Deliverable D4.8

Intermediate multi-cloud application monitoring

Editor(s):	Juncal Alonso
Responsible Partner:	TECNALIA
Status-Version:	Final - v0.1
Date:	30/11/2018
Distribution level (CO, PU):	CO

Project Number:	GA 726755
Project Title:	DECIDE

Title of Deliverable:	Initial multi-cloud application monitoring
Due Date of Delivery to the EC:	30/11/2018

Workpackage responsible for the Deliverable:	WP4 – Continuous deployment and operation
Editor(s):	TECNALIA
Contributor(s):	Juncal Alonso, Gorka Benguria, Marisa Escalante, Maria Jose Lopez, Alberto Molinuevo, Iñaki Etxaniz (TECNALIA), Javier Gavilanes, Antonio Fernandez (Experis)
Reviewer(s):	Iñaki Etxaniz (TECNALIA), Lorenzo Blasi (HPE)
Approved by:	All Partners
Recommended/mandatory readers:	WP2, WP3, WP5, WP6.

Abstract:	This deliverable describes the intermediate version of the technical details of the engine implemented for continuously monitoring multi-cloud applications enumerating the off-shelf software products selected for the implementation, how they are integrated and the implemented interfaces.
Keyword List:	Monitoring, working conditions, micro-service, multi-cloud application.
Licensing information:	The software is released under MIT license. The document itself is delivered as a description for the European Commission about the released software, so it is not public.
Disclaimer	This deliverable reflects only the author's views and views and the Commission is not responsible for any use that may be made of the information contained therein

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	16/10/2018	First TOC and assignments.	TECNALIA
v0.2	29/10/2018	Sections updated: Section 1, 2.1.1, Abstract, Table 1, Figure 1, 2, 3, references.	TECNALIA
v0.3	05/11/2018	Sections updated: Section 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.2.5, Annex 1, Annex 2, conclusions.	TECNALIA
v0.4	15/11/2018	Sections updated: Section 2.2.2. Section 3.1, Annex 1	TECNALIA, Experis
v0.5	19/11/2018	Sections updated with ADAPT MM new naming convention: Section 2, Annex 2.	TECNALIA
v0.6	21/11/2018	Sections updated: Section 3.1, figures updated in section 2	Experis
v0.7	22/11/2018	Sections updated: figures updated in section 2	TECNALIA
v0.8	26/11/2018	Sections updated: minor updates in 2.2.3, section 3.2.	TECNALIA, Experis
v0.9	28/11/2018	Review comments addressed.	TECNALIA, Experis
V1.0	30/11/2018	Ready for submission	TECNALIA

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

EC	European Commission
M12	Month twelve
M24	Month twenty-four
M30	Month thirty
D	Deliverable
NFR	Non Functional Requirement
F	Functionality
MTBF	Mean Time Between Failures
MTTR	Mean Time to Recovery
SRC	Source Code
VH	Violation Handlers
ADAPT MM	ADAPT Monitoring Manager
MCSLA SLOs	Multi Cloud Service Level Agreement Service Level Objective
WP	Work Package
MIT	Massachusetts Institute of Technology
REST	Representational State Transfer
API	Application Programming Interface
URL	Uniform Resource Local
ADAPT DO	ADAPT Deployment Orchestrator
URI	Uniform Resource Identifier
JSON	JavaScript Object Notation
CRUD	Create, Read, Update, Remove
ACSmI	Advance Cloud Service meta-Intermediator
UI	User Interface
TOML	Tom's Obvious, Minimal Language

Executive Summary

This document describes the two components, ADAPT Monitoring Manager (MM) and Violations Handler (VH), forming the current version of the Intermediate multi-cloud application monitoring prototype, which monitors the deployed multi-cloud application in order to verify its working conditions and manages the actions to be performed when a violation occurs.

The document presents the missions, scopes, functional descriptions, technical approaches, download and installation instructions, and user manuals of these components comprising the prototype.

This document describes the M24 release of the ADAPT Monitoring Manager and the Violation Handler components and updates the one delivered with the M12 release [1]. It will be updated again in M30, including the new functionalities implemented for the final version.

The current release of ADAPT Monitoring Manager implements the following new features with respect to the M12 version prototype:

- ADAPT MM controller sub-component, which manages the actions to be executed by ADAPT monitoring, i.e: calls to the correspondent sub-components. This component includes the interface to receive the request to start the monitoring process from ADAPT DO.
- Automatic configuration of ADAPT MM data collection sub-component through the implementation of an ad-hoc library.
- ADAPT MM violations detection sub-component, including the integration with the DECIDE MCSLA editor tool.
- New dashboards in ADAPT MM UI, including the metrics for the availability and performance NFRs.
- Integration with the Application Description, which is the main DECIDE integration mechanism between the different tools.
- Integration with the DevOps framework.

The current release of the Violation Handler implements the following new features with respect to the M12 version prototype:

- Retrieval of the technological risk from the application description. The previous version obtained this value as well, but it took it from a testing database created for that purpose. The current version accesses the actual application description, located in a Git repository, to get this value.
- Notification of a violation. Upon reception of a violation, this component sends an email to the operator with information about what happened.
- Orchestrate the redeployment workflow. Depending on the value of the technological risk, the Violations Handler decides which workflow to trigger. If the risk is “low”, the application will be automatically redeployed, whereas if the risk is “high”, the developer must confirm the redeployment.

1 Introduction

1.1 About this deliverable

This document is a complement to the software prototype of the same name -Intermediate multi-cloud application monitoring- and which are delivered at the same time, specified at the head of the document, namely 30/11/2018.

1.2 Document structure

This document is divided into two main sections, describing the two components of the prototype. Architectural and implementation details of the components and how to use and download them are described in detail in the following sections. Section 2 is dedicated to ADAPT Monitoring Manager, and section 3 describes the Violations Handler. The document also includes two Annexes, one describing the Telegraf parser library implemented in the context of DECIDE, and the other one specifying the iStartStopmonitoring and iStorealertinformation API.

2 ADAPT Monitoring Manager

2.1 Implementation

2.1.1 Functional description

The DECIDE ADAPT MM components monitor the deployed multi-cloud based application and verify that the non-functional requirements and the SLOs are being fulfilled. If a violation of any of the NFRs or SLOs defined in the MCSLA is detected, ADAPT MM components will inform the violations handler component -described in section 3- which will generate the proper actions depending on each situation and context. If the violation occurs, information saying that the working conditions are not met will be sent to the operator. Besides, if the application is low technology risk, the "adaptation" process will be launched, through the violations handler component.

The main functionalities of the ADAPT MM components are:

F1. Collect data from the deployed multi-cloud application: Once a multi-cloud application is deployed on certain resources, ADAPT MM shall start the monitoring process. The ADAPT MM component needs to get the data from the deployed components to collect information of their service level metrics at real time. The data related to the cloud services where the components of Multi-cloud application are deployed will be collected by ACSmI. ADAPT MM informs ACSmI so it can start the monitoring and assessment of the CSPs SLOs. The NFRs to be covered in the context of DECIDE project are: Performance, Availability, Scalability, Security /Legal, Cost, Location, Technology risk and State. From these, ADAPT monitoring will assess the SLOs of Performance, Availability and Scalability of the Application MCSLA.

F2. Store the data collected from the deployed multi-cloud application: To assess the required working conditions of the multi-cloud application, ADAPT MM will deal with data at real time. These time series will be stored for further analysis of the data (i.e. aggregation, min-max values, etc.).

F3. Create the aggregated data to be assessed: from the raw metrics (e.g. http response time), the ADAPT MM component will need to create aggregated data to assess the violations. That is the final measured NFR (e.g. Availability) calculated per component and the aggregated value for the multi-cloud application as a whole.

F4. Visualize the measurements: DECIDE ADAPT MM will provide the user with a graphical interface to visualize the monitored data of the multi-cloud application. DECIDE ADAPT MM will also provide monitoring information of the violations occurred.

F5. Detect when a violation occurs: DECIDE ADAPT MM will detect violations on the working conditions (MCSLA SLOs), send the corresponding alert to the operator (through the VH) and launch the "adaptation" process through a call to the Violations Handler. The adaptation process launched will include the actions directed to stop the monitoring of the previous components and resources.

The existing approaches for application monitoring [2] do not tackle the monitoring different elements of the application (components) and which are deployed onto disperse cloud resources. They just face the problem of monitoring single components, or types of components (i.e. software applications). In DECIDE ADAPT MM, the working conditions of a multi-cloud application with respect to specific NFRs (but with an extendable approach) will be supported and addressed at different levels [3] [4] (software, resources) and combining them at run time.

ADAPT MM will be implemented following an incremental approach, adding features in the successive releases of the tool (D4.7 was already delivered in M12, and D4.9 will be delivered in M30). In this second release, all the main functionalities are implemented, using an example application (Socks Shop, see section 2.1.2.1) as a target. The coverage of each of the functionalities varies, and will be completed in the third release, due in M30.

The following table details the relationship between the functional requirements -indicated in deliverable D4.2 [5] and updated for year 2- and the implemented functionalities, with a description of the coverage for each functionality.

Requirements covered by the prototype:

Table 1. Functionality covered by the M24 ADAPT monitoring prototype.

Functionality	Req. ID	Coverage
F1	WP4-MR3, WP4-MR10, WP4-MR13, WP4-MR13	In this release, ADAPT MM is able to receive the request from ADAPT DO to start the monitoring. It reads the SLOs included in the Application Description to determine the NFRs to be monitored, and it can monitor the following NFRs: Performance and availability at application level.
F2	WP4-MR3, WP4-MR10, WP4-MR13, WP4-MR13	The current release configures automatically the output of the agents (based on the information of the Application Description) which collects the metrics, so these metrics are stored in the corresponding data base (time series data base) and tagged to their corresponding application.
F3	WP4-MR3, WP4-MR14	The component gets the raw real time metrics to generate the aggregated value (performance, availability) per component and aggregates the value to calculate the metrics for the application. Different metrics can be configured if needed in the future.
F4	WP4-MR3, WP4-MR11	The current prototype adds new dashboards to the UI for the availability metrics.
F5	WP4-MR2, WP4-MR3	The current prototype reads the SLO established in the MCSLA at application level, compares it with the real time value and generates a request to the VH if a violation of the SLO is found (availability and performance).

Detailed functionality that this prototype offers:

The delivered prototype is the intermediate version of the ADAPT MM component and contains the following functionalities:

1. Get the request from ADAPT DO, to start the monitoring process.
2. Get raw metrics for availability and performance NFRs: For this intermediate version, ADAPT MM is focused on getting measurements related to availability and performance of the components of the micro-services of the multi cloud application Sock Shop (see section 2.1.2.1 for details). Availability of the multi-cloud application will be impacted by the metrics related to the availability of the software component (or micro-service) themselves and by the aggregation formula used to calculate the final MCSLA.

   The raw metrics gathered to be monitored for Availability and Performance for this prototype are:

   - *http_response_code* (int): The code received from the http request.

- *result_type* (string): The possible results types are success, timeout, response_string_mismatch, connection_failed.
- *Response time* (float): Component response time.

These metrics are monitored per multi-cloud app component (see section 2.1.2.1 for Sock Shop app details). The compilation of these raw metrics is monitored in order to set up the final Availability [3] and Performance metric of the multi-cloud compiled. These compiled metrics for Availability are:

- MTBF: Mean time between failures.
- MTTR Mean time to recover.

Availability at software level in DECIDE.

Some considerations:

- "availability" is considered as a function of time, defined as the probability that a system is operating correctly and is available to perform its function at the instant of time "t".
- Metrics for availability at software (SW) level in DECIDE:
  - A = MTBF/(MTBF+MTTR)
- Availability is calculated per component and aggregated based on the MCSLA definition.

Performance at software level in DECIDE.

Some considerations:

- "Performance" is an indicator of how well a software system or component meets its requirements for timeliness. Timeliness is measured in terms of response time or throughput.
- The response time is the time required to respond to a request. It may be the time required for a single transaction, or the end-to-end time for a user task.
- The throughput of a system is the number of requests that can be processed in some specified time interval.
- Performance can be set up at hardware and software level
- Metrics for performance at software level in DECIDE:
  - Response time: Response time is measured at component level and aggregated based on the MCSLA definition.
  - Throughput: Throughput measurements are not included in this version of the prototype. They will be included in the next version of the prototype.

3. Calculates the value of the availability and performance metrics of the multi-cloud application, based on the aggregation expression established in the MCSLA.
4. Assesses the measured values for Availability and Performance at application level.
5. Shows the availability and performance metrics in a graphical dashboard: The dashboards designed includes the following graphics:
   - Availability dashboard:
     - Graphic 1/2/3/4 (cf. User Manual, section 2.2.3): The code of the response received, per component (each 10s). The possible types of response message are: success, timeout, response string mismatch, connection failed.
   - Performance dashboard

- Graphic 1 (cf. User Manual, section 2.2.3): Mean response time per app component in the same graphic (it includes the current, maximum and minimum values too).
- Graphic 2/3/4/5 (cf. User Manual, section 2.2.3):Mean response time per app component in separated graphics (it includes the current, maximum and minimum values too).
- Graphic 6 (cf. User Manual, section 2.2.3): Maximum response time in the application (considering the different components).
- Graphic 7/8/9/10 (cf. User Manual, section 2.2.3): Maximum response time in the components (considering metrics in the last hour).

2.1.1.1 Fitting into overall DECIDE Architecture

ADAPT MM is one of the final steps in the DECIDE workflow [6]. It supports the operation phase of multi-cloud aware applications by providing means for run-time monitoring of the current deployment, with respect to selected non-functional requirements and SLOs, and re-deployment adaptation when needed. The ADAPT MM is part of the ADAPT key result of the DECIDE architecture, indeed it is one of the components inside this package (see figure 1). In this picture, only the interfaces used and implemented by ADAPT have been depicted. The whole figure with all the interfaces between the different tools can be found in D2.5 [7].

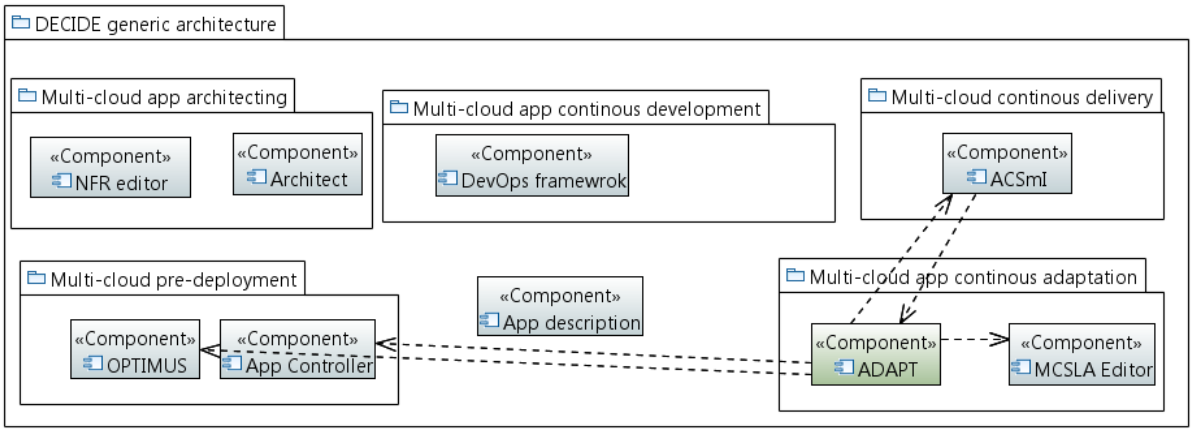

Figure 1. ADAPT monitoring in DECIDE architecture**.**

ADAPT MM interacts with the two other components of ADAPT (Violation Handler and ADAPT deployment orchestrator) as well as with other tools in the DECIDE framework [6]. ADAPT MM will communicate, on the one hand, with the MCSLA editor/ App description to gather the threshold values for the different metrics to be assessed, and with the Violations Handler to provide information about any violation. The ADAPT deployment orchestrator will send the requests to start/stop the monitoring and ADAPT MM will request ACSmI to start /stop the monitoring of the CSP SLOs.

In the following picture, the general architecture of the ADAPT MM components is shown. For a detailed description, see D4.2 [5]:

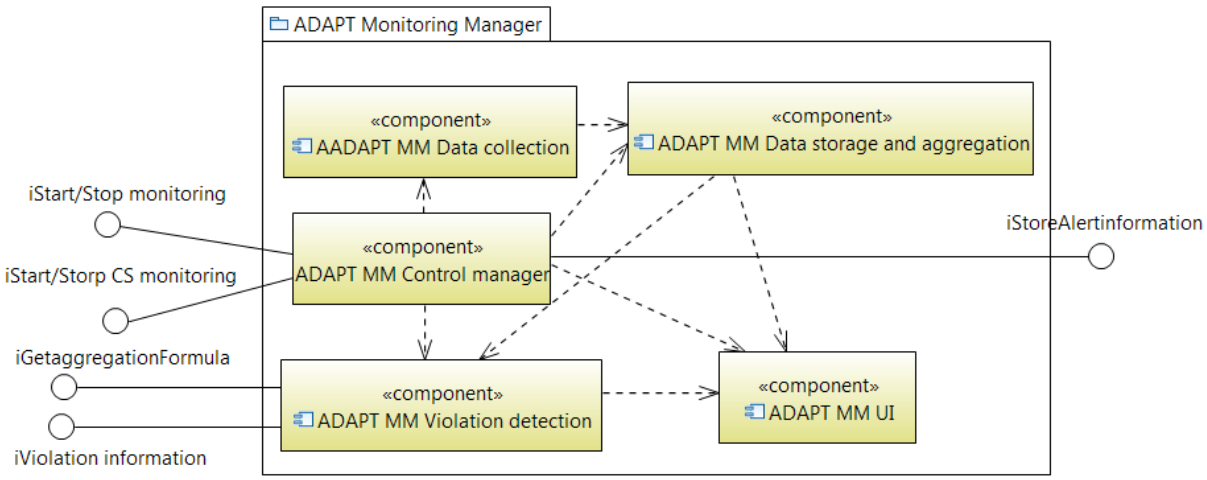

Figure 2. General architecture of ADAPT monitoring.

2.1.2 Technical description

This section describes the technical details of the implemented software for the current prototype of ADAPT MM.

2.1.2.1 Prototype architecture

The current prototype (M24) of ADAPT MM architecture consists of two tiers:

- **Server side**: Four Docker containers deployed on a virtual machine, containing the ADAPT MM data storage and aggregation, the ADAPT MM UI, the ADAPT MM Data collection and ADAPT MM Control manager deployed (this includes the ADAPT MM violations detection). The ADAPT MM Data collection agents get the selected measurements from the different components of the Sock Shop application and store them in the ADAPT MM Data Storage and aggregation. With these metrics, ADAPT MM Control manager generates the final monitored SLA terms to continuously assess them against the SLOs defined in the MCSLA, and detect possible violations (ADAPT MM violation).
- **Client side**: A virtual machine with the Sock Shop application[1] deployed into several Docker containers (each microservice deployed into one docker container). For this prototype and from the "components" architecture for the sock shop application (based on the Docker Compose instance of the application [8]), we have considered the following components to be monitored:
  - Component 1: front-end
  - Component 2: catalogue (the associated *catalogue-db* is not being monitored)
  - Component 3: carts (the associated *carts-db* is not being monitored)
  - Component 4: orders (the associated *orders-db* is not being monitored)
- **Interfaces**: This prototype implements and uses several interfaces:
  - iStartstop monitoring: ADAPT MM Control manager implements this REST interface that enables to start/stop of the monitoring process by other components (i.e. ADAPT DO). The specification of this interface is included in Annex 2.
  - iStorealertinformation: ADAPT MM Control manager implements this REST interface that enables to provide information about the alerts and the corresponding implemented by the VH. The specification of this interface is included in Annex 2.

[1] Shock Shop application is a micro-services-based application used to illustrate micro-services architectures. Generic description is included in [13].

- iGetaggregationformula: ADAPT MM Control manager consumes this interface implemented by MCSLA editor to get the aggregation formula to assess the fulfillment of the MCSLA. This interface is implemented as a JAVA library [9], which is used as part of the ADAPT MM violation detection component.

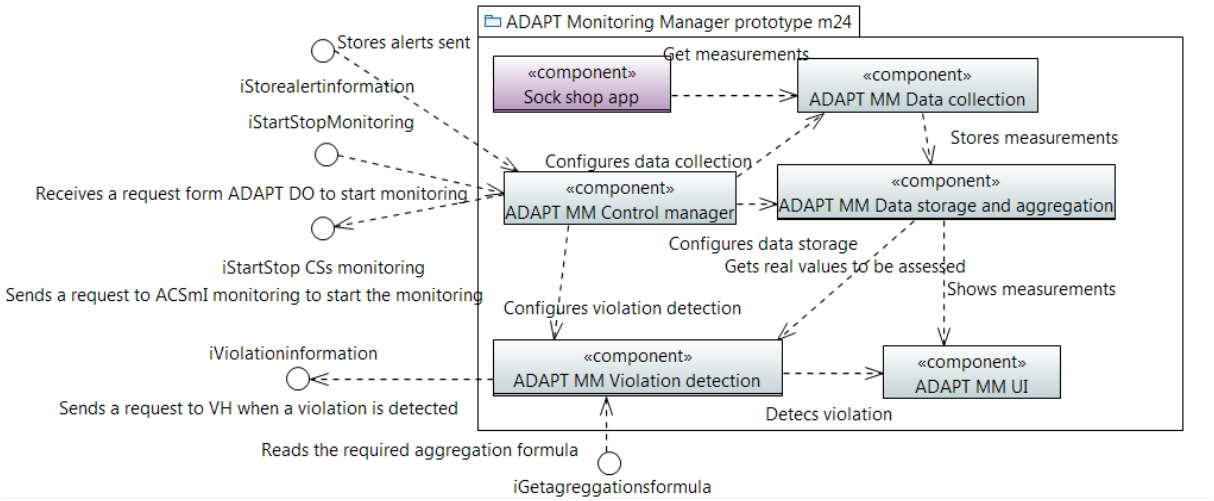

Figure 3. ADAPT Monitoring M24 prototype High level architecture

2.1.2.2 Components description

The current prototype includes the 5 sub-components envisioned for ADAPT monitoring [10].

- **ADAPT MM Data collection**: This sub-component collects the data from the Sock Shop application with respect to its working conditions. In the current prototype, the Data Collection sub-component has been configured to get the availability and performance related metrics from the different components of the Sock Shop application. The Data collection component gets these metrics and stores them in the ADAPT MM Data storage and aggregation every one second. The configuration of this sub-component (based on the information of the components included in the Application Description) is automatically configured.
- **ADAPT MM Data storage and aggregation**: This sub-component is in charge of storing the data collected from the ADAPT MM Data collection and aggregating it to create the actual measures that will be assessed by the ADAPT MM violation detection.
- **ADAPT MM Control manager**: This component manages all the activities of ADAPT MM. It receives the request to start the monitoring of the application. Then ADAPT MM Control manager configures the different agents in the ADAPT MM Data Collection, sets up the ADAPT MM Data storage and aggregation and assesses the values of the different NFRs with respect to the MCSLA SLOs, detecting if a violation has occurred.
- **ADAPT MM UI**: This is the graphical user interface for the ADAPT MM component. In the current version of the prototype, this graphical interface includes the metrics for the visualization of the availability and performance of the different components in real time has been implemented.
- **ADAPT MM Violations detection**: This subcomponent assesses the aggregated value depending on the one established in MCSLA against the measured values.

2.1.2.3 Technical specifications

ADAPT monitoring prototype for M24 has been implemented as a monitoring stack, covering the data collection, data storage, data aggregation, MCSLA assessment and the data visualization.

- **ADAPT MM Data collection**: For the data collection, the Telegraf[2] open source technology is used. Telegraf is a plugin-driven server agent written in Go to collect, process, aggregate, and report metrics. It is a compiled and standalone binary that can be executed on any system with no need for external dependencies. For the purpose of ADAPT MM, the Telegraf plugins have been configured to acquire the following measurements:
    - Input plugins: http_response plugin, which tests HTTP/HTTPS connections to a given list of urls in this case the Sock Shop app components urls), with a given periodicity, from a set interface. This plugin gets the following metrics:
        - response_time (float, seconds)
        - http_response_code (int) (the code received)
        - result_type (string) (success, timeout, response_string_mismatch, connection_failed).
    - Output plugins: Influx DB plugin. This plugin sends the metrics gathered by the agent to a specific Influx DB instance (with a given url). The name of the database, login parameters, etc. needed to be specified.
- **ADAPT MM Data storage and aggregation**: For the storage of the metrics, Influx DB storage technology has been implemented. Influx DB is used as a data store for cases involving large amounts of timestamped data, like monitoring applications or micro-services, which is the case of ADAPT MM. Influx DB supports plugins for data ingestion protocols, like Telegraf. The measurements are injected by the Telegraf plugins in the Influx DB database
- **ADAPT MM Control manager**: The ADAPT MM Control manager has been developed from scratch. It is a Java program that manages the different activities and workflow, to be able to start and programmatically configure all the elements for the monitoring, based on the information from the Application Description. It is deployed using a Jetty server[3].
- **ADAPT MM UI**: For the generation of the user interface where the metrics related to the current working conditions of the application can be monitored, Grafana[4] is used. Grafana is an open platform to visualize data. It provides means to configure specific dashboards with different graphics. For this intermediate version of ADAPT MM, two dashboards - to show the selected metrics per component (micro-services) - have been created. Grafana has been configured to read the information from the Influx DB.
- **ADAPT MM Violations detection**: The ADAPT MM violations detection has been developed as a function inside ADAPT MM Control manager. This function assesses the monitored metrics aggregated against the values stablished in the MCSLA.

2.2 Delivery and usage

2.2.1 Package information

The delivered package consists of the below folder structure.

[2] https://www.influxdata.com/time-series-platform/telegraf/
[3] https://www.eclipse.org/jetty/
[4] https://grafana.com/

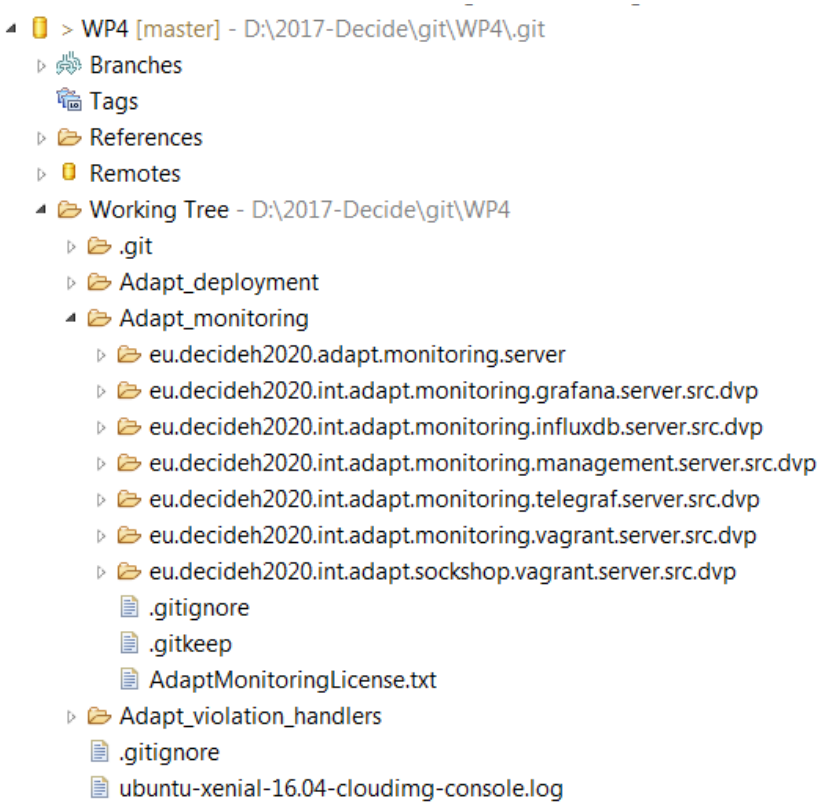

Figure 4. Source folder structure of ADAPT Monitoring component in M24.

ADAPT monitoring folder is composed of seven main sub-folders with the corresponding src folder:

- *eu.decideh2020.int.adapt.monitoring.server*: Contains the source for the ADAPT MM Control manager. (see figure 3).
- *eu.decideh2020.int.adapt.monitoring.grafana.server.src.dvp*: Contains the source for the ADAPT MM UI. (see figure 3) and the corresponding Docker file to generate the container.
- *eu.decideh2020.int.adapt.monitoring.influxdb.server.src.dvp*: Contains the source for the ADAPT MM storage and aggregation (see figure 3) and the corresponding Docker file to generate the container.
- *eu.decideh2020.int.adapt.monitoring.management.server.src.dvp:* Contains the corresponding docker file to generate the container to deploy the ADAPT MM Control manager (see figure 3).
- *eu.decideh2020.int.adapt.monitoring.telegraf.server.src.dvp:* Contains the source for the ADAPT MM Data collection and the corresponding Docker file to generate the container (see figure 3).
- *eu.decideh2020.int.adapt.monitoring.vagrant.server.src.dvp*: Contains the source for the vagrant file to create the virtualized environment (when needed) where ADAPT MM is deployed.
- *eu.decideh2020.int.adapt.shockshop.vagrant.server.src.dvp*: Contains the source for the testing multi-cloud application (Shock Shop application) to be monitored (see Figure 3) and the corresponding Docker file to generate the container.

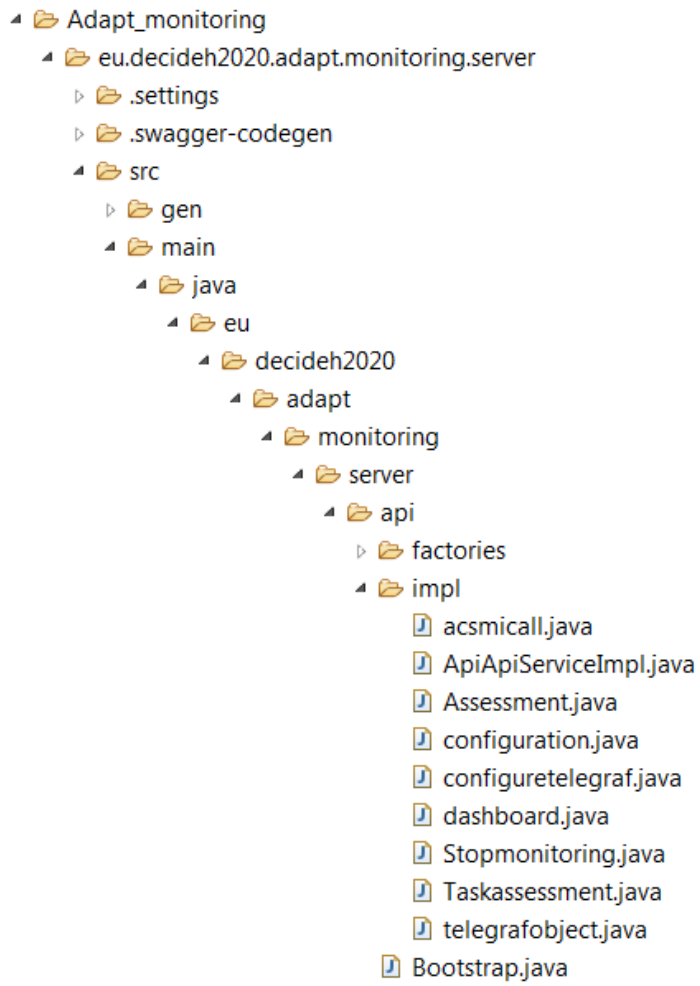

Figure 5. Source folder structure of ADAPT MM Control manager sub-component

Inside the *eu.decideh2020.int.adapt.monitoring.server* the following relevant folders and packages are included:

- src: Contains the source files and their packages. Here the most relevants files are:
  - *ApiApiServiceImpl.java*: Java file with the implementation of the iStartStopmonitoring interface.
  - *configuration.java*: Java file with all the corresponding actions for the initial automatic configuration of the monitoring. This includes reading the Application Description and writing the relevant information on it and launching other internal actions such as configuration of the Telegraf, assessment of the MCSLA.
  - *Confiuretelegraf.java*: Java file that uses the implemented Telegraf parser library[5] (see annex 1) to configure the telegraf.conf file based on the information of the Application description.

[5] This library has been implemented in the context of the DECIDE project to be able to configure the telegraf.conf file programmatically, according to the needs of DECIDE. The telegraf.conf file is written in TOML and up to DECIDE partners' knowledge there is not any open source parser that fulfils DECIDE requirements for the automatic configuration of the telegraf.conf file.

- *dashboard.java*: This function will contain (in next releases) the needed actions to automatically configure the Grafana Dashboards based on the selected NFRs to be monitored.
- *assessment.java*: Java file with all the corresponding actions to assess the MCSLA of the multi-cloud application in real time.
- *acsmicall.java*: Java file to use the ACSmI monitoring REST interface.
- *telegrafobject.java*: Java file to create the object to store the relevant information from the Application description to configure Telegraf.
- *Taskassessment.java*: Java file to create the task for the continuous monitoring until the stop is received.
- *Stopmonitoring.java*: Java file to create the task to perform the needed actions when a stop monitoring request is received.

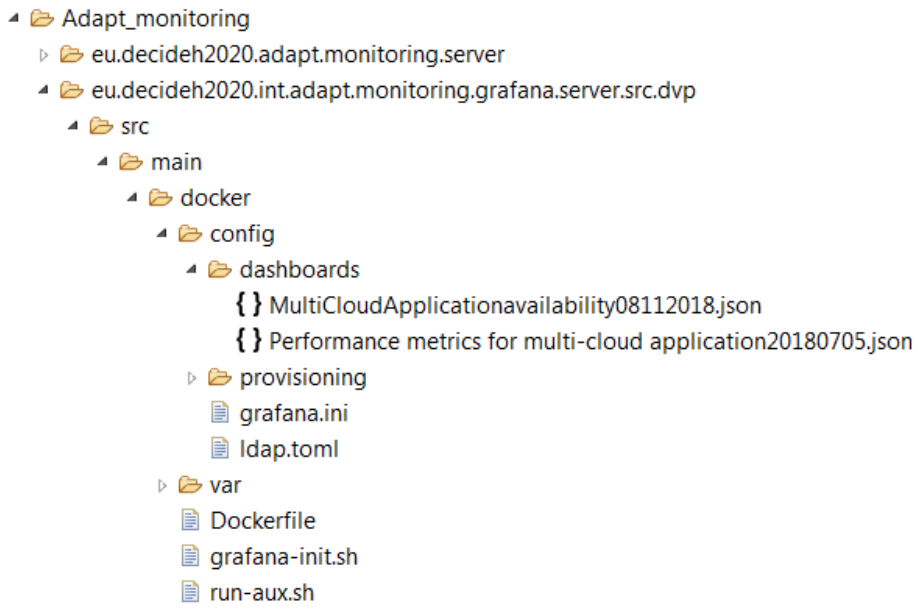

Figure 6. Source folder structure of ADAPT M UI sub-component

Inside the *eu.decideh2020.int.adapt.monitoring.grafana.server.src.dvp* the following relevant folders and packages are included:

- src: Contains the source files and their packages. Here the most relevant files are:
  - *DECIDE MultiCloudApplicationavailability08112018.json / Performance metrics for multi-cloud application20180705.json*: JSON files with the specification of the graphical Dashboards for the DECIDE MM UI.

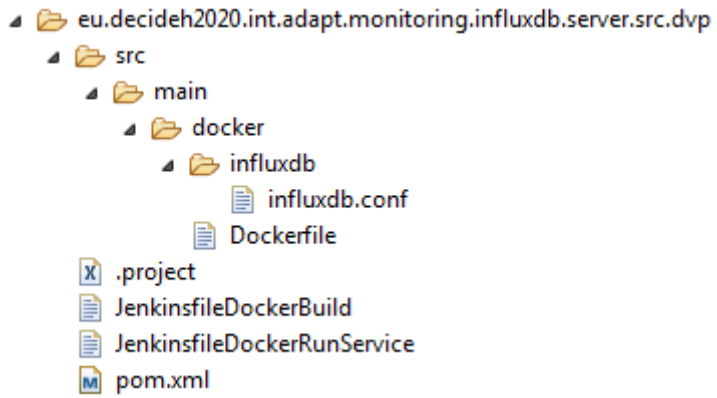

Figure 7. Source folder structure of ADAPT M Data Storage and Aggregation

Inside the *eu.decideh2020.int.adapt.monitoring.influxdb.server.src.*dvp the following relevant folders and packages are included:

- src: Contains the source files and their packages:
  - *Influxdb.conf*: This file contains the configuration of the Data Base to store the selected measurements in DECIDE ADAPT MM.

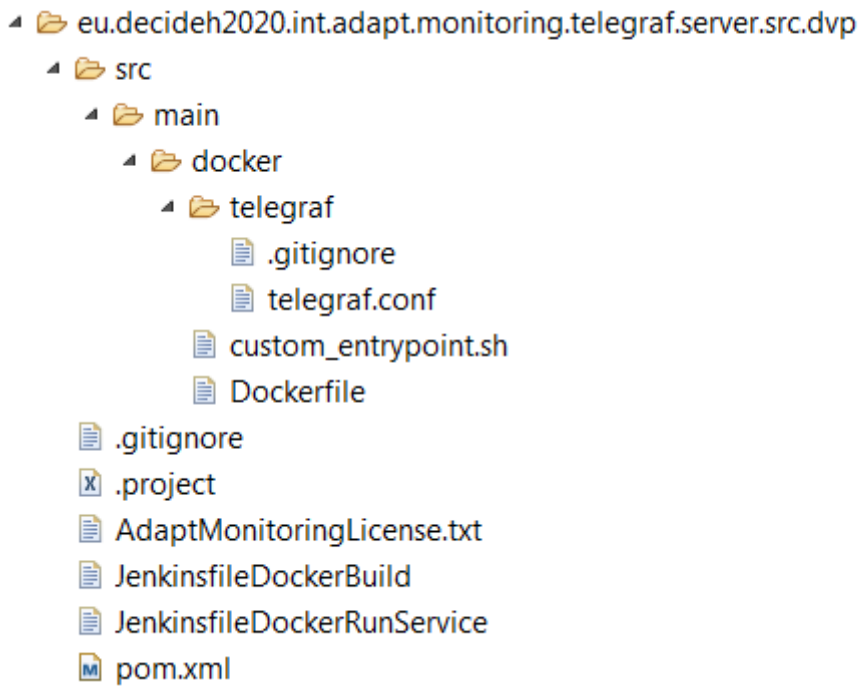

Figure 8. Source folder structure of ADAPT M Data Collection

Inside the *eu.decideh2020.int.adapt.monitoring.telegraf.server.src.dvp* the following relevant folders and packages are included:

- src: Contains the source files and their packages:
  - *telegraf.conf*: This file contains the configuration of the agents where the inputs and output plugins to get the metrics from the application. This file is automatically configured by the ADAPT MM Control manager component.

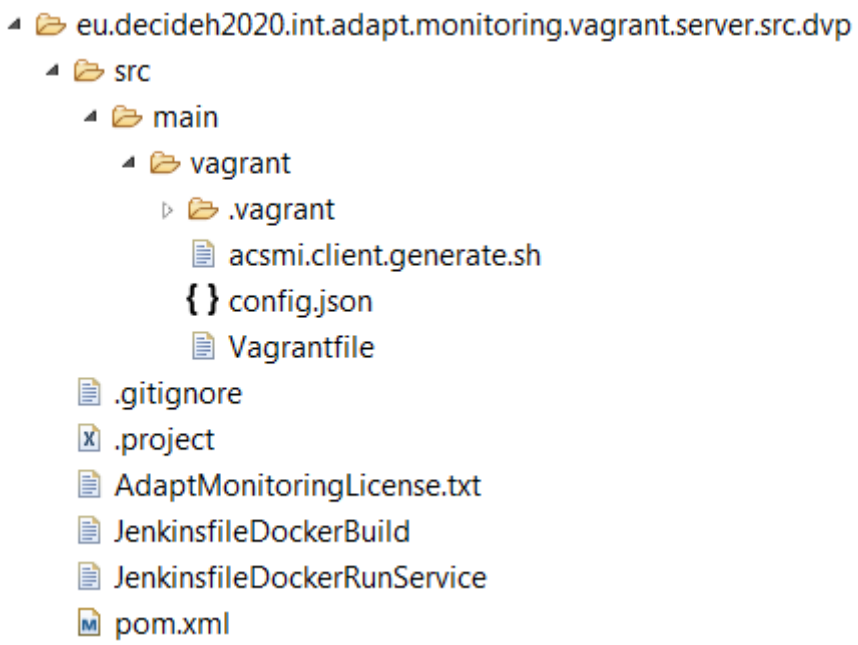

Figure 9. Source folder structure of Vagrant server.

Inside the *eu.decideh2020.int.adapt.monitoring.vagrant.server.src.dvp* the following relevant folders and packages are included:

- src: Contains the source files and their packages:
  - *Vagrant* file: Configuration file to create the virtual environment where to deploy ADAPT MM.

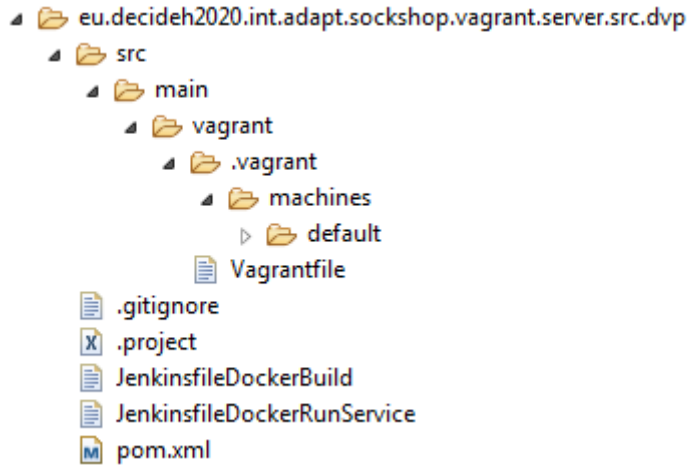

Figure 10. Source folder structure for the Shock Shop application.

Inside the *eu.decideh2020.int.adapt.shockshop.vagrant.server.src.dvp* the following relevant folders and packages are included:

  - *Vagrant file*: Configuration file to facilitate the setting up of the Shock Shop application to be monitored (automatizing the creation of a virtual machine with the corresponding operating system and the containers server). This release includes the Sock Shop application as a testing application that is being monitored for testing purposes.

2.2.2 Installation instructions

The prototype has been installed and tested using the following software (see Pre-requirements section):

- Docker to create the containers of the different sub-components.

The following steps need to be executed to get the prototype up and running:

1. Download the src from the DECIDE repository (DECIDE_components/ADAPT/Monitoring tag M24).
   `Git clone https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git`
2. Insert the credentials and the git path for the git repository to be used to store the Application description[6] in the *configuration.java* file.

```
2  *Copyright (c) 2017 Tecnalia.
31 package eu.decideh2020.adapt.monitoring.server.api.impl;
32 import java.io.IOException;
50 public class configuration {
51     //Create a List of telegraf objects to create the corresponding data for telegraf configuration
52             static List <telegrafobject> telegrafdata = new ArrayList<telegrafobject>();
53     void apply () {
54         //load and read appdescription with the app-controller
55         System.out.println("temp folder is: "+System.getProperty("java.io.tmpdir"));
56         String localAppDesc = "";
57         localAppDesc = System.getProperty("java.io.tmpdir");
58         localAppDesc = localAppDesc+"\\"+System.currentTimeMillis();
59         System.out.println("localAppDescPath is: "+ localAppDesc);
60         Path localappdescpath=Paths.get(localAppDesc);
61         //gitRef to be changed to the common gitlab
62         //String gitRef="https://git.code.tecnalia.com/decide/SockShop_AppDescription.git";
63         //Local DECIDE.json
64         String gitRef="https://git.code.tecnalia.com/j2018b/106776.git";
65         //PUBLIC decide repo
66         //String gitRef="https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git";
67         //HPE's DECIDE.json
68         //String gitRef="https://git.code.tecnalia.com/decide/adapt-do-demo/blob/master/Socks-shop/DECIDE.json";
69
70         //String gitusername="";
71         //String gitpassword="";
```

Figure 11. Excerpt of the configuration.java file where the credentials for the git where the application description is stored are set up

3. Each of the containers of the different sub-components needs to be built

```
 Directorio de D:\2017-Decide\git\WP4\Adapt_monitoring

05/07/2018  12:08    <DIR>          .
05/07/2018  12:08    <DIR>          ..
21/11/2017  16:56               125 .gitignore
30/10/2017  10:12                 0 .gitkeep
17/11/2017  13:34             1.750 AdaptMonitoringLicense.txt
09/11/2018  12:44    <DIR>          eu.decideh2020.adapt.monitoring.server
19/12/2017  15:35    <DIR>          eu.decideh2020.int.adapt.monitoring.grafana.
server.src.dvp
19/12/2017  15:35    <DIR>          eu.decideh2020.int.adapt.monitoring.influxdb
.server.src.dvp
14/06/2018  10:30    <DIR>          eu.decideh2020.int.adapt.monitoring.manageme
nt.server.src.dvp
14/06/2018  10:30    <DIR>          eu.decideh2020.int.adapt.monitoring.telegraf
.server.src.dvp
```

Figure 12. ADAPT monitoring folders structure

a) ADAPT MM data collection (Telegraf) container:
   i. Switch to the folder where the Docker file is located (Telegraf container).

[6] The application description JSON file (DECIDE.json) used as input for this intermediate ADAPT monitoring is included in the source code inside the eu.decideh2020.int.adapt.monitoring.management.server.src.dvp folder

```
Cd D:\2017-
Decide\git\WP4\Adapt_monitoring\eu.decideh2020.int.adapt.mo
nitoring.telegraf.server.src.dvp[7]
```

```
 Directorio de D:\2017-Decide\git\WP4\Adapt_monitoring\eu.decideh2020.int.adapt.
monitoring.telegraf.server.src.dvp\src\main

14/06/2018  10:30    <DIR>          .
14/06/2018  10:30    <DIR>          ..
03/07/2018  11:19    <DIR>          docker
               0 archivos              0 bytes
               3 dirs  239.428.030.464 bytes libres
```

Figure 13. Telegraf docker file location

ii. Build the Docker image with the following arguments (Telegraf container):

```
docker build "/docker.telegraf.server", args: "-t
tecnalia/eu.decideh2020.adapt.monitoring.telegraf.server"
```

iii. Run the docker image with the following arguments (Telegraf container):

```
Docker                                                  run
"tecnalia/eu.decideh2020.adapt.monitoring.telegraf.server",
args:          "-d          --restart=always          --name
eu.decideh2020.adapt.monitoring.telegraf.server  --add-host
influxdb:172.18.0.1 --add-host sockshop:172.26.252.81"
```

In the case of the telegraf container, the influxdb Database where the agents are storing the metrics needs to be specified (IP of its docker container) as well as the IP address of the application to be monitored (its docker container IP address). In this case the current release incorporates a testing application (Sock Shop) for testing purposes[8].

b) ADAPT MM Data Storage and aggregation (Influx DB) container:

i. Switch to the folder where the Docker file is located (InfluxDB container).

```
Cd D:\2017-
Decide\git\WP4\Adapt_monitoring\eu.decideh2020.int.adapt.mo
nitoring.influxDB.server.src.dvp
```

```
 Directorio de D:\2017-Decide\git\WP4\Adapt_monitoring\eu.decideh2020.int.adapt.
monitoring.influxdb.server.src.dvp\src\main

19/12/2017  15:35    <DIR>          .
19/12/2017  15:35    <DIR>          ..
15/05/2018  11:00    <DIR>          docker
               0 archivos              0 bytes
               3 dirs  238.831.718.400 bytes libres
```

Figure 14. Influx DB docker file location

ii. Build the docker image with the following arguments (InfluxDB container):

[7] Please notice that the path is the one used to clone the DECIDE public git repository

[8] To change the application to be monitored, this IP address should be accordingly changed.

```
docker build "/docker.influxdb.server", args: "-t
tecnalia/eu.decideh2020.adapt.monitoring.influxdb.server"
```

iii. Run the Docker image with the following arguments (InfluxDB container):

```
docker run
"tecnalia/eu.decideh2020.adapt.monitoring.influxdb.se
rver", args: "-d -p 8086:8086 --restart=always -e
INFLUXDB_DB='decideh2020adapt' --name
eu.decideh2020.adapt.monitoring.influxdb.server"
```

In the case of the influxdb container the mapping of the ports needs to be configured as well as the name of the influx DB.

c) ADAPT MM UI (Grafana) container:

i. Switch to the folder where the Docker file is located (Grafana container).

```
Cd D:\2017-
Decide\git\WP4\Adapt_monitoring\eu.decideh2020.int.adapt.mo
nitoring.grafana.server.src.dvp
```

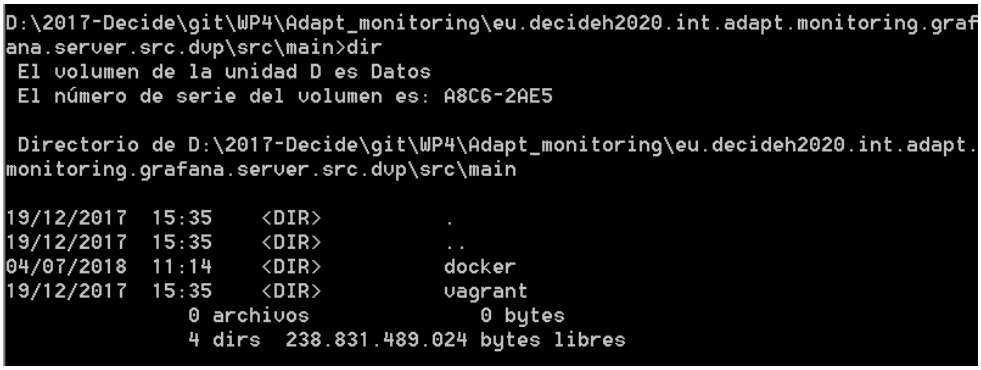

Figure 15. Grafana docker file location

ii. Build the Docker image with the following arguments (Grafana container):

```
docker build "/docker.grafana.server", args: "-t
tecnalia/eu.decideh2020.adapt.monitoring.grafana.server"
```

iii. Run the Docker image with the following arguments (Grafana container):

```
docker run
"tecnalia/eu.decideh2020.adapt.monitoring.grafana.ser
ver", args: "-d -p 3000:3000 --restart=always -e
'GF_SERVER_ROOT_URL=http://grafana.esilab.org'  -e
'GF_SECURITY_ADMIN_PASSWORD=admin'  --name
eu.decideh2020.adapt.monitoring.grafana.server --add-
host influxdb:172.18.0.1 --add-host
sockshop:172.26.252.81"
```

In the case of the grafana container the mapping of the ports needs to be configured, the url of the server, the corresponding admin password, the related influxdb to be connected to the Grafana server with its container IP and the IP of the container where the sockshop testing application is deployed.

d) ADAPT MM Control manager container:

i. Switch to the folder where the Docker file is located (ADAPT monitoring container).

```
Cd D:\2017-
Decide\git\WP4\Adapt_monitoring\eu.decideh2020.int.adapt.mo
nitoring.management.server.src.dvp
```

Figure 16. ADAPT M manager docker file location

ii. Build the Docker image with the following arguments (ADAPT monitoring container):

```
docker build "/docker.monitoring.server", args: "-t
tecnalia/eu.decideh2020.adapt.monitoring.management.server
--build-arg GIT_CREDENTIALS=$CREDENTIALS_GIT "
```

In the case of the ADAPT monitoring container the last version of the implemented code needs to be downloaded from git, and for this the GIT credentials are needed.

iii. Run the Docker image with the following arguments (ADAPT monitoring container):

```
docker run
"tecnalia/eu.decideh2020.adapt.monitoring.management.
server", args: "-d -p 10080:8080 --restart=always --
name
eu.decideh2020.adapt.monitoring.management.server"
```

In the case of the ADAPT monitoring container the mapping of the ports needs to be configured.

Once the containers are created and running,the ADAPT monitoring is up and running.

The current version of the ADAPT monitoring, monitors the availability and performance related metrics of the Sock Shop application. For that reason, an instance of the Sock Shop application needs to be deployed:

1. Download the src of the Sock Shop app.

```
git clone https://github.com/microservices-demo/microservices-demo
cd microservices-demo
```

Figure 17. Download the sample application to be monitored.

2. Install the Vagrant plugin for the Docker compose.

```
$ vagrant plugin install vagrant-docker-compose
```

Figure 18. Install Docker compose plugin in vagrant.

3. Execute vagrant up and assign it to the Virtual Box "provider":

```
vagrant up --provider=virtualbox
```

Figure 19. Executing vagrant up and assigning it to the Virtual Box

Now the Sock Shop application is up and running.

2.2.2.1 Pre-Requirements

The following lists the minimum requirements to get the component working.

- Machine or virtual machine with the O.S (Ubuntu Xenial 16.04) and the Docker server.
- JRE

2.2.3 User Manual

2.2.3.1 ADAPT MM API

ADAPT MM is started directly from the ADAPT DO component through the iStartStopmonitoring REST interface. In order to test it independently, the starting of the monitoring process needs to be launched externally by a call to the iStartStopmonitoring REST interface. This can be done using the Swagger editor[9], loading the corresponding interface json (included in in the software package). To access the corresponding json to be loaded in the swagger editor:

`start` **¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.**

In the swagger editor the host needs to be correspondingly changed to the IP where the host is deployed in each case.

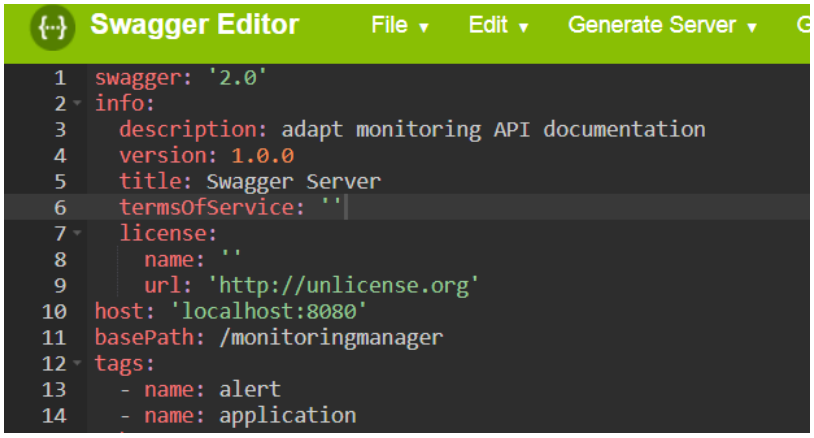

Figure 20. Host to be changed in the ADAPT monitoring interface

[9] https://editor.swagger.io/

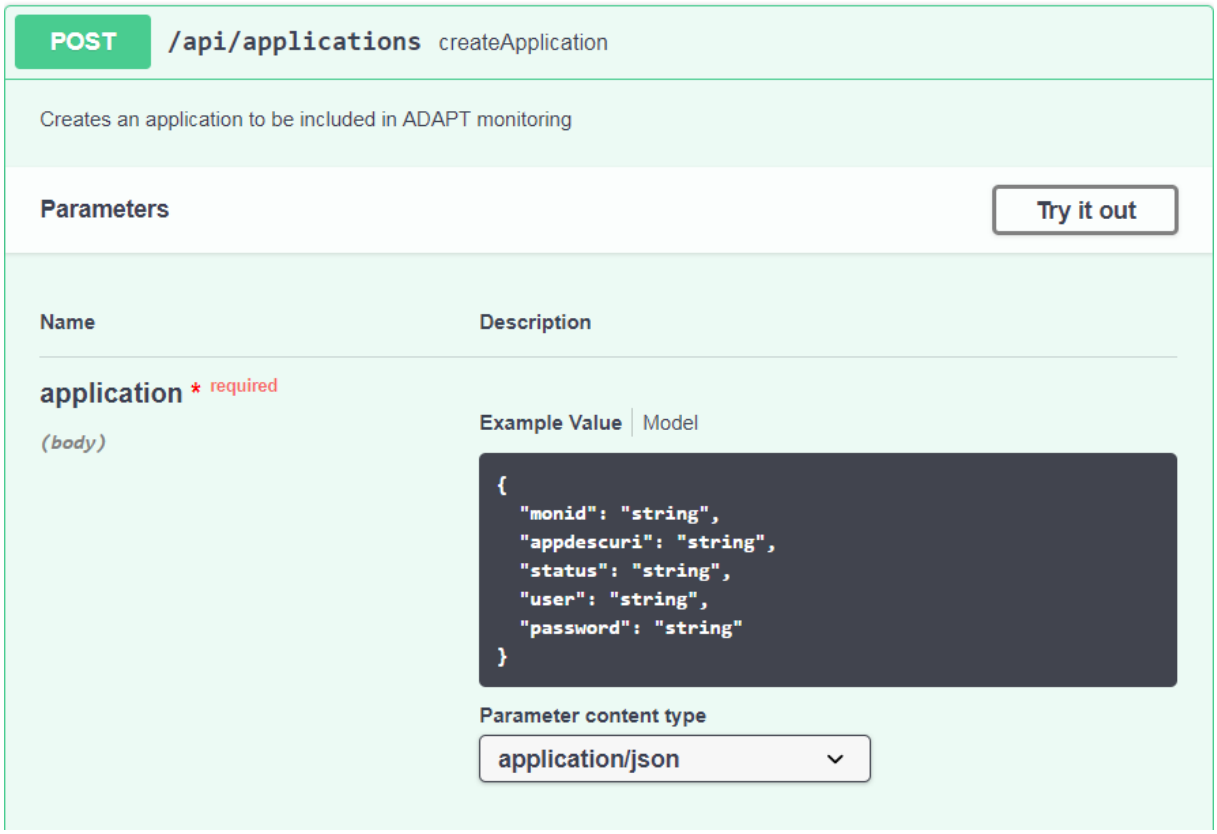

Figure 21. POST method of iStartStop interface

For testing purposes, the app_description uri and user and password used in the current version are hardcoded in the current prototype, so that the application description accessed (appdescuri) is fixed. The user and password are needed to access the git where the application description is stored (see point 2 in the installation instructions).

The monitoring is launched, and the app description is accordingly changed. In figure 22 the response that the interface provides is shown. The *monid* is completed with *applicationinstanceid* (this information is created by ADAPT DO and stored in the App Description). The status of the application is changed to “monitored”.

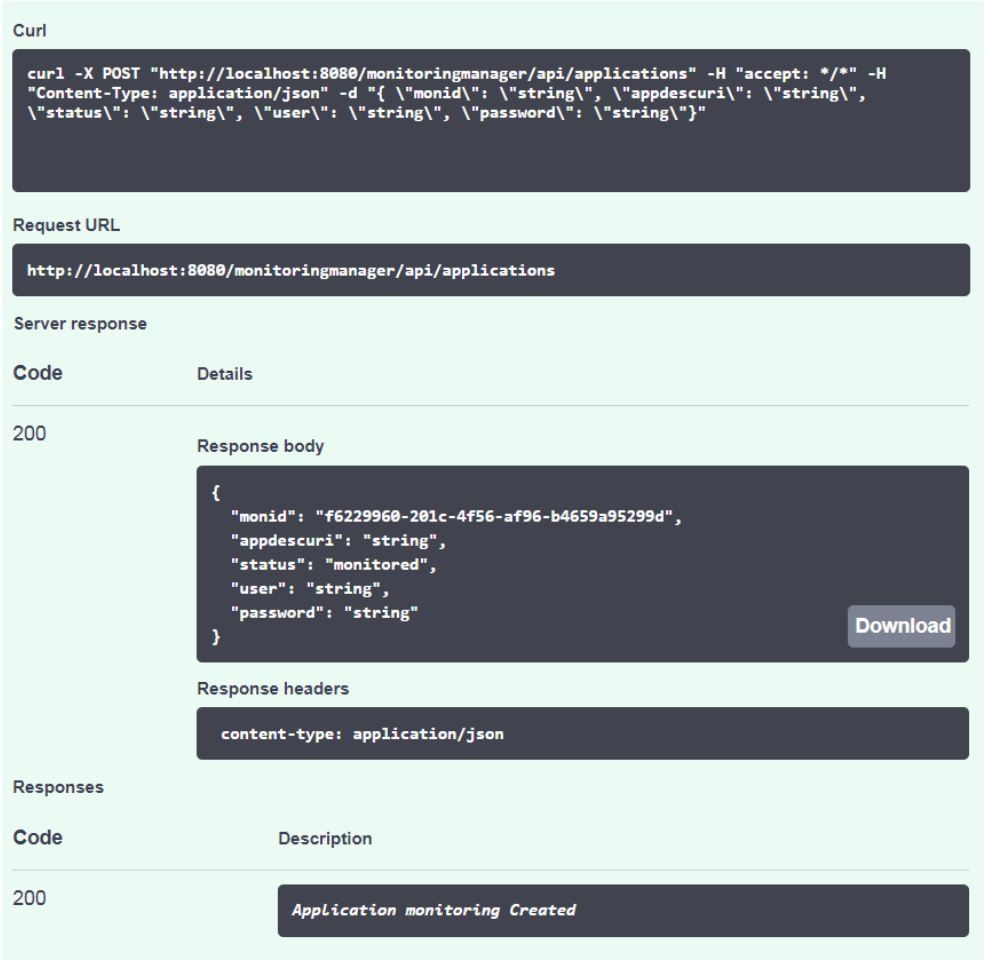

Figure 22. Start monitoring response

ADAPT MM Control manager configures the monitoring of the application based on the information included in the App Description file which is the main information sharing mechanism in DECIDE (see D4.2 [5] and D2.4 [6] for details). For testing purposes, the Application Description accessed by this prototype is included in the software package (see installation instructions). The access rights and user management will be managed by the DevOps Framework in DECIDE. For testing purposes, the access to the git has been hardcoded in the ADAPT MM component (see installation instructions for details).

2.2.3.2 ADAPT MM UI

Once the status received is “monitored”, the ADAPT monitoring UI can be accessed to see the monitored application:

```
http: //IP where the container is deployed:10080
```

First, the user needs to be logged:

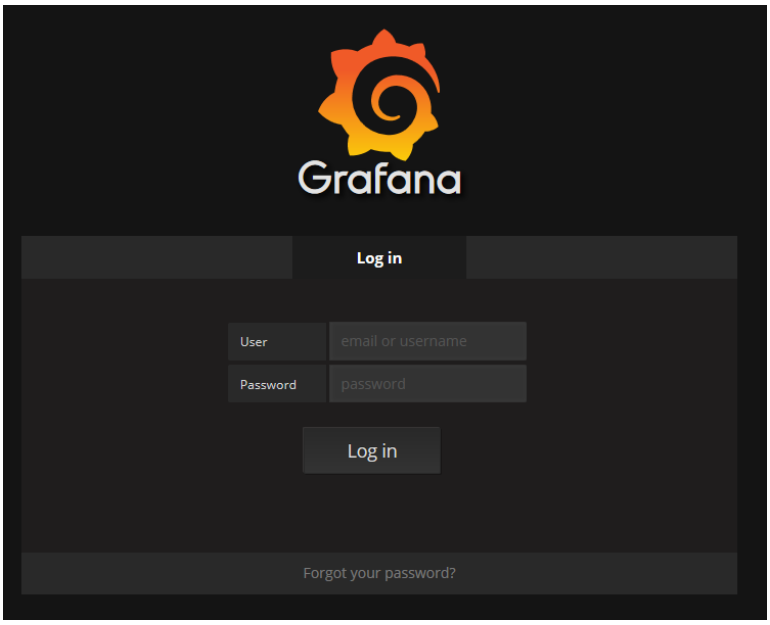

Figure 23. Grafana Log in interface.

A default user has been set up for the ADAPT MM:

- User: decide
- Password: decide

For a better visualization of the metrics, user may change the preferences of the dashboard to the "light" mode:

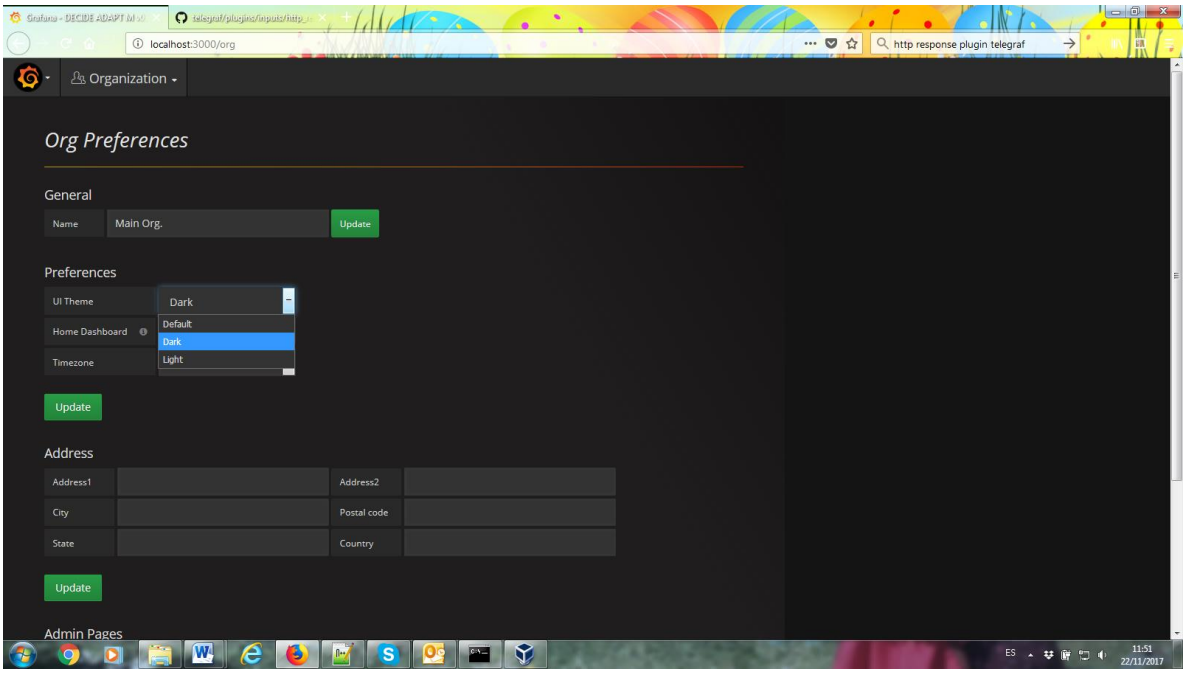

Figure 24. Preferences menu

Then, in the dashboards tab, the user needs to select the multi-cloud Application availability and Performance metrics for multi-cloud application (current version of the ADAPT monitoring UI M24):

Figure 25. Dahsboards tab

In the dashboard, the established metrics are monitored for the Sock Shop application.

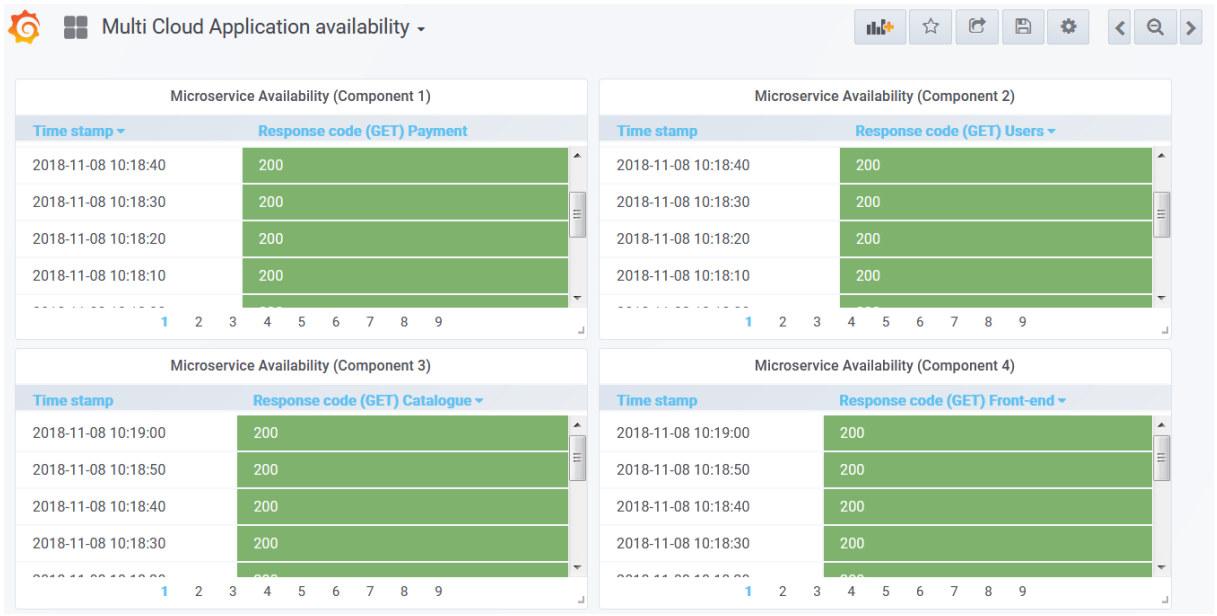

Figure 26. ADAPT Monitoring availability dashboard.

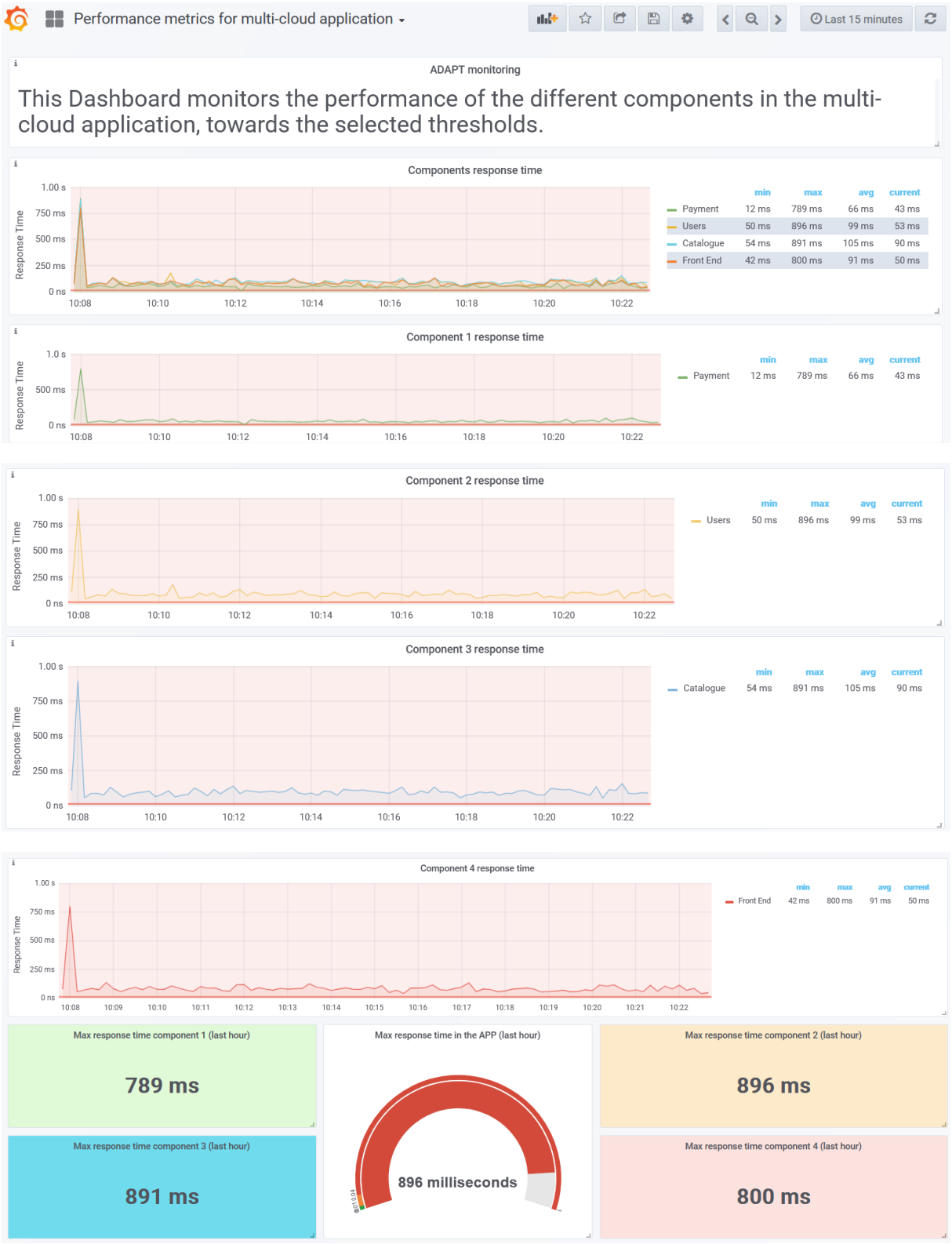

Figure 27 ADAPT monitoring performance dashboard

These Dashboards are automatically shown in the DevOps framework when ADAPT MM is used in the integrated DECIDE environment. The link to the corresponding UI is also included in the corresponding field in the application description (monitoring/url) and loaded by the DevOps framework.

2.2.4 Licensing information

This component is offered under MIT license.

2.2.5 Download

The source code is available in the EC portal for deliverables, included in the zip file for D4.8.

This intermediate release is available in the DECIDE open git repository, more precisely at the following address:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components

DECIDE_components/ADAPT/Monitoring (tag M24)

3 Violations Handler

3.1 Implementation

3.1.1 Functional description

The DECIDE ADAPT Violations Handler is in charge of receiving a violation alert from ADAPT monitoring or ACSmI monitoring and, depending on the application's technological risk, they will notify the operator and

a) automatically trigger a new simulation through OPTIMUS (low technological risk) or
b) notify the operator and give them a chance to modify settings before triggering a new simulation (in case of high technological risk).

The Violations Handler will also archive the violations to keep a history of alerts.

The main functionalities of the ADAPT Violations Handler are:

F1. Receive violation alerts: when a violation occurs, the monitoring components (ADAPT monitoring and ACSmI monitoring) will notify the violation handlers, which will process the violation.

F2. Retrieve technological risk: when the violation handlers receive a violation, they will retrieve the application's technological risk from the Application Description, in order to decide what action to carry out.

F3. Evaluate action to carry out: if the application's technological risk is low, the violation handler will automatically trigger a new simulation and the application will be redeployed according to the highest-ranked deployment schema. If it is high, the operator will have to confirm the deployment schema according to which the application will be redeployed. In both cases, the operator has to be notified of the violation.

F4: Notify operator: when a violation is received, the operator will receive an email containing relevant information about the alert.

F5: Send OPTIMUS a simulation order (low technological risk): if a violation has occurred, and the technological risk is low, the Violations Handler will automatically trigger a new simulation through OPTIMUS. The highest-ranked result will be passed to ADAPT, which will automatically redeploy the application according to it.

F6: Send OPTIMUS a simulation order (high technological risk): if a violation occurs but the technological risk is high, OPTIMUS will be triggered, but it will present the results of the new simulation to the user, who will have to select one of them before passing it to ADAPT Violations Handler will be implemented following an incremental approach adding features in the different releases of the tool (the first one was released on M12 and reported in D4.7 [1], and the last one will be reported in D4.9 on M30). In this second release, all expected functionalities are covered.

The following table details the relationship between the functional requirements - indicated in deliverable D4.2 [5] and updated for year 2- and the implemented functionalities, with a description of the coverage for each functionality.

Requirements covered by the prototype:

Table 2. Functionality covered by the M24 Violations Handler prototype.

Functionality	Req. ID	Coverage
F1	WP4-MR1, WP4-REQ15	The prototype is able to receive an alert from ADAPT and ACSml monitoring and extract the relevant information from it. Besides, it stores the history of alerts to be visualized from the ADAPT UI.
F2	WP4-MR11	The prototype retrieves the technological risk from the Application Description.
F3	WP4-MR8, WP4-MR9	Depending on the technological risk of the application, the prototype decides the action that will be carried out.
F4	WP4-MR9, WP4-MR10	The prototype sends the user an email with violation information when a violation is detected.
F5	WP4-MR1, WP4-MR8	The prototype sends an order to start a simulation to OPTIMUS and instructs ADAPT to automatically redeploy the application according to the highest-ranked schema.
F6	WP4-MR1, WP4-MR9	The prototype sends an order to start a simulation to OPTIMUS. Then, it presents the results of the simulation to the user, for them to choose their preferred deployment option.

Detailed functionality that this prototype offers:

This prototype is the second version of the Violations Handler component and offers the following functionalities:

- Receive and process alert: the prototype receives a violation and is able to parse this violation to extract the relevant information.
- Retrieve technological risk: the prototype checks the technological risk of the component associated to the received alert. The technological risk is a variable that is stored in the Application Description.
- Store the violation: the prototype stores in a MySQL database the history of violations to be visualized from the ADAPT UI.
- Decide action to carry out: depending on the value of the technological risk, the prototype will decide what action will be carried out (automatic redeployment if the risk is low, and manual redeployment if it is high).
- Send an email: the prototype sends an email to the user when a violation is received, with relevant information about the violation.
- Trigger OPTIMUS: when a violation occurs, the prototype triggers OPTIMUS to obtain a new deployment schema.
- Trigger ADAPT: after a violation, if the technological risk is low, the prototype calls ADAPT once OPTIMUS has obtained a new deployment schema, so the application is redeployed.

3.1.1.1 Fitting into overall DECIDE Architecture

The Violations Handler is the component that is in charge of receiving and managing violations and orchestrating the redeployment workflows. It is part of the ADAPT Key Result, located inside the ADAPT package in Figure 1 (as it has been noted, that picture depicts only the interfaces used or implemented by ADAPT. The whole figure with all the interfaces between the different tools can be found in D2.5 [7]).

Within ADAPT, the Violations Handler interacts with ADAPT M Violation detection to receive MCSLA violations, with ADAPT UI to visualize the history of violations, and with ADAPT DO to trigger a new deployment. It also interacts with other KRs: with OPTIMUS, to trigger a new simulation and to select the preferred schema in case of a high technological risk, and with ACSmI Monitoring to receive CSP violations. It also obtains the application's technological risk through the Application Description.

The following figure shows the general architecture of this component. For a more detailed description, see D4.2 [5]:

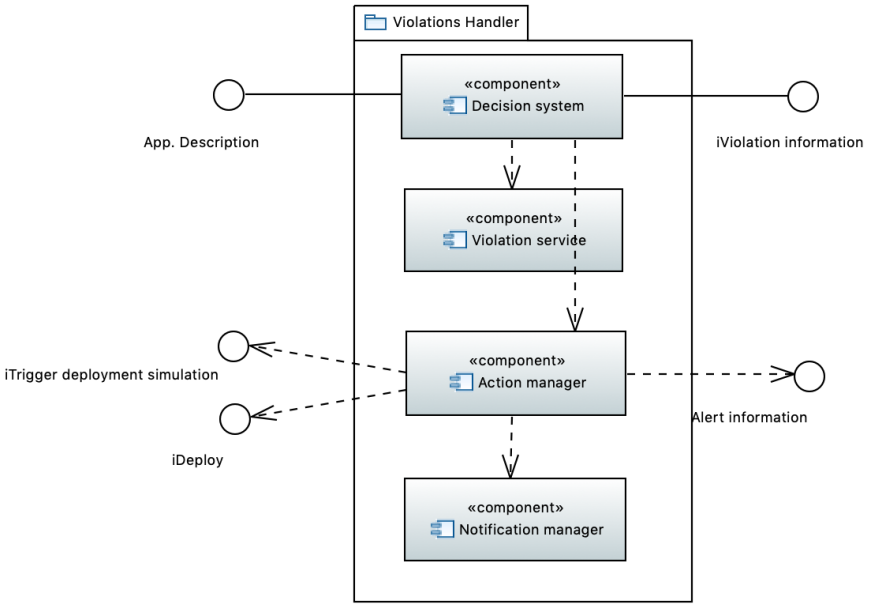

Figure 28. General architecture of the Violations Handler

3.1.2 Technical description

This section describes the technical details of ADAPT Violation handler.

3.1.2.1 Prototype architecture

From the proposed architecture that can be found in deliverable D4.1 [10] and that is shown in Figure 28. General architecture of the Violations Handler, the Violation Handler tool is composed of four main microservices, the **Decision system**, the **Action manager**, the **Violation service** and the **Notification manager**. All of these components are packaged within the Violation Handler framework, which is deployed on a Docker container and accessed through a REST API interface. Finally, the Violation Handler also stores and retrieves information from a MySQL database.

3.1.2.2 Components description

- **Decision system:** It is responsible of receiving the violation and determine which decision should be performed. For making this decision successfully, it should check the technological risk value saved in the application description by the developer. In case the technological risk is high, the action will require user authorization. With all this information, it infers a result and communicates with the ***Action manager*** notifying which action should take place. Additionally, it contacts with the ***Violation service*** for storing the violation information in the historic database, and with the **Notification manager** for sending an email to the user with the violation information.
- **Action manager:** This component must be able to contact with OPTIMUS and ADAPT microservices, trigger a new simulation process and handle the redeployment of the application when a new schema is generated and saved into the application description.

- **Violation service:** This service contacts with the violation repository and saves each violation received in the system. The interface allows to explore the violations history, providing a certain application URI or service id as a query parameter.
- **Notification manager:** Retrieves the email from the application description and reports the violation details to the developer. Every time a new violation is received in the system, a report is sent to the user.

3.1.2.3 Technical specifications

The Violation handler component is an independent microservice mainly developed using Java programming language, implemented using Spring Boot framework. It offers a restful API interface modelled using Swagger editor that enables the communication this the submodules contained in the component. This interface provides a suitable interaction layer for reporting alerts and consulting the database that contains all the violations.

API interface definition

The definition of the API interface is presented in figure below. The controller is deployed under */violationhandler* base URL.

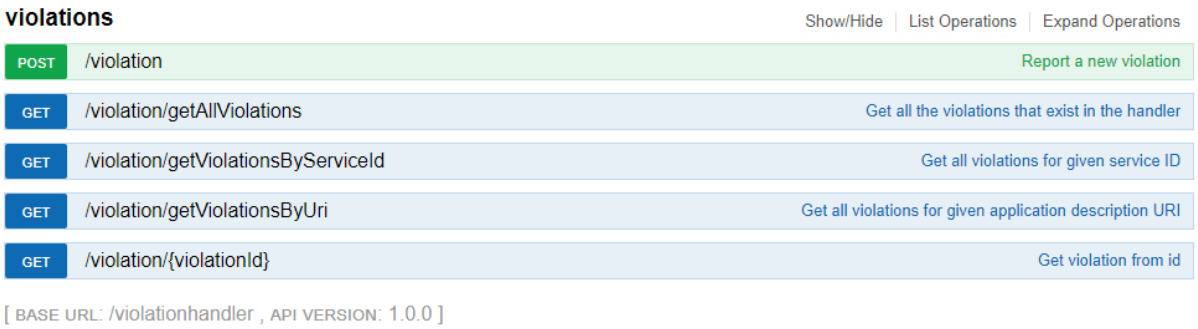

Figure 29. API interface definition for Violation Handler component.

For reporting a new violation in the system, both monitoring components, ACSmI and ADAPT, must send a POST request with the violation object in the body. The rest of the HTTP calls enables to communicate with the violation MySQL database and to filter the violations historic list.

After reporting a new violation, the decision system extracts the relevant information from it, and infers the subsequent action. In parallel, it extracts the developer email address from the application description and contacts with the Notification manager.

The Violation model object defined in the component is composed by different attributes, as it is presented in the figure below:

Violation {
app_desc_uri (string, *optional*): Uri of the application description location,
id (integer, *optional*),
nfr (string, *optional*): Affected NFR name,
service_id (string, *optional*): Id of the serice where violation occurs,
status (string, *optional*): Status of the violation,
timestamp (integer, *optional*): Timestamp with the violation monitoring time,
type (integer, *optional*): Violation source type, SLA (0) or CSP (1)
}

Figure 30. Model of the Violation object handled by the API requests.

Decision system

This service is responsible of deciding the best approach for handling the violation. First of all, it pulls the application description stored in the git repository, using the *app_desc_uri* violation attribute. For this, it has integrated the *AppManager* library component, which provides an intuitive set of functions for interacting with the application description information. Afterwards, it reads the *highTechnologicalRisk* JSON field, which is a *boolean* type. In case it is tagged as true, then the redeployment of the service needs user confirmation, so the email notification will be the only output. On the other hand, if the technological risk is low, then the redeployment process must be triggered automatically without the developer's consent. This redeployment process is delegated to the actions manager. Finally, the violation is saved in the database, so it can be tracked in the future by any DECIDE tool.

Actions manager

The main action performed by this component is the orchestration of the redeployment workflow in case of a low technological risk. The standard workflow behaviour for a deployment request follow a linear process, as shown in **¡Error! No se encuentra el origen de la referencia.** (the user icon indicates when user intervention is needed). Firstly, the user should execute the simulation (OPTIMUS) for generating the deployment schema that fits the best with the application microservices. Thus, the user must pick one schema for the recommended offered by OPTIMUS output. Then, after the MCSLA is created and the CSPs involved in the schema are contracted, the user must give ADAPT the order to redeploy the application.

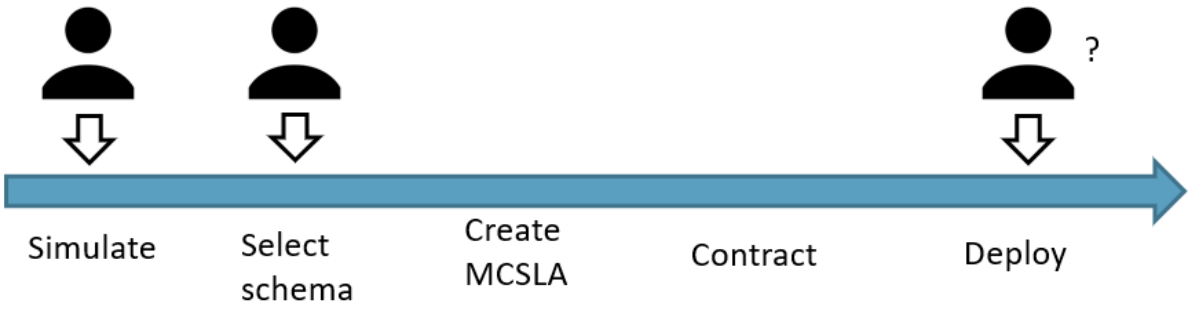

Figure 31. Deployment workflow

The redeployment process triggered by the Violations Handler (for low technological risk) must work similar to the standard deployment procedure. For that, the VH must substitute the user in those actions that are triggered by said user. Thus, the VH handler calls OPTIMUS'

iTriggerDeploymentSimulation interface for generating a new deployment schema. When the schema is ready, the Violations Handler is notified and ADAPT's *iDeploy* interface is executed.

Violation Service - MySQL database

For managing the violation history for a certain application or microservice, every input received in the component is stored inside a MySQL database. All the violation objects can be retrieved using queries calling the API interface methods. The Violation service is responsible of contacting with the violation repository.

Notification manager

This module is responsible for sending the violation information to the developer email. The email address is retrieved from the *application description*. The email sender module has been implemented using the Spring Boot Mail module, contained within Spring Boot framework.

3.2 Delivery and usage

3.2.1 Package information

The Violation handler component has two modules, the server that handles the violation reception and the MySQL database where all the reports are stored.

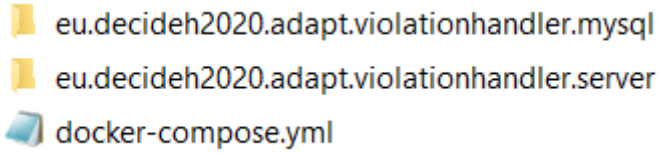

Figure 32. Source code structure of the Violation handler

In the directory structure we can distinguish two packages. The ***src/gen*** folder contains all the API interface definitions and model entities created in Swagger. The ***src/main/java*** directory contains the violation handler code and extends the generated Java classes for implementing the business logic.

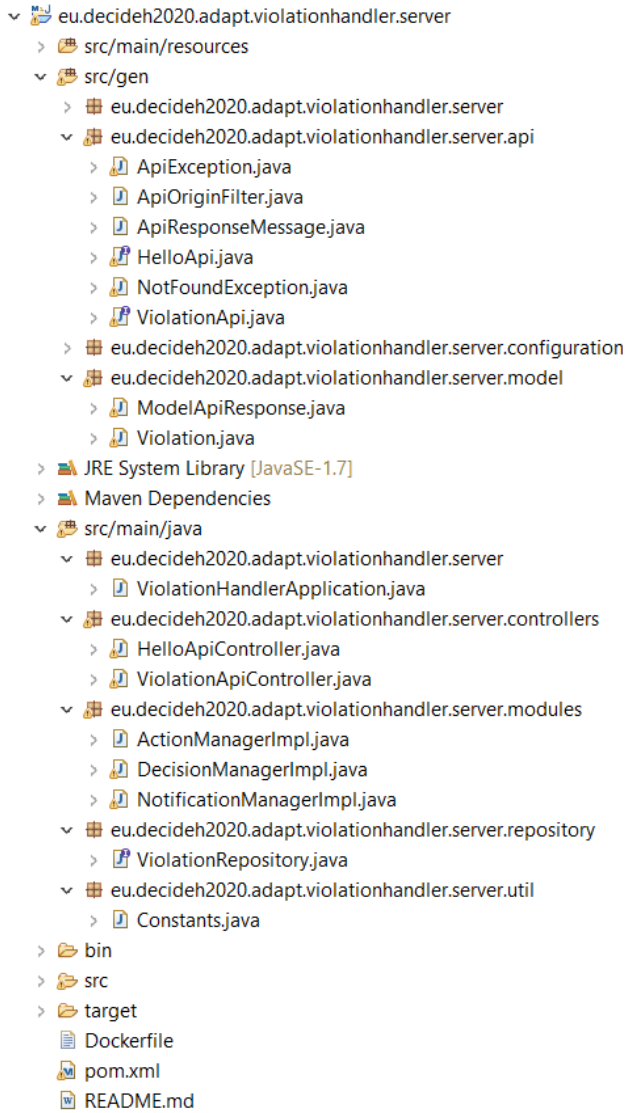

Figure 33. Structure of the *eu.decideh2020.adapt.violationhandler.server* module

The controller package contains the API implementation for the Violations Handler. This controller directly communicates with the rest of the modules. Furthermore, the modules package has all the subcomponents classes such as the *DecisionManager*, *ActionManager* and *NotificationManager*. To sum up, the *ViolationRepository* defines a CRUD (Create, Read, Update, Delete) interface for retrieving information from the MySQL database.

The main class is *ViolationHandlerApplication.java*, which initializes the Spring Boot application when it is invoked. The Violation entity model is used by all the API calls, and it is implemented in the *Violation.java* class, inside the generated files directory

The classes under the *util* package helps to map the received JSON to functions to ease data read and handling. It also manages the constant values of the component.

Finally, the *application.properties* file contains the deployment configuration for several environments.

MySQL database

The database is composed by one table with all the violations reported to the system. The database is named db_*violations*, which contains the table called *violation* with all the rows. The database architecture is presented in Figure 34. Figure 35 details the violation attributes and the variable type for each of them.

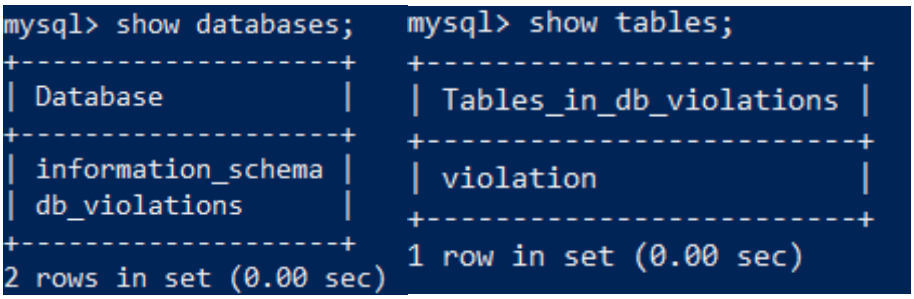

Figure 34. Database *db_violations* and *violation* table in MySQL shell

The *ViolationRepository* class is responsible of query the violations table and filter depending on the input parameter. In this version, there are two queries for accessing the data. The MySQL query for obtaining all the violations detected for a certain *service_id* is:

```
SELECT * FROM violation WHERE service_id = 'my_id'
```

The query for obtaining all the violations detected for a specific application description is:

```
SELECT * FROM violation WHERE app_desc_uri = 'my_uri'
```

These queries are executed by the controller when the *getSimulationsByServiceId* and *getViolationsByUri* are requested. The API is described in Figure 39.

3.2.2 Installation instructions

The Violations Handler have been developed in Java and is supported by a MySQL database. To install it, it is first required to have a Docker environment configured with the following deployment steps:

1. First, it is necessary to compile the Violations Handler repository cloned locally. For this, you should execute:
   •`git clone`
   `https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git`

2. For building the image, you must navigate to the root folder and build it as follows:
   `a. docker-compose build -f docker-compose-public.yml --no-cache`

3. Additionally, there is a docker-compose file which automatically deploys the Violations Handler server and the MySQL database. It also rebuilds the server if necessary:
   - `docker-compose up -d`

```
version: "3"
services:
    mysql_vh:
        image: mysql:5.5.59
        container_name: mysql_vh
        environment:
            MYSQL_ROOT_PASSWORD: root
            MYSQL_USER: admin
            MYSQL_PASSWORD: decide
            MYSQL_DATABASE: db_violations
        ports:
            - 3306:3306
    violationhandler:
        build: ./
        container_name: violationhandler
        depends_on:
            - mysql_vh
        ports:
            - 5000:5000
networks:
  decide:
    external: true
```

Figure 35 Docker compose configuration file.

```
IMAGE                                PORTS                    NAMES
violationhandler_violationhandler    0.0.0.0:5000->5000/tcp   violationhandler
mysql:5.5.59                         0.0.0.0:3306->3306/tcp   mysql_vh
```

Figure 36. Violations Handler's docker containers running

3.2.3 User Manual

The application is started by sending an HTTP POST request to ***violationhandler/violation***, with the following message body format describing the violation object. The API also allows to explore the violations stored, being able to query the results sending the application URI or the service id. A request body example is presented in Figure 38.

```
{
  "app_desc_uri": "http://github.com/my-user/my-project",
  "email": "abc123@gmail.com",
  "id": 0,
  "nfr": "availability",
  "service_id": "a1b2c3d4e5f6",
  "status": "detected",
  "timestamp": 0,
  "type": "0"
}
```


Figure 37. HTTP POST request body example containing the violation information

The Violations Handler has integrated the Swagger UI that enables to explore and consume the API interface directly from the browser. It can be accessed under http://localhost:5000/violationhandler.

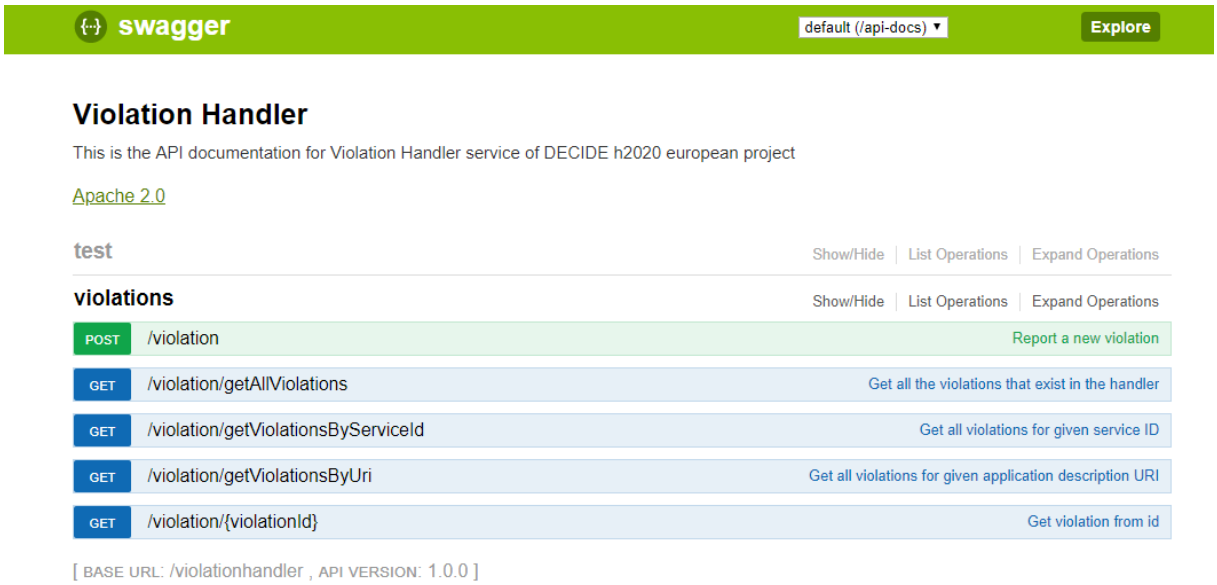

Figure 38. Swagger UI integrated within the Violation Handler.

To sum up, the component has been deployed in the DECIDE integration environment, and the microservice it is located in http://85.91.40.245:8095/violationhandler

3.2.4 Licensing information

This component is offered under the MIT license.

3.2.5 Download

The source code is available in the EC portal for deliverables, included in the zip file for D4.8.

This release is available in the DECIDE open git repository, more precisely at the following address:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git

DECIDE_Components/ADAPT/ViolationsHandler (tag M24)

4 Conclusions

In this document, the intermediate multi-cloud application monitoring is presented. The functional and the technical details of the components comprising this version: ADAPT Monitoring Manager and Violations Handlers are described and their installation processes and usage guides are detailed as well. Both components released provide new functionalities with respect to initial versions. ADAPT Monitoring Manager incorporates the ADAPT MM controller sub-component, Automatic configuration of ADAPT MM data collection sub-component, ADAPT MM violations detection sub-component, New dashboards in ADAPT MM UI, and integration with other components in DECIDE (App Controller and DevOps Framework). ADAPT VH includes the retrieval of information from the app description, the notification of violations functionality and the orchestration of the re-deployment workflow. The next release of this deliverable (D4.9 [11]) will include new functionalities and improvements for the current components so that the requirements defined in D4.2 [5] are fully covered.

In the next release, the envisioned new functionalities to be implemented for **ADAPT Monitoring** are:

- New metrics (performance throughput and scalability) will be included to support the different DECIDE NFRs and MCSLA monitoring.
- Automatic generation of the ADAPT MM UI dashboards
- Calls to ACSmI monitoring and VH
- Automatic reading of the influxDB

As for the **Violations Handler**, even though the basic set of functionalities is covered, the next release will focus on refining the component and improving the integration with the other DECIDE components.

5 References

[1] DECIDE Consortium, “D4.7-Initial multi-cloud application monitoring,” 2017.

[2] DECIDE consortium, “DECIDE DoA (Description of action),” 2016.

[3] C. McLuckie, “Perspective on multi-cloud (part 3 of 3) — Availability and multi-cloud,” [Online]. Available: https://blog.heptio.com/perspective-on-multi-cloud-part-3-of-3-availability-and-multi-cloud-5018762d2702. [Accessed 22 11 2018].

[4] ARTIST Consortium, “D7.1 Definition and extension of performance,” 2014.

[5] DECIDE Consortium, “D4.2-Intermediate DECIDE ADAPT Architecture,” 2018.

[6] DECIDE consortium, “D2.4 Detailed architecture v1,” 2017.

[7] DECIDE consortium, “D2.5-DECIDE detailed architecture v2,” 2018.

[8] WeaveWorks, “Sock Shop - Docker compose,” [Online]. Available: https://microservices-demo.github.io/deployment/monitoring-docker-compose.html. [Accessed 22 11 2018].

[9] DECIDE consortium, “D3.14-Intermediate multi-cloud native application composite CSLA definition,” 2018.

[10] DECIDE Consortium, “D4.1 Initial DECIDE ADAPT Architecture,” 2017.

[11] DECIDE Consortium, “D4.9-Final multi-cloud application monitoring,” 2019.

[12] Weaveworks, “Sock Shop: A microservices demo application,” [Online]. Available: https://microservices-demo.github.io/. [Accessed 25 10 2017].

6 Annex1: Telegraf parser library

This library has been implemented in the context of the DECIDE project to be able to configure the telegraf.conf file programmatically, according to the needs of DECIDE. The telegraf.conf file is written in TOML[10] format and up to DECIDE partners' knowledge there is not any open source parser that fulfils DECIDE requirements for the automatic configuration of the telegraf.conf file.

6.1 Pre-requirements

The only pre-requirement is to have Telegraf_ConfigParser libary (JAR file) installed in the Maven repository.

6.2 Downloading the Telegraf_ConfigParser library

Download the library from the DECIDE public repository (https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git):

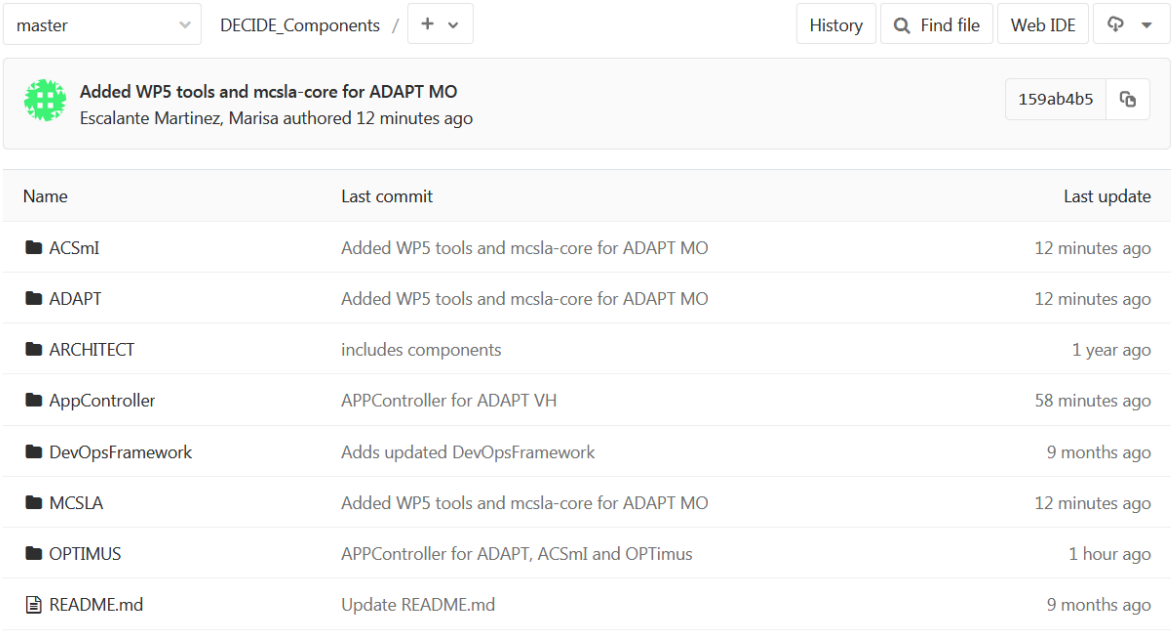

Figure 39. File structure for the Telegraf_ConfigParser folder.

Accessible in "ACSmI/Telegraf_ConfigParser/dist":

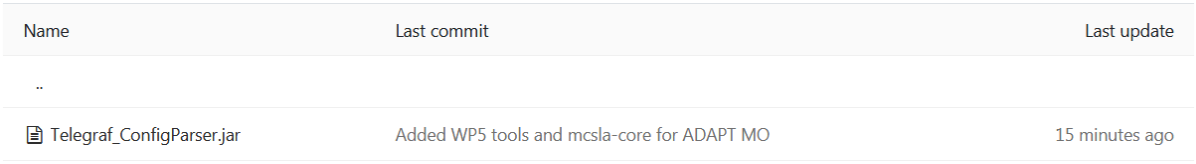

Figure 40. Telegraf_ConfigParser library

6.3 Installing the library in the Maven repository

Execute the following command in cmd:

[10] https://en.wikipedia.org/wiki/TOML

```
mvn install:install-file -Dfile=C:\D\Telegraf_ConfigParser.jar -DgroupId=eu.decideh2020 -DartifactId=eu.decideh2020.telegraf.configparser -Dversion=0.0.1-SNAPSHOT -Dpackaging=jar -DgeneratePom=true
```

Update the pom.xml in the corresponding project:

```
<dependency>
	<groupId>eu.decideh2020</groupId>
	<artifactId>eu.decideh2020.telegraf.configparser</artifactId>
	<version>0.0.1-SNAPSHOT</version>
</dependency>
```

6.4 Components and plugins

6.4.1 Generic component structure

Telegraf_ConfigParser collects the basic structure of the Telegraf configuration file ("telegraf.conf") with the following components structure:

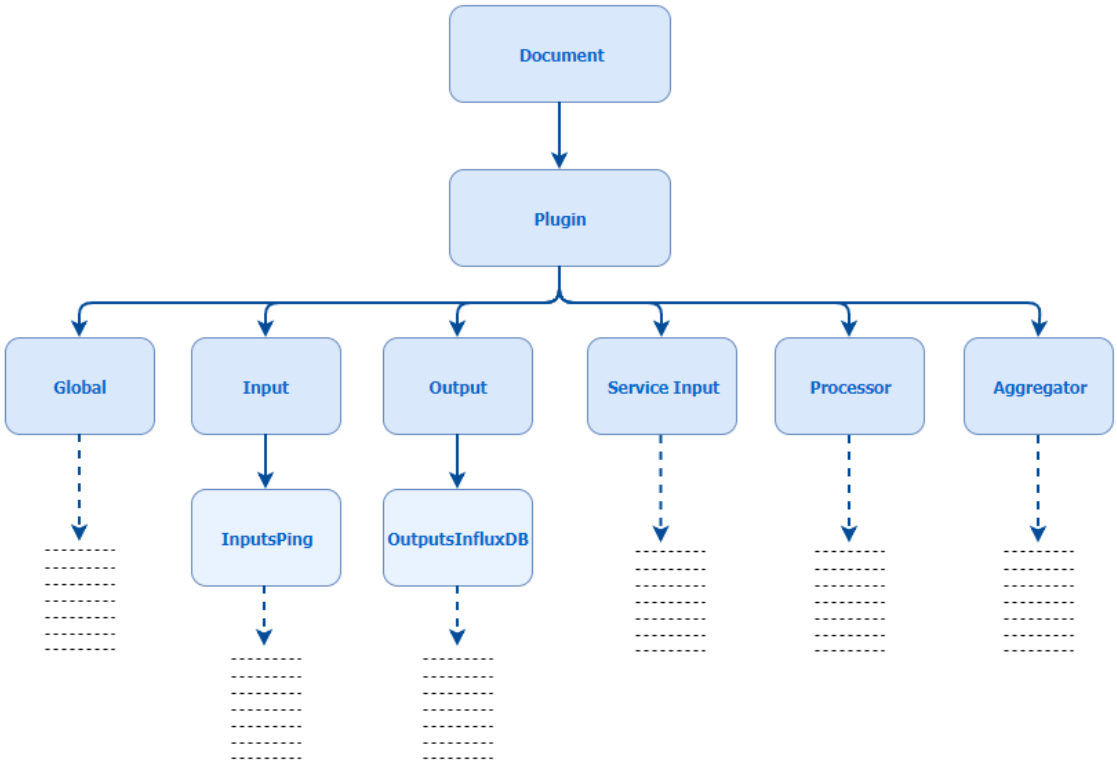

Figure 41. Telegraf.conf components structure

The Telegraf configuration file is collected into a "Document" object, with a list of "Plugin" objects. This "Plugin" object contains the following information:

- Global.

- Input.
- Output.
- Service Input.
- Processor.
- Aggregator.

6.4.2 General structure of a Telegraf Plugin.

The plugins inside the Telegraf configuration file are described using TOML[11] format. This library is focused into three of these plugins which are used by ADAPT monitoring and ACSmI monitoring.

- **HTTP Response**: Input plugin type. Corresponds to the "InputsHTTPResponse" component/class.
- **Ping**: Input plugin type. Corresponds to the "InputsPing" component/class.
- **InfluxDB**: Output plugin type. Corresponds to the OutputsInfluxDB" component/class.

6.4.3 "HTTP Response" plugin structure

Telegraf.conf file contains the following information for this plugin:

```
# HTTP/HTTPS request given an address a method and a timeout

[[inputs.http_response]]

  ## Server address (default http://localhost) Sock Shop Front End

  address = "http://sockshop.integration.decideh2020.lan.esilab.org:8082/customers"

  ## Set response_timeout (default 5 seconds)

  response_timeout = "5s"

  ## HTTP Request Method

  method = "GET"

  ## Whether to follow redirects from the server (defaults to false)

  follow_redirects = true

  ## HTTP Request Headers (all values must be strings)

  # [inputs.http_response.headers]

  #   Host = "github.com"


  ## Optional HTTP Request Body

  # body = '''

  # {'fake':'data'}

  # '''
```

[11] https://en.wikipedia.org/wiki/TOML

```
## Optional substring or regex match in body of the response
## response_string_match = "\"service_status\": \"up\""
## response_string_match = "ok"
## response_string_match = "\".*_status\".?:.?\"up\""

## Optional SSL Config
# ssl_ca = "/etc/telegraf/ca.pem"
# ssl_cert = "/etc/telegraf/cert.pem"
# ssl_key = "/etc/telegraf/key.pem"
## Use SSL but skip chain & host verification
# insecure_skip_verify = false
```

6.4.4 "Ping" plugin structure

Telegraf.conf file contains the following information for this plugin:

```
# # Ping given url(s) and return statistics
# [[inputs.ping]]
#   ## NOTE: this plugin forks the ping command. You may need to set capabilities
#   ## via setcap cap_net_raw+p /bin/ping
#   #
#   ## List of urls to ping
#   urls = ["www.google.com"] # required
#   ## number of pings to send per collection (ping -c <COUNT>)
#   count = 1
#   ## interval, in s, at which to ping. 0 == default (ping -i <PING_INTERVAL>)
#   ping_interval = 1.0
#   ## per-ping timeout, in s. 0 == no timeout (ping -W <TIMEOUT>)
#   timeout = 1.0
#   ## interface to send ping from (ping -I <INTERFACE>)
#   interface = ""
```

6.4.5 "InfluxDB" plugin structure

Telegraf.conf file contains the following information for this plugin:

```
# Configuration for influxdb server to send metrics to

[[outputs.influxdb]]

  ## The HTTP or UDP URL for your InfluxDB instance.  Each item should be

  ## of the form:

  ##   scheme "://" host [ ":" port]

  ##

  ## Multiple urls can be specified as part of the same cluster,

  ## this means that only ONE of the urls will be written to each interval.

  urls = ["http://localhost:8086"] # required

  ## The target database for metrics (telegraf will create it if not exists).

  database = "decideh2020adapt" # required


  ## Name of existing retention policy to write to.  Empty string writes to

  ## the default retention policy.

  retention_policy = ""

  ## Write consistency (clusters only), can be: "any", "one", "quorum", "all"

  write_consistency = "any"


  ## Write timeout (for the InfluxDB client), formatted as a string.

  ## If not provided, will default to 5s. 0s means no timeout (not recommended).

  timeout = "5s"

  # username = "telegraf"

  # password = "metricsmetricsmetricsmetrics"

  ## Set the user agent for HTTP POSTs (can be useful for log differentiation)

  # user_agent = "telegraf"

  ## Set UDP payload size, defaults to InfluxDB UDP Client default (512 bytes)

  # udp_payload = 512


  ## Optional SSL Config
```

```
# ssl_ca = "/etc/telegraf/ca.pem"

# ssl_cert = "/etc/telegraf/cert.pem"

# ssl_key = "/etc/telegraf/key.pem"

## Use SSL but skip chain & host verification

# insecure_skip_verify = false
```

6.4.6 Operations on the Telegraf.conf file

Telegraf_ConfigParser implements a package to manage the different operations which can be performed on the Telegraf.conf file:

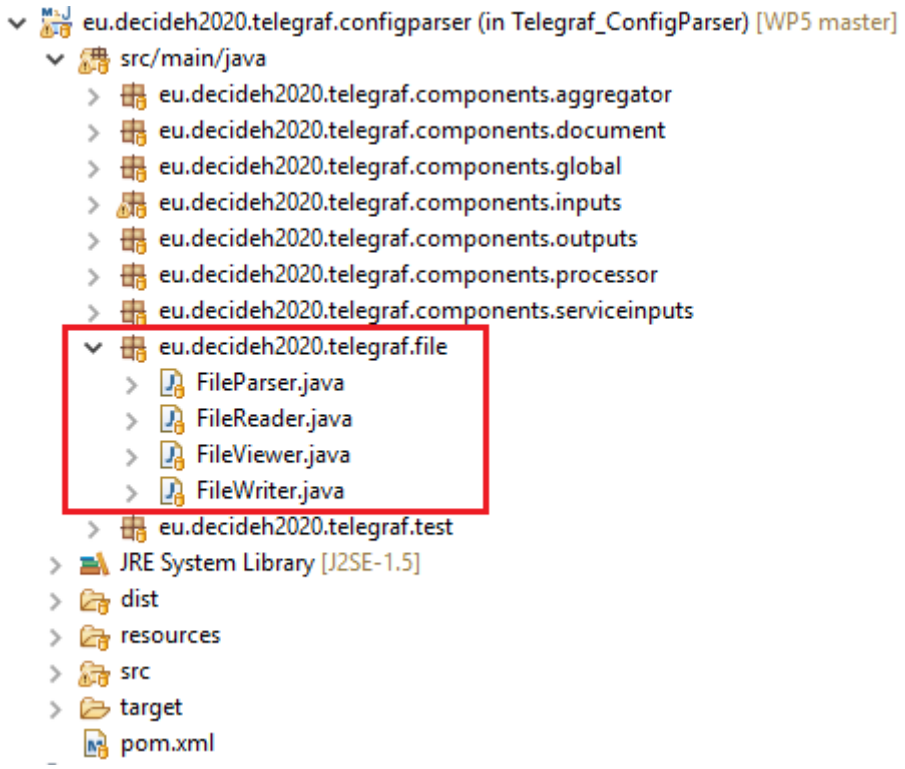

Figure 42. Java classes which implements the different operations to be performed in a Telegraf.conf file

Following these operations are described.:

- FileReader: Manages the reading operation in the configuration file.
- FileWriter: Manages the writing operations in the configuration file: Conversion of a plugin object to the file format, conversion of a document object into the file format, writing the file located in an absolute path, writing the file located in an absolute path or relative path (to be selected).
- FileParser: This class allows to operate in the "Documents" object: Commenting/uncommenting one/all the plugin properties, Commenting/uncommenting one/all the plugins, modify a property with one/several values, duplicate a plugin, delete a plugin, count the number if plugins in the file with the same name.
- FileViewer: This class allows to perform the operations to view in the console the configuration file loaded: visualize the list inside the configuration file, visualize the document object.

6.5 Usage example

Example for loading a telegraf.conf file:

```
try {
    FileParser parser = new FileParser("etc/telegraf/telegraf.conf");
} catch (IOException ioe) {
      ioe.printStackTrace();
} catch (ClassNotFoundException cnfe) {
      cnfe.printStackTrace();
} catch (IllegalAccessException iae) {
      iae.printStackTrace();
} catch (InstantiationException ie) {
      ie.printStackTrace();
} catch (CloneNotSupportedException cnse) {
      cnse.printStackTrace();
}
```

Perform the following operations:

1) Duplicates the “inputs.ping” twice.
2) Modify the “count” property of the original “inputs.ping” (index = 1).
3) Modify the “timeout” property of the first duplicated plugin “inputs.ping” (index = 2).
4) Delete the original “inputs.ping” (index = 1).
5) View the result in the console.
6) Write the new data in the telegraf configuration file.

```
try {
    FileParser parser = new FileParser("etc/telegraf/telegraf.conf");

    // 1) Duplicates the “inputs.ping” twice.
    parser.duplicatePlugin("inputs.ping");
    parser.duplicatePlugin("inputs.ping");

    // 2) Modify the  “count” property of the original “inputs.ping” (index
= 1).
    parser.changePropertyValue("inputs.ping", "count", "3");
```

```
    // 3) Modify the  “timeout” property of the first duplicated plugin
“inputs.ping” (index = 2).

    parser.changePropertyValue("inputs.ping", 2, "timeout", "1.5");


    // 4) Delete the original “inputs.ping” (index = 1).

    parser.deletePlugin("inputs.ping");


    // 5) View the result in the console

    FileViewer.viewDocumentConsole(parser.getDocument());

    // 6) Write the new data in the telegraf configuration file.

    FileWriter.writeFile(

            "etc/telegraf/telegraf.conf", parser.getDocument());

} catch (Exception e) {

      e.printStackTrace();

}
```

7 Annex2: iStarstopmonitoring and iStorealertinformation interface specification

7.1 Overview

This API contains the methods to access the ADAPT MM functionalities.

7.1.1 Version information

Version : 0.0.1

7.1.2 URI scheme

Host : localhost:8080
BasePath : /monitoringmanager
Schemes : HTTP

7.1.3 Tags

application : adapt application tag

7.2 PathsPaths

7.2.1 createAlert

`POST /api/alert`

7.2.1.1 Description

Creates an alert to be included in ADAPT monitoring

7.2.1.2 Parameters

Type	Name	Schema
Body	**Alert** *required*	Alert

7.2.1.3 Responses

HTTP Code	Description	Schema
200	Alert Created	Alert
401	Unauthorized	No Content

7.2.1.4 Consumes

- `alert/json`

7.2.1.5 Produces

- `\*/*`[12]

7.2.1.6 Tags

- alert

[12] It does not produce any element

7.2.2 createApplication

`POST /api/applications`

7.2.2.1 Description

Creates an application to be included in ADAPT monitoring

7.2.2.2 Parameters

Type	Name	Schema
Body	**application** *required*	Application

7.2.2.3 Responses

HTTP Code	Description	Schema
200	Application monitoring Created	Application
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

7.2.2.4 Consumes

- `application/json`

7.2.2.5 Produces

- `\*/*`

7.2.2.6 Tags

- application

7.2.3 getAllApplications

`GET /api/applications`

7.2.3.1 Description

Gets the list of all the applications in ADAPT monitoring, including their id and the status

7.2.3.2 Responses

HTTP Code	Description	Schema
200	The list of all the monitoring elements in ADAPT monitoring is being returned	< Application > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

7.2.3.3 Consumes

- `application/json`

7.2.3.4 Produces

- */*

7.2.3.5 Tags

- application

7.2.4 getApplicationstatus

`GET /api/applications/{appdescuri}`

7.2.4.1 Description

Gets the status wrt monitoring of a given application

7.2.4.2 Parameters

Type	Name	Description	Schema
Path	**appdescuri** *required*	application description uri	string

7.2.4.3 Responses

HTTP Code	Description	Schema
200	Specific app monitoring is returned	Application
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

7.2.4.4 Consumes

- application/json

7.2.4.5 Produces

- */*

7.2.4.6 Tags

- application

7.2.5 updateApplication

`PUT /api/applications/{appdescuri}`

7.2.5.1 Description

Updates the monitoring status of a given application

7.2.5.2 Parameters

Type	Name	Description	Schema
Path	**appdescuri** *required*	application description uri	string

7.2.5.3 Responses

HTTP Code	Description	Schema

200	Status updated	Application
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

7.2.5.4 Consumes

- `application/json`

7.2.5.5 Produces

- `\*/*`

7.2.5.6 Tags

- application

7.2.6 deleteApplication

`DELETE /api/applications/{appdescuri}`

7.2.6.1 Parameters

Type	Name	Description	Schema
Path	**appdescuri** *required*	application description uri	string

7.2.6.2 Responses

HTTP Code	Description	Schema
200	App deleted form the monitoring list	No Content
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

7.2.6.3 Consumes

- `application/json`

7.2.6.4 Produces

- `\*/*`

7.2.6.5 Tags

- application

7.3 Definitions

7.3.1 Alert

Name	Schema
alid *required*	string
altype *required*	string
appdescuri *required*	string

7.3.2 Application

Name	Schema
appdescuri *required*	string
monid *optional*	string
password *optional*	string
status *optional*	string
user *optional*	string