



MORPHEMIC

Specifications of Morphemic User Interfaces

Modelling and Orchestrating heterogeneous Resources and Polymorphic applications for Holistic Execution and adaptation of Models In the Cloud

H2020-ICT-2018-2020
Leadership in Enabling and Industrial Technologies: Information and Communication Technologies

Grant Agreement Number
871643

Duration
1 January 2020 –
31 December 2022

www.morphemic.cloud

Deliverable reference
D5.1

Date
1 March 2020 – 31 August 2020

Responsible partner
Softteam

Editor(s)
Kaïs Chaabouni

Reviewers
Ciro Formisano, Paweł Skrzypek

Distribution
Public

Availability
morphemic.cloud

Executive summary

This document outlines the specification phase for Morphemic User Interface which includes the requirements gathering, the architecture and the description of the graphical user interface. Morphemic User Interfaces include the tools available for the user to access all Morphemic Platform features from modelling Cross-Cloud applications in CAMEL language to managing cloud deployment, optimization and monitoring. The CAMEL Modelling task is performed by CAMEL Designer tool which is a module for Modelio modelling tool. The managing of the execution task of applications deployment and the polymorphic adaptation is implemented by the Web User Interface Client which includes managing heterogeneous resources such as cloud offers and material accelerators, managing CAMEL Models, optimizing, deploying and monitoring the Cross-Cloud Application. The specifications take into consideration the quality of the user experience and the use case studies requirements as they represent the first users of Morphemic Platform.

Author(s)

Kaïs Chaabouni, Alexandros Raikos, Alicja Reniewicz, Andreas Tsagkaropoulos, Benjamin Leroy, Ferath Kherif, Maxime Compastie, Robert Gdowski, Alessandra Bagnato, Etienne Brosse



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871643

Table of Contents

1	Introduction	4
1.1	Structure of the document	4
1.2	Motivation.....	4
1.3	Deliverable Scope	4
1.4	Related deliverables	4
1.5	Specification Methodology	5
1.6	Definitions, Acronyms and Abbreviations	5
2	Use Case Requirements for Morphemic UI	6
2.1	Use Case 1 - Virtualized base station for 5G cloud-RAN (IS-Wireless)	6
2.2	Use Case 2 - e-BrainScience (CHUV)	7
2.3	Use Case 3 - Computational Fluid Dynamics simulation (ICON).....	8
3	Requirements for Morphemic UI features.....	9
3.1	Requirements for CAMEL Modelling	9
3.1.1	Application Model Management.....	9
3.1.2	Creating new CAMEL Model	10
3.1.3	CAMEL Modelling Environment.....	10
3.1.4	CAMEL Deployment Model.....	11
3.1.5	Requirement model	11
3.1.6	Utility Function	12
3.1.7	CAMEL Metric Model.....	12
3.1.8	Other CAMEL Model related requirements	13
3.2	Requirements for Resource Management	14
3.2.1	Cloud Subscription Requirements	14
3.2.2	Cloud Offers Requirements.....	14
3.2.3	User-supplied Nodes Requirements	15
3.3	Requirements for Application Runtime Status	15
3.3.1	General Runtime Commands	15
3.3.2	Adaptation Plan Determination and Execution	16
3.4	Requirements for Monitoring.....	17
3.5	Requirements for Administration.....	17
3.5.1	User management.....	17
3.5.2	Keeping traces of user's activities	18
3.6	Application management	18
3.7	Requirements for UI Quality of Experience.....	18
4	UI Architecture overview	20
4.1	General Architecture	20
4.2	Web User Interface	20
4.3	User interactions with User Interface	21



4.3.1	User interactions with CAMEL Designer.....	21
4.3.2	User interactions with Morphemic Web App	22
5	CAMEL Designer: Modelio environment for CAMEL Modelling	23
5.1	Overview of “CAMEL Designer” by “Modelio”	23
5.2	CAMEL Designer Modelling Environment	23
5.3	CAMEL Models Management	24
5.4	CAMEL Deployment specification.....	25
5.5	CAMEL Requirement specification	26
5.6	CAMEL Metric specification.....	27
5.7	CAMEL Other specification	28
5.8	CAMEL Models Import/Export	28
6	Specifications of Morphemic Web App Interface	30
6.1	Web User Interface Environment.....	30
6.2	Authentication and user management	30
6.3	Applications and models management.....	33
6.3.1	Applications Models Management.....	33
6.3.2	Application Deployment and Process View	36
6.4	Resources management.....	38
6.4.1	Inspecting the provisioned instances	38
6.4.2	Adding, editing or removing the connection to a cloud provider	39
6.4.3	Adding, editing or removing an instance manually configured (user-supplied node).....	40
6.4.4	Instances inspection	41
6.5	Monitoring	42
7	Final remarks and future plan	44
	References	45

1 Introduction

1.1 Structure of the document

This document analyses the specifications of Morphemic User Interface (UI) and is structured as follows:

- Introduction to the Deliverable (section 1);
- Requirements of the Use Case studies regarding Morphemic UI (section 2);
- Requirements that express the functionalities and the constraints that should be provided by the User Interface according to the several features of Morphemic Platform (section 3);
- General architecture of Morphemic UI (section 4);
- Specification of CAMEL Modelling User Interface (section 5);
- Specification of Morphemic Web Client User Interface (section 6);
- Final remarks and future plan (section 7).

1.2 Motivation

The deliverable “D5.1 User Interfaces Specification” provides a deeper understanding about the user interactions amidst the cross-cloud deployment and polymorphic adaptation process from providing the Cross-Cloud Model (CAMEL Model) of the application to the optimization and adaptive deployment of the application.

Moreover, this deliverable provides a glimpse of the final tools and environments that will be exploited by the end users. Often great tools that bring significant added value can be discarded due to the complicated nature of their user interface interaction which prevents users from having efficient access and exploitation of the full functionalities provided by the final product. This project aims to achieve great added value and innovation in polymorphic adaptation which is the ability of Morphemic Platform to optimize and adapt the deployment configuration of cross-cloud applications by its underneath components. Moreover, it aims to achieve a great user experience with this product to allow users to fully benefit from this innovative concept. This early deliverable will push for better convergence of the work carried out on different features in Morphemic project by offering the UI tool that ensures the testing of the different components. In addition, this will facilitate an early release of a prototype that provides sufficient functionalities to test the use cases and have early feedback about the nature of the required final product.

1.3 Deliverable Scope

This deliverable describes the specification phase of the User Interface and concerns all user interactions with Morphemic Platform that can be regarded as a user input such as providing CAMEL Model or adding resources. Furthermore, the user interface specifications include displaying information to the user for visualization purpose such as monitoring or consulting the runtime status. Morphemic Platform covers several features from modelling cross-cloud Applications (CAMEL Modelling), continuous and autonomous optimization and deployment, and providing access to several cloud capabilities. Therefore, the deliverable includes specifications of the web and desktop user interfaces which contains a description of the menus, forms, tools and the navigation through the different pages. This document is considered as the reference for the UI specification and is intended for all parties involved in UI development or other component that interact with the UI.

1.4 Related deliverables

The deliverable “D5.1 User Interfaces Specification” is an early release of the Morphemic User Interface specifications as the specification of the final User Interface release will evolve during Morphemic project period. Thus, the second deliverable “D5.2 User Interface Guidelines” scheduled for the end of the project will contain the description of the development phase along with guidelines about how to use the released Morphemic Platform User Interface.

Moreover, the deliverable “D6.1 Industrial Requirements Analysis” provides the industrial requirements that cover all Morphemic Platform capabilities including features that have an impact on the user interface.

Furthermore, the upcoming deliverable “D4.1 Architecture of pre-processor and proactive reconfiguration” scheduled for the end of the 12th month of the project will present the architecture of Morphemic platform including how the user interface is integrated with the other components.

1.5 Specification Methodology

The specification methodology consists of gathering the functional and non-functional requirements of the UI with their priorities from: (1) Use Case partners who will be using and validating Morphemic Platform (2) and from other partners who are involved in developing other Morphemic features that are related to the UI. Moreover, the Morphemic team responsible for the UI feature classified the requirements into categories that represent the main functionalities of the platform. These requirements are then translated into detailed specifications about the architecture, menus and web pages. Furthermore, these specifications are illustrated with “Mock-ups” that show an overview of the UI expected at the end of the development. These Mock-ups are validated by the members of Morphemic UI feature team members and then by the entire consortium during a plenary meeting in order to start with a large consensus on the basic functionalities of the UI. After the validation of the specifications and Mock-ups, the development of the User Interface (expected to be reported on deliverable “D5.2 User Interface Guidelines”) will be focusing on delivering a first prototype to be tested by users for feedback and for improving the UI in the next releases.

1.6 Definitions, Acronyms and Abbreviations

Term	Definition
AMI	Amazon Machine Images
ArchiMate	ArchiMate (Architecture-aniMate) is an Enterprise Architecture modelling language
AWS	Amazon Web Services
App	Application
BFF	Back for Frontend
BPMN	Business Process Model and Notation
CAMEL	Cloud Application Modelling and Execution Language
CFD	Computational Fluid Dynamics
CHUV	Centre Hospitalier Universitaire Vaudois
CPU	Central Processing Unit
Cross-Cloud Application	Application that runs on more than one Cloud provider at the same time
DSL	Domain-specific language
DU	Distributed Unit
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
IDE	Integrated Development Environment
IP	Internet Protocol
OS	Operating System
PNF	Physical Network Function
RAM	Random Access Memory
RAN	Radio Access Network
RU	Radio Unit
SLA	Service Level Agreement
SPA	Single Page Application is a web application loaded once as a single web page in the browser and where only components and data are re-loaded and no page reloading is required during use
SPM	Statistical Parametric Mapping
UI	User Interface
UI Mock-up	UI model / graphic design used for prototyping and validation of specification.
UI Prototype	An early release of the UI Software that serves for evaluating and improving its future releases.
UI Software	Every software tool intended for user input or for providing information to the user
UML	Unified Modeling Language
VM	Virtual Machine
VNF	Virtual Network Function
Web App	Web Application
XML	eXtensible Markup Language
XMI	XML Metadata Interchange

2 Use Case Requirements for Morphemic UI

Morphemic project relies on three use case studies in order to validate its solution of continuous optimizing and polymorphing cloud-based applications. Therefore, the requirements elicitation for the User Interface will begin with extracting the requirements expressed by the Use Case partners regarding the features that they need in the User Interface in order to be able to test their use case. The requirements expressed by Use Case partners, along with other requirements expressed from other technology providers and academic partners will be considered to generate a full list of requirements for Morphemic UI features detailed in the next section.

The Use Cases cover several areas of industry: medical research, telecommunication and fluid mechanics' research.

These use case studies are provided by different stakeholders:

- IS-Wireless is an SME based on Poland with expertise in developing algorithms, protocols and tools for 4G and 5G mobile networks.
- Centre Hospitalier Universitaire Vaudois (CHUV): Lausanne university hospital is one of the five Swiss university hospitals. Through its collaboration with the Faculty of Biology and Medicine of the University of Lausanne, CHUV plays a key role in the areas of medical care, bio Prototype medical research and education.
- ICON (ICON Technology & Process Consulting Limited) operates in the high-tech field of Computational Fluid Dynamics (CFD) and provides blue-chip multi-sector engineering companies, their suppliers and consultants with the ability to predict fluid flow using 3D computer simulation.

More details about the use case studies are provided in the deliverable “D6.1 Industrial Requirements Analysis”.

2.1 Use Case 1 - Virtualized base station for 5G cloud-RAN (IS-Wireless)

The use case provided by “IS-Wireless” is targeting a deployment of virtualized RAN components (VNF) within a networking slice as an enabler for 5G communication. Virtualized RAN components will be deployed in relation to the physical network function (PNF) containing a physical antenna. The deployment constraints for virtualized network functions (VNFs), DU and RU components in our case, such as latency, gives enough leeway to deploy them within Edge-Cloud continuum, thus addressing various dynamic scenarios.

Three roles have been identified for using Morphemic UI and handling the life cycle of the slice:

1. Slice Engineer;
2. Slice Administrator;
3. Infrastructure Administrator.

Slice Engineer role is to create the description of the slice and its constituent VNFs, their running requirements, interconnectivity between each other, 3rd party applications and PNFs as well as their dependencies. Slice Engineer also needs to be able to configure each component of the slice as well as be able to monitor running slices and react in case of the abnormal behaviour. It would include tweaking topology, dependencies between VNFs, and/or requirements for deployment.

Slice Administrator defines high level goals such as SLA (Service Level Agreement) and the efficiency of the running service. He/she would need to be able to inspect the operational status of the slice at run time to make decision whether and how to change the utility function and the constraints.

Infrastructure Administrator onboards a suitable cloud/service infrastructure to Morphemic Platform. The infrastructure aims at serving deployed application components. It could be a public, private Clouds as well as PNFs.

Table 1: Use Case 1 - Virtualized base station for 5G cloud-RAN (IS-Wireless): Requirements for Morphemic UI enumerates the list of requirements for Use Case 1 regarding Morphemic UI.

Table 1: Use Case 1 - Virtualized base station for 5G cloud-RAN (IS-Wireless): Requirements for Morphemic UI

ID	Requirement	Description	Actor	Priority
UC-UI1.1	Application and topology description	The user should be able to identify topologies of given software components, together with the description of their requirements (CPU, RAM, etc.) and dependencies (communication between components, booting order, etc.) in order to translate them into CAMEL Model.	Slice Engineer	High
UC-UI1.2	Adding CAMEL model	The user should be able to upload existing CAMEL model.	Slice Engineer	High

UC-UI1.3	Instantiation time configuration	The UI should allow user to configure software components during the instantiation time. Used to pass arbitrary configuration scripts to setup deployment specific parameters e.g. IP of other software components.	Slice Engineer	Medium
UC-UI1.4	Run-time configuration	The UI should allow configuration of software components at runtime. Configuration of parameters related to the deployment context change.	Slice Engineer	Low
UC-UI1.5	App performance monitoring	The UI should allow user to monitor performance of the software components at runtime.	Slice Engineer	High
UC-UI1.6	App topology performance monitoring	The UI should allow user to see the performance of the entire slice in the form of statistics and metrics.	Slice Administrator	High
UC-UI1.7	Utility function and constraints editing	The user should be able to tweak the parameters of the utility function and constraints to change the deployment of the slice.	Slice Administrator	Medium
UC-UI1.8	Adding clouds	The UI should allow user to add new cloud providers (e.g. a private cloud of the client).	Infrastructure Administrator	High
UC-UI1.9	Adding external functions	The user should be able to add BYON (Bring Your Own Node) machine which will also provide to non-virtualized external network functions (PNF) to the cloud-RAN deployment.	Infrastructure Administrator	High

2.2 Use Case 2 - e-BrainScience (CHUV)

The use case “e-BrainScience” of “Centre Hospitalier Universitaire Vaudois” (CHUV) is targeting deployment of three applications with different types of services and execution environments:

- Image pre-processing pipeline (batch processing / private cloud);
- SPM on web (interactive / public or private cloud);
- Federated learning (interactive and batch processing / private and public cloud).

Two roles, associated with deployment of Morphemic platform, have been identified:

1. the Infrastructure administrator and Service support,
2. the Research/clinical lab manager.

The Infrastructure administrator and Service support is in charge, in each research or clinical centre, for the configuration, deployment and incident management of the local node of the federated infrastructure and the community cloud. The Research/clinical lab managers, in each participating centre, are responsible of the configuration and monitoring of three applications included in the e-BrainScience use case.

Table 2: Use Case 2 - e-BrainScience (CHUV): Requirements for Morphemic UI enumerates the list of requirements for Use Case 2 regarding Morphemic UI.

Table 2: Use Case 2 - e-BrainScience (CHUV): Requirements for Morphemic UI

ID	Requirement	Description	Actor	Priority
UC-UI2.1	App deployment configuration	The user should be able to enter the requirements for the deployment of the application: storage size, number of workers.	Research/clinical lab manager	High
UC-UI2.2	Crash/malfunctioning detection	The user should be informed of crash or malfunctioning of the deployment by Morphemic.	Infrastructure administrator and Service support	High

UC-UI2.3	App performance monitoring	The UI should provide performance monitoring for the user.	Infrastructure administrator and Service support	High
UC-UI2.4	Start/stop/restart application	The user should be able to start, stop and restart application on optimised environment.	Infrastructure administrator and Service support	High
UC-UI2.5	Utility function and constraints editing	The user should be able to set and modify utility function, constraints.	Research/clinical lab manager	Medium
UC-UI2.6	User privileges editing	The administrator should be able to manage user privileges for accessing Morphemic features.	Infrastructure administrator Service support	Medium
UC-UI2.7	Set-up cloud infrastructure	Set-up federated infrastructure: Select locations of public cloud and local nodes. Set the network connectivity between the central node and the local nodes.	Infrastructure administrator Service support	High

2.3 Use Case 3 - Computational Fluid Dynamics simulation (ICON)

The Computational Fluid Dynamics (CFD) simulation use case is proposed by “ICON” and includes the deployment of two types of CFD cases: the high-fidelity simulations and the low fidelity simulations. These cases require different types of resources and can be deployed differently in a cost-effective manner. More details about the high-fidelity and low-fidelity simulations are provided in the deliverable “D6.1 Industrial Requirements Analysis”. The requirements for the User Interface used to manage the deployments of the CFD simulations is listed in the Table 3: Use Case 3 - Computational Fluid Dynamics simulation (ICON): Requirements for Morphemic UI below.

Table 3: Use Case 3 - Computational Fluid Dynamics simulation (ICON): Requirements for Morphemic UI

ID	Requirement	Description	Actor	Priority
UC-UI3.1	Select and load a case to be run by the deployed application	The user should be able to select and load a case prepared to be run with the deployed application: iconCFD, OpenFOAM, etc.	CFD Engineer	Medium
UC-UI3.2	Selection of the appropriate deployment resources based on user preferences	The user can add constraints to be respected such as response time and cost which will impact the selection of the deployed resources such as the right amount of CPU resources.	CFD Engineer	Medium
UC-UI3.6	Cost estimation	The UI should provide cost estimation based on the most cost-effective resources for the job in the time allocated.	Project Manager	Medium
UC-UI3.7	Time estimation	The UI should provide time estimation for the job to be completed.	Project Manager	Medium

3 Requirements for Morphemic UI features

The requirement elicitation for Morphemic UI has permitted to acquire a set of requirements that cover multiple UI features. These requirements can be categorized in the following UI features: CAMEL Modelling, Cloud Resources Management, Application Runtime Status, Monitoring, User and Application Management, and UI Quality of Experience. The requirements are organized according to a specific template described as follows: each requirement is identified by a unique ID and is assigned a priority level to identify the most urgent needs. Some of the requirements have already been covered by the User Interface of Melodic¹ platform which will be the main component for deploying the cross-cloud applications in Morphemic platform. Since the features covered by Melodic UI will be part of the functionalities provided by Morphemic UI, it will be possible to extensively reuse part of the solutions already developed in Melodic. On the other hand, the requirements that are not covered in Melodic UI will be developed as completely new features. Therefore, the nature of the coverage in Melodic UI is also specified in the requirements to be adapted in Morphemic UI. For example, CAMEL Modelling has been realized during Melodic project using CAMEL Eclipse² DSL Editor, whereas the aim of Morphemic project is to achieve more user-friendly environment requiring less coding and more intuitive graphical design.

3.1 Requirements for CAMEL Modelling

The deployment and polymorphic adaptation of cross-cloud applications in Morphemic project is based on CAMEL models. The Cloud Application Modelling and Execution Language (CAMEL)³ is a multi-domain-specific language (DSL)⁴ allowing users to specify multiple aspects/domains related to multi-/cross-cloud applications, such as deployment, requirements, metrics, execution, etc. Furthermore, CAMEL will be extended in this project to cover the polymorphic modelling concepts that allow applications to have several possible deployment configurations.

3.1.1 Application Model Management

Morphemic UI should be able to manage multiple applications and therefore manage multiple CAMEL models. The following table (Table 4: Model Management requirements) enumerates the list of the requirements regarding CAMEL models management. Melodic covers some of these model management requirements by uploading the CAMEL file in XMI format. These high priority requirements will be copied in Morphemic by uploading the XMI file to register a new Application in Morphemic UI.

Table 4: Model Management requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CM01	Use CAMEL as default language	The UI should use CAMEL as the default language for modelling Cross-Cloud Applications.	High	Covered
CM02	Add a new CAMEL model	The user should be able to create a new CAMEL Model.	High	Covered
CM03	Delete a CAMEL model	The user should be able to delete an existing CAMEL model.	Medium	Covered
CM04	Update a CAMEL model	The user should be able to modify an existing CAMEL model such as modifying the configurations of the software components, the requirements, the utility function, etc.	Medium	Covered via CAMEL DSL Editor
CM05	Duplicate a CAMEL model	The UI should allow user to copy an existing CAMEL model to facilitate having different versions of the model of the same application.	Low	Not Covered

¹ <https://melodic.cloud/>

² <https://www.eclipse.org/ide/>

³ <http://camel-dsl.org/documentation/>

⁴ https://en.wikipedia.org/wiki/Domain-specific_language

CM06	Show an overview of a selected CAMEL model	The user can see an overview of the selected CAMEL model such as the CAMEL DSL code and the associated XMI file.	Medium	Not Covered
CM07	Ability to manage many CAMEL models	The user should be able to store many CAMEL models in the Morphemic Platform and choose which model he wants to deploy.	Medium	Covered

3.1.2 Creating new CAMEL Model

Creating and designing new CAMEL models is a crucial part of Morphemic requirements because it represents the description of several domains of the cross-cloud application upon which the polymorphic adaptation will be based on. Table 5: Creating new CAMEL Model requirements lists the requirements regarding creating new CAMEL models. The requirement CM08 is already covered in Melodic UI where a new CAMEL model is created by uploading an XMI file and this will be replicated in Morphemic UI.

Table 5: Creating new CAMEL Model requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CM08	Create a new CAMEL Model by uploading XMI file	The user should be able to add a new CAMEL Model in the list of the stored CAMEL Models managed by Morphemic Platform by uploading the CAMEL Model file in XMI format	High	Covered
CM09	Create a new CAMEL Model by uploading CAMEL DSL file	The user can add a new CAMEL Model in the list of the stored CAMEL Models managed by Morphemic Platform by uploading the CAMEL Model file written in CAMEL DSL language	Low	Not Covered
CM10	Create a new CAMEL Model with easy and user-friendly UI	The user can create a CAMEL Model for his application by using a simplified CAMEL Designer without having to use the CAMEL DSL editor	Medium	Not Covered

3.1.3 CAMEL Modelling Environment

The requirements about the CAMEL modelling environment refer to the tools available for the user to design the entirety of CAMEL model or by re-using existing elements of CAMEL models. Table 6: CAMEL Modelling Environment requirements contains the list of the requirements towards the CAMEL modelling environment.

Table 6: CAMEL Modelling Environment requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CM11	Integrated and easy web UI for creating a CAMEL Model	The user can design a new CAMEL Model from the unified Web UI without having to download other tools	Low	Not Covered
CM13	Enable the re-use of application components in new applications	The UI should allow the storage, search & selection of CAMEL components to be reused in other CAMEL Models	Medium	Not Covered
CM14	Enable the re-use of other CAMEL elements like metrics, attributes & units	The UI should allow the storage, search & selection of CAMEL elements such as metrics, attributes & units to be reused in other CAMEL Models	Medium	Not Covered

CM50	CAMEL syntax checking	The UI should include syntax checking and inform the user if there is an error in his CAMEL modelling	Medium	Covered in CAMEL DSL Editor
CM51	CAMEL modelling tips and suggestions	The UI can provide tips and suggestions for the user to help with the modelling task	Low	Not covered
CM52	Graphical environment for CAMEL modelling	The UI design environment should be based on graphical modelling tools instead of code-based tool.	Medium	Not covered

3.1.4 CAMEL Deployment Model

The requirements for CAMEL deployment are referring to the ability of creating the software components to be deployed and the communications between them (see Table 7: CAMEL Deployment Model requirements). This feature is covered in Melodic project with the use of CAMEL DSL Editor based on Eclipse IDE and will afterward be feasible using CAMEL Designer Module for Modelio.

Table 7: CAMEL Deployment Model requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CM15	Specify CAMEL Deployment Model	The user should be able to specify deployment model (software components, their configuration and requirements). Application components and references to their requirements are specified in the requirement model as well as the communication between them.	High	Covered with CAMEL DSL Editor
CM16	Add Software Components in the Deployment Model	The user should be able to add Software components in the Deployment Model by adding their configurations such as script configuration, requirements for each Software Component such as the resources required, horizontal scaling and images.	High	Covered with CAMEL DSL Editor
CM17	Specify application components definitions	The user should be able to define configurations for each application component. For each type of application component, a dedicated form for configuration is required: script, docker, serverless, hardware accelerator.	High	Covered with CAMEL DSL Editor
CM18	Specify communication	The user should be able to define the communication dependencies between components.	High	Covered with CAMEL DSL Editor

3.1.5 Requirement model

The requirement model in CAMEL Model refers to requirements regarding the cloud provider, the cloud image, the cloud location, the horizontal scale, the service level constraints, etc. The requirement model requirements are listed in Table 8: Requirement model requirements and are provided with CAMEL DSL Eclipse based Editor and will be developed in CAMEL Designer module for Modelio.

Table 8: Requirement model requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CM19	Specify requirements for application	The user should be able to specify all types of available requirements (OS, provider, location, image, horizontal scale, resource, etc.)	High	Covered with CAMEL DSL Editor
CM20	Specify sets of requirements	The UI should allow user to create a group of selected requirements to be used as description for the requirements of a specific Software Component	High	Covered with CAMEL DSL Editor

3.1.6 Utility Function

Table 9: Utility Optimization Function requirements contains the requirements for the utility function which represents the equation to optimize for the deployment such as the average response time and the cost of the deployment on the cloud. This utility function feature will be extended in the context of Morphemic in order to be evolved as an outcome of high-level policies provided by the user instead of a complex mathematical equation.

Table 9: Utility Optimization Function requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CM61	Specify utility function via high level policies	The user should be able to specify a set of high-level policies for the utility function such as: “as fast as possible” or “within 3 days and with a cost less than 120€”. These requirements are then translated to the mathematical utility function.	High	Not Covered
CM62	Specify utility function via precise mathematical formula	The user should be able to specify the mathematical utility function in the CAMEL Model such as '1/(WorkerCardinality * WorkerPrice)'	High	Covered with CAMEL DSL Editor

3.1.7 CAMEL Metric Model

Table 10: CAMEL Metric Model requirements shows the list of requirements for the Metric model in CAMEL which like the previous sections are covered by CAMEL DSL Eclipse based editor and will be implemented in CAMEL Designer Modelio module.

Table 10: CAMEL Metric Model requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CM23	Specify Metric Model	The user should be able to create a Metric model where metrics, variable, sensors, constraints, etc. are specified.	High	Covered with CAMEL DSL Editor
CM24	Specify measurable attributes	The user should be able to specify a measurable attribute and choose type from types available in CAMEL MetaData Model.	High	Covered with CAMEL DSL Editor
CM25	Specify metric templates used in optimization	The user should be able to specify a new metric template: choose attribute from all defined attributes (see previous requirement), choose unit from Unit Template CAMEL Model and value type from Type Template CAMEL Model.	High	Covered with CAMEL DSL Editor
CM26	Specify variables used in optimization	The user should be able to create new variable: choose template from all defined templates (see previous requirement) and choose component from all defined components.	High	Covered with CAMEL DSL Editor
CM27	Specify sensors used for metrics collection	The user should be able to specify a new sensor.	High	Covered with CAMEL DSL Editor
CM28	Specify constraint models	The user should be able to create constraints through the usage of metrics or variables and mathematical operators (e.g. '>', '<').	High	Covered with CAMEL DSL Editor
CM29	Specify window for metrics	The user should be able to create a measurement window by specifying its type, size type, time size and type unit.	High	Covered with CAMEL DSL Editor

CM30	Specify schedule for metrics	The user should be able to create a measurement schedule by specifying interval and time unit.	High	Covered with CAMEL DSL Editor
CM31	Specify composite metrics	The user should be able to create a composite metric by specifying a template and formula (via metrics.)	High	Covered with CAMEL DSL Editor
CM32	Specify composite metric context	The user should be able to create a composite metric context by specifying metrics (from defined composite metrics), grouping, window (from defined windows), schedule (from defined schedules).	High	Covered with CAMEL DSL Editor
CM33	Specify raw metrics	The user should be able to create a raw metric by specifying template from the list of available templates.	High	Covered with CAMEL DSL Editor
CM34	Specify object context	The user should be able to create an object context for components from deployment type model.	High	Covered with CAMEL DSL Editor
CM35	Specify raw metric context	The user should be able to create a raw metric context by specifying metric (from available raw metrics), sensor (from available sensors), schedule (from available sensors) and object context (from available object contexts).	High	Covered with CAMEL DSL Editor

3.1.8 Other CAMEL Model related requirements

This section contains a set of requirements that concern other model types in CAMEL (see Table 11: Other CAMEL Model related requirements). As the case of the previous model types, most of these model types are also covered in CAMEL DSL editor and will be implemented in CAMEL Designer module for Modelio.

Table 11: Other CAMEL Model related requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CM36	Enable the specification of units	The user should be able to add unit of measurement with regard to the measurement values produced in the context of metrics	Medium	Covered with CAMEL DSL Editor
CM37	Enable the specification of value types	The user should be able to add numeric or non-numeric type values. The value types can be used to denote the range of values that metrics (variables) can take	Medium	Covered with CAMEL DSL Editor
CM38	Enable the specification of security elements	The user should be able to specify security controls, domains, attributes and metrics	Medium	Covered with CAMEL DSL Editor
CM39	Enable the specification of organisation models	The user should be able to specify organisation elements such as organisations, users, etc	Low	Covered with CAMEL DSL Editor
CM40	Enable the specification of Metadata model	The user should be able to specify MetaData model which contains Cloud concepts, big data concepts, utility notions, and the list of (non-functional) properties that a cloud application can have	Medium	Covered with CAMEL DSL Editor
CM41	Integration with Metadata schema editor	The UI should include an editor for extending the metadata schema (support for attributes tree, class and subclass properties and definitions). This metadata schema is used as the background knowledge for extending the expressivity of the CAMEL model	Medium	Not Covered

3.2 Requirements for Resource Management

The Morphemic platform features the polymorphic adaptation of applications to be operated in the cloud and edge. This adaptation requires the specifications and the qualifications of the infrastructure resources involved in the process. This section details the requirements on the interface enabling the regular Morphemic user to manage those resources.

3.2.1 Cloud Subscription Requirements

To define the cloud services a Morphemic instance should be allowed to interact with, the user interface should enable a regular user to specify the basic settings of a cloud account. Those ones should contain the cloud service to be use, the access credentials, and other parameters that are not impacted by the adaptation process. These specifications are referred as cloud subscription. Consequently, the user should be able to perform basic operations to manage those subscriptions, including listing them, adding a new one, removing one and updating the configuration. Those requirements are exposed below in the Table 12: Cloud subscription requirements.

Table 12: Cloud subscription requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CR01	Add new cloud provider Definition	The user should be able to add new cloud provider definition such as AWS, OpenStack, Azure, etc.	Medium	Covered
CR02	See the list cloud providers	The user should be able to list the configured cloud providers.	Medium	Covered
CR03	Modify a Cloud provider definition	The user should be able to update the configuration of the cloud provider.	Medium	Covered
CR04	Delete a Cloud provider Definition	The user should be able to delete the configuration of a cloud provider.	Medium	Covered

3.2.2 Cloud Offers Requirements

Each cloud provider comes with their own characteristics that will impact the behaviour of the application as much as the of the Morphemic one. They include the available cloud regions, the dimensioning of VM type, their cost, their eligibility to operate has hardware accelerator, the software available as VM images. The user should be able to assess the operational capabilities of the cloud providers he has subscribed to. He should also be able to assess the operability. Those requirements are consolidated in the Table 13: Cloud offers requirements.

Table 13: Cloud offers requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CR05	See list of available offers per Cloud provider	The user should be able to see the list of available Cloud Offers per Cloud provider. The exposed peculiarities should at least assess the type of hardware instances, the location, the system images and prices. Facultatively, it could indicate the support for hardware acceleration.	Medium	Covered
CR06	See list of available hardware instances with their configurations	The user should be able to see the list of available hardware instances with their technical specifications. They should include at least the number of cores, RAM memory, operated system images, and the hosting cloud region. Facultatively, it could refer to the support of hardware acceleration and the disk storage.	Medium	Covered
CR07	See list of available locations / Regions	The user should be able to see the list of the operable regions of a cloud subscription.	Medium	Covered
CR08	See list of available images stored in Cloud account	The user should be able to see list of available images stored in Cloud account such as AMI in AWS.	Medium	Covered

CR09	Show the operability status of resources	The UI should list configured resource such as public and private clouds when configured, and indicate clearly if they are ready to operate, if they are undergoing any configuration operation, or lost.	Medium	Covered
------	--	---	--------	---------

3.2.3 User-supplied Nodes Requirements

The Morphemic platform should not be limited to the management of infrastructure resource from cloud providers. Instead, the user should be able to provide infrastructure resources he has under his direct custody, such as on edge devices and on-premise infrastructure. In that specific case, the user should be in charge of acquiring these resources while the Morphemic platform should be limited to their registration management and to their operation. Therefore, the UI should permit the inspection, the addition and the removal of their definitions. The following Table 14: User-supplied nodes requirements exposes these requirements.

Table 14: User-supplied nodes requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
CR10	Add local nodes	A user should be able to add resources he has access to and that are not operated by a cloud provider (such as local node or edge devices).	Medium	Covered
CR11	See list of assessed local nodes	The UI should show the list of available local nodes with their configurations such as number of Cores, RAM.	Medium	Covered
CR12	Modify properties of local cloud node	The user should be able to update the configuration of a user-supplied node.	Medium	Covered
CR13	Delete a local node	The user should be able to remove the definition of a node he has provided if he doesn't want to use it anymore.	Medium	Covered

3.3 Requirements for Application Runtime Status

Once the CAMEL model of an application is specified, Morphemic platform is able to instantiate the application and support it along its life cycle operations. They include the initial deployment, the adaptation and the deallocation. This section describes how the user interface enables the user to turn an application specification into a working instance, manage the instances of different application and control the adaptation process.

3.3.1 General Runtime Commands

The management of application instances requires exposing the list of application instances to the user, and to enable its modification. They should not only include the allocation and the removal of instances, but also interact with the adaptation process, the constraints and the utility function triggering it. The relevant data of the previous and current adaptation process should be accessible to the user. These requirements are synthesized under the Table 15: General runtime commands requirements below.

Table 15: General runtime commands requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
AR01	Create a new Application based on selected CAMEL Model	The user should be able to create a new cross-cloud application by launching the deployment of CAMEL Model from the list of the stored CAMEL Models in the Morphemic UI catalogue.	Medium	Single application only
AR02	Ability to manage multiple Applications	The user should be able to manage multiple Applications in the same Morphemic Platform UI.	Medium	Not covered
AR03	General overview of the deployment	The user should be able to see a general overview of the deployment configuration adaptation phases such as the process view in Melodic UI which contains the phases:	Medium	Covered

	configuration adaptation process	fetching offers, generating constraint problem, reasoning, deploying, updating.		
AR04	Pause, stop, resume the polymorphic adaptation in each phase of the process	The user could pause, stop, resume the adaptation process of the application in each phase of the process.	Low	Not covered
AR05	Check the execution history of applications.	The UI should provide a search interface over applications with a time horizon which enables to check how well the applications behaved in the past and which things went wrong while they were executed and being provisioned.	Low	Not covered
AR06	Ability to modify requirements, constraints and utility function	The user could modify requirements, constraints and utility function from the original CAMEL Model and restart the whole process of optimization and redeployment.	Low	Not covered

3.3.2 Adaptation Plan Determination and Execution

The polymorphic and proactive adaptation features represent key assets for the platform. They drive the determination of the necessary configuration actions to be enforced on applications instances. This process is expected to be complex and the user should therefore be informed of the current status of it in a comprehensive and transparent way. Once the configuration actions to be performed have been determined, the Morphemic platform must commit them on the application. Those actions can include time-consuming processing (e.g. the preparation of components, the allocation of infrastructure resources and their configuration) that will impact the operability of the application. Therefore, the user should be able to investigate the content of the adaptation plan and monitor its execution. The following Table 16: Deployment plan determination requirements contains the requirements formalizing these expectations.

Table 16: Deployment plan determination requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
AR07	Indicate when the process is on "Fetching offers" phase	The user interface should inform the user when the Morphemic platform is currently retrieving offers from cloud service providers during the adaptation process.	Low	Covered
AR08	Indicate when the process is on "Generating Optimization Problem" phase	The UI should inform the user when the Morphemic platform is converting the requirement expressed by him and the information from the current technical context into an optimization problem to be submitted to a solver.	Low	Covered
AR09	Indicate when the process is on "Resolving Optimization Problem" phase	The UI should inform the user when the Morphemic platform is currently solving the prepared optimization problem to determine the necessary reconfiguration actions.	Low	Covered
AR10	Indicate when the process is on "Deploying" phase	The user should be able to monitor the execution of the application deployment and adaptation process.	Medium	Covered
AR11	Check the execution reconfiguration action the application	The user should be able to inspect the undertaken reconfiguration actions, including detailed logs and their duration, as well as the context triggering the reconfiguration.	Medium	Not Covered

3.4 Requirements for Monitoring

The user of the Morphemic platform should be able to visualize monitoring metrics which are retrieved from the monitoring agents. Based on these, the user can verify the correct operation of the application and validate the actions which are undertaken by the platform to maintain a reasonable quality of service. In Morphemic, Grafana⁵ is used to fulfil these goals (as was also done in the User Interface of Melodic), which will be appropriately configured by use-case adopters. The requirements concerning the access to monitoring information are provided in Table 17: Monitoring requirements.

Table 17: Monitoring requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
M01	Access to Application metrics	The user should be able to access aggregated hardware-level and application level metrics (average CPU/RAM usage, number of users, requests per second, etc.)	Medium	Redirected to Grafana
M02	Allow the user to view statistics on the operation of the Morphemic platform	The user should be able to view a timeline with statistics on the reconfiguration actions, the time it took for them, the time to spawn instances per provider and other statistics (also allow modification of these statistics later).	Medium	Covered by Grafana
M03	Embedding Grafana monitoring views in Morphemic UI	The UI should be able to reuse the Grafana monitoring view which has already been developed as part of Melodic by embedding Grafana views inside Morphemic UI.	Medium	Covered by Grafana

3.5 Requirements for Administration

3.5.1 User management

Morphemic platform will be used by multiple companies and organizations with varied size and complexity, thus a proper user management mechanism is required. In particular, Morphemic UI should enable to manage users' accounts, merge users into groups and set permissions to particular UI views for each group. Moreover, users can have roles which specify their privileges. Most of these functionalities are implemented in Melodic, but in Morphemic these functionalities are planned to be extended by adding the possibility of grouping users. These requirements are described in Table 18: User management requirements.

Table 18: User management requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
UM01	Possibility to create 2 types of users: common user and administrator	The administrator should be able to create 2 types of users: <ul style="list-style-type: none"> Common user – can perform application deployment and has all permission related to deployment; Administrator – has permission characteristic for common user and also can manage users (see UM02) and group of users (see UM03). 	Medium	Covered
UM02	Administrator can manage users	Administrator can create/read/update/delete/lock/unlock user accounts.	Medium	Covered
UM03	Possibility to manage group of users	Administrator can create/read/delete group of users, set permissions for particular views for group, update group of users (add/delete users, change permissions to particular views).	Low	Not covered

⁵ <https://grafana.com/>

3.5.2 Keeping traces of user's activities

Table 19: Keeping user activity history requirements shows the list of requirements regarding the ability to keep trace of user's activities such as: creating applications, adding resources, starting deployment, etc. Such a feature is important also for security reasons as it enables to track and discover suspicious activities in a certain Melodic platform installation. This feature is not covered in Melodic but will be fully added to Morphemic Platform.

Table 19: Keeping user activity history requirements

ID	User type	Requirement	Description	Priority	Coverage in Melodic UI
AC01	Admin	View list of activities in the platform	The administrator should be able to see all the activities in the platform and which user is responsible for which activity.	Medium	Not covered
AC02	Admin	Grouping activities per user	The administrator should be able to group activities per user such as: creating applications, adding resources, starting deployment.	Medium	Not covered
AC03	User	User can see his own activities	The user could access his own history of activities.	Medium	Not covered

3.6 Application management

The main role of Morphemic is application deployment, so from the User Interface, it should be possible to create and manage applications. Morphemic is designed as platform for multiple application deployments and this aspect will be also covered in the User Interface. All features from application management area are listed in Table 20: Application management requirements.

Table 20: Application management requirements

ID	Requirement	Description	Priority	Coverage in Melodic UI
AM01	Possibility to manage of application deployments	The user should be able to see applications that are already deployed or currently being deployed, start new deployment, un-deploy application.	Medium	Partially covered (create and view)
AM02	Possibility to control which application deployment is presented now on screen	The UI should allow the user to switch between applications and then display the selected application which deployment is presented in the current context	Medium	Not covered

3.7 Requirements for UI Quality of Experience

The “Quality of Experience (QoE)”[1] of the UI reflects the impression that the user is getting from navigating through the UI and accessing to the several features provided by Morphemic Platform. When managing deployed applications through the Morphemic platform, a clear and consistent quality of experience must be introduced by the user interface in order to maximise usability, information retrievability, contextual awareness and overall visual clarity for each aspect of the corresponding deployed application. Navigating through the platform should enhance the efficacy of user ergonomics and require no additional effort to accommodate novice and experienced users alike. Table 21: Quality requirements enumerate the list of requirements regarding the UI quality.

Table 21: Quality requirements

ID	Requirement	Description
Q01	Easy to use without reading extensive documentation	A user with average devops/cloud knowledge should be able to use Morphemic Platform without having to read a detailed guideline
Q02	Keep user always aware about his current viewed section	The user should always be aware about the section that the current web page belongs to (e.g. CAMEL Model, Resource Manager, Security, etc.)
Q03	Accessibility to most features without deep navigation	The UI should keep most of the features accessible without deep navigation into multiple levels in order to keep the user aware about his current position in the menus of the app
Q04	Keep user always aware about the backend process during Runtime	The UI should always keep the user informed in a transparent way about the current phase of deployment / reconfiguration
Q05	Web UI compatibility with modern browsers	The part of the UI implemented as a Web UI should be compatible with modern web browsers (i.e. Firefox > 74, Chrome > 80, Edge > 80, Safari > 13)

4 UI Architecture overview

4.1 General Architecture

Morphemic Platform is based in a big part on Melodic Core components for optimizing and deploying cross-cloud applications. Morphemic Platform User Interfaces are composed of two tools:

1. CAMEL Designer modelling tool for designing CAMEL models
2. And Morphemic Platform Web App for deploying of applications based on CAMEL models created with “CAMEL Designer”.

“CAMEL Designer” is a module for the modelling Environment Modelio⁶ which allows to design user-oriented diagrams easy to explore and edit.

The Morphemic Platform is designed according to the “Microservice Architecture” which is useful for handling integration between several heterogeneous components and facilitates collaborative development and maintainability for each component. The web user interface for Morphemic Platform (illustrated by Figure 1) is composed of a web client app which is the **Frontend** and an intermediary component called **Back for Front (BFF)**[2] component which acts in a similar way as an “API gateway” or a “Reverse Proxy” as it orchestrates incoming requests and communications with microservices. Thus, the Back for Front (BFF) will be a single point which the Web App client needs to communicate with. The microservices part contain the list of components that handle the deployment and polymorphic adaptation and the handling of the platform data.

Although the final specification of this part will be described in a dedicated deliverable for the Architecture of Morphemic Platform, it can be broken down into these two parts:

- The “Upperware” is the intelligent part responsible for processing and reasoning and will be an extended version of Melodic Core. The Upperware contains new components that act as a pre-processor for Melodic with the addition of forecasting modules for the proactive adaptation feature. The Melodic Core modules are extended to cover these modifications. These modules are responsible for generating constraint problems, optimizing allocation resources, creating the optimized application configuration to be deployed, processing events and delivering monitoring information, etc. More information about Melodic Core is available at the Melodic Project Deliverable: “D3.5 Melodic Upperware”⁷.
- The “Executionware” is responsible for execution which represents the tasks related to scheduling, performing deployment and using resources and cloud offers.

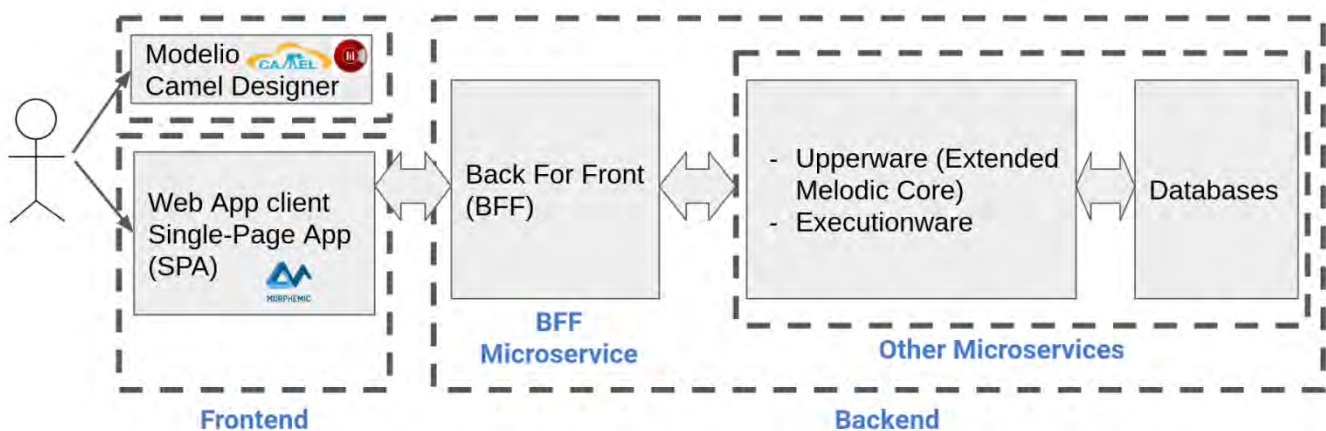


Figure 1: Architecture of Morphemic Web App

4.2 Web User Interface

The Frontend web client is represented by Single Page App (SPA)[3] which is a modern trend in building feature-rich web applications and provides better user experience. This web interface can be thought as a portal to Morphemic

⁶ <https://www.modelio.org/>

⁷ <https://melodic.cloud/wp-content/uploads/2020/03/D3.5-v2.6.pdf>

platform by giving the user access to all commands necessary for polymorphic deployment and adaptation of the targeted cloud-based application. Moreover, the user interface provides information about the state of the deployed applications as well as useful performance indicators. These features can be broken down into the following domains:

- CAMEL Models Management;
- Cloud Resources Management;
- Application Runtime Process View;
- Monitoring;
- Administration;
- Settings.

Like the backend, the frontend can also be thought as “micro-frontends” [4] manifested by an integration of domain specific components and in-app routing to the several views as it is shown in Figure 2. All important features of Morphemic Platform are accessible via several entries in the left slide-menu with entries reserved only for administrator privileges such as the User Management Menu. The header section shows the selected application and its current status. The header contains also the user logged in the current session. The header information should be visible at all time for the user. For more details about the Web Client environment see the specifications on section 6.

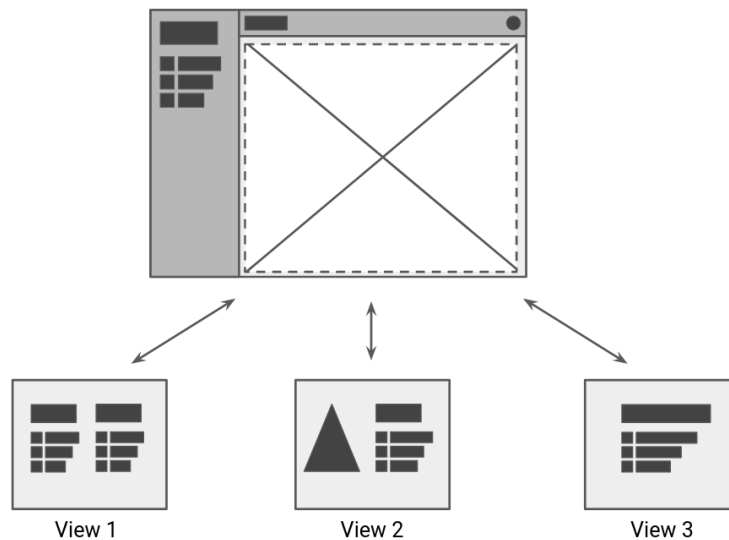


Figure 2: Frontend in-app routing to views

4.3 User interactions with User Interface

4.3.1 User interactions with CAMEL Designer

Figure 3 shows the use case scenarios of CAMEL Designer and the interaction between the user and the Design Process. First, the user creates a new CAMEL Model in Modelio environment or import existing CAMEL Model file. Then, the user updates his CAMEL model by creating CAMEL sub-models such as Deployment Model, Requirement Model, Metric Model, etc. Finally, the user exports the final CAMEL Model into XMI file to be uploaded in Morphemic Web App in order to create a new Application Model.

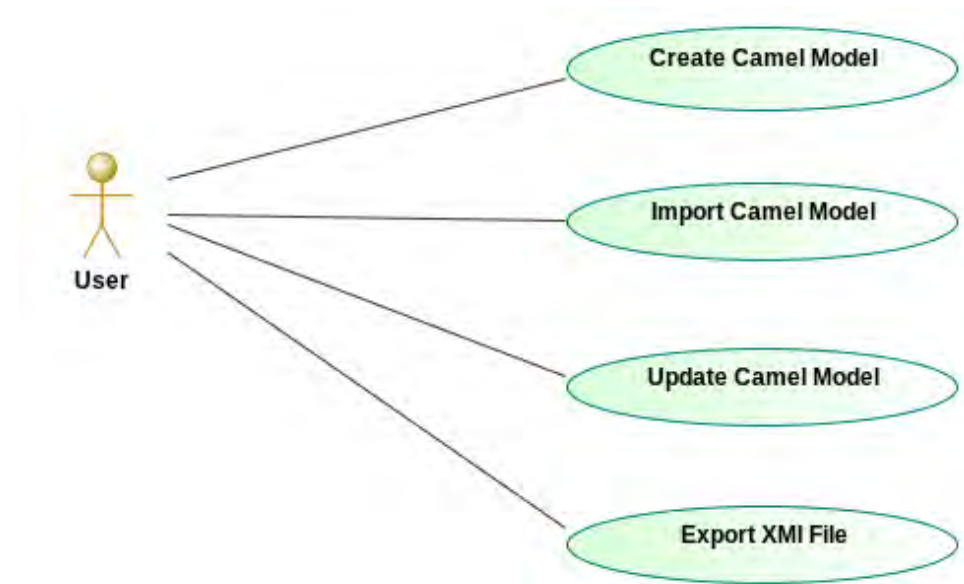


Figure 3: Use Case Diagram for CAMEL Designer

4.3.2 User interactions with Morphemic Web App

Figure 4 shows the Morphemic Web App use case scenarios. At first, the user of Morphemic Web App adds offers of deployment in the platform such as cloud offers, hardware accelerators or user supplied nodes. Then, the user creates a new Application by uploading a new CAMEL file in XMI format. The user can then update the model with high level policies that determine the utility function, metrics and constraints. After finishing updating the model, the user has now an application model ready for optimization and deployment. The user can start the deployment which begins by finding the optimal solution and proceed with deploying the proposed solution. At any time, the user can check the result of the deployment and the current status of the deployed application. Finally, the user can stop the deployment process.

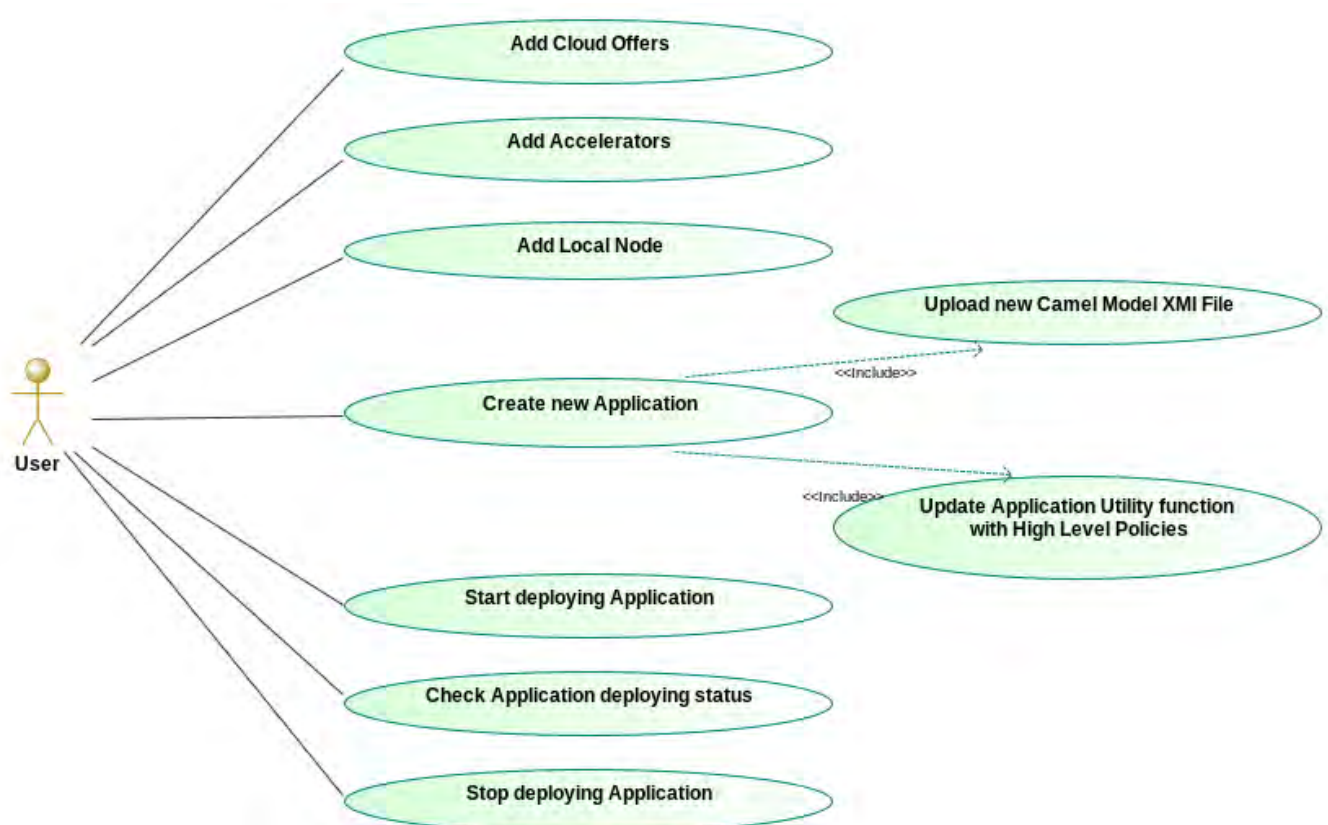


Figure 4: Use Case Diagram for Morphemic Web App Client

5 CAMEL Designer: Modelio environment for CAMEL Modelling

5.1 Overview of “CAMEL Designer” by “Modelio”

CAMEL Designer is an open source module for graphically creating, editing and exporting CAMEL Models in an ergonomic, intuitive and user-friendly environment. CAMEL Designer module is used via Modelio tool community or commercial version and has in both environments all the necessary tools for creating the CAMEL model and all its sub models and components.

Modelio is a modelling tool for a large number of standards for System Engineering, Software Development and Enterprise Architecture. For instance, Modelio includes the standard **Business Process Model and Notation (BPMN)** for modelling business processes, the standard **Unified Modeling Language (UML)** and the **ArchiMate** language for Enterprise Architecture. Modelio offers services and diagrams for modelling and can assist with model auditing and consistency verification. Furthermore, for additional functionalities that require the extension of Modelio modelling capabilities or extending its metamodels, the Modelio Module API is used to create a new module to be deployed into a Modelio project.

Figure 5 illustrates how CAMEL Designer is implemented as module to be deployed on Modelio. CAMEL Designer module includes a UML profile for the CAMEL Language which is implemented by associating stereotypes based on CAMEL concepts to different UML meta-classes such as “Class”, “Package”, “Attribute”, etc. These elements of the CAMEL Profile are created by additional “Commands” provided by CAMEL Designer module. In addition to these commands, the user can design graphically with the diagram tools and with the help of a property page widget for accessing and editing model elements properties. Further details about the modelling environment of CAMEL Designer are provided in the next section. Moreover, CAMEL Designer invokes the CAMEL DSL API⁸ for parsing CAMEL files during import and serializing CAMEL files during export.

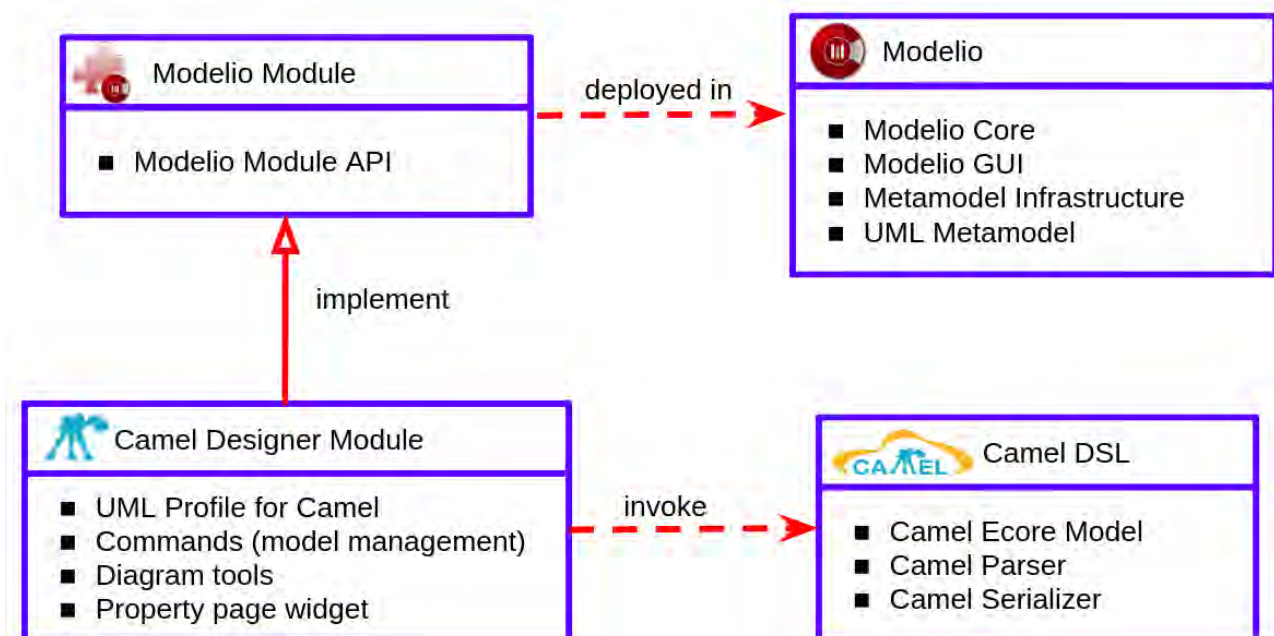


Figure 5: CAMEL Designer Module and Modelio Overview

5.2 CAMEL Designer Modelling Environment

Figure 6 illustrates the modelling environment provided by Modelio which contains a model explorer depicted on the left side of Figure 6. This model explorer shows the hierarchy of the persisted model elements and allows to create, delete and copy/paste other model elements. On the right side of Figure 6, the diagram represents how the elements in the model are represented and linked together. In addition, a set of tools is provided for each diagram to allow the user to modify the model such as adding new elements, properties, dependencies or just customizing the visual appearance of the elements illustrated in the diagram. Moreover, other commands can be called directly on elements from the model

⁸ <http://CAMEL-dsl.org/prototype/>

explorer by right clicking the selected element and executing the command which may bring changes to the model. Finally, on the bottom of Figure 6, the properties section displays a list of properties of the selected element and allows the user to edit these properties.

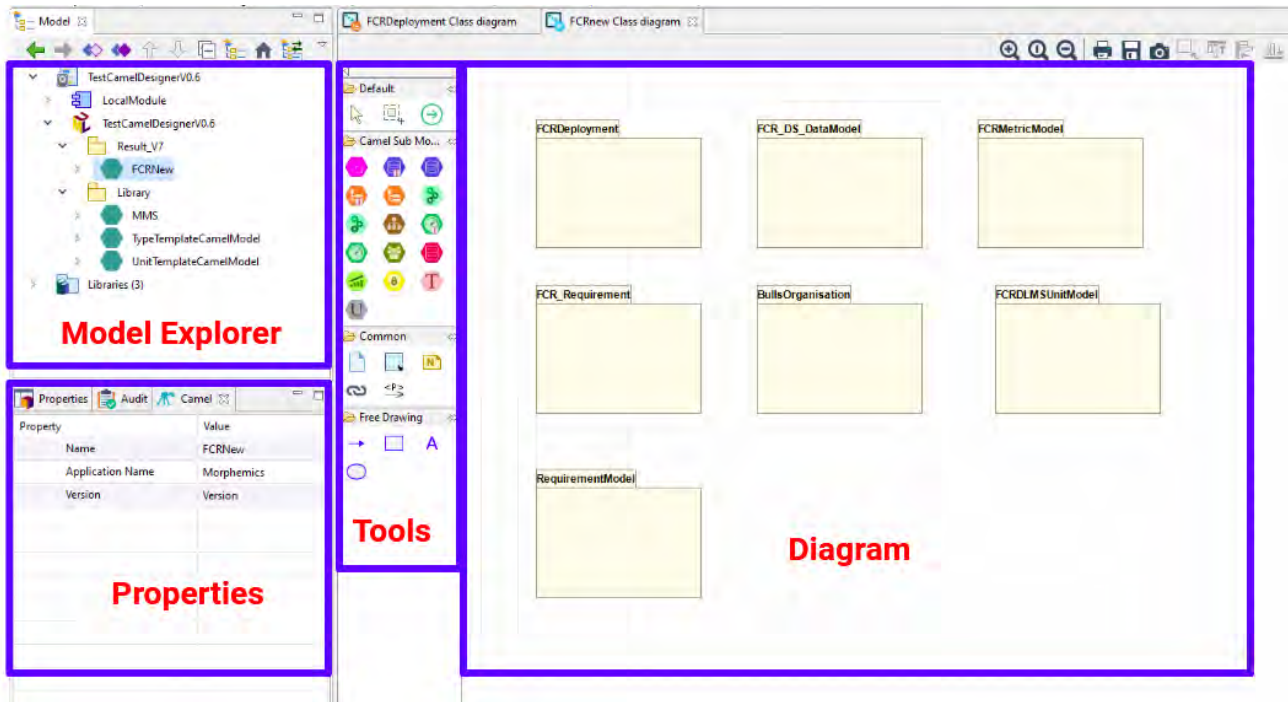


Figure 6: Modelio Environment for CAMEL Designer

5.3 CAMEL Models Management

Related Requirements

CM01 / CM02 / CM03 / CM04 / CM05 / CM06 / CM07 / CM10 / CM13 / CM14 / CM50 / CM51 / CM52

Typical Scenario

1. User creates a new CAMEL model;
2. A new CAMEL model is created and a first diagram is opened;
3. User uses CAMEL tools and commands to edit the CAMEL model;
4. User saves the model;
5. User Exports the CAMEL model in XMI format to be used by Morphemic Platform Web Client UI.

Interface Description

The first step of starting with CAMEL Designer is to create and open Modelio project and deploy CAMEL Designer module into the project. In order to create a new CAMEL Model, simply right click a package where you wish to create your CAMEL model and select the command “Create a new CAMEL Model” (see Figure 7). In a similar way, this applies to importing existing CAMEL Models files on CAMEL or XMI format. As a result, a new CAMEL Model is created, and a new diagram is opened to edit the newly created CAMEL Model.

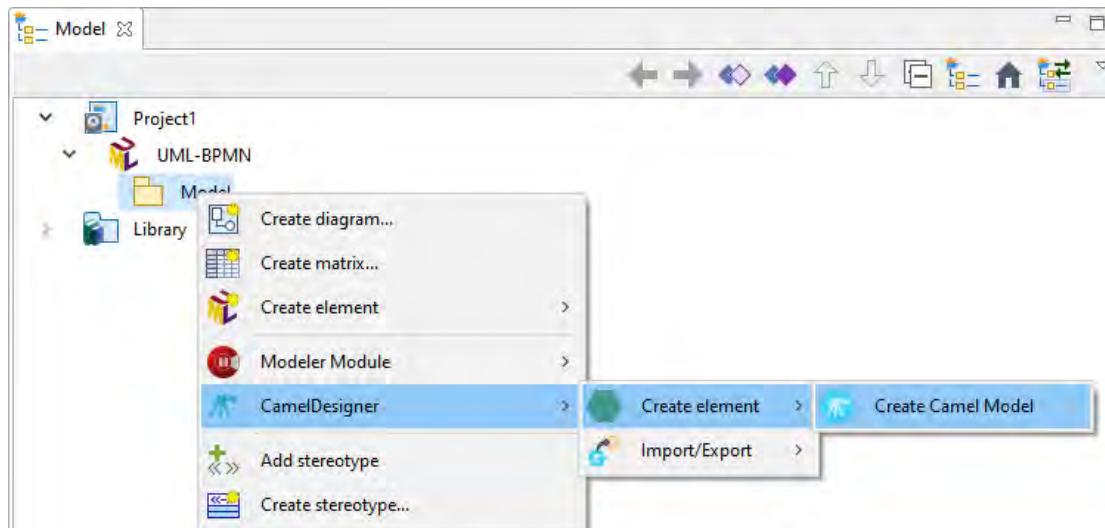


Figure 7: Create a CAMEL Model Command

After having successfully created a CAMEL model, you can then create sub-models on CAMEL model root by clicking on the tools displayed on the palette tools on the left of the CAMEL Model diagram. For example, Figure 8 shows the palette tools on the left where it is used to create the Requirement Model and the Type Model.

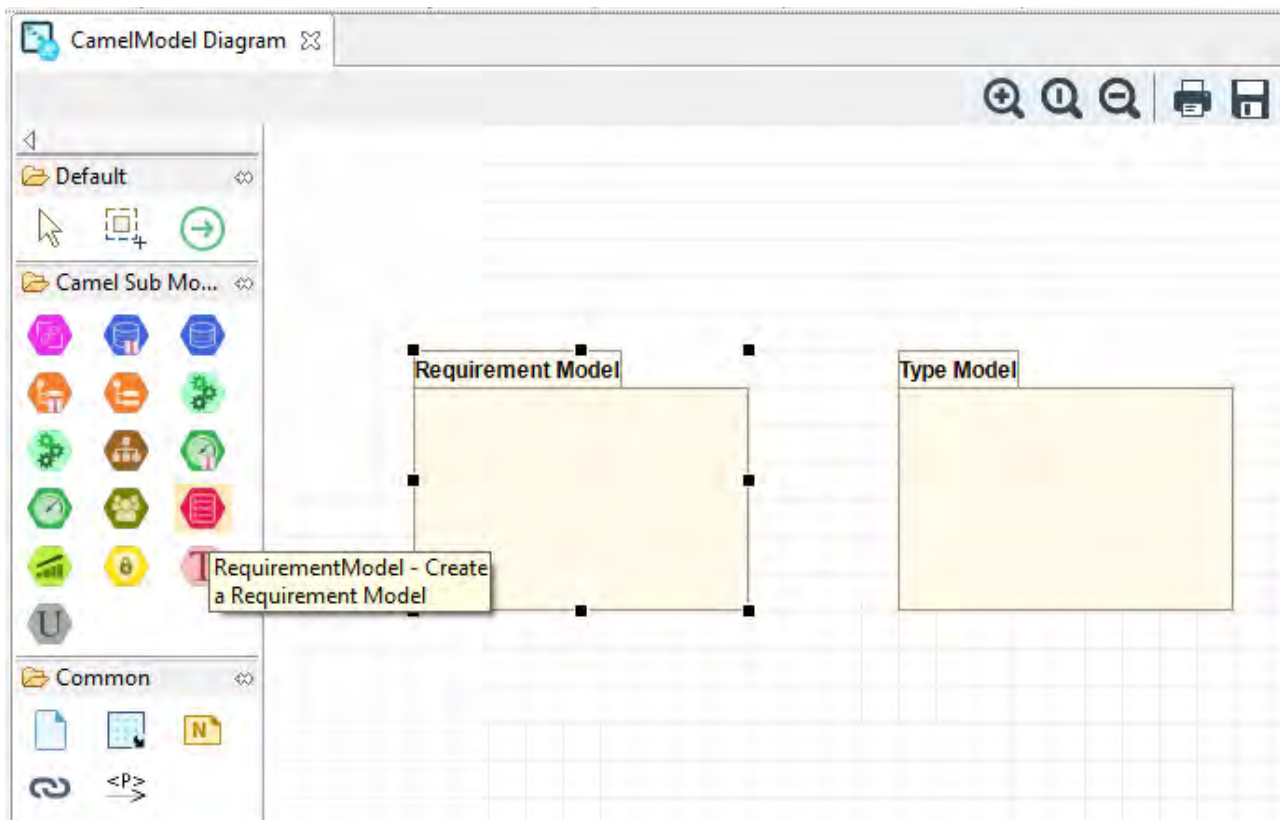


Figure 8: CAMEL Model Diagram Tools

5.4 CAMEL Deployment specification

Related Requirements

UC-UI1.1 / UC-UI1.XXX / UC-UI1.3 / UC-UI1.4 / UC-UI2.1 / CM10 / CM13 / CM14 / CM15 / CM16 / CM17 / CM18 / CM50 / CM51 / CM52

Typical Scenario

1. In an existing CAMEL model, the user creates a new deployment model and diagram;
2. By using the deployment CAMEL tools, the user can:
 - a. Create components to deploy;
 - b. Specify their properties (configuration scripts, communication ports, etc.);
 - c. Connect the components through their ports.

Interface Description

The first step of starting with CAMEL Designer is to create and open Modelio/Camel project with a CAMEL model containing a Deployment model and an associated Deployment diagram (cf. section 5.3). Once your CAMEL model is initialized, the user can create a Deployment model and diagram as depicted in Figure 8. By using the tools, as shown in Figure 6, the deployment model can be enriched with dedicated deployment concepts as depicted in Figure 9.

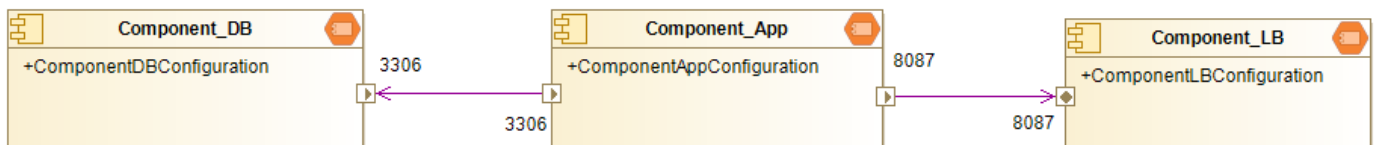


Figure 9: Simple CAMEL Component deployment

5.5 CAMEL Requirement specification

Related Requirements

CM10 / CM13 / CM14 / CM19 / CM20 / CM50 / CM51 / CM52 / AR06

Typical Scenario

1. In an existing CAMEL model, user creates a new requirement model and diagram;
2. By using the deployment CAMEL tools, the user can:
 - a. Create the dedicated requirements;
 - b. Specify their properties;
 - c. Regroup some of these requirements inside a dedicated requirement set;
 - d. Reference the requirement set from a component.

Interface Description

If the user successfully created an initial CAMEL model under Modelio (cf. section 5.3), the user can create a Requirement model and diagram as depicted in Figure 8. By using the tools provided by the CAMEL Requirement diagram, the user is able to specify the CAMEL concepts related to its Requirement aspects. These aspects could include the definition of several requirements, their grouping inside a “RequirementSet” and finally their reference by a software component for example, as depicted in Figure 10.

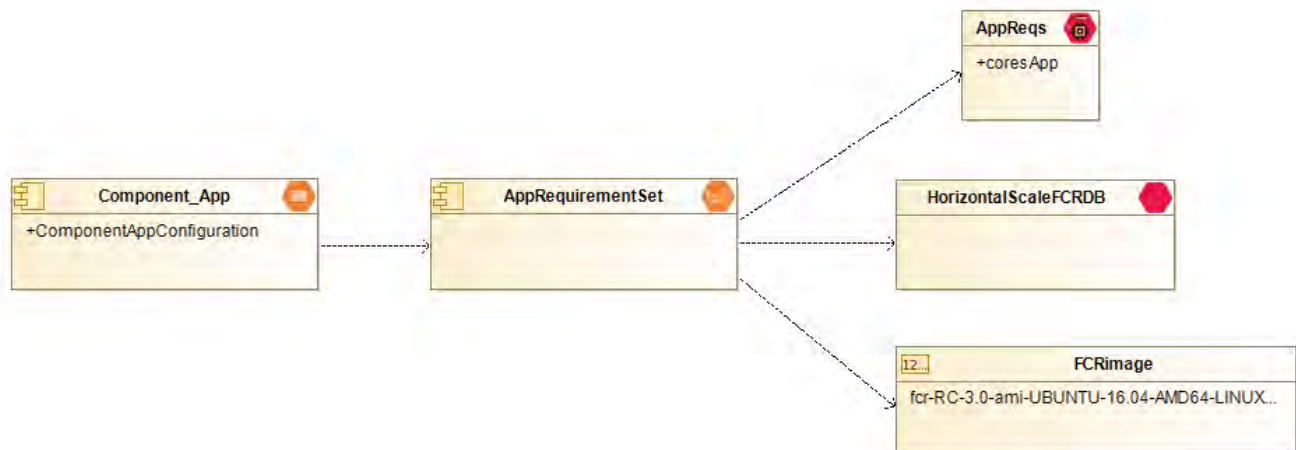


Figure 10: Three Requirements grouped inside a RequirementSet referenced by a Component

5.6 CAMEL Metric specification

Related Requirements

UC-UI1.7 / UC-UI2.5 / UC-UI3.2 CM03 / CM10 / CM13 / CM14 / CM23 / CM24 / CM25 / CM26 / CM27 / CM28 / CM29 / CM30 / CM31 / CM32 / CM33 / CM34 / CM35 / CM50 / CM51 / CM52 / CM61 / CM62 / AR06

Typical Scenario

1. In an existing CAMEL model, user creates new “Metric”, “Type”, and “Unit” models and diagrams;
2. By using the CAMEL diagram tools, the user can:
 - a. Create the necessary environment for an optimisation specification i.e. the needed CAMEL “Unit”, “Type”.
 - b. Define the different “Measurable Attribute” and “Template”, which their associated “Unit” and “Type” to collect the needed information.
 - c. Specify the “Metrics” and the “Constraints” associated to the previous “Metric” for optimisation purpose.

Interface Description

In order to create an Optimisation specification, several CAMEL concepts need to be specified. First of all, information needs to be defined in order to be collected. For example, Figure 11 depicts the definition of a “Template” named CardinalityTemplate. This latter is based on a cardinality “Measurable Attribute” and is a positive integer (ZeroToHundredInteger) representing the number of instances.

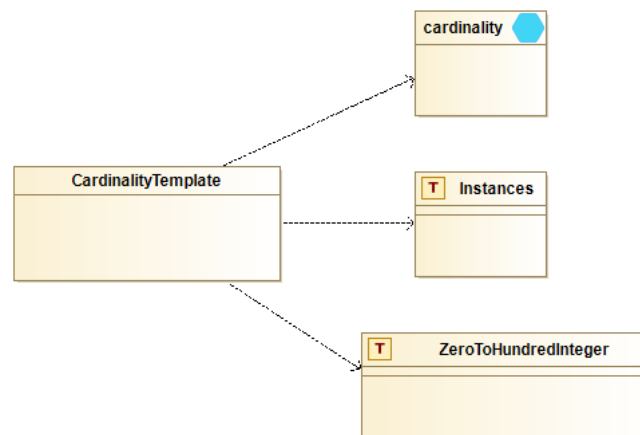


Figure 11: CardinalityTemplate “Template” specification

Then an AppActCardinality “Variable”, see Figure 12, extending the predefined CardinalityTemplate “Template”, is applied to the Component-App “Software Component” to count the number of instances of this particular software component.

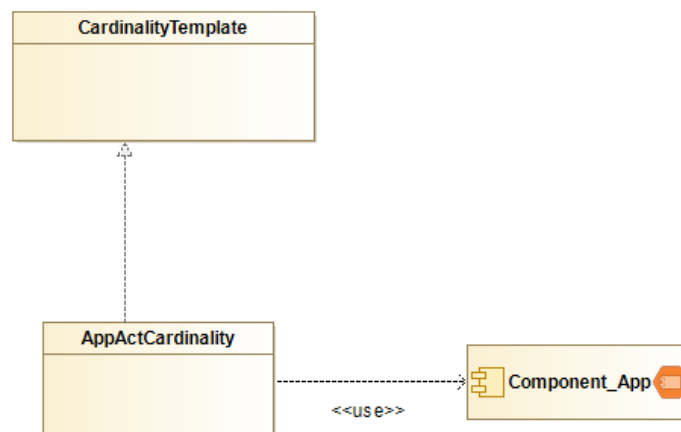


Figure 12: AppCardinality “Variable” definition

5.7 CAMEL Other specification

Related Requirements

UC-UI1.9 / CM10 / CM13 / CM14 / CM36 / CM37 / CM38 / CM39 / CM40 / CM41 / CM50 / CM51 / CM52

Typical Scenario

1. In an existing CAMEL model, user can create several other aspects of the CAMEL language by using dedicated diagram and tools.

Interface Description

By using the different diagrams provided by CAMEL designer, the user can specify, for example, “Locations” as depicted in Figure 13 or “Unit” and “Dimension” as shown in Figure 14.

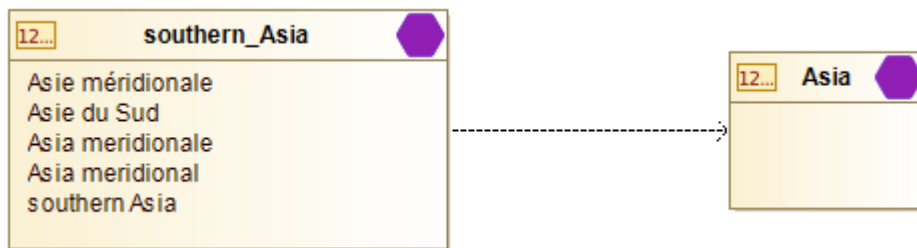


Figure 13: CAMEL Location specification

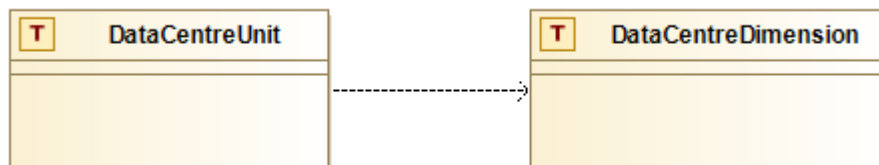


Figure 14: CAMEL Unit and Dimension definition

5.8 CAMEL Models Import/Export

Related Requirements

UC-UI1.2 / UC-UI3.1 / CM08 / CM09

Typical Scenario

1. User Exports the CAMEL model in XMI format to be used by Morpheic Platform Web Client UI.

Interface Description

After completing the design of the CAMEL Models, you can export the model in CAMEL or XMI format, by right clicking on the model explorer and use the command “Export CAMEL Model” (see Figure 15). Then a dedicated window is opened to help to specify the name and the location of the exported file.

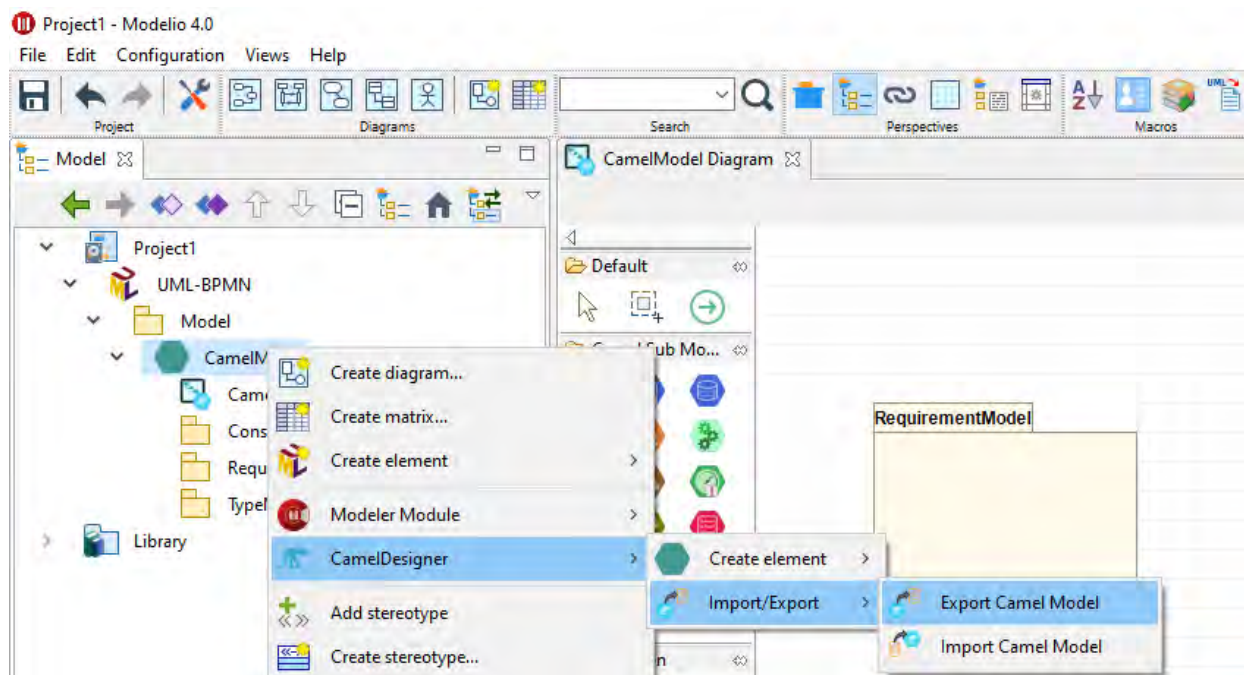


Figure 15: Export CAMEL Model

6 Specifications of Morphemic Web App Interface

6.1 Web User Interface Environment

The mock-up of the User Interface Web Client, depicted in Figure 16, shows the sections accessible through the left slide menu. Morphemic Platform can manage multiple applications; therefore, it is important to define unambiguously the context in which the user is currently working. For this reason, the top header of the Interface depicts the current selected application and the logged in user. The left side menu shows the accessible sections categorized into two groups: *Deployment and Settings*. The former group, along with the home page, which provides an overview and pertinent information about the platform and the selected application, includes the following sections:

- “Activity”, which refers to the history of activities of the user
- “Applications”, which provides mechanisms for managing multiple applications, their models and their runtime processes
- “Resources”, which aims at handling all kind of Cloud or user supplied node resources to be exploited for the deployment of the application
- “Monitoring”, which allows users to keep track of the application performance and to visualize different metrics.

The group called “Settings” includes:

- “Platform Settings”, which provides general settings to the platform.
- “Your Account”, which allows users to update their profile information
- “User Management”, accessible only by administrators, which allows to manage users access to the platform privileges.

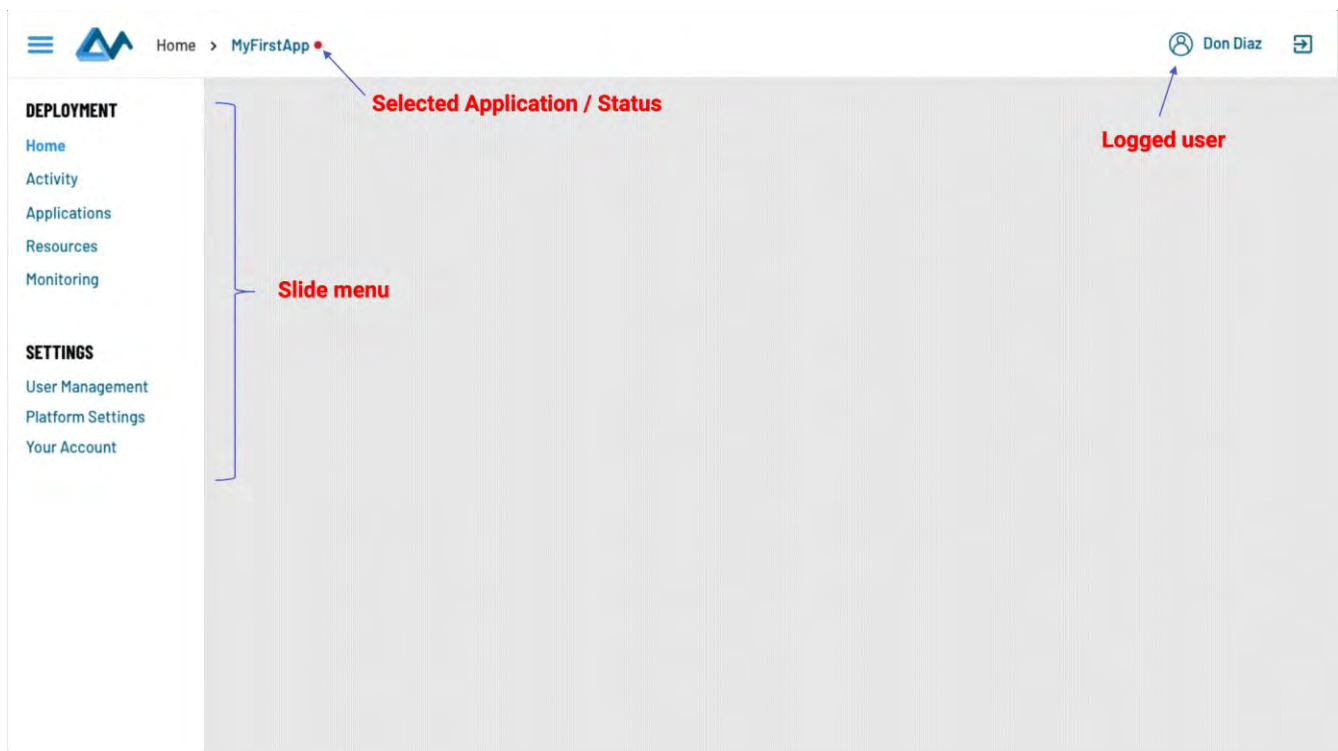


Figure 16: UI skeleton mock-up

6.2 Authentication and user management

Introduction

Morphemic UI allows several users to collaborate on the platform. There are different roles for users such as administrator and regular users where administrator have more privileges than regular users such as managing users' access and roles on the platform.

Related Requirements

UC-UI2.6 / UM01 / UM02 / UM03 / AC01 / AC02 / AC03

Typical Scenario

1. Administrator logs in;
2. Administrator checks list of users;
3. Administrator checks users' activities;
4. Administrator adds a new user by inviting him by email;
5. User clicks link to create his account by providing password;
6. User checks his activities.

Interface Description

After opening the User Interface page, the user is automatically redirected to Login page if he is not yet logged in (see Figure 17). After logging in, the user is redirected to the home page. The user can see the list of his activities accessible by the left sidebar menu as it is highlighted by Figure 18. He can sort or filter his activities by column. The activities represent the different types of events that are related to the user such as adding resources, adding applications, starting deployment, etc. Moreover, the user can access his own profile page where he can update his personal information, password and his profile picture (see Figure 19).

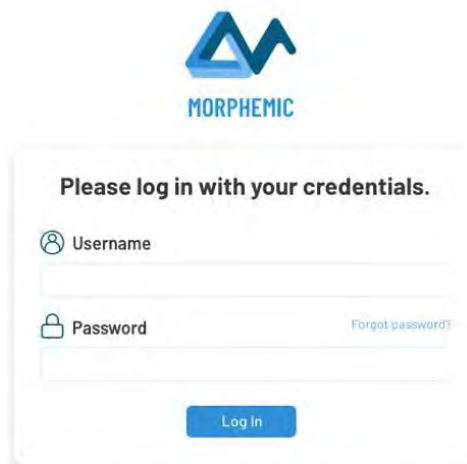
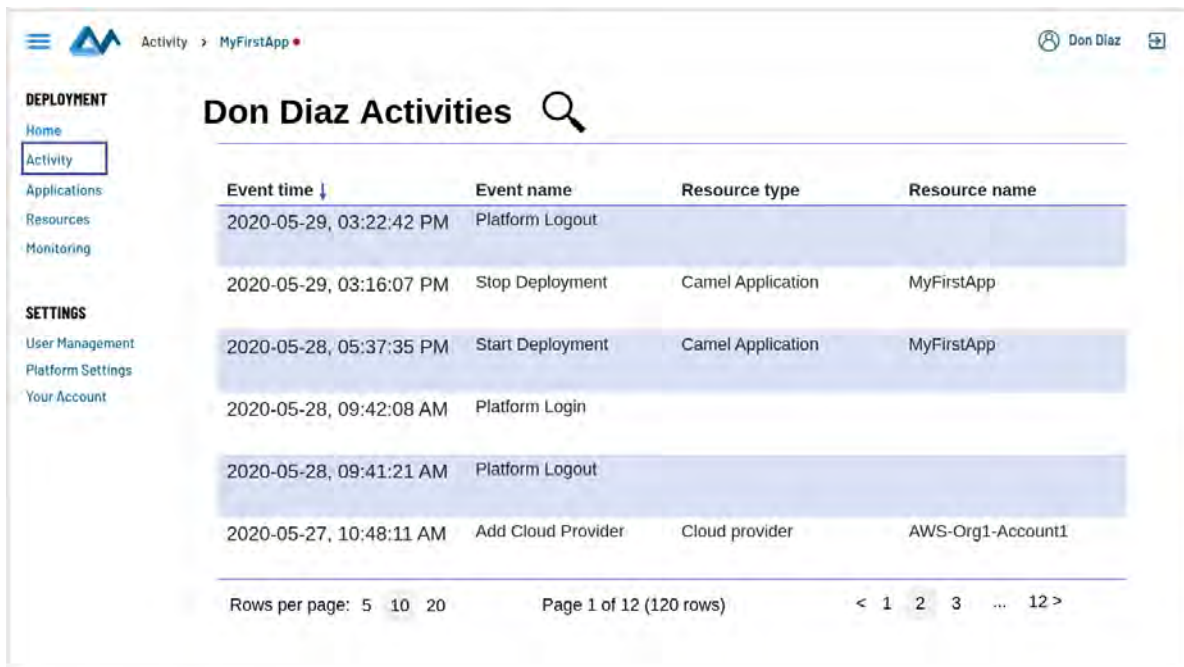


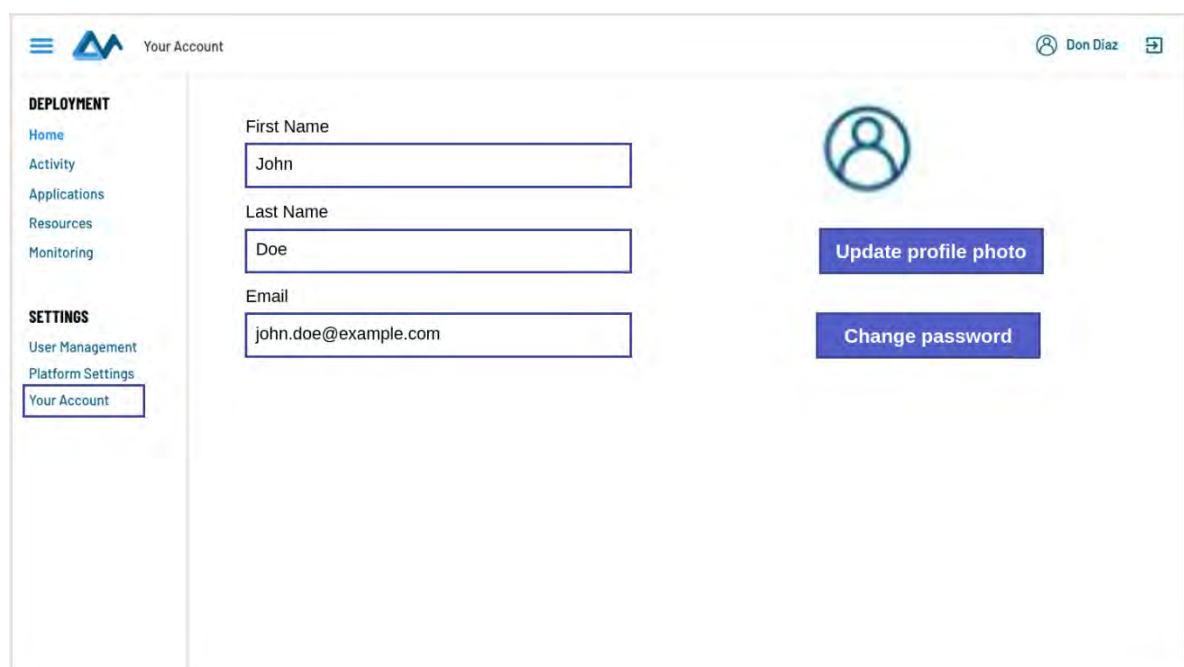
Figure 17: Login page mock-up



Event time ↓	Event name	Resource type	Resource name
2020-05-29, 03:22:42 PM	Platform Logout		
2020-05-29, 03:16:07 PM	Stop Deployment	Camel Application	MyFirstApp
2020-05-28, 05:37:35 PM	Start Deployment	Camel Application	MyFirstApp
2020-05-28, 09:42:08 AM	Platform Login		
2020-05-28, 09:41:21 AM	Platform Logout		
2020-05-27, 10:48:11 AM	Add Cloud Provider	Cloud provider	AWS-Org1-Account1

Rows per page: 5 10 20 Page 1 of 12 (120 rows) < 1 2 3 ... 12 >

Figure 18: User's activities page mock-up



First Name
John

Last Name
Doe

Email
john.doe@example.com

Update profile photo

Change password

Figure 19: User's profile mock-up

If the logged-in user is an administrator, he can access the “User Management” menu where he can see the accounts of all users, add new user, edit and delete existent users (see Figure 20). The administrator is also able to see the history of activities of all users, and to filter or sort the different types of activities in order to be able to see which activity has been performed by which user (see Figure 21).

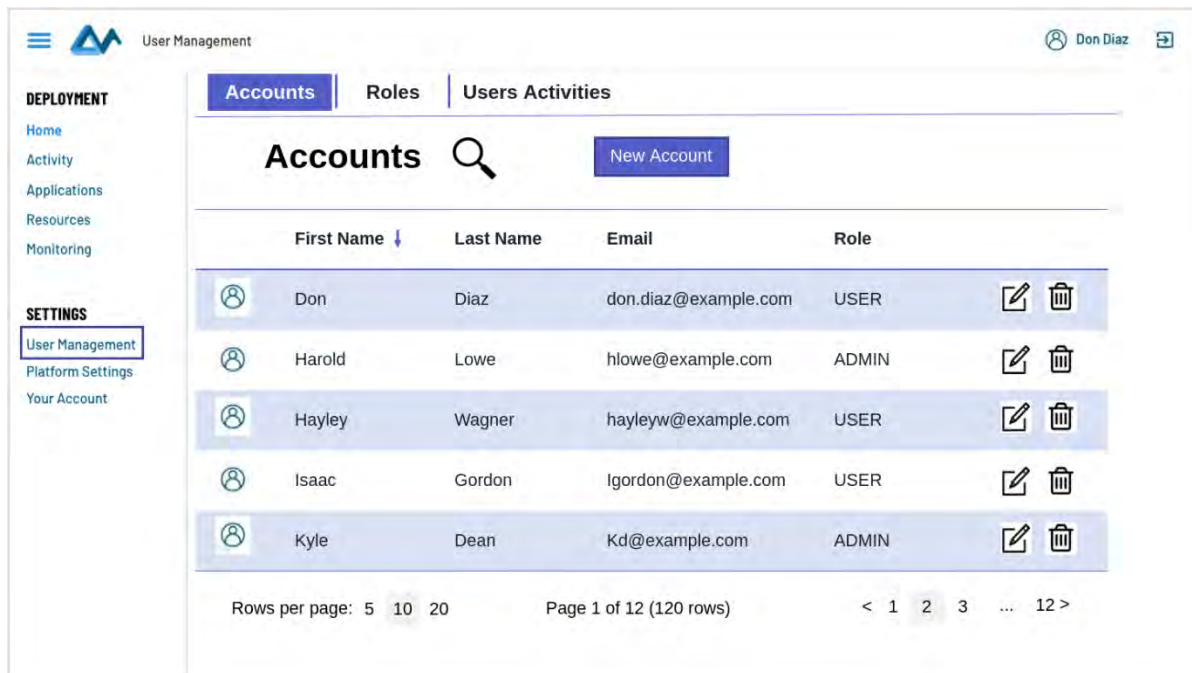


Figure 20: Administrator users' accounts page mock-up

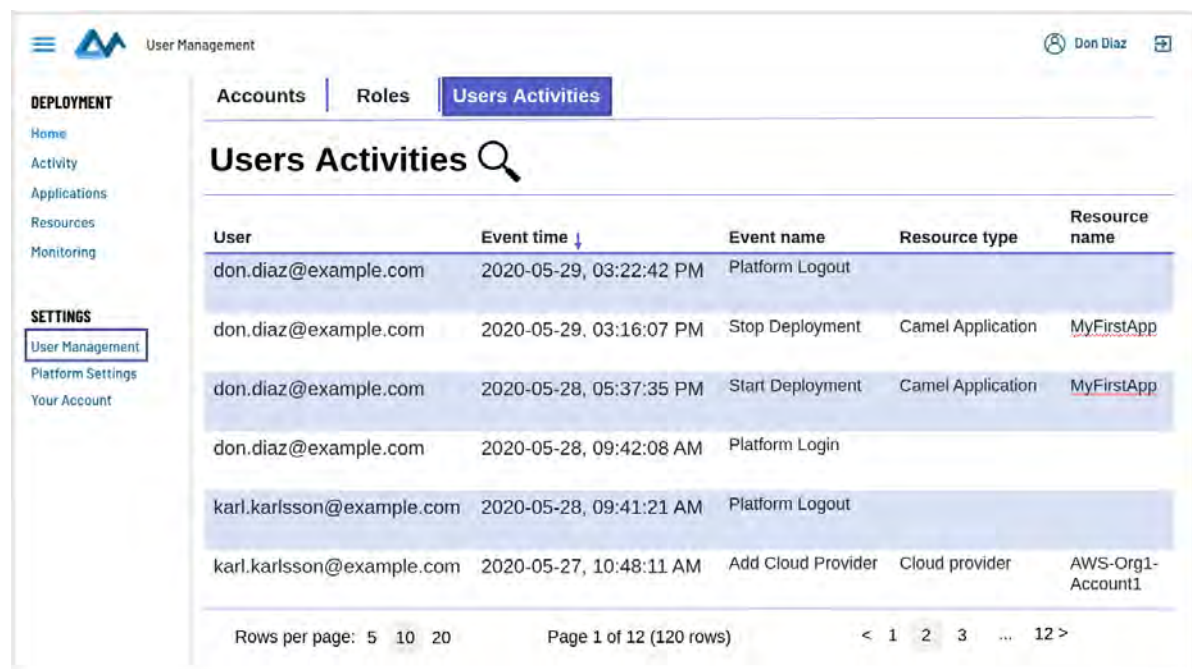


Figure 21: Administrator users' activities page mock-up

6.3 Applications and models management

This section is about the management of applications and models, it includes the management of CAMEL models, the management of multiple applications, the deploying and runtime process of applications.

6.3.1 Applications Models Management

Introduction

In Morpheic Platform, every Application is represented by a Cross-Cloud Model described in CAMEL Language. Thus, managing Applications in Morpheic Platform User Interface is equivalent to managing the CAMEL models of

these applications. For instance, to add a new application, a CAMEL file can be simply uploaded and it will represent the content and configuration of the application.

Related Requirements

UC-UI1.1 / UC-UI1.2 / UC-UI1.XXX / UC-UI1.3 / UC-UI1.4 / UC-UI1.7 / UC-UI1.8 / UC-UI1.9 / UC-UI2.7 / CM01 / CM02 / CM03 / CM04 / CM07 / CM08 / CM12 / AR01 / AR02 / AM01 / AM02

Typical Scenario

1. Precondition: User logs into Morphemic Platform;
2. UI shows current selected Application;
3. User adds a new Application by uploading a CAMEL model serialized as XMI file;
4. User checks the lists of registered Applications CAMEL models;
5. User switch to another registered Application;
6. User deletes a registered Application.

Interface Description

When the user clicks on “Applications” menu, he is presented with 4 panels that reflect the management cycle of applications in Morphemic Platform (see Figure 22). The Figure illustrates the “All Applications” panel where the user can see the current selected application along with all registered applications which the user can switch to. The current selected application is also highlighted in the header as always to keep the user informed about the current context of deployment. In order to create a new application, the user clicks on “New Application” button and a pop-up window appears for uploading a CAMEL File in XMI format (see Figure 23).

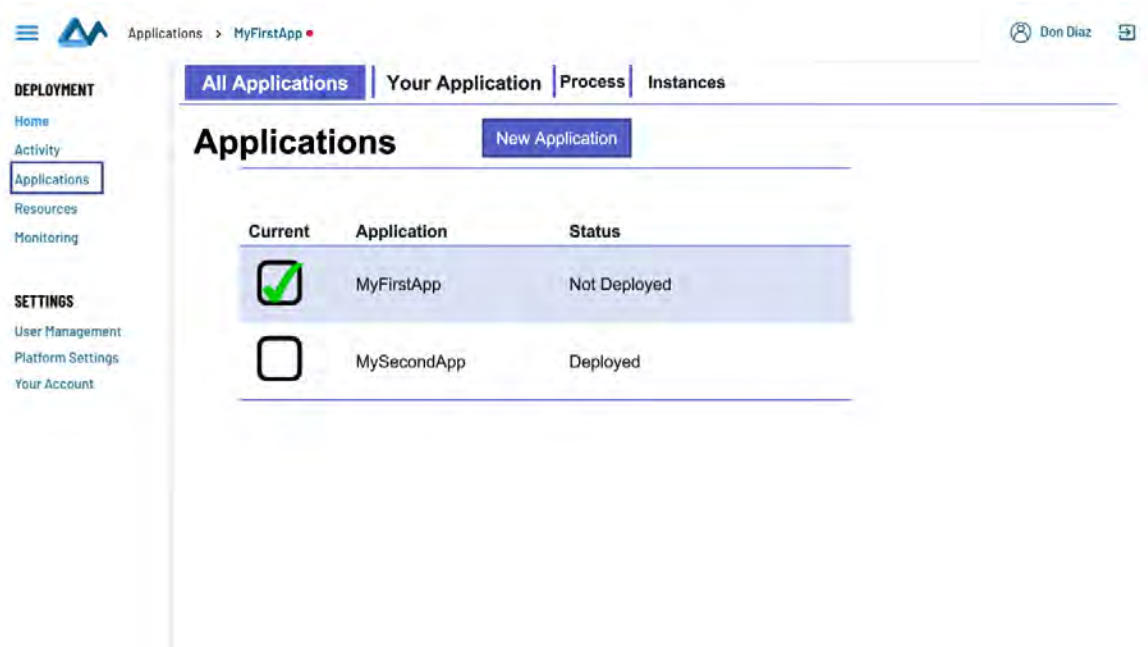


Figure 22: “All Applications” page

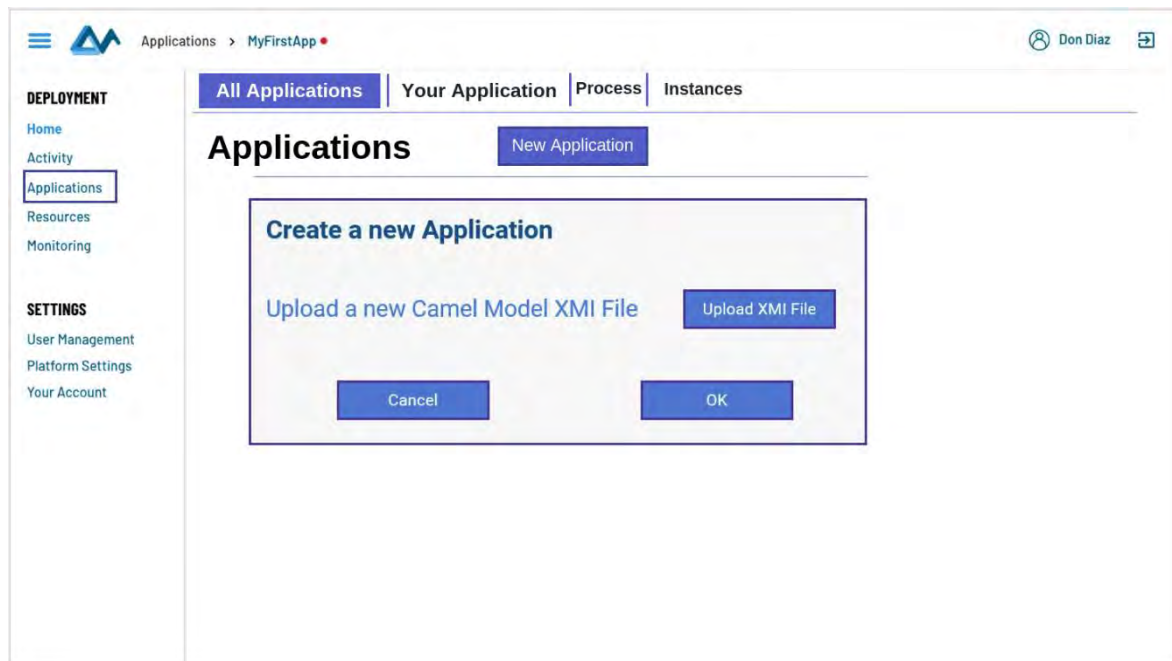


Figure 23: Create a new Application pop-up window

The next panel in the “Applications” section is “Your Application” panel (see Figure 24). In this panel, an overview of the CAMEL model associated to the selected application is depicted. Moreover, it is possible for the user to refine his model by selecting a set of “Optimization Policies” with their respected weights in order to ensure the priorities of optimizing the deployment of applications (see Figure 25). Finally, the user should specify which resources can be used for deployment from the list of the registered resources (see Figure 26).

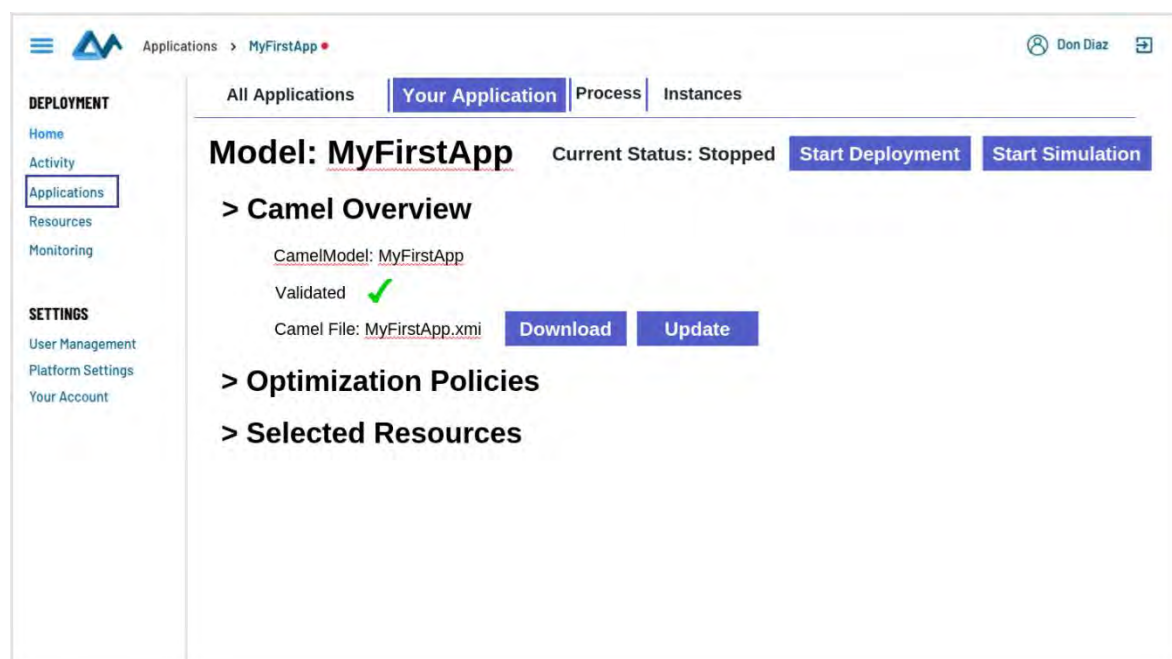


Figure 24: Your Application CAMEL Overview

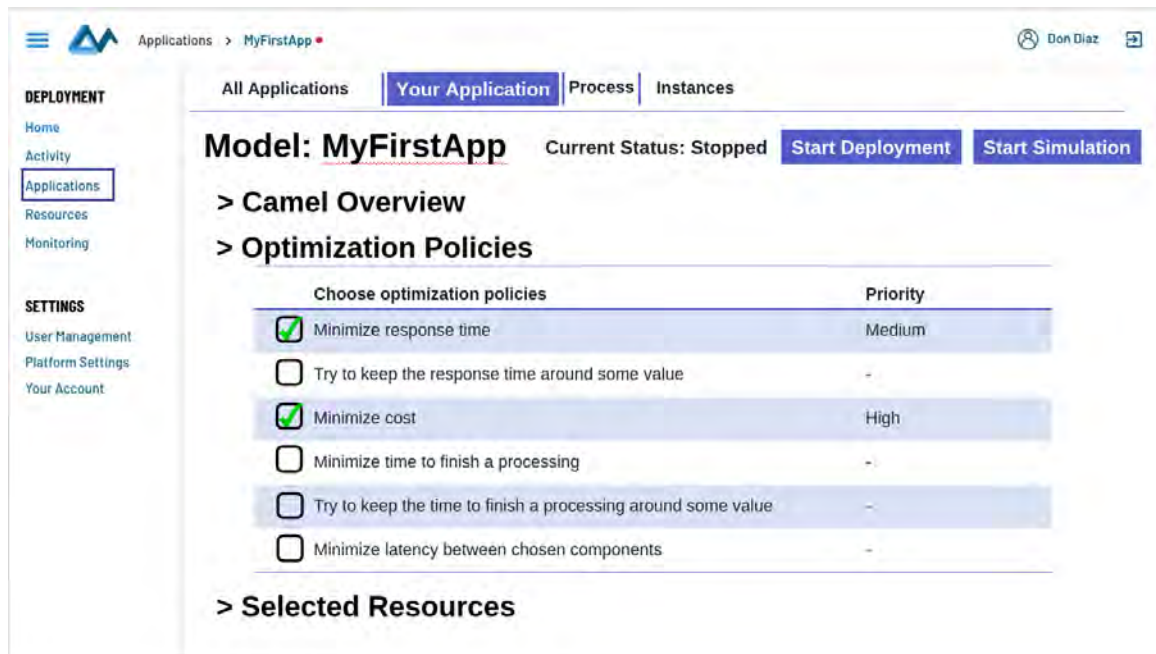


Figure 25: Your Application Optimization Policies

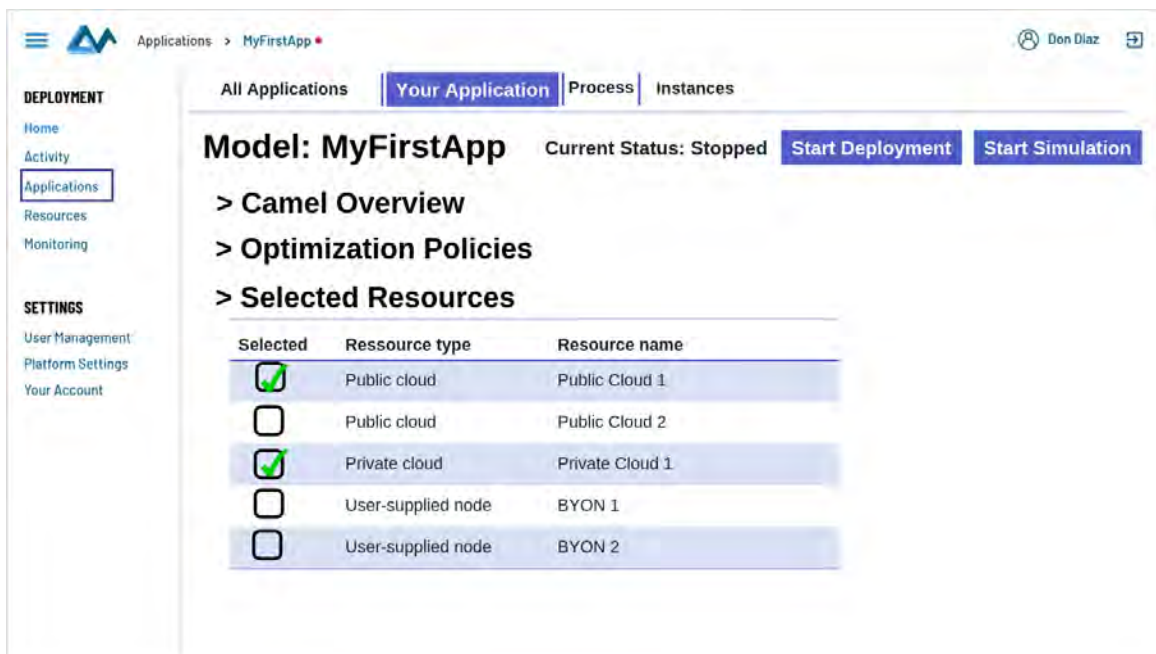


Figure 26: Your Application Selected Resources

6.3.2 Application Deployment and Process View

Introduction

The application deployment process view allows the user to have an idea about the current phase of the deployment and to have information related to the current phase.

Related Requirements

UC-UI2.4 / AR03 / AR05 / AR07 / AR08 / AR09 / AR10 / AR11

Typical Scenario

1. Precondition: User had created an application;
2. User starts deployment;

3. User checks the process view of deployment;
4. User checks information of each process view phase;
5. User checks the list of deployed instances for the running application;
6. User connect to a deployed instance with SSH connection.

Interface Description

After having chosen his application model, his optimization policies and his selected resources, the user can click on “Start Deployment button (see Figure 24, Figure 25, Figure 26). The user is then redirected to the process view panel where he can follow the different phases of the deployment of the application from fetching offers, to reasoning to deployment and reconfiguration as it is illustrated by Figure 27. Each process phase has underlying information such as the total number of offers in the “Fetching offers” phase. Furthermore, the user can check the list of deployed instances in the “Instances” panel depicted in Figure 28, where he can see the relevant details about the instances such as the public IP address. In addition, it is possible for the user to connect with SSH to the deployed instances.

The screenshot shows the 'Application Process Panel' with the 'Process' tab selected. The panel is divided into two main sections: '> Current Process' and '> Processes History'. The 'Current Process' section displays four phases of the deployment process:

- Fetching offers:** Shows the current total number of offers as 299.
- Generating Constraint Problem:** Displays variables such as WorkerCardinality (from 1 to 10), WorkerCores (4), and provider_Component (0).
- Reasoning:** Shows the solution variables, including WorkerCardinality (1), WorkerCores (4), and provider_Component (0).
- Deploying:** Displays the deployment difference, including components to create, delete, or remain.

Figure 27: Application Process Panel

The screenshot shows the 'Application Deployed Instances Panel' with the 'Instances' tab selected. The panel displays a table of virtual machines under the heading 'Virtual Machines'.

No.	Name	Id	State	Public IP	Private IP	Provider	Location	Diagnostic	SSH connection
1	Instance-1	sdsc-sdsc-sdsc-sdsc	RUNNING	18.203.45.152	172.31.15.190	aws	Dublin	Aws	SSH
2	Instance-2	aszdq-g-esdq-gs	RUNNING	18.213.45.152	172.31.15.192	aws	Dublin	Aws	SSH

Below the table, there is a pagination control showing 'Page 1 of 12 (120 rows)' and a 'Rows per page' dropdown set to 10.

Figure 28: Application Deployed Instances Panel

6.4 Resources management

This subsection details how the user interface implements the requirements in compute resource management. The dedicated interface is composed of four different panels:

- A panel exposing the provisioned instances in the infrastructure managing them.
- A panel controlling the configuration of public and private clouds.
- A panel configuring the infrastructure supplied by the user.
- A panel wrapping up the specifications of selected instances.

The instances exposing panel is located on the screen left and is always shown. The last three panels are affected on the screen right and are shown only when they are invoked from the left panel.

6.4.1 Inspecting the provisioned instances

Related Requirements

CR02 / CR09 / CR11 / UC-UI1.XXX

Typical Scenario

1. Precondition: User logs in to Morphemic Platform;
2. UI shows the configured cloud service providers registration;
3. UI lists the instances on each registered cloud service providers;
4. UI lists the instances manually configured by the user (user-supplied nodes).

Interface Description

The user is able to investigate the deployed instance at a glance through the tree-list located in the left of the interface. The first level of the arborescence lists the cloud infrastructures set up by the user and a dedicated item grouping instances directly brought by the user. The provisioned instances are listed on the second level of the arborescence under the cloud they are allocated on. They can also be under the "User-supplied item" if they are supplied by the user. A set of icons permit the user to acknowledge the nature and status of the instances by identifying (i) provisioned and active nodes, (ii) provisioned but unused nodes, and (iii) active nodes providing software acceleration capabilities.

Two buttons allowing the user to declare new resources are located below the tree-list. The first button is intended for the connection to a cloud infrastructure. This includes both public and private cloud providers. The second button enacts the registration of nodes provided by the user. By this mean, the user is able to enlist on-premise and edge resources. The Figure 29 provides an illustration of this panel with a mock-up.

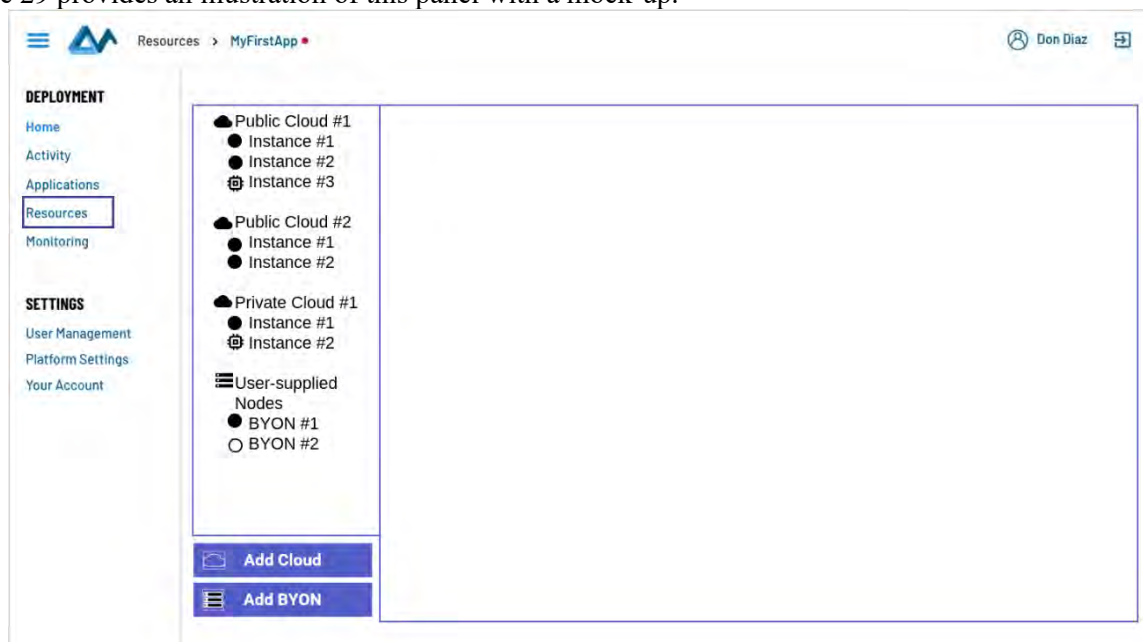


Figure 29: Provisioned Instances Panel

6.4.2 Adding, editing or removing the connection to a cloud provider

Related Requirements

CR01 / CR03 / CR04 / CR05 / CR06 / CR07 / UC-UI1.XXX / UC-UI1.8 / UC-UI2.7

Typical Scenario

1. Precondition: User logs in to Morphemic Platform;
2. User adds a new cloud provider;
3. User edits a cloud provider registration;
4. User remove a cloud provider registration;
5. User checks the list of cloud providers;
6. User checks the list of regions, VM types, hardware, price, etc.
7. User checks the images available with a cloud service provider registration.

Interface Description

The manipulation of the connection configuration to a cloud service provider is achieved through a dedicated panel containing a form to be valued with the configuration details, a widget to browse the available VM types and offers, a widget listing available images and two action buttons. The first button enables to save the provided configuration value, the second permits the removal of the registered connection profile.

Therefore, to register the subscription to a cloud provider, the user must select the dedicated button below the tree-list on the left panel of the interface. The panel containing the configuration for the connection will come empty. The user will be able to save the new configuration by clicking on the dedicated button in the interface. To edit a configuration, the user has to select the name of the cloud whose connection is to be edited from the arborescence in the left panel. The configuration form will come filled with registered values. The user will be able to edit them, before saving them, by using the dedicated button. To remove a cloud configuration, the user is expected to act similarly to the latter case, but then use the removal button in the panel instead of the saving one.

The form contains fields containing basic information about the connection to the cloud provider. The following parameters are to be completed:

- The name of the connection configuration;
- The type of connector to use with respect to the cloud service to interact with;
- The API public key to identify the owner of the cloud account to use;
- The API private key to authenticate the owner of this account;
- An “other parameters” field to enable the user to supply optional or connector-specific connection parameters.

Once these values are supplied, the user will be able to use the save button. Moreover, the browser of cloud offers will be usable: The user will be in capacity to search pattern related to cloud offers and to examine the result of its query in the table below. Each row corresponds to a cloud offers that can be used with the configured cloud offers. The columns provide criteria for the examinations:

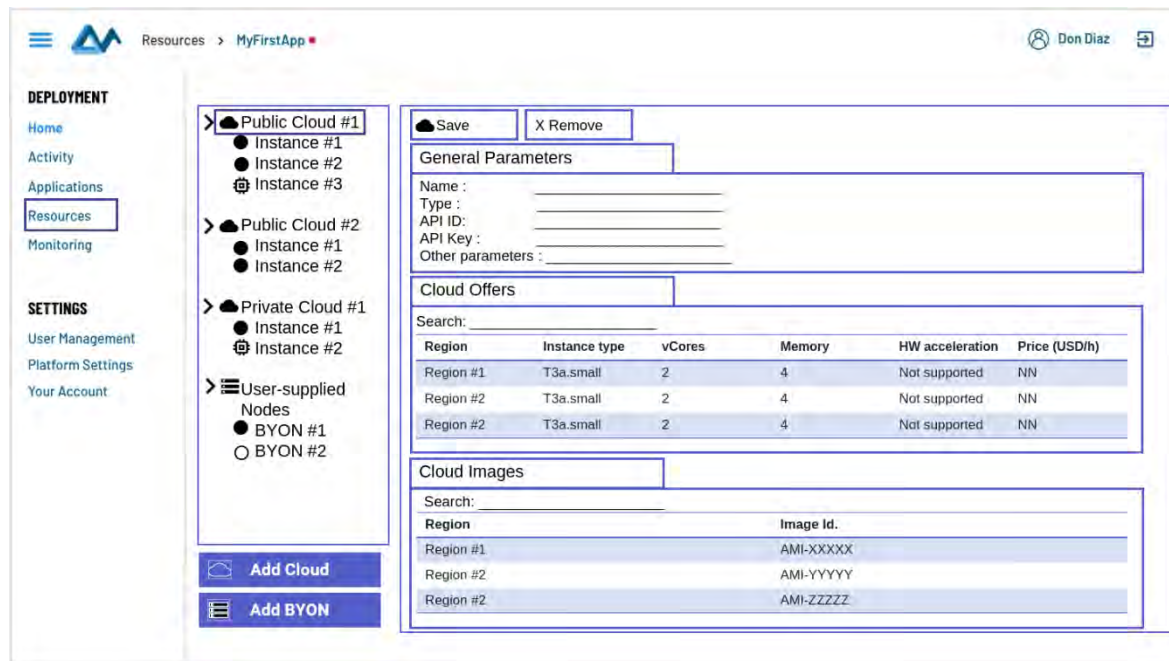
- The region from the cloud service provider;
- The name of the instance type associated to the offer;
- The number of CPUs of the instance type;
- The amount of memory from the instance type;
- The support for hardware accelerator;
- The price.

This browser consequently enacts the user to browse the available regions of a cloud provider.

Similarly, the cloud images browser will also be enabled after to configuration of the cloud providers settings. The user will be able to search and browse the images associated with its account. The browser provides two columns:

1. The region of the cloud service provider hosting the image;
2. The reference to the image.

The Figure 30 illustrates this panel with a mock-up.



DEPLOYMENT

- Home
- Activity
- Applications
- Resources**
- Monitoring

SETTINGS

- User Management
- Platform Settings
- Your Account

Public Cloud #1

- Instance #1
- Instance #2
- ⊗ Instance #3

Public Cloud #2

- Instance #1
- Instance #2

Private Cloud #1

- Instance #1
- ⊗ Instance #2

User-supplied Nodes

- BYON #1
- BYON #2

Save **X Remove**

General Parameters

Name : _____

Type : _____

API ID: _____

API Key : _____

Other parameters : _____

Cloud Offers

Search: _____

Region	Instance type	vCores	Memory	HW acceleration	Price (USD/h)
Region #1	T3a.small	2	4	Not supported	NN
Region #2	T3a.small	2	4	Not supported	NN
Region #2	T3a.small	2	4	Not supported	NN

Cloud Images

Search: _____

Region	Image Id.
Region #1	AMI-XXXXX
Region #2	AMI-YYYYY
Region #2	AMI-ZZZZZ

Add Cloud **Add BYON**

Figure 30: Configuration Panel for Cloud Providers

6.4.3 Adding, editing or removing an instance manually configured (user-supplied node)

Related Requirements

CR08 / CR10 / CR12 / CR13 / UC-UI1.XXX/ UC-UI1.9/ UC-UI2.7

Typical Scenario

1. Precondition: User logs in to Morphemic Platform;
2. User adds a new node under his control;
3. User edits a node under his control;
4. User remove a node under his control.

Interface Description

The management of an infrastructure resource under the control of the user is made into a dedicated panel. From a similar approach to the management of connection to a cloud provider, this panel comes with a form containing the settings to interact with the supplied resource, and two action buttons to save and remove a configuration. The first button enables the saving the provided configuration value, the second permits the removal of the selected resource.

The user can consequently register a new node by selecting the adequate button below the tree-list in the left panel. The panel for user-supplied node configuration will take place on the right side of the screen. The configuration form will be empty. The user will be able to validate its changes by clicking on the save action button. Similarly, to edit an existing registration of a user-supplied node, the user must select the registration located in the dedicated arborescence, and the panel will appear with the configuration form already filled with the current configuration values. The user will then be able to enforce its modification by clicking on the save button. To remove the registration of user-supplied node, the user has to proceed similarly to the edition case, but eventually select the removal action button.

The form contains information to identify and connect to the user-supplied node. The parameter to be configured are the following:

- The name of the resources;
- An URI to connect the resource;
- The selection of a connector to interact with the resource runtime;
- The user field to identify against the resource;
- The private key to authenticate against the resource.

The Figure 31 represents this panel.

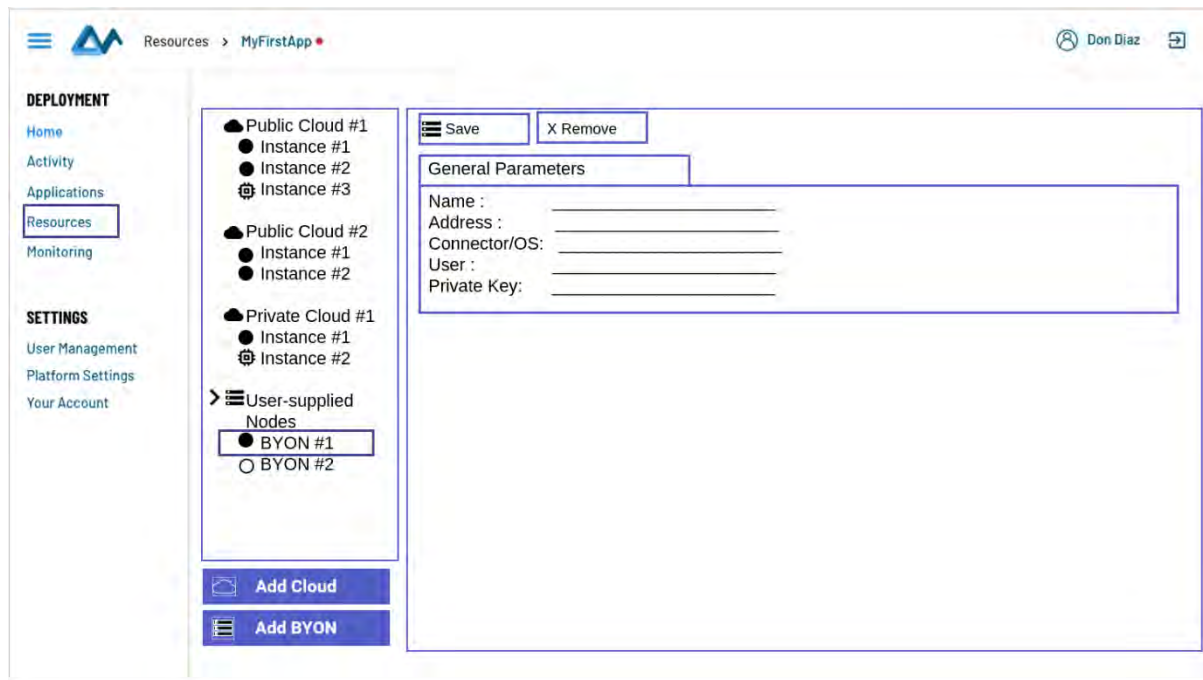


Figure 31: Configuration Panel for User-Supplied Nodes

6.4.4 Instances inspection

Related Requirements

CR09 / CR11

Typical Scenario

1. Precondition: User logs in to Morphemic Platform;
2. The user inspects the nature of a cloud instance.

Interface Description

A dedicated panel permits the inspection of the instances provisioned in the cloud. This panel is only informative and does not contain any action button. Two tables consolidate the information regarding the instance's status.

To inspect a cloud instance status, the user must select it from the tree-list on the left panel. The inspection panel will be loaded on the right side of the screen. The information will be appeared in the related tables.

The panel contains two tables. The first is limited to the information regarding the context of the instance:

- The name of the cloud connection in-use;
- The name of the cloud provider, according to the cloud connector in-use;
- The location of the instance, with respect to the cloud region.

The second panel contains the information of the specification of the instance itself:

- The name of the used VM profile;
- The name of the disk image;
- The number of allocated virtual CPUs;
- The allocated memory;
- The available storage;
- The support of hardware acceleration by the VM type;
- The price of the instance.

The Figure 32 illustrates this panel.

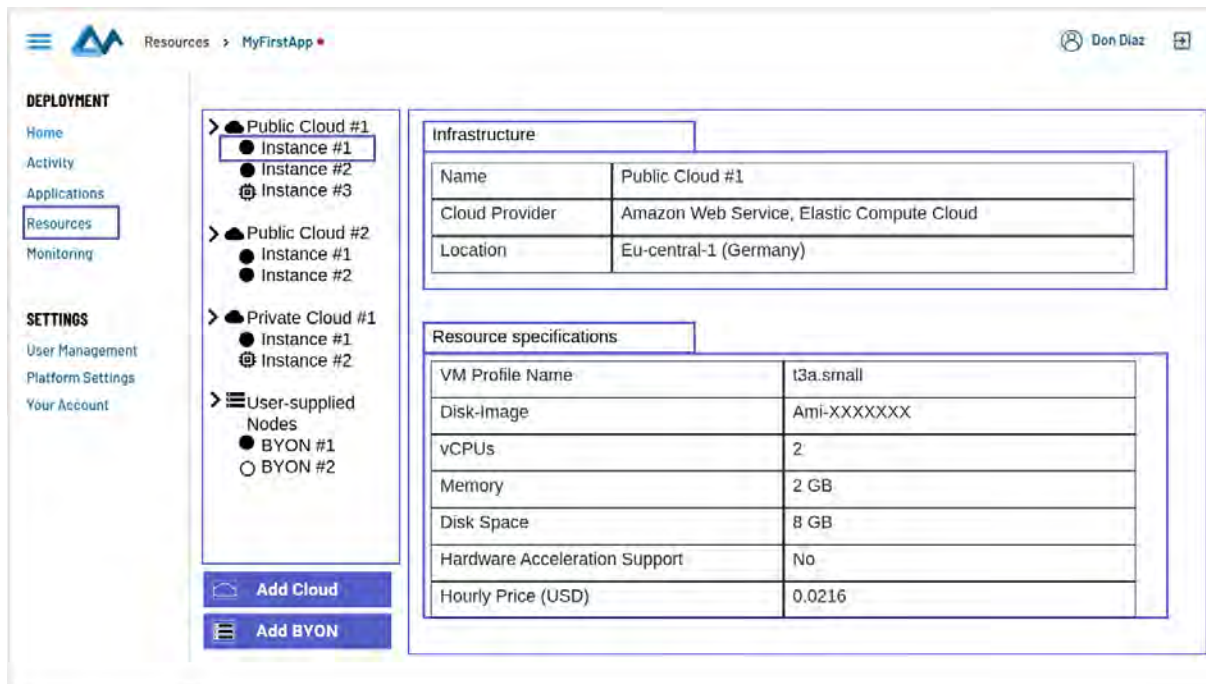


Figure 32: Instance Inspection Panel

6.5 Monitoring

This subsection presents the implementation of the requirements from the user interface related to the visualization of monitoring metrics. The interface is composed of a single screen, which allows the inspection of the available monitoring metrics – that is everything that has been stated in CAMEL and that can be part of an SLO expression. Further, the interface enables the user to select the monitoring metrics which will be used to setup a Grafana dashboard. Monitoring metrics will be collected and processed by EMS.

Related Requirements

UC-UI1.5 / UC-UI1.6 / UC 2.2 / UC 2.3 / M01 / M02 / M03

Typical Scenario

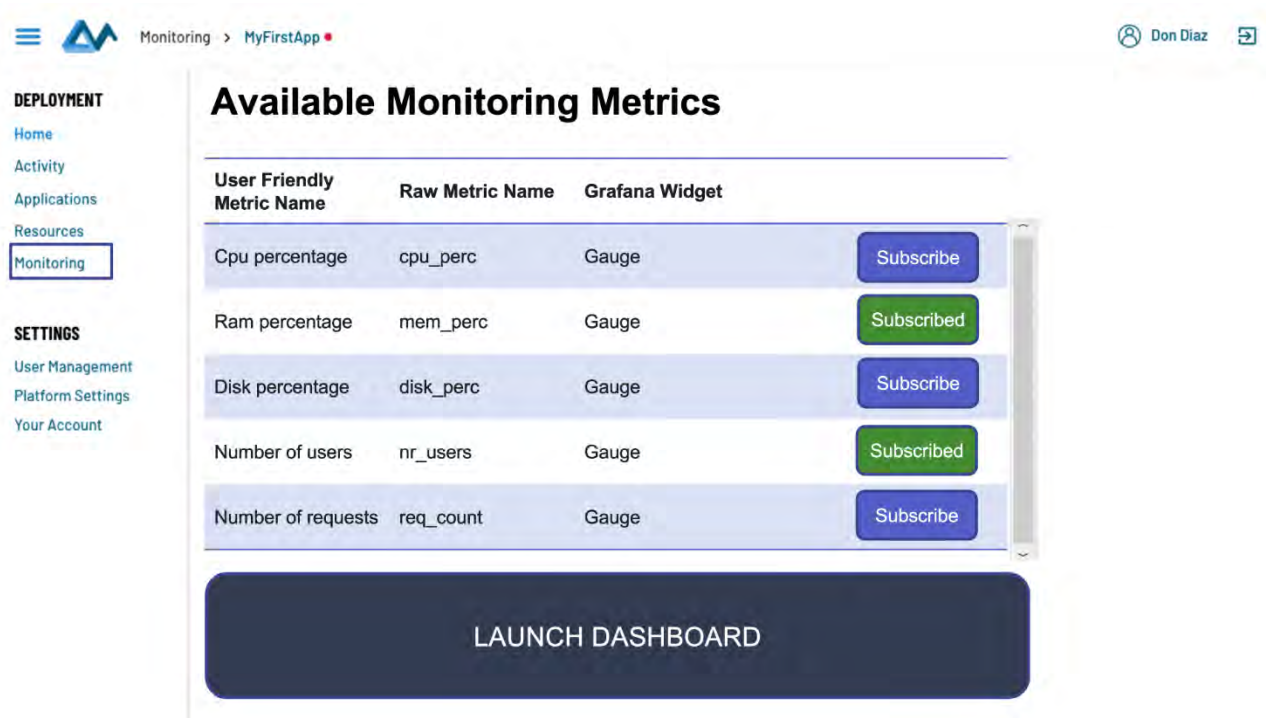
1. Precondition: User logged in and application deployed;
2. User clicks on the 'Monitoring' tab to view information related to the usage of Monitoring metrics;
3. User explores the list of available metrics;
4. User selects metrics to be visualized;
5. User launches dashboard;
6. User redirected to Grafana dashboard;
7. User checks the metrics views.

Interface Description

The Monitoring interface is responsible for the presentation of monitoring metrics, which can be used to create a Grafana dashboard. While the creation of a proper dashboard is expected to be undertaken by each Morpemic adopter, metrics which should be included in it should be selected from a list of available metrics. The panel contains a list of metrics which are retrieved by the User Interface and presented in a list. Each metric has its raw metric name – the metric name which is retrieved by the UI – a user-friendly name (which defaults to the raw metric name), and the type of the Grafana widget which should be used for the depiction of the metric.

Once a user has identified the desired metrics, the ‘Subscribe’ button can be clicked for the respective items, whose colour is then changed to reflect their updated status. Subsequently, the ‘Launch Dashboard’ button can be clicked to initiate the creation of a Grafana dashboard featuring the selected metrics.

The Figure 33 illustrates this panel.



The screenshot shows a web application interface for monitoring metrics. The top navigation bar includes a menu icon, a logo, the text 'Monitoring > MyFirstApp', and a user profile 'Don Diaz'. The left sidebar has two sections: 'DEPLOYMENT' with links for Home, Activity, Applications, Resources, and Monitoring (which is highlighted); and 'SETTINGS' with links for User Management, Platform Settings, and Your Account. The main content area is titled 'Available Monitoring Metrics' and contains a table with the following data:

User Friendly Metric Name	Raw Metric Name	Grafana Widget	
Cpu percentage	cpu_perc	Gauge	Subscribe
Ram percentage	mem_perc	Gauge	Subscribed
Disk percentage	disk_perc	Gauge	Subscribe
Number of users	nr_users	Gauge	Subscribed
Number of requests	req_count	Gauge	Subscribe

Below the table is a large blue button labeled 'LAUNCH DASHBOARD'.

Figure 33: Available Monitoring Metrics

7 Final remarks and future plan

Most of the requirements regarding the user interface of Morphemic Platform have been satisfied. However, few requirements are still partially satisfied, or need clarification, or have been postponed as possible improvements or discarded due to their complexity. Reasons for this are detailed below:

- UC-UI3.6 Cost estimation: giving a cost estimation to the user can be done by starting simulation instead of starting deployment (see Figure 18). This allows user to see the cost estimation during the execution of the simulation.
- UC-UI3.7 respect deadline: similar to UC-UI3.6 the simulation is used to estimate the execution time.
- CM11 Integrated web CAMEL Modeler: this requirement is discarded due to its complexity and its cost of developing a web modeler. It is simply replaced by Modelio modelling environment, and the CAMEL model is then uploaded to Morphemic Platform in XMI format.
- AR04: Pause, stop, resume the general adaptation in each phase of the process. This requirement is not fully satisfied because it is only possible to start the deployment at beginning and stop after the end of deployment process. Based on Melodic deployment process, the deployment cycle cannot be interrupted after the reasoning phase for example because it is implied that the user has agreed to the whole deployment process. Furthermore, if the user wants to check the solution of the reasoning phase without actually deploying any resources, he can use the simulation of the deployment which provides insights about the simulated deployment status without deploying any resources.
- Q01 Easy to use without reading extensive documentation: this requirement is not quantified to determine the degree of ease of use but it can be argued that it is easy to use the conceived UI because it has few commands and the commands are self-explanatory and don't require much explanation or documentation.
- Q02 Keep user always aware about his current viewed section: This requirement is satisfied because the title of the current page and the selected menu are always visible to the user.
- Q03 Accessibility to most features without deep navigation: most of the commands available in the UI are accessible within few clicks to the user.
- Q04 Keep user always aware about the backend process during runtime: this requirement is satisfied by the process view tab in the “Applications” menu.
- Q05 Web UI compatibility with modern browsers: the compatibility with modern web browsers relates to the development part which has not yet started for the Web App Client. However, it will be taken into account during the development process.

In conclusion, the result of this specification phase has permitted to design a UI that satisfies the requirements for important user interactions. These interactions cover CAMEL modelling, resource management, application deployment and monitoring. As it is portrayed in the “Related Requirements” subsections in the above specifications, most of requirements have been covered in the specifications for the user interfaces represented by “CAMEL Designer” Modelio module and the Web Client User Interface for Morphemic UI.

The future plan includes developing CAMEL Designer module for Modelio and the Web User Interface. The development phase is expected to be done in progressive releases where we prioritize the requirements with the high-level priorities. In other words, the priority for CAMEL Designer is to have a first functional prototype for designing a basic CAMEL Model that is sufficient enough for covering the basic concepts needed for modelling the use cases supported by Morphemic project. Similarly, it is expected to release a prototype for the Web User Interface that includes the “Frontend” and “Back for Front” components (see section 4.1) in order to get feedback from the users and add new features and apply improvements in the next releases.



References

- [1] BRUNNSTRÖM, Kjell, BEKER, Sergio Ariel, DE MOOR, Katrien, *et al.* Qualinet white paper on definitions of quality of experience. 2013.
- [2] The Back-end for Front-end Pattern (BFF), https://philcalcado.com/2015/09/18/the_back_end_for_front_end_pattern_bff.html, last accessed 31 August 2020
- [3] The Single Page Interface Manifesto, http://itsnat.sourceforge.net/php/spim/spi_manifesto_en.php, last accessed 31 August 2020
- [4] Micro Frontends, <https://micro-frontends.org/>, last accessed 31 August 2020