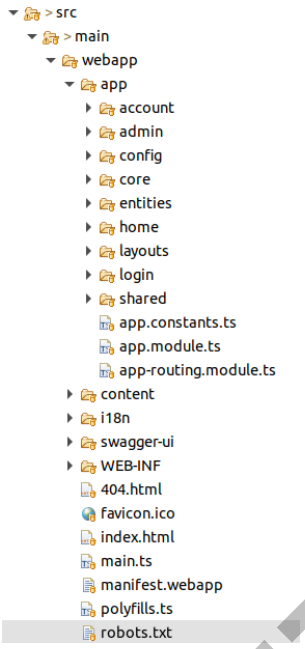

Figure 37: AngularJS package

Each of these packages and typescript files has its main objective and its context within the whole prototype.

The main purpose and composition of each file/component is as follows:

- `app/app.constants.ts`: Global application constants.
- `app/app.module.ts`: Declaration of all the needed modules, providers and components loaded in the web application.
- `app/app-routing.module.ts`: Routing configuration.
- `app/account`: Account management module. It contains components and services related to the user account management.
- `app/admin`: Admin related modules. Api documentation module, Gateway route module and user management module.
- `app/config`: Global configuration and constant typescript files.
- `app/core`: Global utils files, core models and services.
- `app/entities/iecBackend`: All files related to the iecBackend data model. Services, components and models.
- `app/home`: Home component
- `app/layout`: Layout components; error, footer, navbar, main and profile components.
- `app/login`: User Login component.
- `app/shared`: Shared module with common components, directives and pipes.
- `content`: Static files of webapp. Css and images.
- `i18n`: Internationalization files.

1.4 Installation instructions

This project is executed in a docker container. There are docker compose files for each environment development/production. To execute this project in a development environment:

1. Clone repository
   git clone https://git.code.tecnalia.com/piacere/public/the-platform/iop/iec-catalogue.git

2. Run docker compose to start Jhipster registry and MySQL instances

docker-compose --env-file .env.dev -f docker-compose-local-dev.yaml up --build -dFrontends

3. Build and deploy Catalogue backend
   ./mvnw -Pdev,api-docs -DskipTests

4. Build and deploy Catalogue frontend
   ./mvnw -Pdev,webapp,api-docs -DskipTests

Available services after initialization (User with admin role: admin / admin; User with non-admin role: user / user):

Once we successfully deploy the docker-compose, and supposing the following values for SERVER_HOST (192.168.56.1.nip.io) and HTTPS_PORT (8443) we will be able to access the services at:

- https://iec.192.168.56.1.nip.io:8443/ IEC for development
- https://iec.192.168.56.1.nip.io:8443/services/iecbackend/v3/api-docs IEC API

And to those generic services described at development-services:

- https://traefik.192.168.56.1.nip.io:8443/ Traefik dashboard
- https://traefik.192.168.56.1.nip.io:8443/api/http/routers Traefik API
- https://portainer.192.168.56.1.nip.io:8443/ to access Portainer
- https://ca.192.168.56.1.nip.io:8443/ Tecnalia CA

Apart from those, in case we are using the ".env.int" we will have access to additional endpoints

- https://jhipster-registry.192.168.56.1.nip.io:8443/ jhipster registry

1.5 User Manual

The catalogue is not intended to be used directly by the user. It is part of the optimization component and is used behind de scenes (mainly by the IOP) to calculate and provide its results. This use is made via the APIs provided by the IEC and is transparent for the final user.

However, the Catalogue provides a GUI (Graphical User Interface) that allows to check its contents and interact with it. Also, the integrated PIACERE IDE offers a way of show the content of the catalogue in a specific view.

1.5.1 Using the catalogue via webpage

The home page of the Catalogue shows a table of the number of services contained, organized by providers (files) and service types (rows):

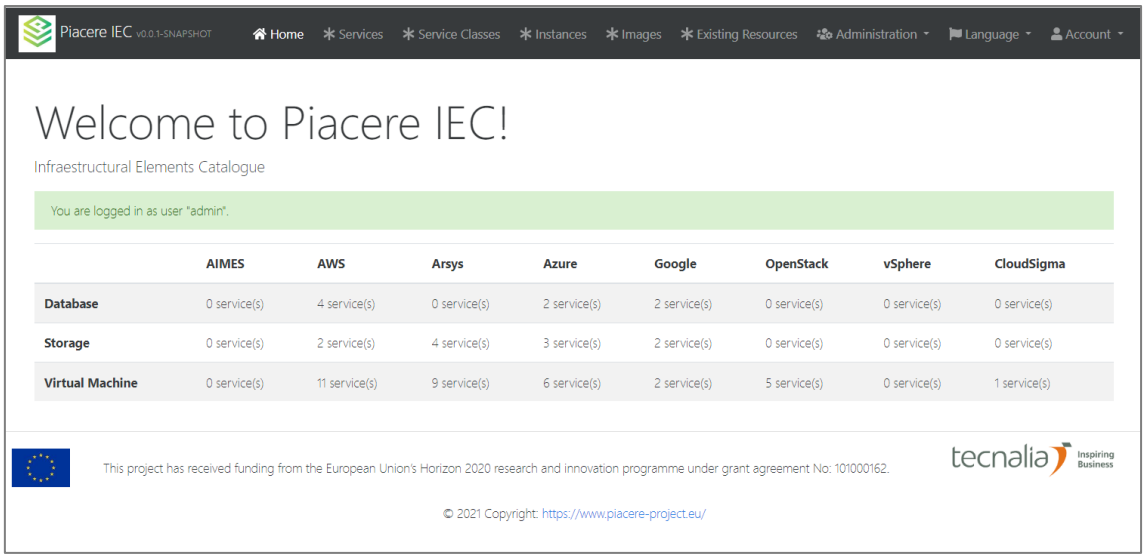

Figure 38: Welcome page of IEC.

The Service Classes option lists the types of services contained in the Catalogue:

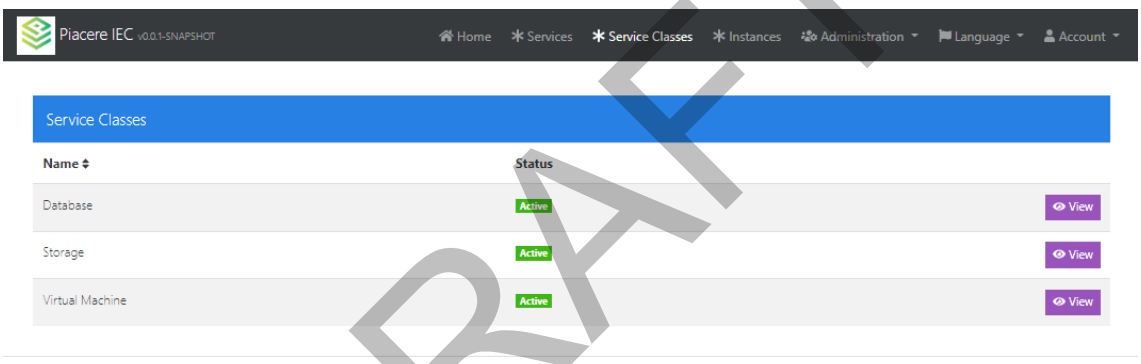

Figure 39: Service Classes page of IEC.

Clicking the "View" button shows the attributes of the selected type of service.

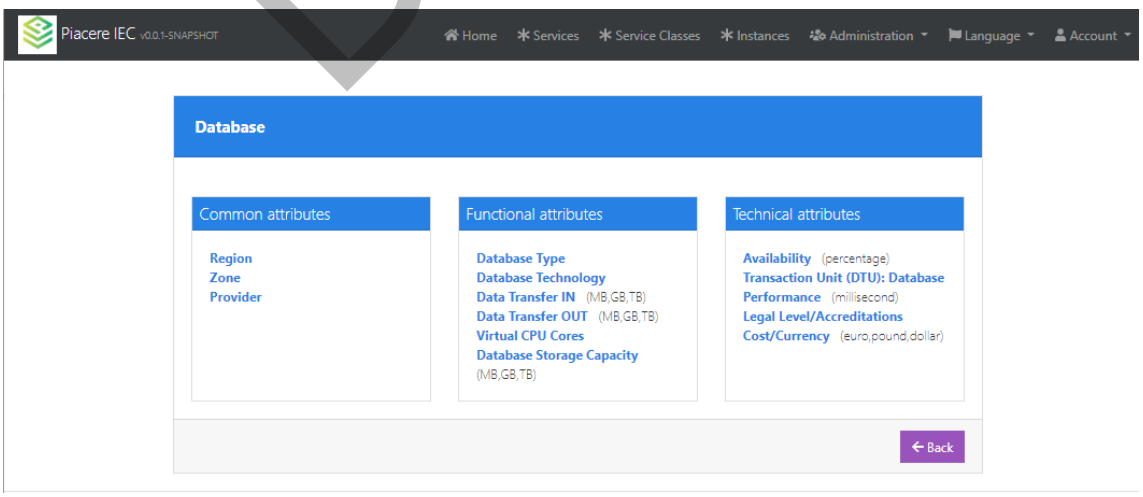

Figure 40: View Service Classes page of the IEC.

The “Services” menu option list all the services in the catalogue. A search box allows to filter by the name of the service.

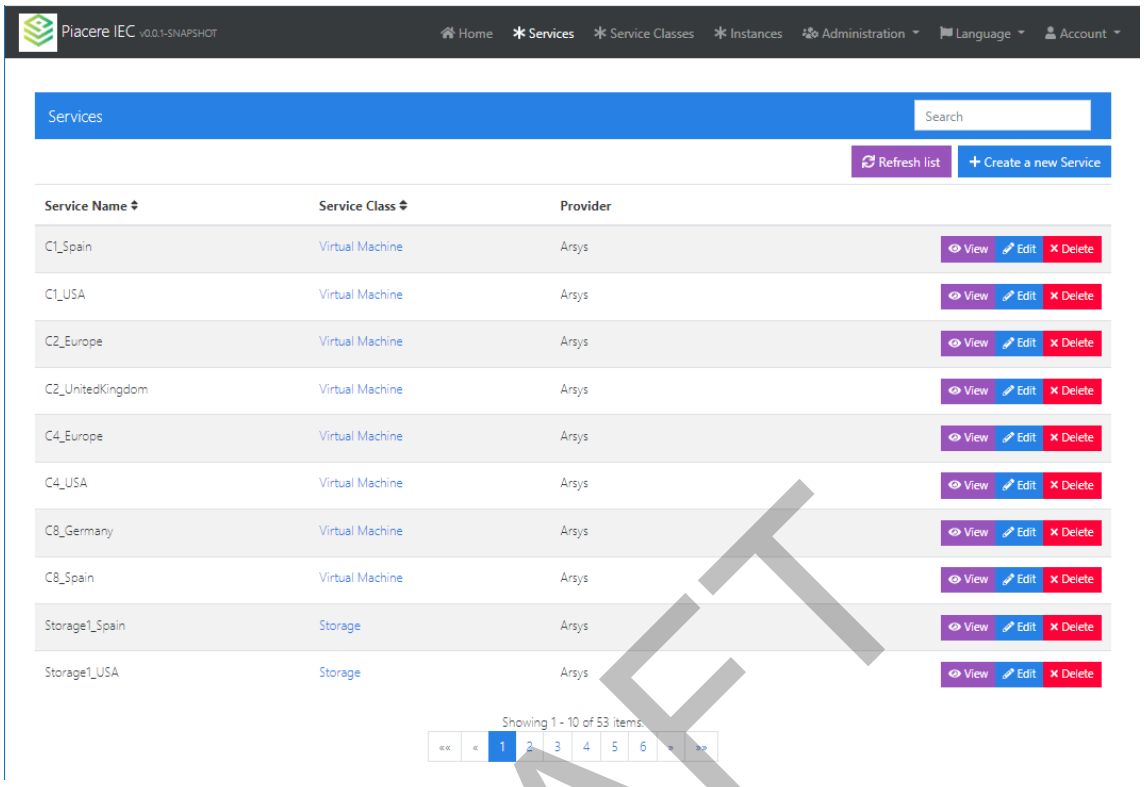

Figure 41: Services page of IEC.

A click on the “View” button of one of the services shows its properties:

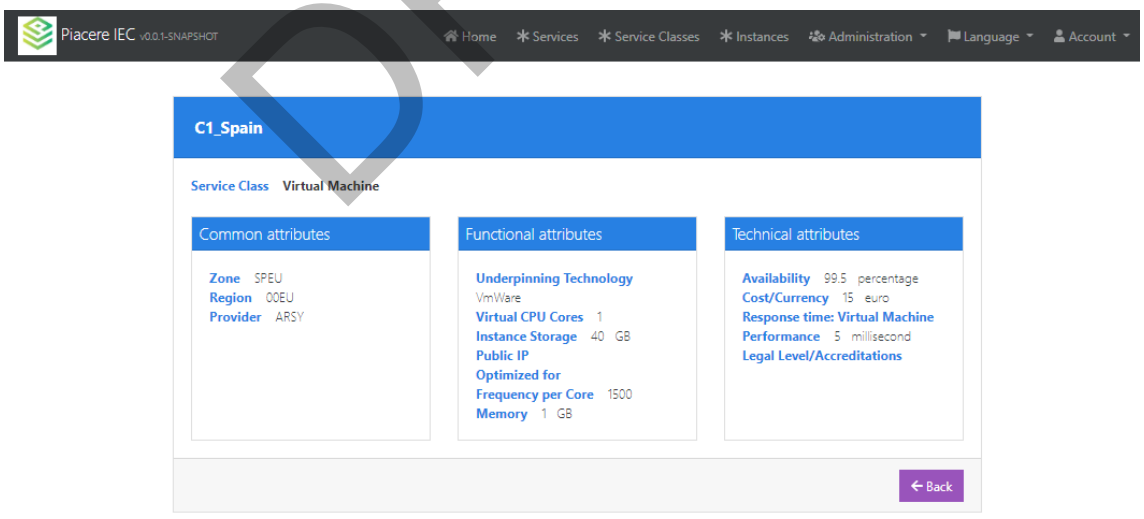

Figure 42: View Service page of IEC.

While a click on the “Edit” button of the service allows to edit its properties:

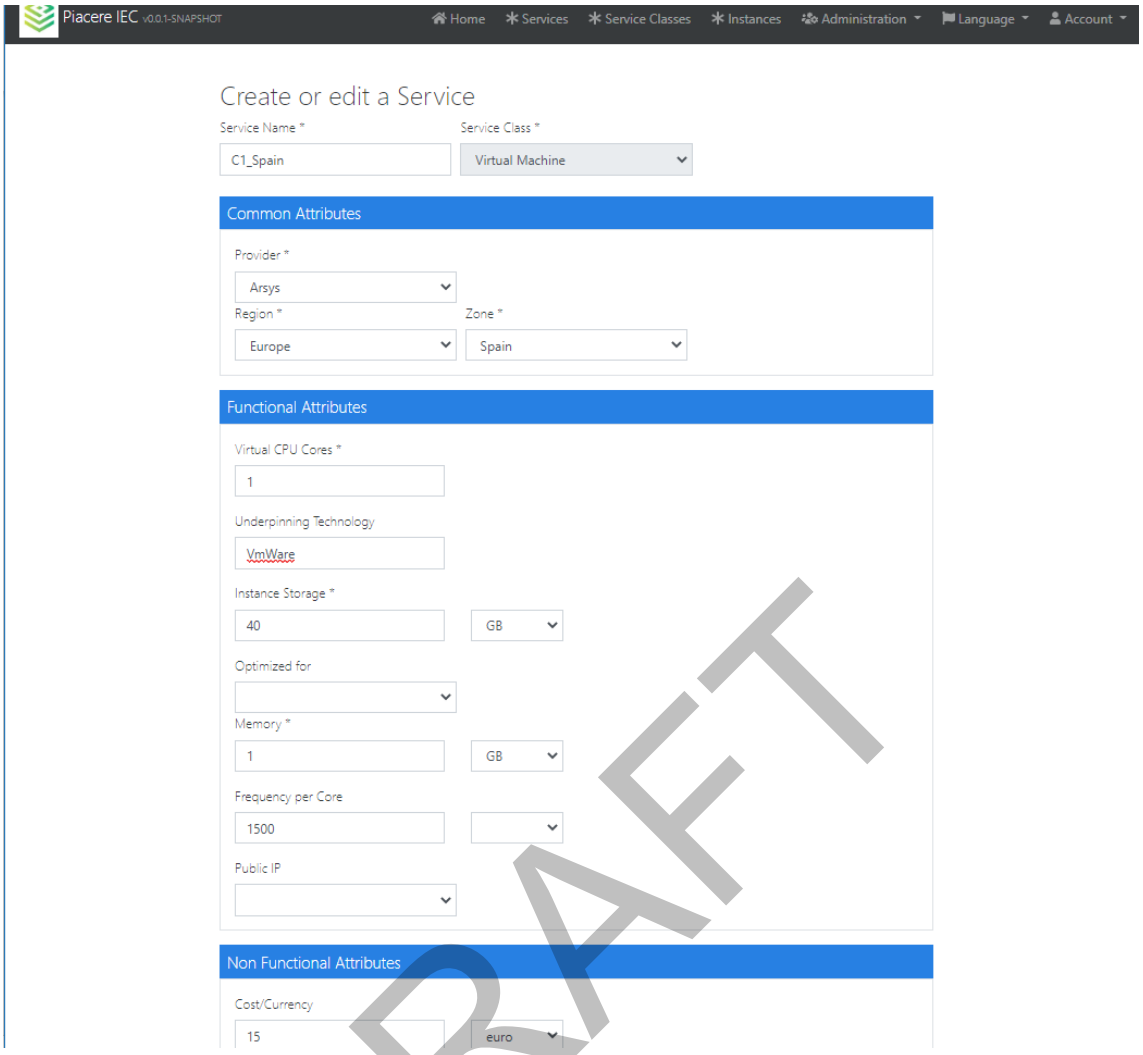

Figure 43: Edit Service page of IEC.

The “Instances” menu option shows a list of instances of the deployed services:

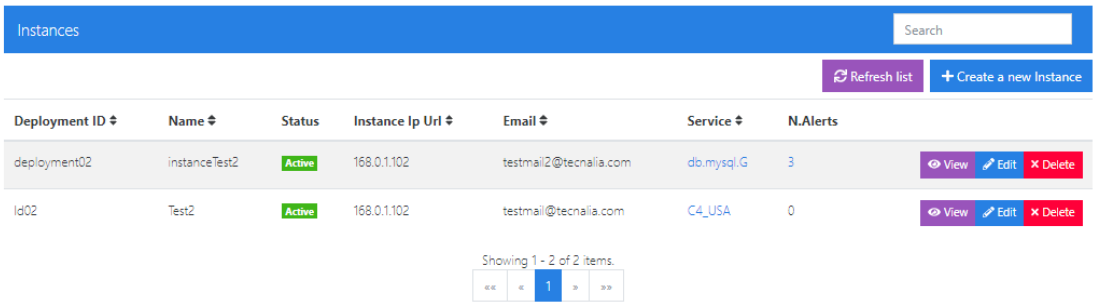

Figure 44: instances page of IEC.

The “Images” menu option shows a list of images defined in the system:

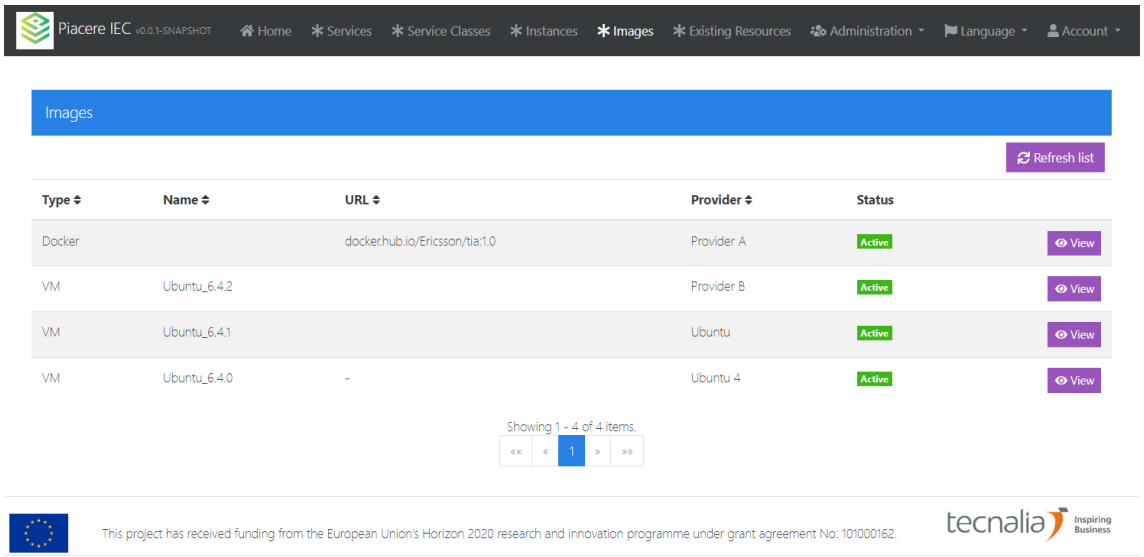

Figure 45: Images page of IEC.

There are two different types of images:

- the Docker images, whose main attribute is a URL from where it can be downloaded (e.g *docker.hub.io/Ericsson/tia:1.0*)
- The Virtual Machine (VM) images, whose name defines the VM (e.g. Ubuntu_6.4.1)

The "Existing Resources" menu option shows a list of the pre-defined resources in the deploy environment. A button there allows the user to create a new Existing Resource.

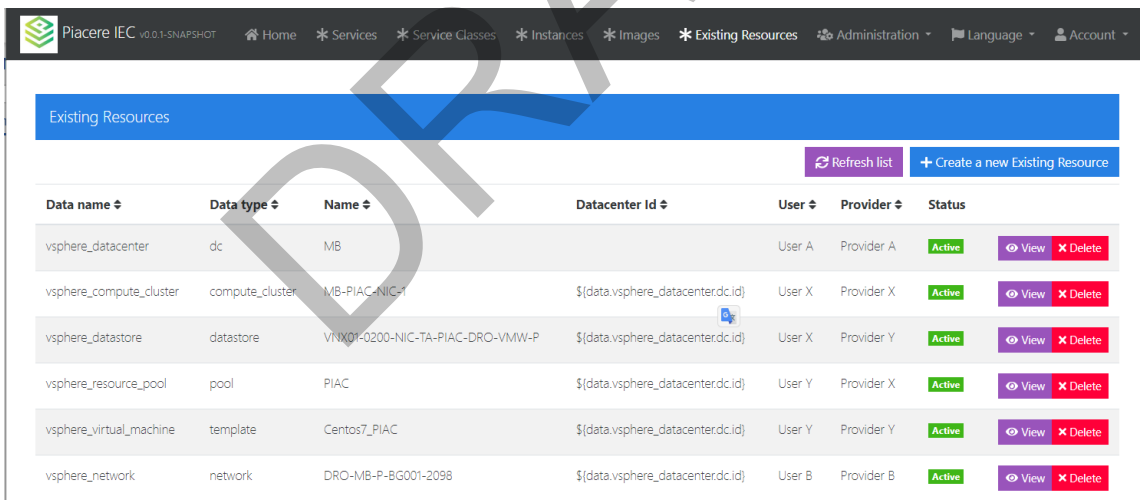

Figure 46: Existing Resources page of IEC

A click on the "View" button of one of the items shows its properties:

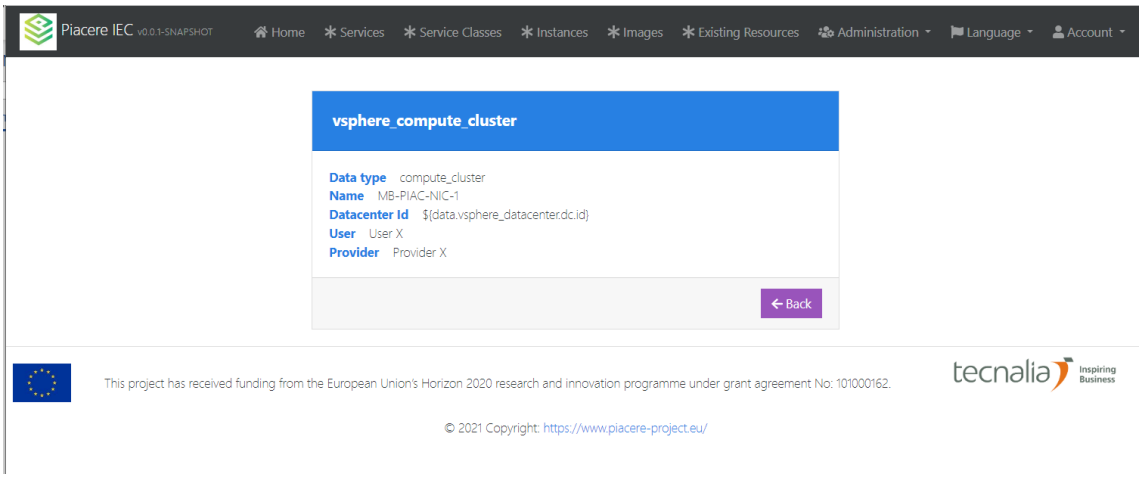

Figure 47: Existing Resources details page

1.5.2 Using the Catalogue from the IDE

Prior to using the Catalogue itself, you need to open the right view.

1. We use the "Window" toolbar menu, click on "Show view" option and select the "PIACERE Infrastructure Elements Catalogue" option.

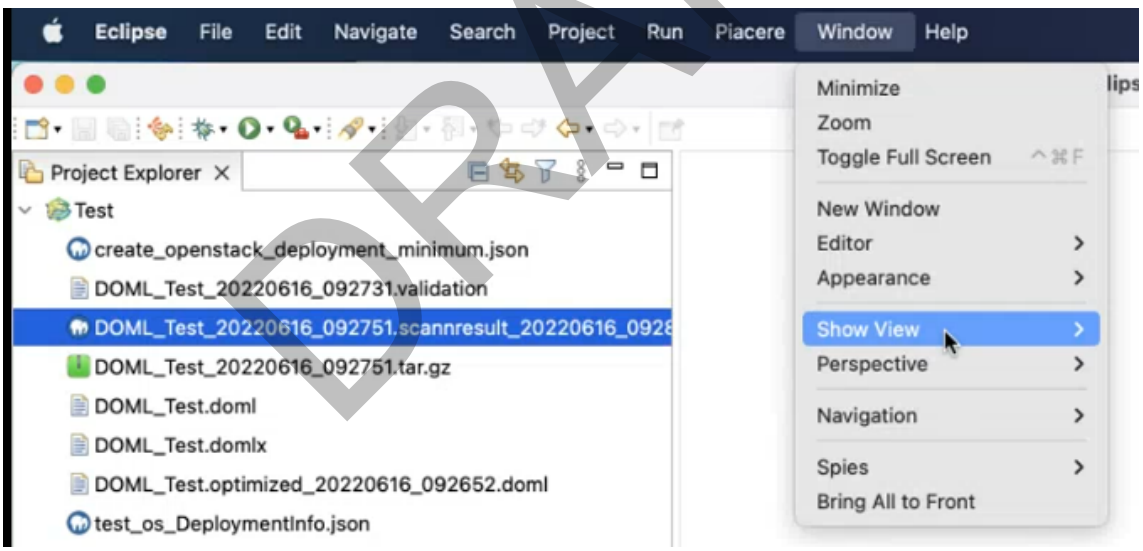

Figure 48: Show view option in the IDE.

2. Once in the view, the different types of elements of the Catalogue are shown in a list.

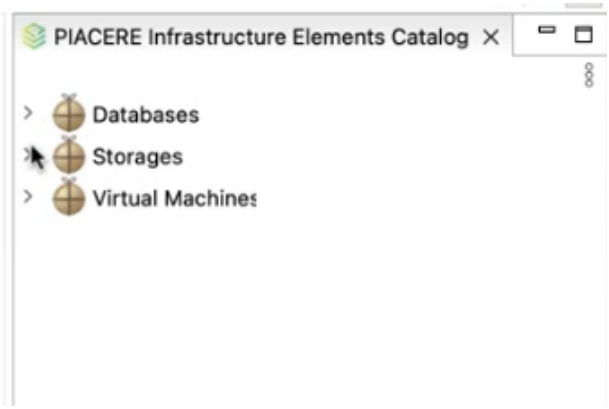

Figure 49: Catalogue View.

3. Clicking in any of these types, all the services of the type are listed (a further click in any of the services shown its properties and values).

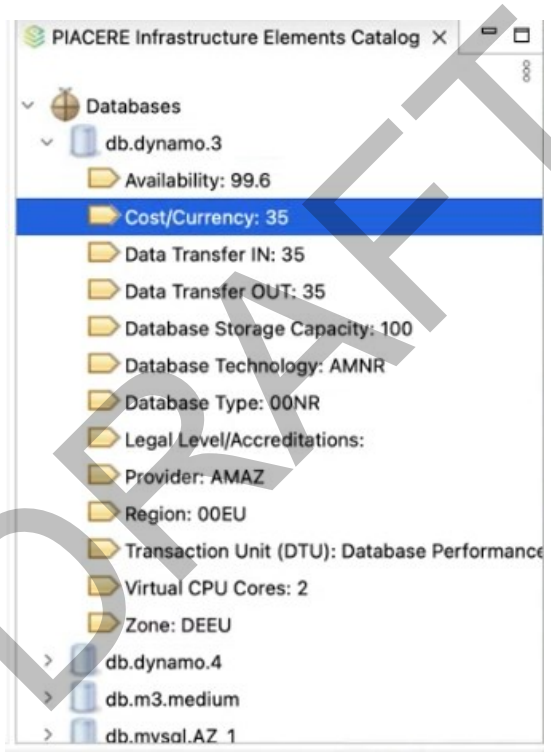

Figure 50: List of services of Databases type.

1.6 Licensing information

This component is offered under Apache 2.0 license. More detailed information can be found in the GitLab repository.

1.7 Download

The code is available at the public GitLab repository of the PIACERE project:

https://git.code.tecnalia.com/piacere/public/the-platform/iop/iec-catalogue

2 IOP Optimizer Implementation

In this appendix, we describe the most important aspects regarding the implementation of the IOP. It should be considered at this point that this information has experienced few modifications in the last six months. For this reason, the content is almost the same as in D5.8. In any case, and for the sake of completeness, we have decided to include it in this last deliverable.

2.1 Fitting into overall PIACERE Architecture

Regarding the fitting of the IOP Optimizer in the PIACERE architecture, as described in D5.8, the component is placed both in Run Time (Figure 51) and Design Time (Figure 52) phases. Having said that, the role of the IOP in both Run Time and Design Phase can be summarized as follows:

- *Role of the IOP in the Run Time phase*: in this phase, the IOP is involved in the seal-healing process when needed asking for a re-deployment, and it is called by the PIACER Runtime Controller.
- *Role of the IOP in the Design Time Phase*: in this case, the IDE can call the IOP as an optional step (user choice), after the DOML Model Checker interaction. Thus, the IOP will return the optimization to the IDE, and the user can accept or not the proposed optimization.

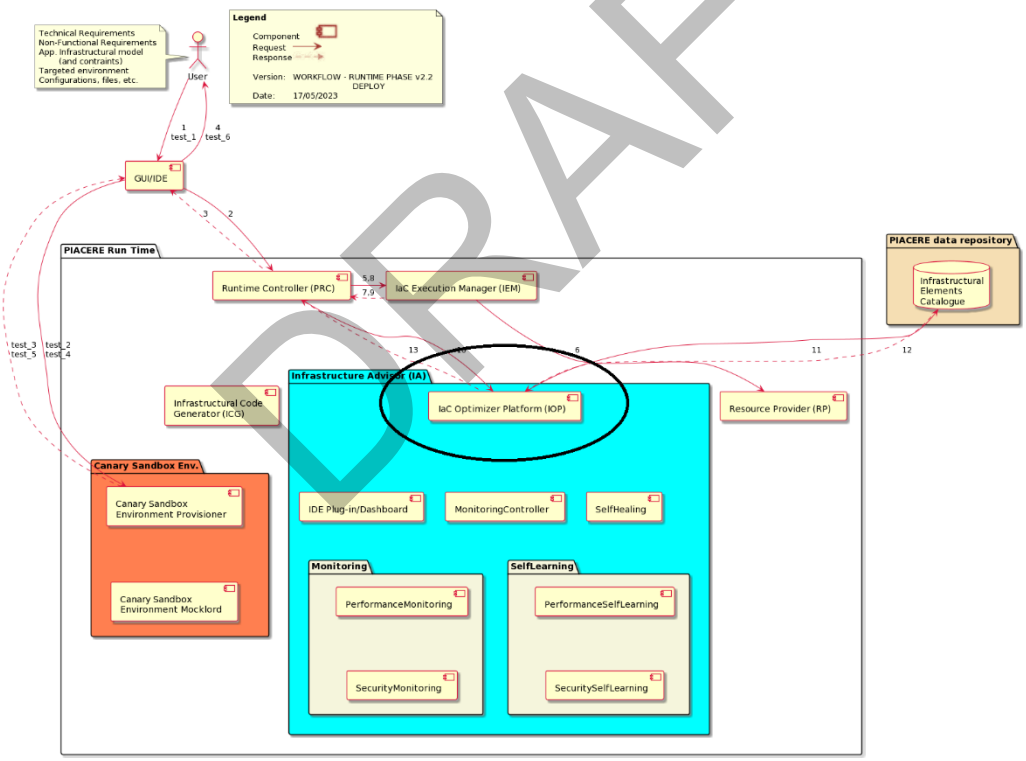

Figure 51: PIACERE Run Time Diagram on its 2.2 version

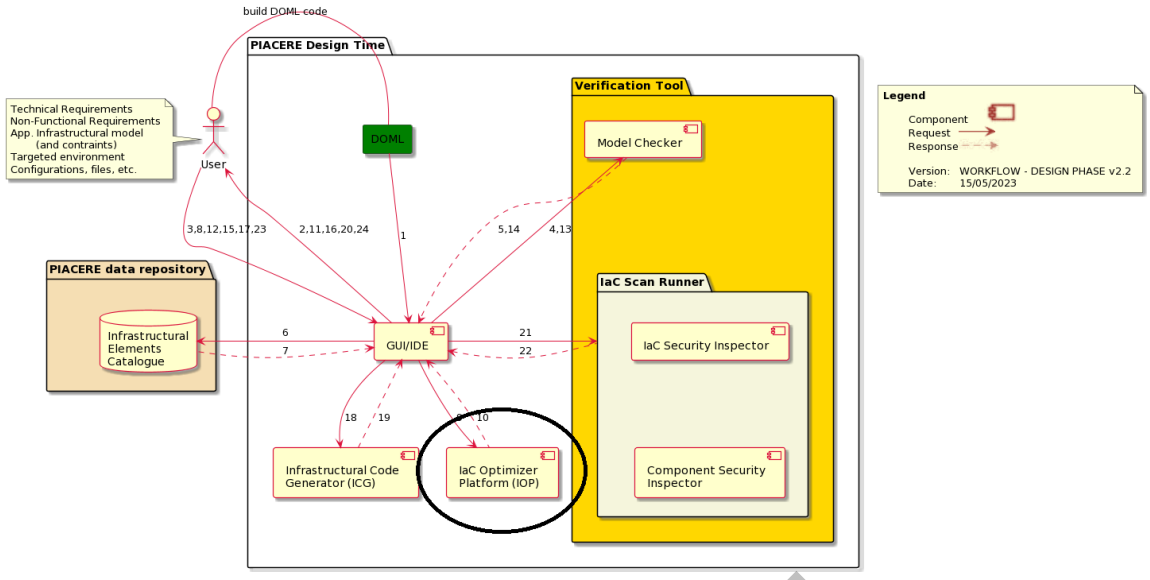

Figure 52: PIACERE Design Time Diagram on its 2.2 version

2.2 Technical description

This subsection is devoted to describing the technical specification of this third version of the IOP Optimizer. First, the main composition of the component is shown and described in Section 2.2.1. *Prototype architecture and components description*. After that, the technical specifications of the developed system are described in Section 2.2.2. *Technical Specifications*

2.2.1 Prototype architecture and components description

First, we depict in Figure 53 the main composition of the IOP Optimizer tool.

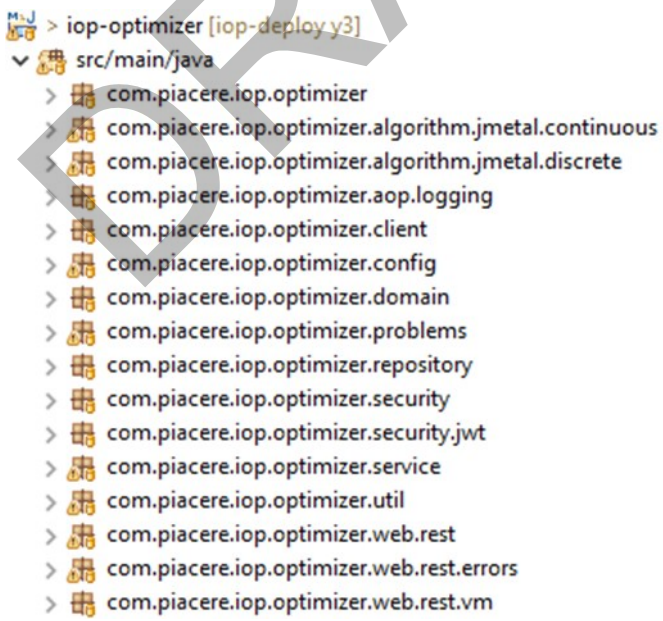

Figure 53: Composition of the IOP Optimizer

Thus, the main components of the Optimizer can be summarized as follows. It should be noted that further detail of this components is provided in the upcoming Section 4.2 *Package information.*

- `Optimizer Service`: the optimizer service is the entry point of the system, and the class that is called by external components. In a nutshell, the optimizer service gathers the input data (provided in DOML language), builds the optimization problem, and initialize both the solving algorithm algorithms and the data reader procedure. Classes devoted to this duty are placed `com.piacere.iop.optimizer.service`.
- `Data Reader`: this component has the main objective of gathering the data from the IEC and process them for being used as optimization variables by the solver. This component conducts some data-processing procedures such as data cleaning, normalization, discretization or integration. Classes devoted to this duty are placed in `com.piacere.iop.optimizer.service`.
- `Solving algorithms`: This is the component in which the solving algorithms are comprised. Despite NSGA-II and NSGA-III methods have been chosen for being the solving techniques used in the IOP, the rest of the testing algorithms have been maintained for further testing purposes. Classes devoted to this duty are placed in `com.piacere.iop.optimizer.algorithm.jmetal.continuous` and `com.piacere.iop.optimizer.algorithm.jmetal.discrete`.
- `Optimization problem building`: This component is dedicated to build the optimization problem for being subsequently solved by the artificial intelligence algorithms. It should be considered that the user preferences demand to the IOP the flexibility of generating optimization problems of different nature (single-objective, multi-objective, considering multi-element optimization…). Classes devoted to this duty are placed in `com.piacere.iop.optimizer.problem`.
- `Solution Processing Mechanism`: the purpose of this component is to obtain the outcomes reached by the solving algorithm and return them to the user in an understandable and ranked form. This component is also in charge of traducing the solutions to DOML format. Classes devoted to this duty are placed in `com.piacere.iop.optimizer.util`.
- `Web Service Component`: the last component is charge of preparing the whole tool for being accessible by external components and tools of the PIACERE project. Classes devoted to this duty are placed in `com.piacere.iop.optimizer`, `com.piacere.iop.optimizer.aop.logging`,`com.piacere.iop.optimizer.client`,`com.piacere.iop.optimizer.config`,`com.piacere.iop.optimizer.domain`,`com.piacere.iop.optimizer.repository`, `com.piacere.iop.optimizer.security`,`com.piacere.iop.optimizer.security.jwt`,`com.piacere.iop.optimizer.web.rest`,`com.piacere.iop.optimizer.web.rest.errors` and `com.piacere.iop.optimizer`. More specifically, the classes devoted to trigger the alerts described in Section 2.1 are placed in `com.piacere.iop.optimizer.web`.

2.2.2 Technical specifications

The IOP Optimizer has been developed using JAVA, which is a high-level, general-purpose, class-based and object-oriented programming language. JAVA was conceived for having few dependencies, in order to be more efficient than other similar languages. Also, JAVA embraces the philosophy known as WORA, which is *Write Once and Run Anywhere*. This philosophy implies that the compiled JAVA code can be run in all platforms that support JAVA without requiring any additional recompilation.

Additionally, jMetal framework, crucial for the optimization algorithms, is imported using MAVEN mechanism, which is a is a software project management and comprehension tool. MAVEN is based on a concept named as Project Object Model (POM). Thus, MAVEN can

manage a build, reporting and documentation in a project from a unique file of information. Regarding the versions, jMetal 5.1 has been used, and 11 JDK of JAVA.

Additionally, following the service-oriented architecture (SOA) approach that is shared among the different pieces of the PIACERE Framework, the logic of the IOP has been packaged as a service using Jhipster Framework. The Jhipster Framework is a free and open-source application generator used to develop web applications and Microservices using Angular and the Spring Framework. Jhipster has many goodness, that drove us to its usage. Some of the key ones were:

- The implementation of REST APIs
- The support of the Open API Specification (OAS)
- The inclusion of different security mechanism
- The horizontal scale capabilities

The Open API specification was quite important as it provides a standard way to describe the REST API. This issue allows us to quickly generate clients in different languages that facilitates the integration of other components inside and outside the PIACERE framework.

Finally, like other components in the PIACERE framework, the component is packaged using the docker technology and its integration in the piacere framework has been specified using the docker-compose choreography mechanisms.

It is worth mentioning that the Docker file used for the containerization of the IOP follows a multistage approach that covers both the compilation and packaging of the source code in a *jar*, and the running of that jar to provide the expected service. This is very beneficial to ensure the homogeneity of the behavior of the service provided by the resulting image.

2.3 Optimizer Delivery and usage

As in the previous annex related with the IOP Optimizer, the delivery and usage of this tool has remained the same in the last six months. In any case, and because this is the last version of this deliverable, we have decided to include this context as annex for the sake of completeness.

2.3.1 Package information

We have depicted in the previous Figure 52 the structure of the IOP Optimizer. For analysing this structure, we divide the explanation into two thematic groups: the first one dedicated with the packages oriented to the IOP optimization problem solving (Section 4.1.1), and the other one dedicated to those packages created for making the whole IOP accessible for external components (Section 4.1.2)

2.3.1.1 Packages related with the optimization problem solving

The main role of the packages that fall within this category is to solve the PIACERE optimization problem described previously. Probably, the most representative examples are the packages coined as `com.piacere.iop.optimizer.algorithm.jmetal.continuous` and `com.piacere.iop.optimizer.algorithm.jmetal.discrete`. We depict in the next Figure 54 the composition of these both packages.

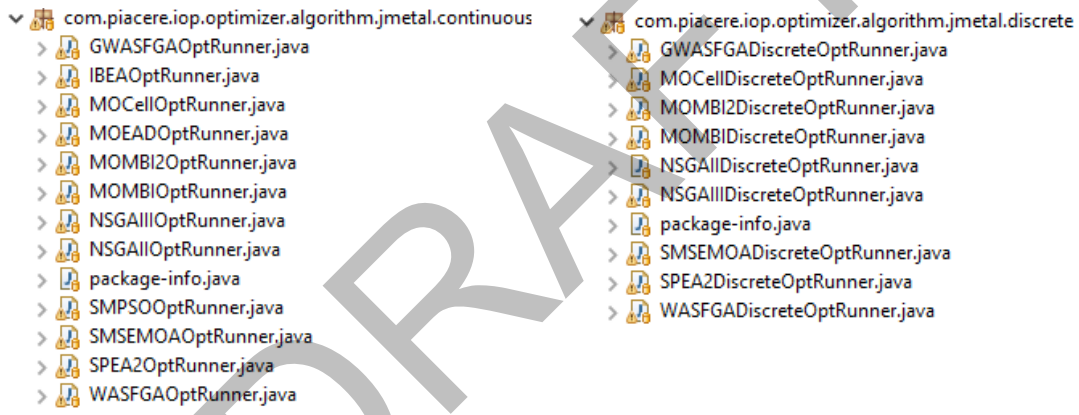

Figure 54: Composition of the packages that contain the jMetal algorithm adaptations

Essentially, these packages contain the jMetal implementations of the algorithms. In this regard, it should be taken into account that the algorithms that have been used for the tool, and the ones employed for solving the PIACERE use cases are the `NSGAIIDiscreteOptRunner.java`, and `NSGAIIIDiscreteOptRunner.java` in their discrete variants. In any case, the rest of the algorithms have been left on their corresponding packages for future additional performance testing purposes.

Additionally, the package `com.piacere.iop.optimizer.problems` contains the classes that define the optimization problem itself. In this package, the four classes regarding the jMetal adaptation of the problem have been considered.

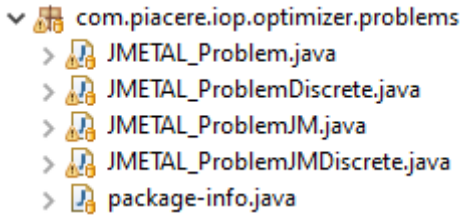

Figure 55: composition of the `com.piacere.iop.optimizer.problems` package

Regarding the output of the results, the class in charge of this crucial aspect is the one coined as `IOPSolutionLustOutput.java`. As shown in Figure 56, `IOPSolutionListOutput.java` is contained in a `com.piacere.iop.optimizer.util` package.

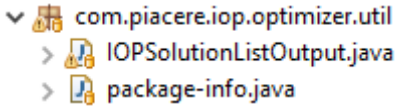

Figure 56: structure of the `com.piacere.iop.optimizer.util` package

Finally, the last of the packages dedicated to the optimization problem solving, is the one coined as `com.piacere.iop.optimizer.service`. This package contains three different classes of a crucial importance for the correct working of the IOP component. In Figure 57 we depict the composition of this package:

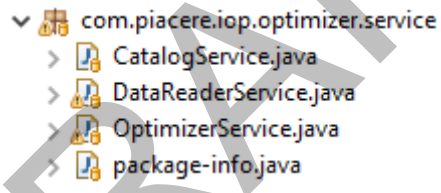

Figure 57:Composition of the package `com.piacere.iop.optimizer.service`

Going deeper on this package, each of the classes that comprise it have the following role within the whole IOP Optimizer:

- `CatalogService`: This class is specifically dedicated to access the IEC for obtaining all the elements and their features for being used for the optimization procedure. Thus, this class deals with all the security aspects needed for acceding the IEC thought the JAVA code.
- `DataReaderService`: Once the data is obtained in its raw form, this `DataReaderService` class is in charge of reading all the information and transform it in optimization variables. This class has also some filtering methods, in order to properly contemplate optimization requirements related, for example, with the region or the provider.
- `OptimizerService`: this class is, probably, the most important one of the whole IOP Optimizer. `OptimizerService.java` is the main class, which orchestrates all the optimization process. This is the entry point of the IOP for both of its execution ways: from the JAVA code or as external service, and it is in charge of receiving the DOML inserted as input and staring the complete pipeline of execution of the IOP. After all the optimization is executed, this class is also the one that returns to the user the main outcome provided by the internal class `IOPSolutionLustOutput.java`.

2.3.1.2 Packages dedicated to make the whole IOP accessible

The packages dedicated to providing the IOP functionality are, in most of cases, generated using Jhipster. In Figure 58 we depict the packages of the IOP, most of them, except those indicated with (1), have been automatically generated based on the initial configuration of the Jhipster. Besides, most of the customization for the service provision is implemented in (2), where the request is received and the jMetal packages are used.

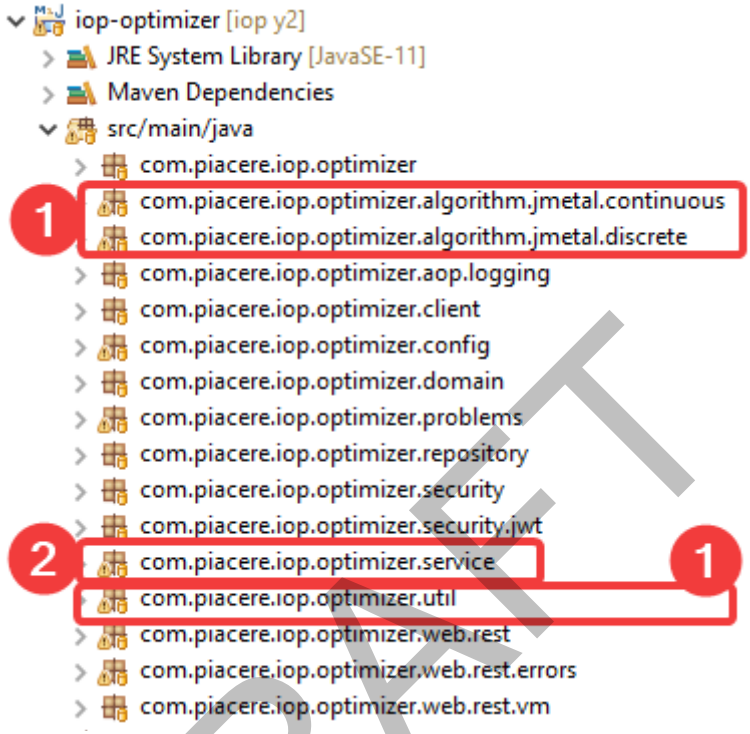

Figure 58: packages generated by jhipster.

Besides, there are some other packages manipulated in other to declare, expose and characterize the IOP services. These are `com.piacere.iop.optimizer.web.rest` and `com.piacere.iop.optimizer.security`. In the first one, we declare the endpoint and the data received and the response sent. The second one the package that allow us to manage the access security together with the YAML configuration of the service. The configuration allows us to define multiple mechanism to specify who can or who cannot access the service. i.e. json web token (JWT), Open ID, etc.

Jhipster users yeoman to generate the microservice elements the configuration used to the generation is stored in .yo-rc.json. This is useful in case we want to regenerate the microservice from scratch or update to a newer version of Jhipster that will help us to keep our microservice with the latest libraries that include the latest security patches.

2.4 Installation instructions

The installation instructions for this third version of the tool are similar to the ones provided for the first version of the IOP Optimizer. The whole tool is built in a compressed folder, which can be imported by any JAVA development framework such as NetBeans of Eclipse. Furthermore, the project can also be imported as a Maven project, being this option even more comfortable than just importing the whole project folder. Related also with this last aspect, jMetal framework is imported to the project and used by the IOP using Maven functionality.

Additionally, the last version of the jMetal framework has been used for this third version of the IOP, which at the time of writing this deliverable, is version 5.1. Regarding the MOEA optimization framework, which was considered in the first version of the prototype, it has been finally discarded from the project as described in previous Section 5.3.3. for this reason, it is not necessary to include it in the `pom.xml` related to the project.

Finally, we depict the dependencies added to the `pom.xml` file for correctly importing jMetal framework to the JAVA project.

```
<dependency>
      <groupId>org.uma.jmetal</groupId>
      <artifactId>jmetal-core</artifactId>
      <version>5.1</version>
  </dependency>

  <dependency>
     <groupId>org.uma.jmetal</groupId>
     <artifactId>jmetal-problem</artifactId>
     <version>5.1</version>
  </dependency>

  <dependency>
      <groupId>org.uma.jmetal</groupId>
      <artifactId>jmetal-algorithm</artifactId>
      <version>5.1</version>
  </dependency>

  <dependency>
      <groupId>org.uma.jmetal</groupId>
      <artifactId>jmetal-exec</artifactId>
      <version>5.1</version>
  </dependency>
```

2.5 User Manual

Three different alternatives are available for properly testing the IOP Optimizer. The first one resorts to the execution of the JAVA project using a JAVA framework and a unitary test. The second alternative is related to using it through the Eclipse based IDE, developed in the context of PIACERE project. Finally, the IOP can also be run via webpage-based platform. These three methods are explained in this Section.

2.5.1 Running the IOP Optimizer via JAVA code

The entry point of the IOP Optimizer if the user wants to run it via JAVA code is the class named as `OptimizerServiceIT.java`, which is part of the package `com.piacere.iop.optimizer.service`, within the test branch of the code. We depict in the following figure an excerpt of the class `OptimizerServiceIT.java`. More concretely, we show the part of the code in which the IOP is called through the `optimizedService.run(call)` method.

```
@Test
void testDefaultOptimize() throws Exception {

    String call = "...";
    String result = optimizerService.run(call);
    System.out.println(result);
    assertThat("ok").isEqualTo("ok");

}
```

Figure 59: An excerpt of the OptimizerServiceIT.java class

As mentioned, the method that runs the IOP is the one coined as `optimizedService.run(call)`. This is the method that it is called by the PIACERE components to make the IOP run and to find the most optimized deployment configurations. Two different important aspects can be seen in this little excerpt of code:

- For properly run the Optimizer, a String should be introduced as input. As explained before, this input should a fully working DOML file, formatted as String, in which the `optimization opt` part should be correctly introduced. As mentioned, this `optimization opt` part must have the following format:

```
optimization opt {
    objectives {
    }
    nonfunctional_requirements {
    }
}
```

  In which objectives contains the `optimization` objectives (such as cost, performance or availability) and `nonfunctional_requirements` slot includes main optimization requirements (such as the maximum cost, the minimum performance or the elements to search).
- The output of the IOP is a String (represented as `result`), which is the same DOML introduced as input (`call` variable). In a nutshell, the output string introduces the main results as an adding in the `optimization opt` section of the input DOML, following this format:

```
solution solX {
    objectives {

    }
    Decisions[]
}
```

  It should be considered that one solution excerpt in introduced by each of the solutions found by the Optimizer. More concretely, this solution part includes an `objectives` part representing the main values for each optimization objective introduced in the input DOML, and a `Decisions` part, which represents the solution itself, that is, the elements chosen from the IEC and which built the deployment configuration.

2.5.2 Running the IOP Optimizer via PIACERE IDE

In addition to executing the code as mentioned in previous section, the IOP can also be used through the PIACERE IDE, developed in the context of this project. In this sense, the IOP

Optimizer is completely integrated with this tool. In the previous Section 2.1.4 - *Advances made regarding the IDE integration* we represent an example of how the IOP should be used in the PIACERE IDE.

2.5.3 Running the IOP Optimizer via OpenAPI

In case we want to execute the IOP optimizer by its OpenAPI specification we have several options, some of them are:

- We can get the OpenAPI specification and import in a testing tool such as Postman
- We can use the API explorer tool in IEC, https://iec.ci.piacere.digital.tecnalia.dev/admin/docs
- We can use any swagger explorer providing a proper CORS configuration.

First step is how to get the OpenAPI specification that describes the IOP service following and standardized syntax. In order to get the OpenAPI specification, we use the same component that as part of the basic features provided by Jhipster, so we can retrieve the Jhipster specification. Thus, to get the specification we can access an endpoint in the IOP, for example: https://iop.ci.piacere.digital.tecnalia.dev/v3/api-docs. This file can be used in any of the methods specified before. Figure 60 shows how the URL is used to import the IOP OpenAPI in Postman.

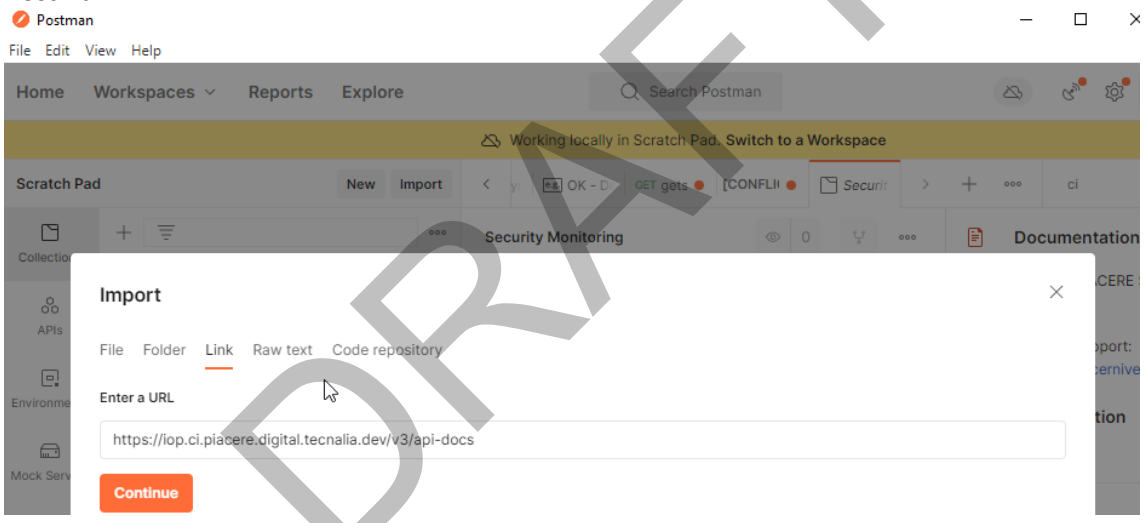

Figure 60: Postman link import

Figure 61 shows how the API looks like in postman after importing.

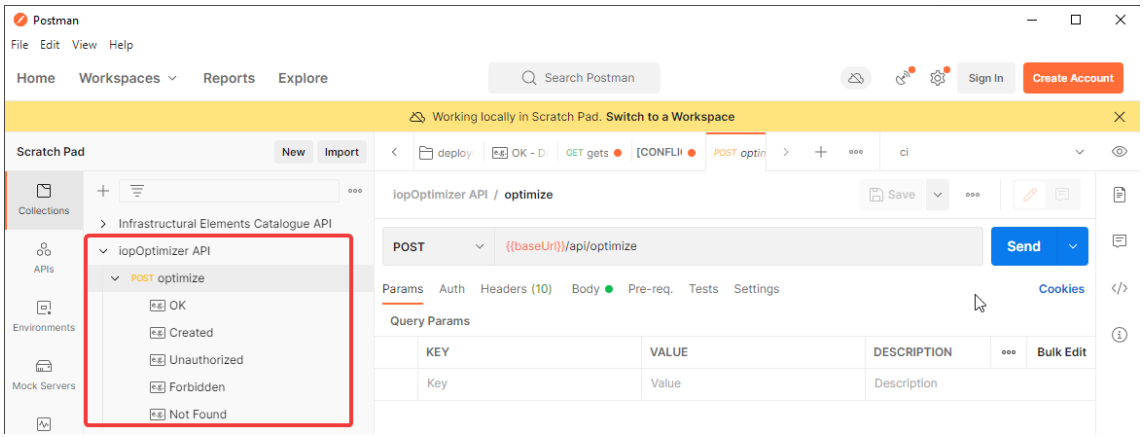

Figure 61: IOP Optimizer API after import

Finally, Figure 62 shows how the IOP API looks like in the IEC Administration API tool.

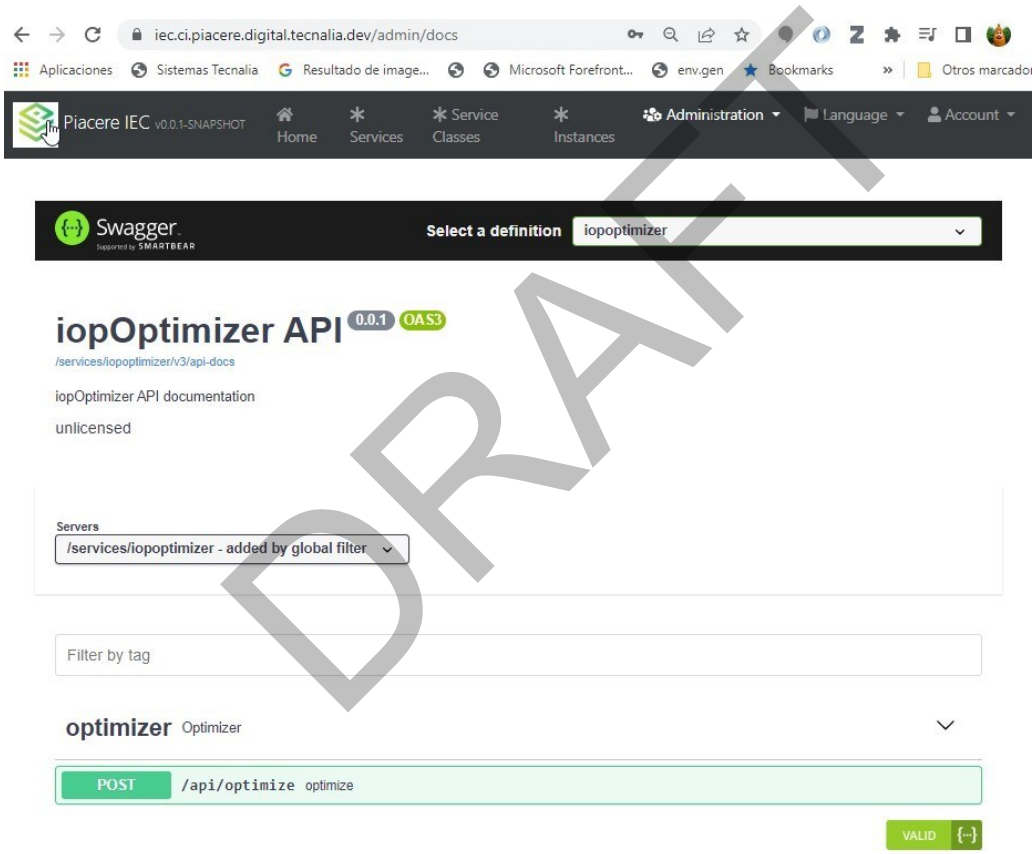

Figure 62: IOP Optimizer API in the Administration API tool in IEC.

2.6 Licensing information

This component is offered under Apache 2.0 license. Detailed information can be found in the GitLab repository.

2.7 Download

The code is available at the public GitLab repository of the PIACERE project:

https://git.code.tecnalia.com/piacere/public/the-platform/iop/optimizer

APPENDIX B: input DOML file for UC3

In this Appendix B, we provide the input DOML file for the preliminary version of the Ericsson use case. The main reason of placing the DOML files here is because of their length:

```
doml uc3_openstack
application app {
    software_component iwg {
        provides { net_info }
    }
    software_component osint {
        provides { osint_info }
        consumes { net_info, get_twitter, ewcf_rest_interface }
    }
    software_component ewcf {
        provides { ewcf_rest_interface }
        consumes { get_firebase }
    }
    saas external_twitter {
        provides { get_twitter @ "https://twitter_api/get" }
    }
    saas external_firebase {
        provides { get_firebase @ "https://firebase_api/get" }
    }
}
infrastructure infra {
    vm igw_vm {
        os "CentOS-7-2111"
        size "small-centos"

        iface igw_vm_oam {
            belongs_to subnet_oam_igw
        }

        iface igw_vm_net1 {
            belongs_to subnet_net1_igw
        }

        iface igw_vm_net2 {
            belongs_to subnet_net2_igw
        }

        credentials ssh_key
    }
    vm osint_vm {
        os "CentOS-7-2111"
        size "small-centos"

        iface osint_vm_oam {
            belongs_to subnet_oam_osint
        }

        iface osint_vm_net1 {
            belongs_to subnet_net1_osint
        }

        iface osint_vm_net3 {
            belongs_to subnet_net3_osint
        }
```

```
    credentials ssh_key
}
    vm ewcf_vm {
    os "CentOS-7-2111"
    size "small-centos"

    iface ewcf_vm_oam {
        belongs_to subnet_oam_ewcf
    }

    iface ewcf_vm_net1 {
        belongs_to subnet_net1_ewcf
    }

    iface ewcf_vm_net3 {
        belongs_to subnet_net3_ewcf
    }

    credentials ssh_key
}

container cont_tia {

    host osint_vm {

        container_port 80
        vm_port 8080
        iface osint_vm_oam

    }
}

container cont_tis {
    host osint_vm {
        container_port 80
        vm_port 8080
        iface osint_vm_oam
    }
}

container cont_twr {
    host osint_vm {
        container_port 81
        vm_port 8081
        iface osint_vm_oam
    }
}

container c1 {
    host igw_vm {
        container_port 82
        vm_port 8082
        iface igw_vm_oam
    }
}

container c2 {
    host igw_vm {
```

```
            container_port 83
            vm_port 8083
            iface igw_vm_oam
        }
    }

    container c3 {
        host igw_vm {
            container_port 84
            vm_port 8084
            iface igw_vm_oam
        }
    }

    container cont_mongodb {
        host ewcf_vm {
            container_port 85
            vm_port 8085
            iface ewcf_vm_oam
        }
    }

    container cont_ewcf {
        host ewcf_vm {
            container_port 86
            vm_port 8086
            iface ewcf_vm_oam
        }
    }

    net oam {
        protocol "TCP/IP"
        cidr "16.0.0.0/24"

        subnet subnet_oam_igw {
            protocol "TCP/IP"
            cidr "16.0.1.0/26"
        }

        subnet subnet_oam_osint {
            protocol "TCP/IP"
            cidr "16.0.1.64/26"
        }

        subnet subnet_oam_ewcf {
            protocol "TCP/IP"
            cidr "16.0.1.128/26"
        }
    }

    net net1 {
        protocol "TCP/IP"
        cidr "16.0.1.0/24"

        // Subnets definition
        subnet subnet_net1_igw {
            connections {
                subnet_net1_osint
            }
```

```
        protocol "TCP/IP"
        cidr "16.0.1.0/25"
    }

    subnet subnet_net1_osint {
        connections {
            subnet_net1_igw
            subnet_net1_ewcf
        }
        protocol "TCP/IP"
        cidr "16.0.1.64/26"
    }

    subnet subnet_net1_ewcf {
        connections {
            subnet_net1_osint
        }
        protocol "TCP/IP"
        cidr "16.0.1.128/26"
    }
}
net net2 {
    protocol "TCP/IP"
    cidr "16.0.2.0/24"

    subnet subnet_net2_igw {
        protocol "TCP/IP"
        cidr "16.0.2.0/25"
    }

}

net net3 {
    protocol "TCP/IP"
    cidr "16.0.3.0/24"

    subnet subnet_net3_osint {
        protocol "TCP/IP"
        cidr "16.0.3.0/25"
    }

    subnet subnet_net3_ewcf {
        protocol "TCP/IP"
        cidr "16.0.3.128/25"
    }

}

// credentials region
key_pair ssh_key {
    user "ubuntu"
    keyfile "/home/ubuntu/.ssh/openstack.key"
    algorithm "RSA"
    bits 4096
}
security_group sg {
    egress icmp {
        protocol "ICMP"
        from_port -1
```

```
            to_port -1
            cidr ["0.0.0.0/0"]
        }
        ingress http {
            protocol "TCP"
            from_port 80
            to_port 80
            cidr ["0.0.0.0/0"]
        }
        ingress https {
            protocol "TCP"
            from_port 443
            to_port 443
            cidr ["0.0.0.0/0"]
        }
        ingress ssh {
            protocol "TCP"
            from_port 22
            to_port 22
            cidr ["0.0.0.0/0"]
        }

        ifaces igw_vm_oam, igw_vm_net1, igw_vm_net2, osint_vm_oam, osint_vm_net1, osint_vm_net3,
ewcf_vm_oam, ewcf_vm_net1, ewcf_vm_net3
    }
}

deployment config1 {
    osint -> osint_vm,
    iwg -> igw_vm,
    ewcf -> ewcf_vm
}

active deployment config1

concretizations {
    concrete_infrastructure con_infra {

        provider openstack {

            vm concrete_osint_vm {
                properties {
                    // Actually, this is not recognized by ICG, so it's useless
                    vm_name = "osint";
                    // vm_flavor property moved to "size" attribute
                    vm_key_name = "ubuntu";
                }
                maps osint_vm
            }

            vm concrete_igw_vm {
                properties {
                    vm_name = "igw";
                    vm_key_name = "ubuntu";
                }
                maps igw_vm
            }

            vm concrete_ewcf_vm {
```

```
            properties {
                vm_name = "ewcf";
                vm_key_name = "ubuntu";
            }
            maps ewcf_vm
        }

        // Concrete Network region
        net concrete_oam {
            properties {
                name = "uc3_oam";
            }
            maps oam
        }

        net concrete_net1 {
            properties {
                name = "uc3_net1";
            }
            maps net1
        }

        net concrete_net2 {
            properties {
                name = "uc3_net2";
            }
            maps net2
        }

        net concrete_net3 {
            properties {
                name = "uc3_net3";
            }
            maps net3
        }

    }
  }
  active con_infra
}
optimization opt {
    objectives {
        "cost" => min
        "performance" => max
        "availability" => max
    }
    nonfunctional_requirements {
        req1 "cost <= 300" max 300.0 => "cost";
        req2 "performance >= 7%" min 7.0 => "performance";
        req3 "elements" => "VM, Storage";
    }
}
```

Using this input DOML, the solution provided by the IOP is the following one:

```
doml uc3_openstack

application app {

    software_component iwg {
```

```
        provides { net_info }
    }
    software_component osint {
        provides { osint_info }
        consumes { net_info, get_twitter, ewcf_rest_interface }
    }
    software_component ewcf {
        provides { ewcf_rest_interface }
        consumes { get_firebase }
    }
    saas external_twitter {
        provides { get_twitter @ "https://twitter_api/get" }
    }
    saas external_firebase {
        provides { get_firebase @ "https://firebase_api/get" }
    }

}

infrastructure infra {
    vm igw_vm {
        os "CentOS-7-2111"
        size "small-centos"

        iface igw_vm_oam {
            belongs_to subnet_oam_igw
        }

        iface igw_vm_net1 {
            belongs_to subnet_net1_igw
        }

        iface igw_vm_net2 {
            belongs_to subnet_net2_igw
        }

        credentials ssh_key
    }

    vm osint_vm {
        os "CentOS-7-2111"
        size "small-centos"

        iface osint_vm_oam {
            belongs_to subnet_oam_osint
        }

        iface osint_vm_net1 {
            belongs_to subnet_net1_osint
        }

        iface osint_vm_net3 {
            belongs_to subnet_net3_osint
        }

        credentials ssh_key
    }

        vm ewcf_vm {
```

```
    os "CentOS-7-2111"
    size "small-centos"

    iface ewcf_vm_oam {
        belongs_to subnet_oam_ewcf
    }

    iface ewcf_vm_net1 {
        belongs_to subnet_net1_ewcf
    }

    iface ewcf_vm_net3 {
        belongs_to subnet_net3_ewcf
    }

    credentials ssh_key
}

container cont_tia {

    host osint_vm {

        container_port 80
        vm_port 8080
        iface osint_vm_oam

    }
}

container cont_tis {
    host osint_vm {
        container_port 80
        vm_port 8080
        iface osint_vm_oam
    }
}

container cont_twr {
    host osint_vm {
        container_port 81
        vm_port 8081
        iface osint_vm_oam
    }
}

container c1 {
    host igw_vm {
        container_port 82
        vm_port 8082
        iface igw_vm_oam
    }
}

container c2 {
    host igw_vm {
        container_port 83
        vm_port 8083
        iface igw_vm_oam
    }
```

```
}

container c3 {
    host igw_vm {
        container_port 84
        vm_port 8084
        iface igw_vm_oam
    }
}

container cont_mongodb {
    host ewcf_vm {
        container_port 85
        vm_port 8085
        iface ewcf_vm_oam
    }
}

container cont_ewcf {
    host ewcf_vm {
        container_port 86
        vm_port 8086
        iface ewcf_vm_oam
    }
}
net oam {
    protocol "TCP/IP"
    cidr "16.0.0.0/24"

    subnet subnet_oam_igw {
        protocol "TCP/IP"
        cidr "16.0.1.0/26"
    }

    subnet subnet_oam_osint {
        protocol "TCP/IP"
        cidr "16.0.1.64/26"
    }

    subnet subnet_oam_ewcf {
        protocol "TCP/IP"
        cidr "16.0.1.128/26"
    }
}

net net1 {
    protocol "TCP/IP"
    cidr "16.0.1.0/24"

    // Subnets definition
    subnet subnet_net1_igw {
        connections {
            subnet_net1_osint
        }
        protocol "TCP/IP"
        cidr "16.0.1.0/25"
    }

    subnet subnet_net1_osint {
```

```
        connections {
            subnet_net1_igw
            subnet_net1_ewcf
        }
        protocol "TCP/IP"
        cidr "16.0.1.64/26"
    }

    subnet subnet_net1_ewcf {
        connections {
            subnet_net1_osint
        }
        protocol "TCP/IP"
        cidr "16.0.1.128/26"
    }
}

net net2 {
    protocol "TCP/IP"
    cidr "16.0.2.0/24"

    subnet subnet_net2_igw {
        protocol "TCP/IP"
        cidr "16.0.2.0/25"
    }

}

net net3 {
    protocol "TCP/IP"
    cidr "16.0.3.0/24"

    subnet subnet_net3_osint {
        protocol "TCP/IP"
        cidr "16.0.3.0/25"
    }

    subnet subnet_net3_ewcf {
        protocol "TCP/IP"
        cidr "16.0.3.128/25"
    }

}

key_pair ssh_key {
    user "ubuntu"
    keyfile "/home/ubuntu/.ssh/openstack.key"
    algorithm "RSA"
    bits 4096
}
security_group sg {
    egress icmp {
        protocol "ICMP"
        from_port -1
        to_port -1
        cidr ["0.0.0.0/0"]
    }
    ingress http {
        protocol "TCP"
```

```
            from_port 80
            to_port 80
            cidr ["0.0.0.0/0"]
        }
        ingress https {
            protocol "TCP"
            from_port 443
            to_port 443
            cidr ["0.0.0.0/0"]
        }
        ingress ssh {
            protocol "TCP"
            from_port 22
            to_port 22
            cidr ["0.0.0.0/0"]
        }

        ifaces igw_vm_oam, igw_vm_net1, igw_vm_net2, osint_vm_oam, osint_vm_net1, osint_vm_net3,
ewcf_vm_oam, ewcf_vm_net1, ewcf_vm_net3
    }
}

deployment config1 {
    osint -> osint_vm,
    iwg -> igw_vm,
    ewcf -> ewcf_vm
}

active deployment config1

concretizations {
    concrete_infrastructure con_infra {

        provider openstack {

            // Concrete computing nodes region

            vm concrete_osint_vm {
                properties {
                    vm_name = "osint";
                    vm_key_name = "ubuntu";
                }
                maps osint_vm
            }

            vm concrete_igw_vm {
                properties {
                    vm_name = "igw";
                    vm_key_name = "ubuntu";
                }
                maps igw_vm
            }

            vm concrete_ewcf_vm {
                properties {
                    vm_name = "ewcf";
                    vm_key_name = "ubuntu";
                }
                maps ewcf_vm
```

```
        }
        net concrete_oam {
            properties {
                name = "uc3_oam";
            }
            maps oam
        }

        net concrete_net1 {
            properties {
                name = "uc3_net1";
            }
            maps net1
        }

        net concrete_net2 {
            properties {
                name = "uc3_net2";
            }
            maps net2
        }

        net concrete_net3 {
            properties {
                name = "uc3_net3";
            }
            maps net3
        }

    }
}

concrete_infrastructure opt_infra1{
    provider multi {
        storage Storage1_USA {
            properties {
                st_flavor = "Storage1_USA";
                st_name = "Storage1_USA";
                st_Availability = 99.5;
                st_Zone = "United_States";
                st_Request_Response_time_Storage_Performance = 5;
                st_Cost_Currency = 6;
                st_Region = "North_America";
                st_Storage_Subtype = "Block";
                st_Storage_Capacity = 40;
                st_Storage_Type = "General";
                st_Storage_Data_Redundancy = "Locally_redundant_storage__LRS";
                st_provider_OU = "Arsys";
            }
        }
        vm t2_nano{
            properties {
                vm_flavor = "t2_nano";
                vm_name = "t2_nano";
                vm_Availability = 98;
                vm_Response_time_Virtual_Machine_Performance = 3;
                vm_Memory = 0.5;
                vm_Zone = "Ireland";
                vm_Frequency_per_Core = 1500;
```

```
                vm_Virtual_CPU_Cores = 1;
                vm_provider_OU = "AWS";
                vm_public_IP_type = "IPv4";
                vm_Cost_Currency = 4.53;
                vm_Instance_Storage = 40;
                vm_Optimized_for = "General_purpose";
                vm_Region = "Europe";
            }
            maps igw_vm
        }
        net opt_network{
            maps oam
        }

    }
}

concrete_infrastructure opt_infra2{
    provider multi {
        storage Storage1_Spain {
            properties {
                st_flavor = "Storage1_Spain";
                st_name = "Storage1_Spain";
                st_Availability = 99.5;
                st_Zone = "Spain";
                st_Request_Response_time_Storage_Performance = 5;
                st_Cost_Currency = 6;
                st_Region = "Europe";
                st_Storage_Subtype = "Block";
                st_Storage_Capacity = 40;
                st_Storage_Type = "General";
                st_provider_OU = "Arsys";
                st_Storage_Data_Redundancy = "Zoneredundant_storage__ZRS";
            }
        }
        vm t2_nano{
            properties {
                vm_flavor = "t2_nano";
                vm_name = "t2_nano";
                vm_Availability = 98;
                vm_Response_time_Virtual_Machine_Performance = 3;
                vm_Memory = 0.5;
                vm_Zone = "Ireland";
                vm_Frequency_per_Core = 1500;
                vm_Virtual_CPU_Cores = 1;
                vm_provider_OU = "AWS";
                vm_public_IP_type = "IPv4";
                vm_Cost_Currency = 4.53;
                vm_Instance_Storage = 40;
                vm_Optimized_for = "General_purpose";
                vm_Region = "Europe";
            }
            maps igw_vm
        }
        net opt_network{
            maps oam
        }

    }
```

```
}

concrete_infrastructure opt_infra3{
    provider multi {
        storage Storage1_USA {
            properties {
                st_flavor = "Storage1_USA";
                st_name = "Storage1_USA";
                st_Availability = 99.5;
                st_Zone = "United_States";
                st_Request_Response_time_Storage_Performance = 5;
                st_Cost_Currency = 6;
                st_Region = "North_America";
                st_Storage_Subtype = "Block";
                st_Storage_Capacity = 40;
                st_Storage_Type = "General";
                st_Storage_Data_Redundancy = "Locally_redundant_storage__LRS";
                st_provider_OU = "Arsys";
            }
        }
        vm m1_tiny{
            properties {
                vm_flavor = "m1_tiny";
                vm_name = "m1_tiny";
                vm_Availability = 98;
                vm_Response_time_Virtual_Machine_Performance = 10;
                vm_Zone = "Spain";
                vm_Memory = 512;
                vm_Frequency_per_Core = 1500;
                vm_Virtual_CPU_Cores = 1;
                vm_provider_OU = "OpenStack";
                vm_public_IP_type = "IPv4";
                vm_Cost_Currency = 10;
                vm_Region = "Europe";
                vm_Instance_Storage = 1;
            }
            maps igw_vm
        }
        net opt_network{
            maps oam
        }

    }
}

concrete_infrastructure opt_infra4{
    provider multi {
        storage Storage1_Spain {
            properties {
                st_flavor = "Storage1_Spain";
                st_name = "Storage1_Spain";
                st_Availability = 99.5;
                st_Zone = "Spain";
                st_Request_Response_time_Storage_Performance = 5;
                st_Cost_Currency = 6;
                st_Region = "Europe";
                st_Storage_Subtype = "Block";
                st_Storage_Capacity = 40;
                st_Storage_Type = "General";
```

```
            st_provider_OU = "Arsys";
            st_Storage_Data_Redundancy = "Zoneredundant_storage__ZRS";
        }
    }
    vm m1_tiny{
        properties {
            vm_flavor = "m1_tiny";
            vm_name = "m1_tiny";
            vm_Availability = 98;
            vm_Response_time_Virtual_Machine_Performance = 10;
            vm_Zone = "Spain";
            vm_Memory = 512;
            vm_Frequency_per_Core = 1500;
            vm_Virtual_CPU_Cores = 1;
            vm_provider_OU = "OpenStack";
            vm_public_IP_type = "IPv4";
            vm_Cost_Currency = 10;
            vm_Region = "Europe";
            vm_Instance_Storage = 1;
        }
        maps igw_vm
    }
    net opt_network{
        maps oam
    }

  }
}

concrete_infrastructure opt_infra5{
    provider multi {
        storage Storage1_Spain {
            properties {
                st_flavor = "Storage1_Spain";
                st_name = "Storage1_Spain";
                st_Availability = 99.5;
                st_Zone = "Spain";
                st_Request_Response_time_Storage_Performance = 5;
                st_Cost_Currency = 6;
                st_Region = "Europe";
                st_Storage_Subtype = "Block";
                st_Storage_Capacity = 40;
                st_Storage_Type = "General";
                st_provider_OU = "Arsys";
                st_Storage_Data_Redundancy = "Zoneredundant_storage__ZRS";
            }
        }
        vm C1_USA{
            properties {
                vm_flavor = "C1_USA";
                vm_name = "C1_USA";
                vm_Availability = 99.5;
                vm_Response_time_Virtual_Machine_Performance = 5;
                vm_Zone = "United_States";
                vm_Memory = 1;
                vm_Frequency_per_Core = 1500;
                vm_Virtual_CPU_Cores = 1;
                vm_provider_OU = "Arsys";
                vm_Underpinning_Technology = "VmWare";
```

```
                vm_Cost_Currency = 15;
                vm_Region = "North_America";
                vm_Instance_Storage = 40;
            }
            maps igw_vm
        }
        net opt_network{
            maps oam
        }

    }
}
active opt_infra1
}
optimization opt {
    objectives {
        "cost" => min
        "performance" => max
        "availability" => max
    }
    nonfunctional_requirements {
        req1 "cost <= 300" max 300.0 => "cost";
        req2 "performance >= 7%" min 7.0 => "performance";
        req3 "elements" => "VM, Storage";
    }
    solution sol1 {
        objectives {
            cost 10.530000000000001 euro
            performance 8.0 metric
            availability 98.75 %
        }
        decisions ["[Storage1_USA, t2.nano]"]
    }
    solution sol2 {
        objectives {
            cost 10.530000000000001 euro
            performance 8.0 metric
            availability 98.75 %
        }
        decisions ["[Storage1_Spain, t2.nano]"]
    }
    solution sol3 {
        objectives {
            cost 16.0 euro
            performance 15.0 metric
            availability 98.75 %
        }
        decisions ["[Storage1_USA, m1.tiny]"]
    }
    solution sol4 {
        objectives {
            cost 16.0 euro
            performance 15.0 metric
            availability 98.75 %
        }
        decisions ["[Storage1_Spain, m1.tiny]"]
    }
    solution sol5 {
        objectives {
```

```
            cost 21.0 euro
            performance 10.0 metric
            availability 99.5 %
        }
        decisions ["[Storage1_Spain, C1_USA]"]
    }

}
```

APPENDIX C. About Gaia-X

In its 2022 architecture document [3], the concepts in the scope of Gaia-X and their relations are described, that is, the Gaia-X conceptual model, shown in section 2.1.1.2. The Gaia-X core concepts are represented in classes. The upper part of the model shows different actors of Gaia-X (highlighted in blue), while the lower part shows elements of commercial trade and the relationship to actors outside Gaia-X.

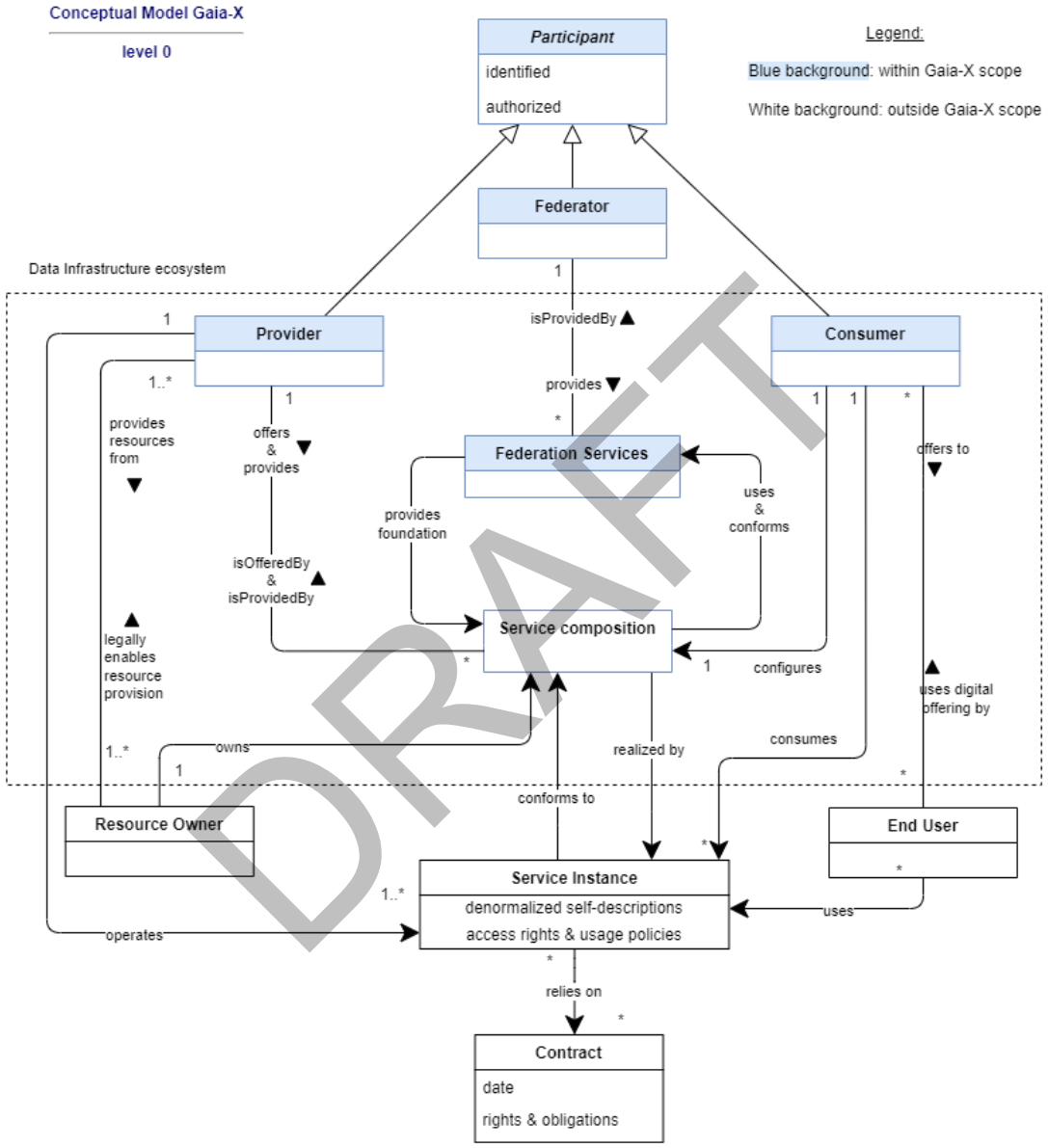

Figure 63: Gaia-X Conceptual model [3]

A *Participant* can take on one or more of the following roles: Provider, Consumer, Federator. Provider and Consumer present the core roles that are in a business-to-business relationship while the Federator enables their interaction.

- A *Provider* is a Participant who provides Resources in the Gaia-X Ecosystem. The Provider defines the Service Offering including terms and conditions as well as technical Policies. Furthermore, it provides the *Service Instance* that includes a Self-Description and associated Policies.

- A *Consumer* is a Participant who searches Service Offerings and consumes Service Instances in the Gaia-X Ecosystem to enable digital offerings for End-Users.
- *Federators* are in charge of the Federation Services and the Federation which are independent of each other. Federators are Gaia-X Participants. There can be one or more Federators per type of Federation Service.
- A *Federation* refers to a loose set of interacting actors that directly or indirectly consume, produce, or provide related Resources.

Resources and Assets describe the goods and objects of a Gaia-X Ecosystem. Resources and Assets compose the Service Offerings.

An *Asset* can be a Data Asset, a Software Asset, a Node or an Interconnection Asset. A set of Policies described in a Self-Description is bound to each Asset. The different categories of Assets are shown in the following figure:

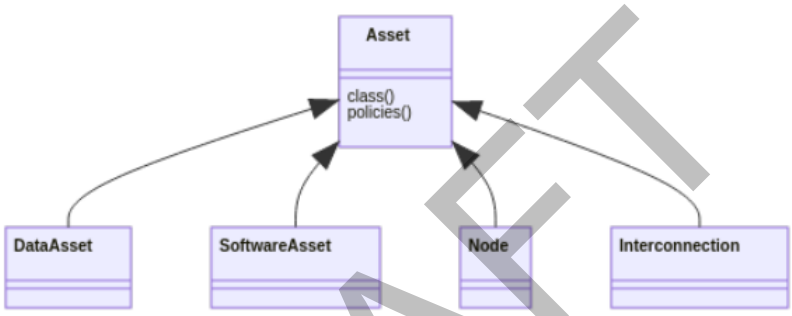

Figure 64: Assets categories [3]

- A *Data Asset* is an Asset that consists of data in any form and necessary information for data sharing.
- A *Node* is an Asset and represents a computational or physical entity that hosts, manipulates, or interacts with other computational or physical entities.
- A *Software Asset* is a form of Assets that consist of non-physical functions.
- An *Interconnection* as an Asset presents the connection between two or more Nodes. These Nodes are usually deployed in different infrastructure domains and owned by different stakeholders, such as Consumers and/or Providers. The Interconnection between the Nodes can be seen as a path which exhibits special characteristics, such as latency, bandwidth and security guarantees, that go beyond the characteristics of a path over the public Internet.

The difference between *Resources* and *Assets* lies in that Resources represent those elements necessary to supply Assets. In other words, they are internal Service Instances not available for order. For example, the running instance that provides a data set is a Resource.

Federation Services are services required for the operational implementation of a Gaia-X Data Ecosystem. They comprise four groups of services that are necessary to enable Federation of Resources, Participants and interactions between Ecosystems. The four service groups are Identity and Trust, Federated Catalogue, Sovereign Data Exchange and Compliance. The Federation Services provide the foundation for Service Offerings.

A *Service Offering* is defined as a set of Resources that a Provider aggregates and publishes as a single entry in a Catalogue. Service Offerings may themselves be aggregated realizing service

composition. The instantiation of a Service Offering is the deliverable of a Provider to a Consumer.

Federated Catalogue

The *Federated Catalogue* [3] constitutes an indexed repository of Gaia-X Self-Descriptions to enable the discovery and selection of Providers and their Service Offerings. The Self-Descriptions are the properties and Claims of Participants and Resources, representing key elements of transparency and trust in Gaia-X.

Self-Descriptions intended for public usage can be published in a Catalogue where they can be found by potential Consumers. The goal of Catalogues in Gaia-X is to enable Consumers to find best-matching offerings and to monitor for relevant changes of the offerings. The Providers decide in a self-sovereign manner which information they want to make public in a Catalogue and which information they only want to share privately.

A Catalogue stores Self-Descriptions both as standalone and as aggregated in a graph data structure. The Self-Description Storage contains the raw published Self-Description files (in the JSON-LD format). The individual Self-Descriptions can reference each other. The Self-Description Graph is the basis for advanced query mechanisms that consider the references between and among Self-Descriptions [3].

Ecosystem-specific Catalogues (e.g., for the automotive domain) and even company-internal Catalogues (with private Self-Descriptions to be used only internally) can be linked to the system of federated Catalogues. The Catalogue federation is used to exchange relevant Self-Descriptions and updates thereof. It is not used to execute queries in a distributed fashion [3].

The system of Federated Catalogues consists of a top-level Catalogue operated by Gaia-X and provides the means to link to Ecosystem-specific Catalogues (e.g., for the automotive domain) and even company-internal Catalogues with private Self-Descriptions to be used only internally. Self-Descriptions in a Catalogue are either loaded directly into a Catalogue or exchanged from another Catalogue through inter-Catalogue synchronization functions [3].

Since Self-Descriptions are protected by cryptographic signatures, they are immutable and cannot be changed once published. This implies that after any changes to a Self-Description, the Participant as the Self-Description issuer has to sign the Self-Description again and release it as a new version.

The possible states for the Self-Description lifecycle are four. All states except “Active” are terminal, which means that no further state transitions are allowed. The states are [3] [21]:

- **Active**: is the default state.
- **End-of-Life**: after a timeout date, e.g., the expiry of a cryptographic signature.
- **Deprecated**: by a newer Self-Description.
- **Revoked**: by the original issuer or a trusted party (e.g., because it contained wrong or fraudulent information).

The Self-Description Graph contains the information imported from the Self-Descriptions that are known to a Catalogue and in an "Active" state [21]. The Self-Description Graph allows for complex queries across Self-Descriptions. To present search results objectively and without discrimination, compliant Catalogues use a query engine with no internal ranking of results. Users can define filters and sort-criteria in their queries. But if some results have no unique ordering according to the defined sort-criteria, they are randomized. Self-Descriptions can be

communicated to the Catalogue by third parties, as the trust verification is independent of the distribution mechanism.

Self-Descriptions can be marked by the issuer as "private" to prevent them from being copied to a public Catalogue by a third party that received the Self-Description over a private channel [3]. The Catalogues have no built-in user interface, instead provide an API that can be used by an external user interface or technical clients [21]. The interfaces of the Gaia-X Federation Services use REST and the OpenAPI specification [22] to describe them.

A *Visitor* is an anonymous user accessing a Catalogue without a known account. Every Non-Visitor user interacts with a Catalogue REST API in the context of a session. Another option to interact with a Catalogue is to use a GUI frontend (e.g., a Gaia-X Portal or a custom GUI implementation) that uses a Catalogue REST API in the background. The interaction between a Catalogue and its GUI frontend is based on an authenticated session for the individual user of the GUI frontend.

Gaia-X Self-Descriptions express characteristics of Resources, Service Offerings and Participants that are linked to their respective Identifiers. Providers are responsible for the creation of Self-Descriptions of their Resources. In addition to self-declared Claims made by Participants, a Self-Description may comprise Credentials issued and signed by trusted parties. Self-Descriptions can be used for [3]:

- Discovery and composition of Service Offerings in a Catalogue
- Tool-assisted evaluation, integration and orchestration of Service Instances/Resources
- Enforcement, continuous validation and trust monitoring
- Negotiation of contractual terms concerning Resources of a Service Offering and Participants

Gaia-X Self-Descriptions are characterized by the following properties [3]:

- Machine-readable and machine-interpretable
- Technology-agnostic
- Adhering to a generalized schema with expressive semantics and validation rules
- Interoperable, following standards in terms of format, structure, and included expressions (semantics)
- Flexible, extensible and future-proof in that new properties can be easily added
- Navigable and referenceable from anywhere in a decentralized fashion
- Accompanied by statements of proof (e.g., certificates or signatures), making them cryptographically trustworthy

The exchange format for Self-Descriptions is JSON-LD. JSON-LD uses JSON encoding to represent subject-predicate-object triples according to the W3C Resource Description Framework (RDF). The relations between Self-Descriptions form a graph with typed edges, which is called the Self-Description Graph. The Catalogues implement a query algorithm on top of the Self-Description Graph [3].

To foster interoperability, Self-Description schemas with optional and mandatory properties and relations are defined. A Self-Description has to state which schemas are used in its metadata. A Self-Description schema corresponds to a class in RDF. The Self-Description schemas form an extensible class hierarchy with inheritance [3].