Project Acronym:	HosmartAI
Grant Agreement number:	101016834 (H2020-DT-2020-1 – Innovation Action)
Project Full Title:	Hospital Smart development based on AI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016834

DELIVERABLE

D4.2 – Platform Architecture Design and Open APIs – Second version

Dissemination level:	PU -Public
Type of deliverable:	R -Report
Contractual date of delivery:	31 January 2023
Deliverable leader:	ITCL
Status - version, date:	Final – v1.0, 2023-01-31
Keywords:	Platform architecture, Open APIs

Executive Summary

This document presents the second version of the HosmartAI Platform Architecture and Open APIs. The first version of the HosmartAI architecture from D4.1 are analysed, identifying all the possible changes in the implementation, communication between pilots or internal elements, and transforming the new requirements into elements to be added to the new version of the architecture.

Considering that in the future it will be necessary to analyse the data inputs and outputs of each pilot to develop a common API to standardise the flow of information between the pilots and the platform, several tools have been studied to perform this function.

This deliverable also serves as documentation of all the tools to generate the necessary OpenAPIs, which in the next phase will be needed to finish linking all the elements that make up the architecture.

With this task clear, the architecture has been modified to meet emerging needs. An important change has been the inclusion of Nexus to manage the repositories because as time goes by, the number of tools and elements used is increasing and a powerful tool is needed to manage all the elements.

Changes have also been made to the security of the whole system, including new tools to increase the overall security level.

Deliverable leader:	Sergio Chico, Daniel Lozano (ITCL)
Contributors:	Sergio Chico, Daniel Lozano (ITCL) Pauline Loygue, Khaldoun Al Agha, Guy Pujolle (GC) Vasilis Charisis, Stelios Hadjidimitriou, Georgios Apostolidis, Ioannis Dimaridis (AUTH) Axel Kuhn, Gregorio Meyer, Luca Gilardi (EXYS) Makis Karadimas (INTRA)
Reviewers:	Manos Georgoudakis (TMA) Oliver Brinkmann (ETHZ)
Approved by:	Athanasios Poulakidas, Irene Diamantopoulou (INTRA)

Document History

Version	Date	Contributor(s)	Description
0.1	2022-02-24	Daniel Lozano (ITCL)	Document creation
0.2	2022-07-15	Sergio Chico (ITCL)	Open API section
0.3	2022-09-18	Daniel Lozano (ITCL)	New components section
0.4	2022-10-24	Sergio Chico (ITCL)	Section assignment
0.5	2022-12-27	Sergio Chico (ITCL) Daniel Lozano (ITCL) Adrián Gil (ITCL)	Chapter 2 (Short definition of platforms) Chapter 3 (OpenAPI generators)
0.6	2023-01-09	Pauline Loygue (GC) Khaldoun Al Agha (GC) Guy Pujolle (GC)	Router specification
0.7	2023-01-10	Axel Kuhn (EXYS) Gregorio Meyer (EXYS) Luca Gilardi (EXYS)	Section 4.2.2 Security VM update
0.8	2023-01-11	Daniel Lozano (ITCL) Sergio Chico (ITCL)	Format and section fixing review
0.9	2023-01-18	Daniel Lozano (ITCL) Sergio Chico (ITCL)	Fix review issues noted by ETHZ
0.10	2023-01-23	Daniel Lozano (ITCL)	Fix review issues noted by TMA
1.0	2023-01-31	INTRA	QA and creation of the final submitted version

Table of Contents

List of Figures

List of Tables

Definitions, Acronyms and Abbreviations

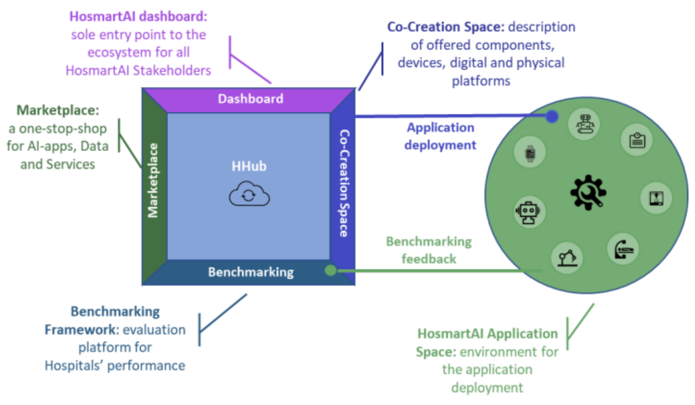
Acronym/ Abbreviation	Title
API	Application Programming Interface
DoA	Description of Action
EHR	Electronic Health Record
HHub	HosmartAI Hub
HL7	Health Level 7
HL-FAIR	Health Level 7 – Fast Healthcare Interoperability Resources
KPI	Key Performance Indicator
RAF	Reference Architecture Framework
SME	Subject Matter Expert
WP	Work Package

Term	Definition
Consortium	Group of beneficiaries that have signed the Consortium Agreement and the Grant Agreement (either directly as Coordinator or by accession through Form A).
Consortium Agreement	Contractual document signed by all the beneficiaries (and not the EC), explaining how the Consortium is managed and works together.
Deliverable Leader	Responsible for ensuring that the content of the deliverable meets the required expectations, both from a contractual point of view and in terms of usage within the project. Is also responsible for ensuring that the deliverable follows the deliverable process and is delivered on time.
Description of Action	Annex 1 to the Grant Agreement. It contains information on the work packages, deliverables, milestones, resources, and costs of the beneficiaries, as well as a text with a detailed description of the action. The DoA is made of Part A (structured data collected in web forms and Workplan tables) and Part B (text document describing the action elements).
Dissemination	EC term for the communication of information to a wide audience.
Grant Agreement	The contractual document which defines the contractual scope of the HosmartAI project. It is signed between the EC and the beneficiaries.

1 Introduction

1.1 Project Information

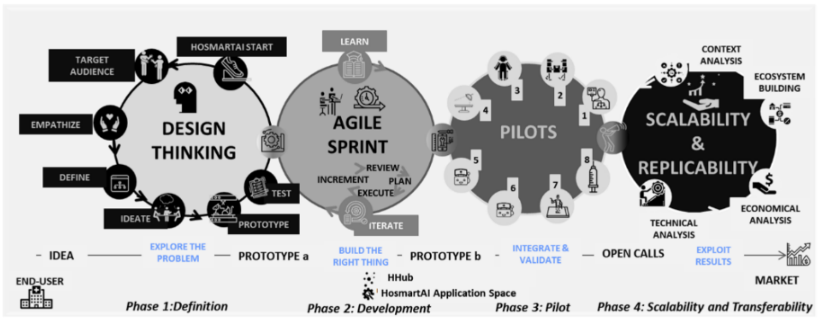
VISION	The HosmartAI vision is a strong, efficient, sustainable, and resilient European **Healthcare system** benefiting from the capacities to generate an impact of the technology European Stakeholders (SMEs, Research centers, Digital Hubs, and Universities).
MISSION	The HosmartAI mission is to guarantee the **integration** of Digital and Robot technologies in new Healthcare environments and the possibility to analyze their benefits by providing an **environment** where digital healthcare tool providers will be able to design and develop AI solutions as well as a space for the instantiation and deployment of AI solutions.


HosmartAI dashboard: sole entry point to the ecosystem for all HosmartAI Stakeholders
Co-Creation Space: description of offered components, devices, digital and physical platforms
Marketplace: a one-stop-shop for AI-apps, Data and Services
Dashboard
HHub
Marketplace
Co-Creation Space
Benchmarking
Application deployment
Benchmarking feedback
Benchmarking Framework: evaluation platform for Hospitals' performance
HosmartAI Application Space: environment for the application deployment

HosmartAI will create a common open Integration **Platform** with the necessary tools to facilitate and measure the benefits of integrating digital technologies (robotics and AI) in the healthcare system.

A central **hub** will offer multifaceted lasting functionalities (Marketplace, Co-creation space, Benchmarking) to healthcare stakeholders, combined with a collection of methods, tools, and solutions to integrate and deploy AI-enabled solutions. The **Benchmarking** tool will promote the adoption in new settings while enabling a meeting place for technology providers and end-users.

Eight Large-Scale Pilots will implement and evaluate improvements in medical diagnosis, surgical interventions, prevention and treatment of diseases, and support for rehabilitation and long-term care in several Hospital and care settings. The project will target different **medical** aspects or manifestations such as Cancer (Pilot #1, #2, and #8); Gastrointestinal (GI) disorders (Pilot #1); cardiovascular diseases (Pilot #1, #4, #,5 and #7); Thoracic Disorders (Pilot #5); Neurological diseases (Pilot #3); Elderly Care and Neuropsychological Rehabilitation (Pilot #6); Foetal Growth Restriction (FGR) and Prematurity (Pilot #1).



To ensure a user-centered approach, harmonization in the process (e.g., regarding ethical aspects, standardization, and robustness both from a technical and social and healthcare perspective), the **living lab** methodology will be employed. HosmartAI will identify the appropriate instruments (**KPI**) that measure efficiency without undermining access or quality of care. Liaison and cooperation activities with relevant stakeholders and **open calls** will enable ecosystem building and industrial clustering.

HosmartAI brings together a **consortium** of leading organizations (3 large enterprises, 8 SMEs, 5 hospitals, 4 universities, 2 research centers and 2 associations – see Table 1) along with several more committed organizations (Letters of Support provided).

Table 1: The HosmartAI consortium.

Number[1]	Name	Short name
1 (CO)	INTRASOFT INTERNATIONAL SA	**INTRA**
1.1 (TP)	INTRASOFT INTERNATIONAL SA	**INTRA-LU**
2	PHILIPS MEDICAL SYSTEMS NEDERLAND BV	**PHILIPS**
3	VIMAR SPA	**VIMAR**
4	GREEN COMMUNICATIONS SAS	**GC**
5	TELEMATIC MEDICAL APPLICATIONS EMPORIA KAI ANAPTIXI PROIONTON TILIATRIKIS MONOPROSOPIKI ETAIRIA PERIORISMENIS EYTHINIS	**TMA**
6	ECLEXYS SAGL	**EXYS**
7	F6S NETWORK IRELAND LIMITED	**F6S**
7.1 (TP)	F6S NETWORK LIMITED	**F6S-UK**
8	PHARMECONS EASY ACCESS LTD	**PhE**
9	TERAGLOBUS LATVIA SIA	**TGLV**
10	NINETY ONE GMBH	**91**
11	EIT HEALTH GERMANY GMBH	**EIT**
12	UNIVERZITETNI KLINICNI CENTER MARIBOR	**UKCM**
13	SAN CAMILLO IRCCS SRL	**IRCCS**
14	SERVICIO MADRILENO DE SALUD	**SERMAS**
14.1 (TP)	FUNDACION PARA LA INVESTIGACION BIOMEDICA DEL HOSPITAL UNIVERSITARIO LA PAZ	**FIBHULP**
15	CENTRE HOSPITALIER UNIVERSITAIRE DE LIEGE	**CHUL**
16	PANEPISTIMIAKO GENIKO NOSOKOMEIO THESSALONIKIS AXEPA	**AHEPA**
17	VRIJE UNIVERSITEIT BRUSSEL	**VUB**
18	ARISTOTELIO PANEPISTIMIO THESSALONIKIS	**AUTH**
19	EIDGENOESSISCHE TECHNISCHE HOCHSCHULE ZUERICH	**ETHZ**
20	UNIVERZA V MARIBORU	**UM**

[1] CO: Coordinator. TP: linked third party.

Number[1]	Name	Short name
21	INSTITUTO TECNOLÓGICO DE CASTILLA Y LEON	**ITCL**
22	FUNDACION INTRAS	**INTRAS**
23	ASSOCIATION EUROPEAN FEDERATION FORMEDICAL INFORMATICS	**EFMI**
24	FEDERATION EUROPEENNE DES HOPITAUX ET DES SOINS DE SANTE	**HOPE**

1.2 Document Scope

The deliverable aims to provide the second version of the HosmartAI architecture, based on all the changes and new necessities, in addition to the analysis of the technical requirements for each pilot since the first version was created. Also, this deliverable provides other key elements, like security, data privacy and interoperability methods to complete the HosmartAI architecture.

In addition, this deliverable aims to specify the communication between elements of the architecture through the OpenAPI specification.

1.3 Document Structure

This document is comprised of the following chapters:

Chapter 1 presents an introduction to the project and the document.

Chapter 2 describes the platforms integrated with the HosmartAI platform.

Chapter 3 documents the tools used for the OpenAPI specification.

Chapter 4 explains and describes the second version of the HosmartAI architecture.

Chapter 5 offers some concluding remarks.

2 Integrated Open Platforms

2.1 Digital Platforms

A digital platform is a software that manages one or more big functionalities. This software could be hosted by a third party and could be installed in the HosmartAI HUB or in any of the pilot's developments. Platforms created by partners for the HosmartAI project are also considered Digital platforms.

2.1.1 Acumos AI Platform

Acumos AI is an open-source platform and framework that makes it easy to build, share, and deploy AI apps. Acumos standardizes the infrastructure stack and components required to run an out-of-the-box general AI environment.

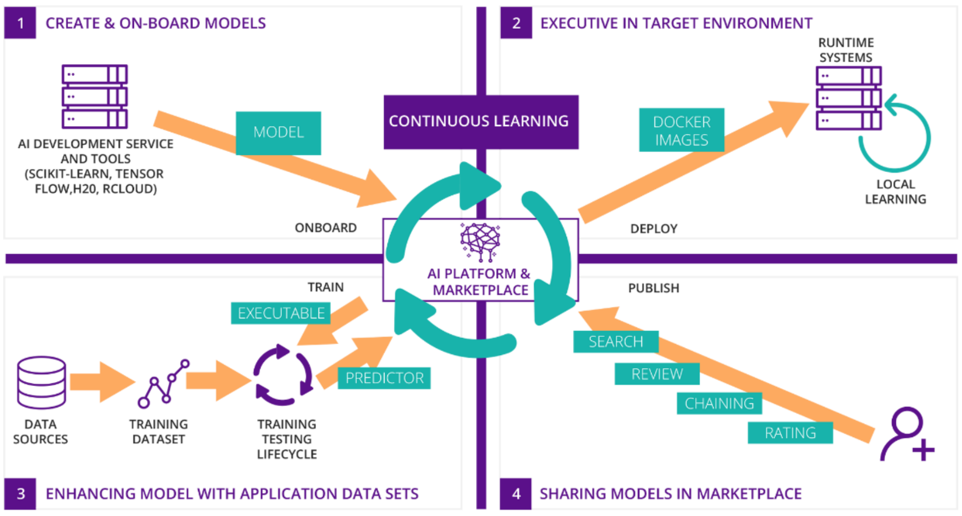

1 CREATE & ON-BOARD MODELS
AI DEVELOPMENT SERVICE AND TOOLS (SCIKIT-LEARN, TENSOR FLOW,H20, RCLOUD)
MODEL
ONBOARD
CONTINUOUS LEARNING
2 EXECUTIVE IN TARGET ENVIRONMENT
RUNTIME SYSTEMS
DOCKER IMAGES
LOCAL LEARNING
DEPLOY
AI PLATFORM & MARKETPLACE
TRAIN
EXECUTABLE
PREDICTOR
DATA SOURCES
TRAINING DATASET
TRAINING TESTING LIFECYCLE
3 ENHANCING MODEL WITH APPLICATION DATA SETS
PUBLISH
SEARCH
REVIEW
CHAINING
RATING
4 SHARING MODELS IN MARKETPLACE

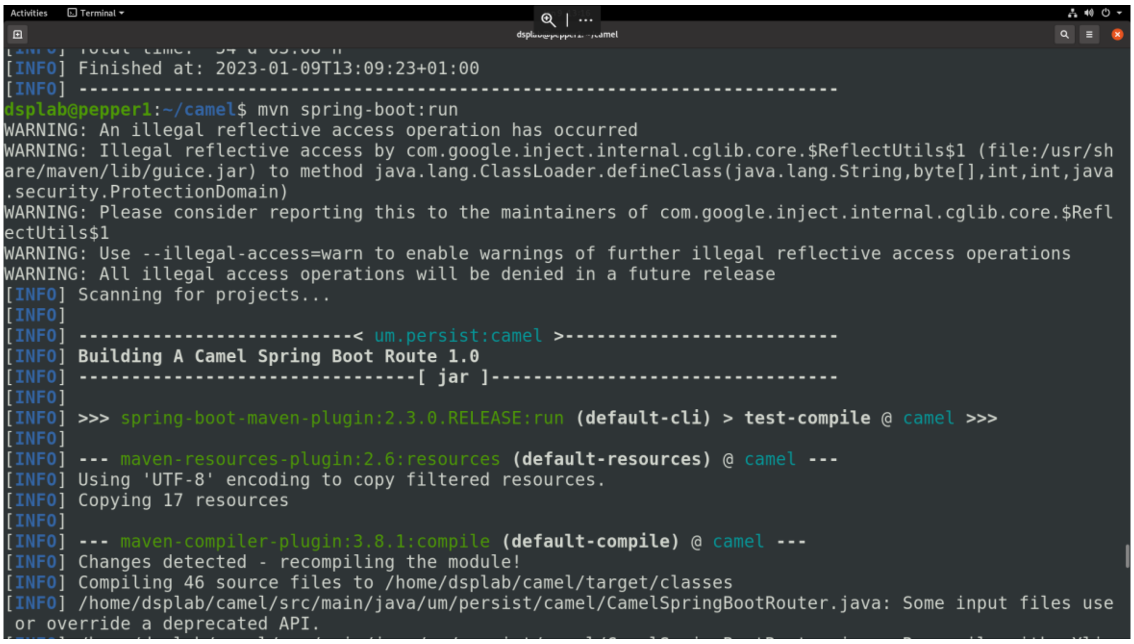
Figure 1: Acumos architecture.

In HosmartAI, the Acumos AI Platform is used as a tool to import AI4EU models, which can later be updated, adapted to HosmartAI use cases and deployed to the cloud or on-premises infrastructure.

2.1.2 Apache Maven

Apache Maven [[REF-06] is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. Maven addresses two aspects of building software: how software is built and its dependencies. It works as follows, an XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging. Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository and stores them in a local cache. This local cache of downloaded artifacts can also be updated with artifacts created by local projects. Public repositories can also be updated.

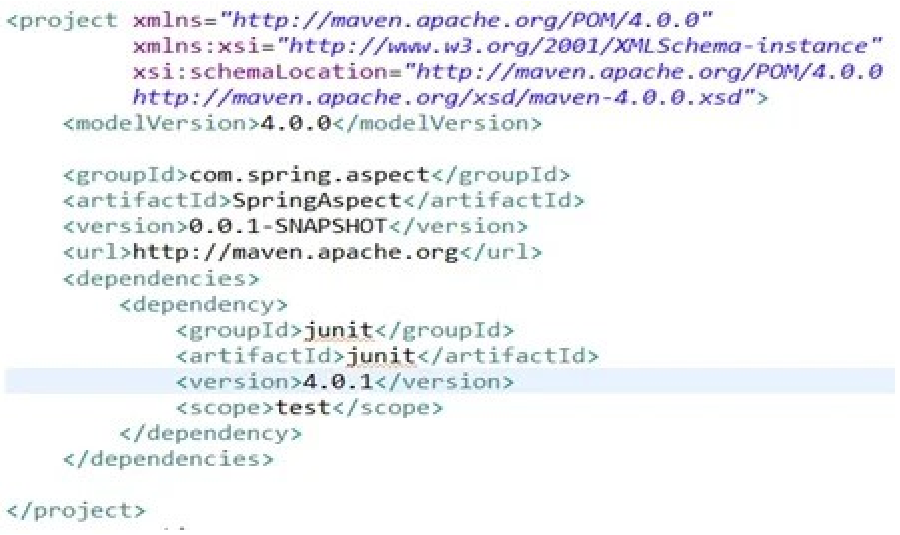
UM uses Maven to build, test and run Apache Camel and Apache Spring Boot Java-based applications that represent microservices for the HOSMARTAI project. For development, Maven is executed in the Linux-based terminal with its default terminal logger.



```
[INFO] Finished at: 2023-01-09T13:09:23+01:00
[INFO] ------------------------------------------------------------------------
dsplab@pepper1:~/camel$ mvn spring-boot:run
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1 (file:/usr/sh
are/maven/lib/guice.jar) to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java
.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.$Refl
ectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------------< um.persist:camel >--------------------------
[INFO] Building A Camel Spring Boot Route 1.0
[INFO] --------------------------------[ jar ]--------------------------------
[INFO]
[INFO] >>> spring-boot-maven-plugin:2.3.0.RELEASE:run (default-cli) > test-compile @ camel >>>
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ camel ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 17 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ camel ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 46 source files to /home/dsplab/camel/target/classes
[INFO] /home/dsplab/camel/src/main/java/um/persist/camel/CamelSpringBootRouter.java: Some input files use
 or override a deprecated API.
```

Figure 2: CLI interface for the Maven deployment & implementation in pilot 5.

The main benefit of the use of Maven is its plugin-based architecture that allows it to make use of any application controllable through standard input, i.e., POMs. An example POM of an XML specification is highlighted in the figure below.



```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.spring.aspect</groupId>
    <artifactId>SpringAspect</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.0.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

Figure 3: example of POM file in Pilot 5.

POM is an XML representation of a Maven project held in a file named `pom.xml`. The POM contains all necessary information about a project, as well as configurations of plugins to be used during the build process. It is the declarative manifestation of the "who", "what", and "where", while the build lifecycle is the "when" and "how". A project contains configuration files, as well as the developers involved and the roles they play, the defect tracking system, the organization and licenses, the URL of where the project lives, the project's dependencies, and all the other little pieces that come into play to give code life. It is a one-stop shop for all things concerning the project. In fact, in the Maven world, a project does not need to contain any code at all, merely a `pom.xml`.

2.1.3 Apache Camel and Spring Boot

Apache Camel [REF-07] is an open-source integration framework that empowers you to quickly and easily integrate various systems consuming or producing data. Apache Camel provides to the HosmartAI platform the base objects, commonly needed implementations, debugging tools, a configuration system.

Apache Spring Boot [REF-08] is a tool that allows to set up a Spring-based application with minimal configuration and setup. It creates stand-alone Spring applications, provides dependencies and configuration to simplify the building process.

Camel support for Spring Boot provides auto-configuration of the Camel and starters for many Camel components.

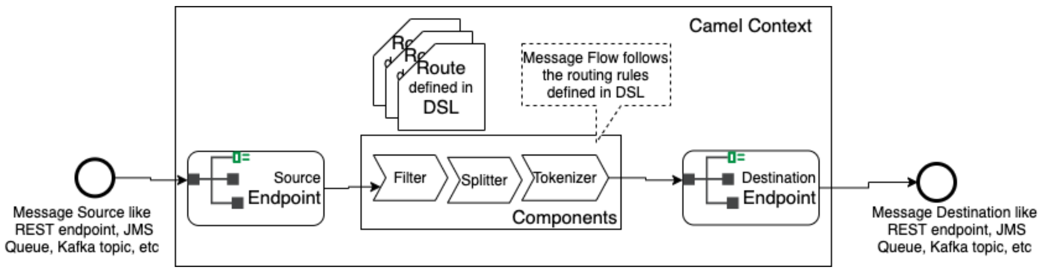
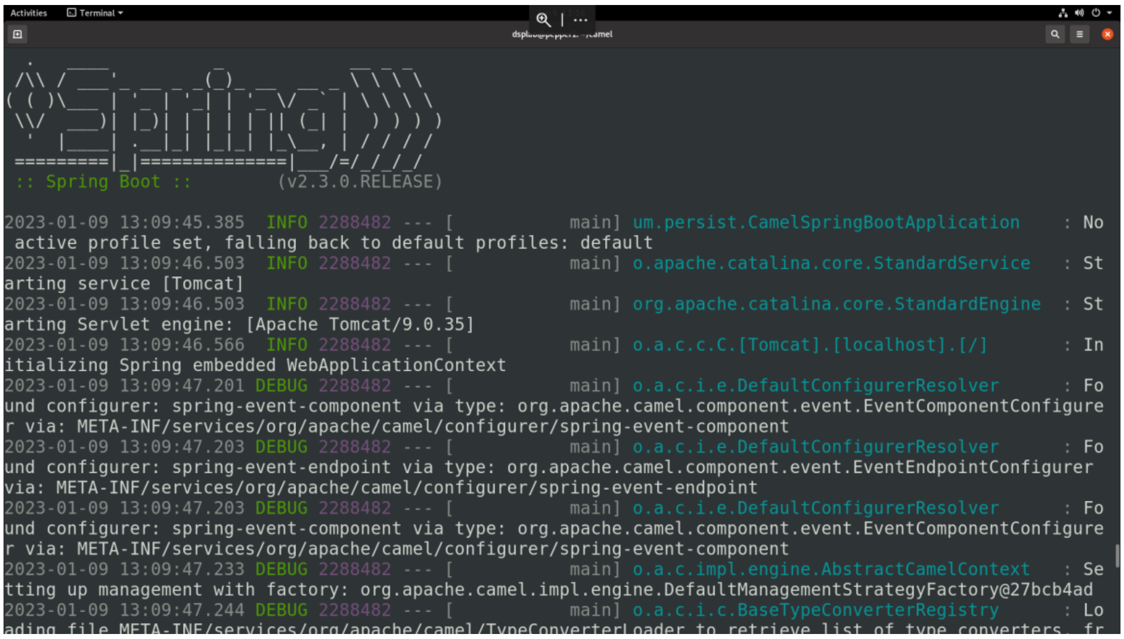

Camel Context
Route defined in DSL
Message Flow follows the routing rules defined in DSL
Source Endpoint
Filter
Splitter
Tokenizer
Components
Destination Endpoint
Message Source like REST endpoint, JMS Queue, Kafka topic, etc
Message Destination like REST endpoint, JMS Queue, Kafka topic, etc

Figure 4: Generic approach to the Apache Camel Spring Boot.

Spring Boot applications that are deployed at the UM side contain a typical Spring Boot structure with pom.xml which contains dependency descriptors for each needed component that is used to deploy a service. Some of those are the Swagger, Apache Camel, and Apache Tomcat.



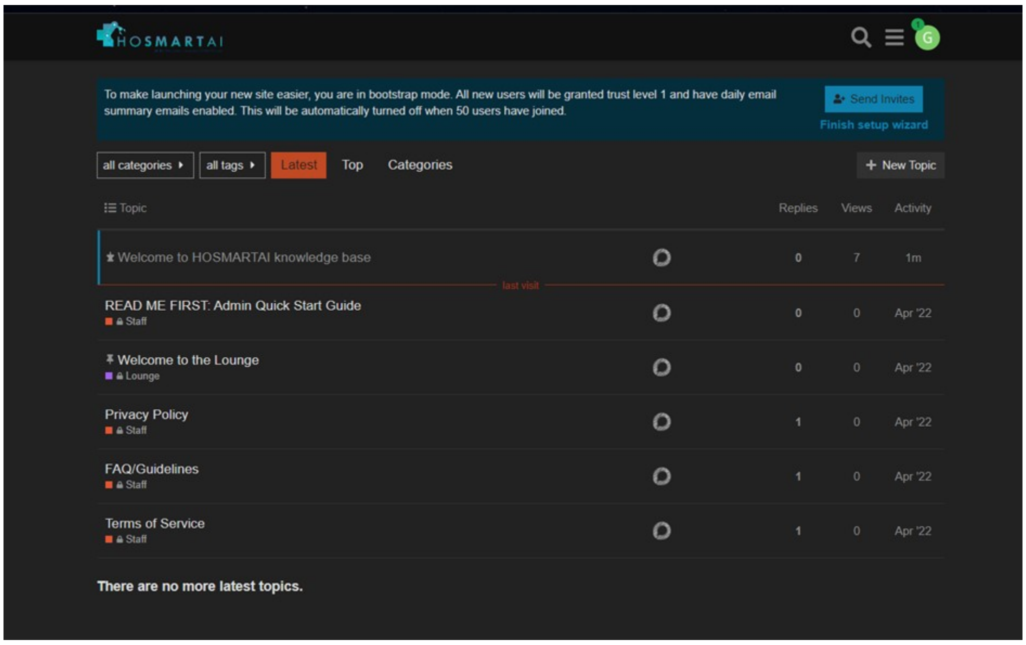
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.3.0.RELEASE)

2023-01-09 13:09:45.385  INFO 2288482 --- [           main] um.persist.CamelSpringBootApplication    : No
 active profile set, falling back to default profiles: default
2023-01-09 13:09:46.503  INFO 2288482 --- [           main] o.apache.catalina.core.StandardService   : St
arting service [Tomcat]
2023-01-09 13:09:46.503  INFO 2288482 --- [           main] org.apache.catalina.core.StandardEngine  : St
arting Servlet engine: [Apache Tomcat/9.0.35]
2023-01-09 13:09:46.566  INFO 2288482 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : In
itializing Spring embedded WebApplicationContext
2023-01-09 13:09:47.201 DEBUG 2288482 --- [           main] o.a.c.i.e.DefaultConfigurerResolver      : Fo
und configurer: spring-event-component via type: org.apache.camel.component.event.EventComponentConfigure
r via: META-INF/services/org/apache/camel/configurer/spring-event-component
2023-01-09 13:09:47.203 DEBUG 2288482 --- [           main] o.a.c.i.e.DefaultConfigurerResolver      : Fo
und configurer: spring-event-endpoint via type: org.apache.camel.component.event.EventEndpointConfigurer
via: META-INF/services/org/apache/camel/configurer/spring-event-endpoint
2023-01-09 13:09:47.203 DEBUG 2288482 --- [           main] o.a.c.i.e.DefaultConfigurerResolver      : Fo
und configurer: spring-event-component via type: org.apache.camel.component.event.EventComponentConfigure
r via: META-INF/services/org/apache/camel/configurer/spring-event-component
2023-01-09 13:09:47.233 DEBUG 2288482 --- [           main] o.a.c.impl.engine.AbstractCamelContext   : Se
tting up management with factory: org.apache.camel.impl.engine.DefaultManagementStrategyFactory@27bcb4ad
2023-01-09 13:09:47.244 DEBUG 2288482 --- [           main] o.a.c.i.c.BaseTypeConverterRegistry      : Lo
ading file META-INF/services/org/apache/camel/TypeConverterLoader to retrieve list of type converters, fr
```

Figure 5: CLI of Apache Camel Spring Boot implementation for Pilot 5.

One of the main benefits of the Apache Camel Spring Boot [REF-09] deployment is the use of Camel routes which are detected automatically in the Spring application context. Namely Spring Boot component provides auto-configuration for Apache Camel. The opinionated auto-configuration of the Camel context deployed in HosmartAI auto-detects Camel routes available in the Spring context and registers the key Camel utilities (like producer template, consumer template and the type of converter) as beans.

2.1.4 Discourse

Discourse is an open-source forum software. The main features that interest in the project include support for categorization and tagging of discussions, configurable access control, live updates, expanding link previews, infinite scrolling, and real-time notifications. It allows for a high level of customizability via its plugin architecture and its theming system.



To make launching your new site easier, you are in bootstrap mode. All new users will be granted trust level 1 and have daily email summary emails enabled. This will be automatically turned off when 50 users have joined.

Topic	Replies	Views	Activity
Welcome to HOSMARTAI knowledge base	0	7	1m
READ ME FIRST: Admin Quick Start Guide Staff	0	0	Apr '22
Welcome to the Lounge Lounge	0	0	Apr '22
Privacy Policy Staff	1	0	Apr '22
FAQ/Guidelines Staff	1	0	Apr '22
Terms of Service Staff	1	0	Apr '22

There are no more latest topics.

Figure 6: Discourse screen.

The purpose of Discourse is to act as both a knowledge base and a discussion forum. More specifically, it will contain information on HosmartAI devices, modules, services and practices. Users will also be possible to ask and discuss issues, queries related to HosmartAI offerings.

Pinned posts under categories and topics serve as knowledge base. Single Sign - On Authentication to Discourse is possible due to integration with the central Keycloak server while access is possible both from the HosmartAI Hub and from external URL.

2.1.5 Docker

Docker [REF-04] is a set of platforms as a service (PaaS) product that uses OS-level virtualization to deliver software in packages called containers. All the containers are isolated from one another and bundle their own software, libraries, and configuration files. Also, they can communicate with each other through well-defined channels. Because all the containers share the services of a single operating system kernel, they use fewer resources than virtual machines.

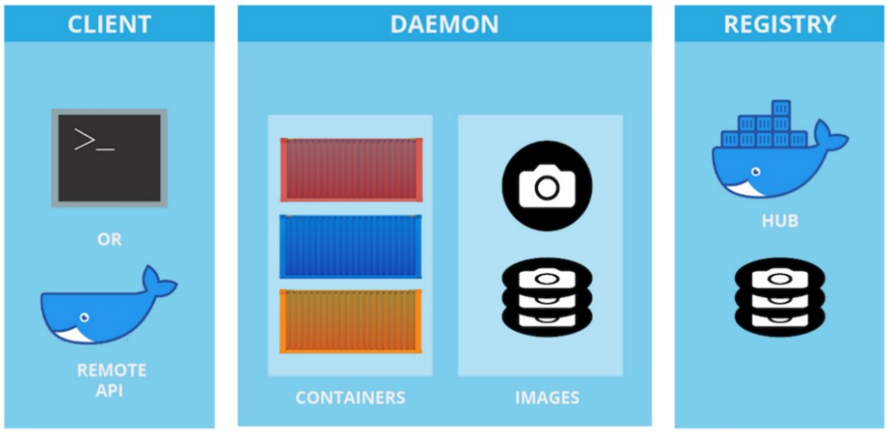
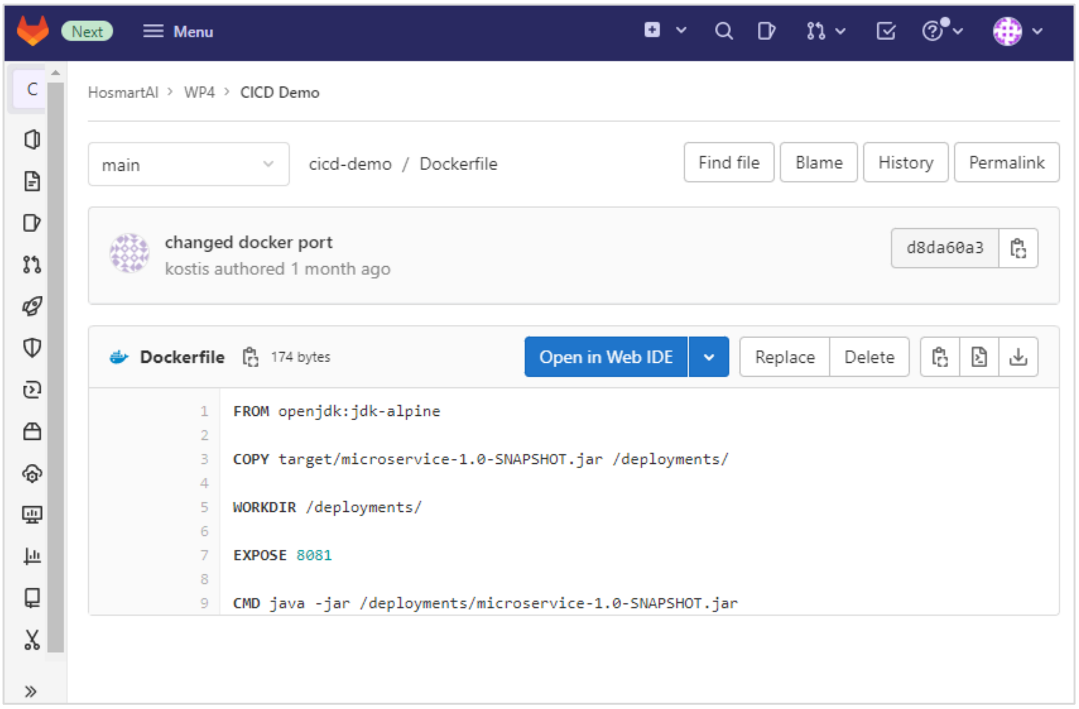


Figure 7: Docker components.

UM can deploy production-ready applications as Docker containers. This enables fast deployment of the same application for the future, ease of creating new instances, faster migrations and maintaining of applications.

2.1.5.1 Docker in Platform

Each component that belongs to or is associated with the HosmartAI platform is expected to use a Dockerfile, which contains the instructions on how it is built. We can find an example of such a Dockerfile in the GitLab repository of an example component that uses the CI/CD.



```
FROM openjdk:jdk-alpine

COPY target/microservice-1.0-SNAPSHOT.jar /deployments/

WORKDIR /deployments/

EXPOSE 8081

CMD java -jar /deployments/microservice-1.0-SNAPSHOT.jar
```

Figure 8: Dockerfile example.

The Dockerfile is used by Docker to build an image that can be used to deploy a container. Many HosmartAI components consist of multiple containers which can be deployed at once by including them in a Docker Compose [REF-05] file.

2.1.5.2 Docker in Pilot 5 [UM]

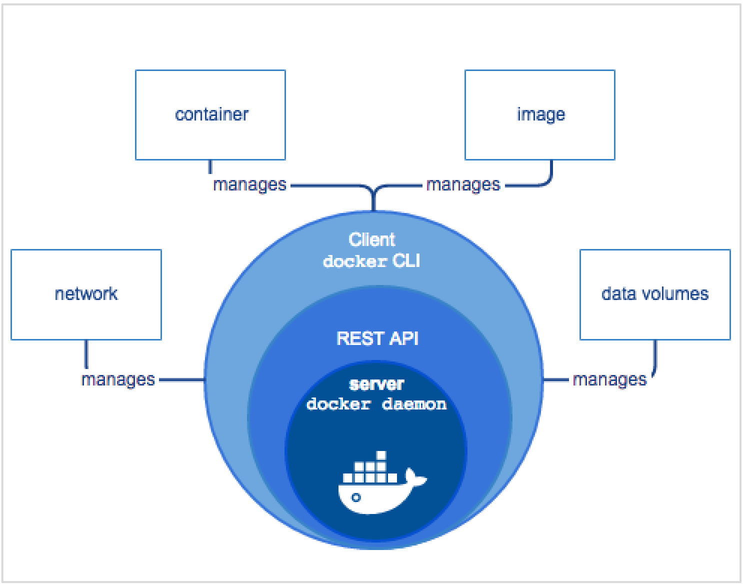


Figure 9: Dockerfile implementation outline for Pilot 5.

In pilot 5 all the services, services such as camel-spring-boot framework, speech recognition, speech synthesis, gesture generation and SLAM framework are dockerized to ensure replicability of the targeted environment and rapid deployment/re-deployment of the services. Namely, Docker is based on Linux and, as such, has the Linux kernel in every container, regardless of the system it is running on. This means that the environment remains stable, despite different updates and updates to the repositories, on any system or device.

2.1.6 GreenSoft

Green Communications provides ready-to-use edge platforms for the easy setup of an edge-based solution with connectivity, edge cloud and services among which a blockchain. Edge clouds operating at multiple locations can be connected and synchronized to create a large-scale Internet of Edges (IoE).

An edge cloud running a blockchain will be deployed at Pilot #5 and Pilot #6. Although the edge cloud of both pilots could connect, they will remain isolated in the context of the project (not connected by the Internet of Edges).

The objective is to connect pepper robots, collect and log its activities using the blockchain service that is embedded onto the edge cloud.

Green Communications' edge solution is composed of two main components, the GreenSoft Operating System and the YOI platform. The following sections describe the GreenSoft while the YOI platform is described in Section 2.2.12.

2.1.6.1 The GreenSoft

Green Communications' software (the GreenSoft) is the Operating System of Green Communications' Edge Cloud platform. The GreenSoft allows hardware to:

i. connect with nearby devices to create a dynamic and self-configuring wireless network with quality of service (QoS).
ii. access and share a common edge cloud with edge-based services.

GreenSoft is a set of software components dedicated to wireless networking. It features a wide range of applications, from low-level programs to web applications for end users. GreenSoft's low-level programs are mostly routing software for mesh networks, but also feature an SNMP module and helpers for Zeroconf networking, among others. High-level utilities feature web applications such as a chat, a network setup app, or a live network visualization tool.

2.1.6.2 Edge network

GreenSoft features an intelligent routing protocol implemented as a user space daemon in charge of the following tasks:

- It detects the other devices that are part of the network.
- It estimates QoS properties for each link.
- It computes (possibly indirect) routes to other devices and sets the system's routing table up accordingly (thus ensuring that every device forward data properly, and that any network host can reach any other host).
- When some devices forward data from the mesh network to other networks (e.g., the Internet), it ensures that all network hosts may reach these other networks.

GreenSoft features a **handoff manager** program. This is a user-space daemon that helps routers provide access points to regular Wi-Fi users. These users, though outside the core mesh network, may associate to the access points and get regular network connectivity through the mesh network. The handoff manager ensures users can move from one access point to another without disrupting their connections.

The handoff manager performs the following tasks:

- It either acts as a distributed DHCP server or as a DHCP relay, relaying DHCP requests from users to a designated DHCP server.
- It snoops on DHCP transactions and informs the network accordingly so routers can map MAC addresses to IP addresses.
- It snoops on Wi-Fi association and disassociation events, so routers can detect handoffs.
- It updates the system's routing tables accordingly and configures access point interfaces to act as a gateway to associated users.

In addition, GreenSoft features an mDNS helper. This is a user-space daemon that ensures Zeroconf works. In practice, this means that devices that run the GreenSoft may advertise Zeroconf services to other devices and users; and that users may also advertise their own Zeroconf services to the network (including other users).

2.1.6.3 Edge Cloud

GreenSoft features a web framework to provide users with several web applications (chat, network setup, live network visualization tool, etc.). The application of interest in the HosmartAI project is the blockchain. This framework uses client-side JavaScript and static HTTP, except for three dynamic HTTP resources:

- A resource that provides a GraphML (XML) representation of the current network. The network visualization tool uses this resource. A custom GreenSoft program provides this resource to the web server using the SCGI protocol.
- A resource that converts GET and POST HTTP requests including JSON data to SNMP GetBulk and Set requests. The network setup app uses this resource. A custom GreenSoft program provides the SNMP/JSON converter to the web server using the CGI protocol. Also note that GreenSoft features a Net-SNMP module that implements the SNMP configuration backend.
- A resource that maps XMPP traffic to HTTP using the BOSH protocol. The chat app uses this resource. Green Communications' routers (YOI) rely on the ejabberd XMPP server for this resource.

GreenSoft web apps therefore need a web server. Any software can provide this server if it supports SCGI and CGI (Green Communications' routers use Nginx with fcgiwrap for CGI). One can easily use the webserver to provide local content to users. One can also easily develop new applications to integrate into the framework.

2.1.7 HAPI-FHIR Platform

HAPI FHIR is a complete implementation of the HL7 FHIR standard for healthcare interoperability in Java. HAPI FHIR defines a class model for each resource and data type defined in the FHIR specification, which in turn can be encoded in XML or JSON, for exchange via REST APIs.

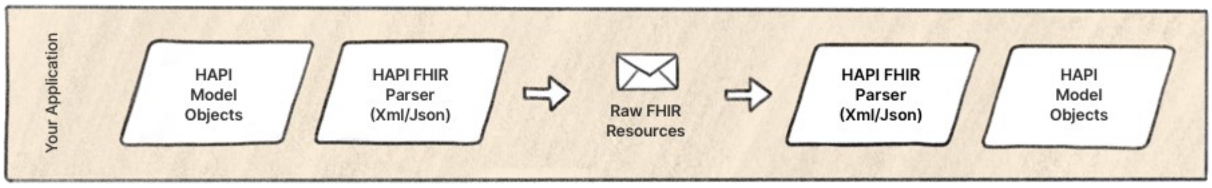


Figure 10: HAPI FHHIR process.

UM deploys the HAPI FHIR server based on the HAPI FHIR JPA SERVER [REF-10] which is running the HL7 DSTU3 model for resources. This project is a fully contained FHIR server, supporting all standard operations (read/create/delete). A conceptual architecture is outlined below.

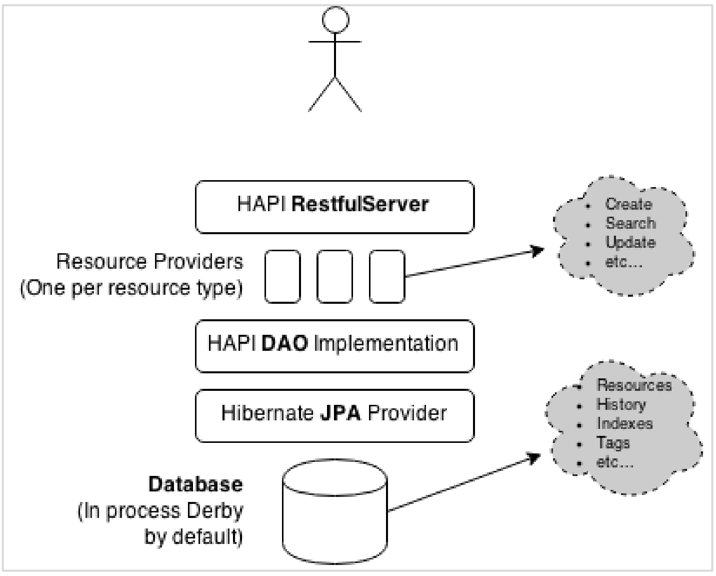
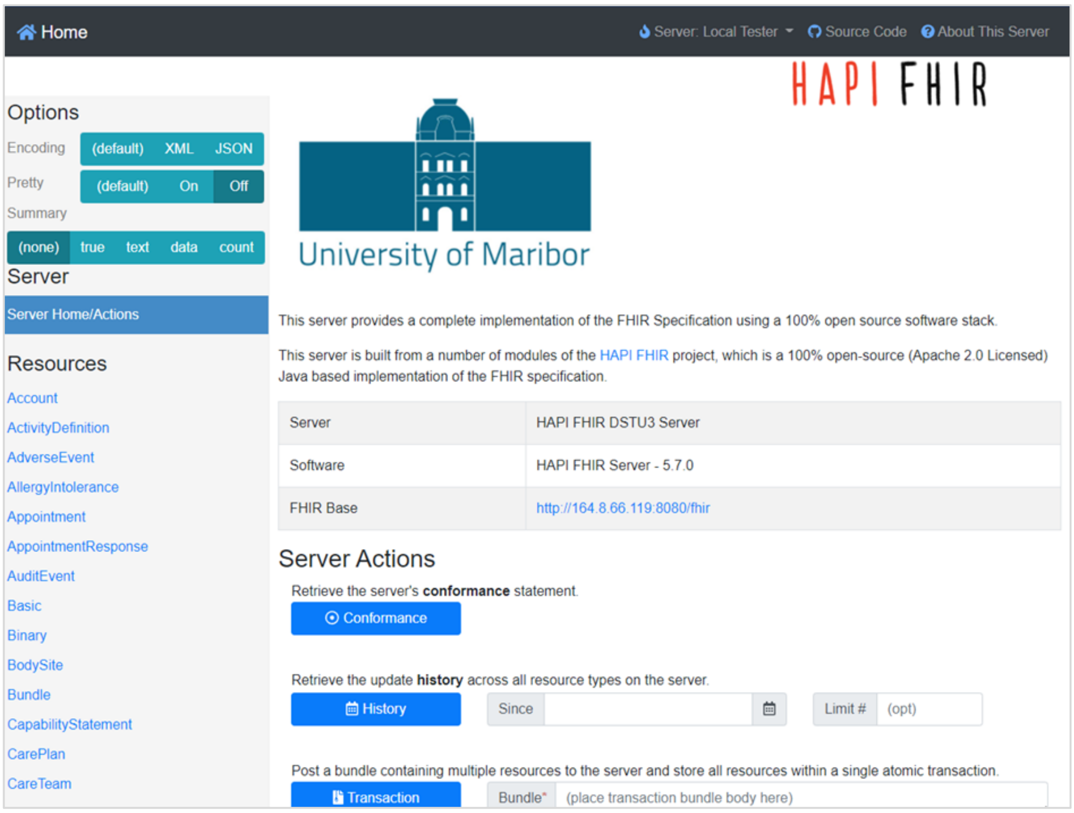


Figure 11: HAPI FHIR JPA Architecture used in Pilot 5.

The HAPI JPA Server has the following components:

- A RESTful server Resource Provider is provided for each resource type in each release of FHIR.
- HAPI DAOs implement all the database business logic relating to the storage, indexing, and retrieval of FHIR resources, using the underlying JPA API.
- **Database:** The RESTful server uses an embedded Derby database, but can be configured to talk to several databases: e.g., MS SQL Server, PostgreSQL, Oracle, DB2, etc.

UM HAPI FHIR server offers Swagger UI as the REST API offering endpoints for work with the FHIR resources. UM HAPI FHIR server is running as a Spring Boot application on top of Java and is implemented using the DB2 database.



This server provides a complete implementation of the FHIR Specification using a 100% open source software stack.

This server is built from a number of modules of the HAPI FHIR project, which is a 100% open-source (Apache 2.0 Licensed) Java based implementation of the FHIR specification.

Server	HAPI FHIR DSTU3 Server
Software	HAPI FHIR Server - 5.7.0
FHIR Base	http://164.8.66.119:8080/fhir

Server Actions

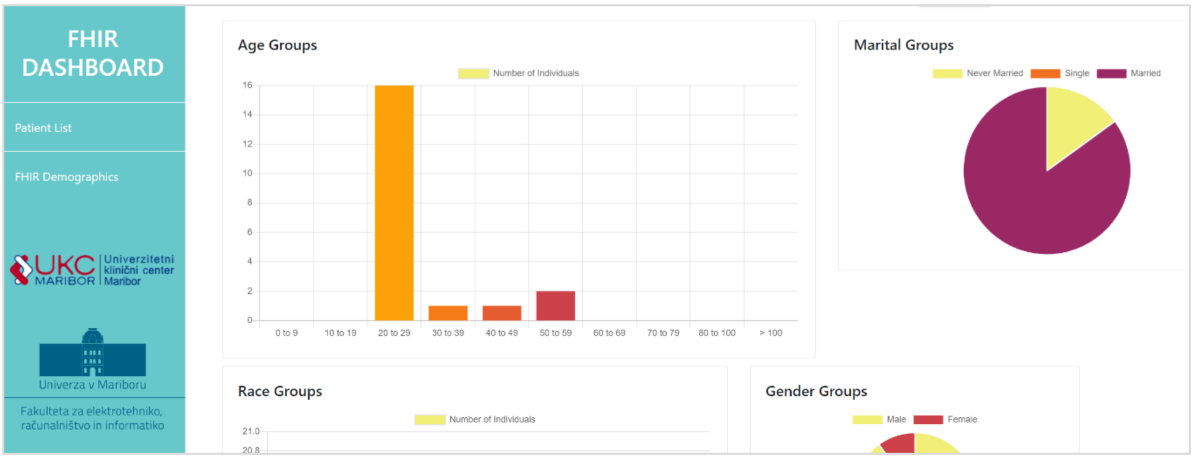
Retrieve the server's **conformance** statement.

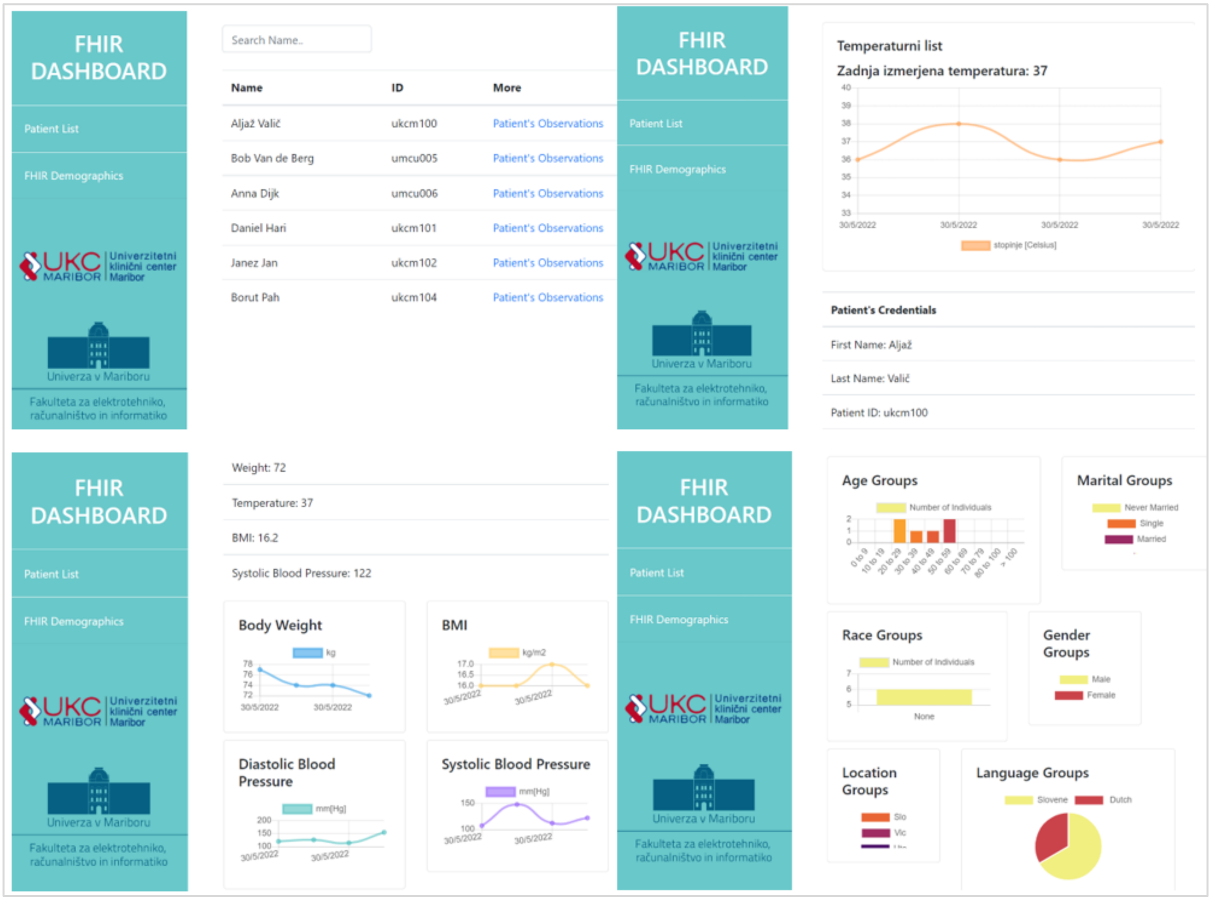
Retrieve the update **history** across all resource types on the server.

Post a bundle containing multiple resources to the server and store all resources within a single atomic transaction.

Figure 12: UM's HAPI FHIR implementation in Pilot 5.

Together with the HAPI FHIR server, UM provides a dashboard of the FHIR collected resources. The dashboard can be used to show a list of patients, their observational data (latest observation UUID, date of observation, status, weight, BMI, systolic blood pressure), and more specific credentials (UUID, patient id, gender, address, birth date) as well as graphs to help clinicians to visualize collected patient data over time.





Name	ID	More
Aljaž Valič	ukcm100	Patient's Observations
Bob Van de Berg	umcu005	Patient's Observations
Anna Dijk	umcu006	Patient's Observations
Daniel Hari	ukcm101	Patient's Observations
Janez Jan	ukcm102	Patient's Observations
Borut Pah	ukcm104	Patient's Observations

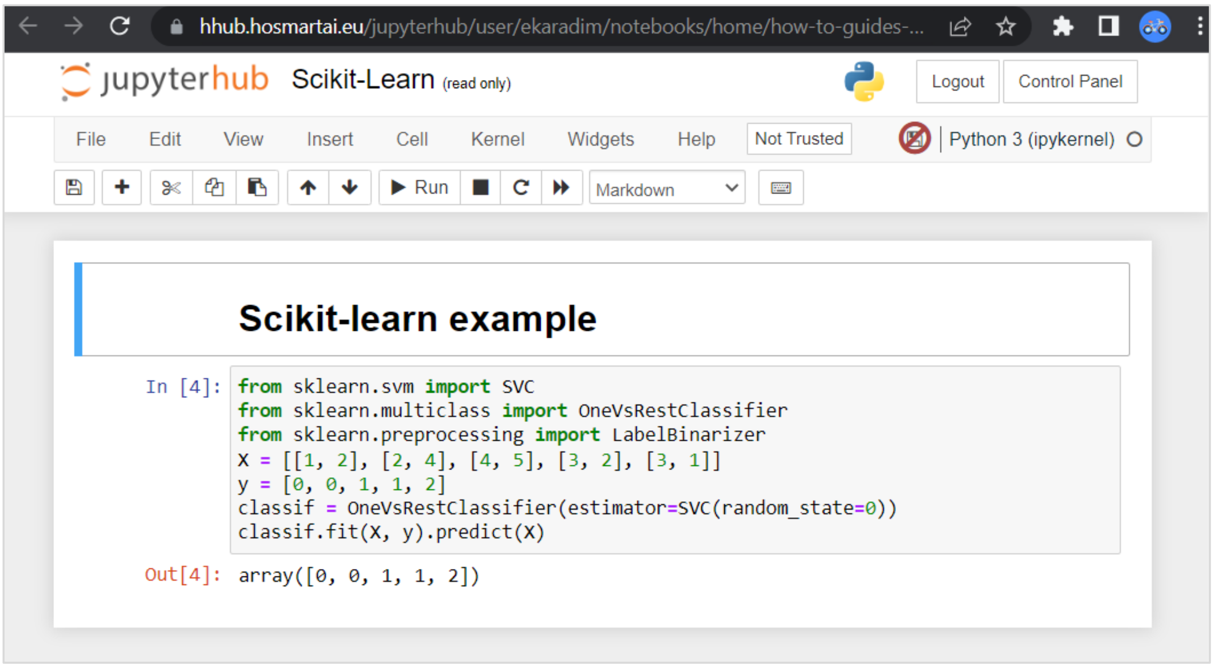
Patient's Credentials
First Name: Aljaž
Last Name: Valič
Patient ID: ukcm100

Weight: 72
Temperature: 37
BMI: 16.2
Systolic Blood Pressure: 122

Figure 13: Dashboards used to visualize the clinical and data collected in Pilot 5.

2.1.8 JupyterHub

JupyterHub is a multi-user proxy server that interconnects several Jupyter Notebook instances. It can be hosted in the cloud (HosmartAI platform) or on own hardware and allows you to use a shared Notebook environment.


hhub.hosmartai.eu/jupyterhub/user/ekaradim/notebooks/home/how-to-guides-...
jupyterhub Scikit-Learn (read only)
Logout
Control Panel
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

Scikit-learn example

```
In [4]: from sklearn.svm import SVC
        from sklearn.multiclass import OneVsRestClassifier
        from sklearn.preprocessing import LabelBinarizer
        X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
        y = [0, 0, 1, 1, 2]
        classif = OneVsRestClassifier(estimator=SVC(random_state=0))
        classif.fit(X, y).predict(X)

Out[4]: array([0, 0, 1, 1, 2])
```

Figure 14: Jupyter example.

In HosmartAI, JupyterHub is used as a collaboration space, mostly for Jupyter Notebooks, but also for running Python and other scripts on the Linux terminal. Jupyter Notebooks can be used to run examples in Python that use Apache Kafka, Spark, or MongoDB, which are in separate VMs. It provides access to AI tools that have been installed on the same VM, such as Scikit-Learn PySpark, TensorFlow and Pandas.

2.1.9 Keycloak

Keycloak is an open-source software product that enables single sign-on (IdP) with Identity Management and Access Management for modern applications and services. This software is written in Java and supports by default the SAML v2 and OpenID Connect (OIDC) / OAuth2 identity federation protocols.

From a conceptual perspective, the intention of the tool is to facilitate the protection of applications and services with little or no encryption. An IdP allows an application (often called a Service Provider or SP) to delegate its authentication.

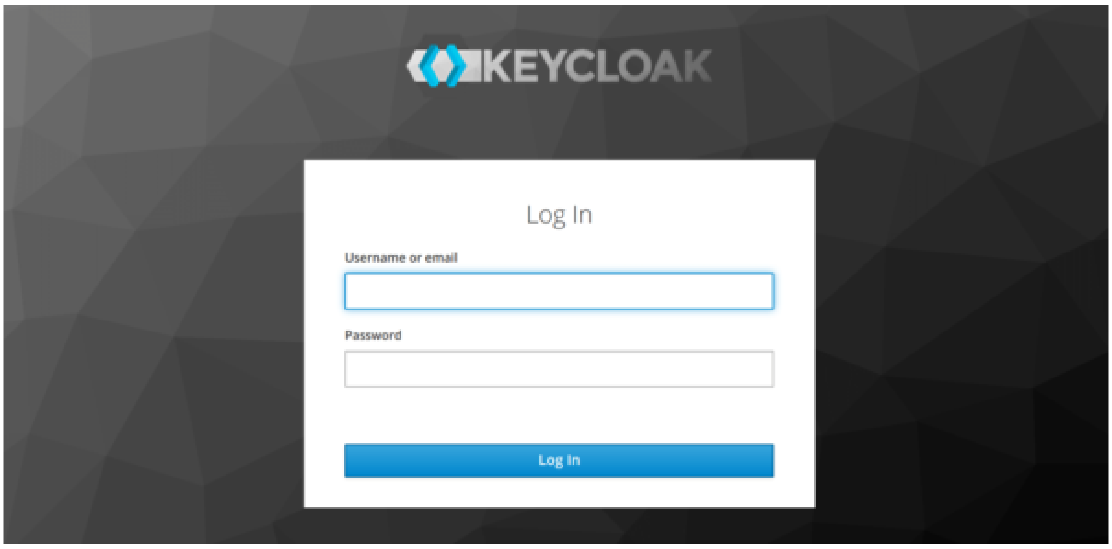


Figure 15: Login screen.

2.1.9.1 Keycloak in Platform

Keycloak has been configured to handle the login requests from various platform components and can be configured for external components as well. For example, when someone tries to access JupyterHub using HosmartAI credentials, they are presented with a form to provide their credentials, which is generated by Keycloak. Upon successful login, the user is redirected back to the component they initially requested to access, e.g., JupyterHub.

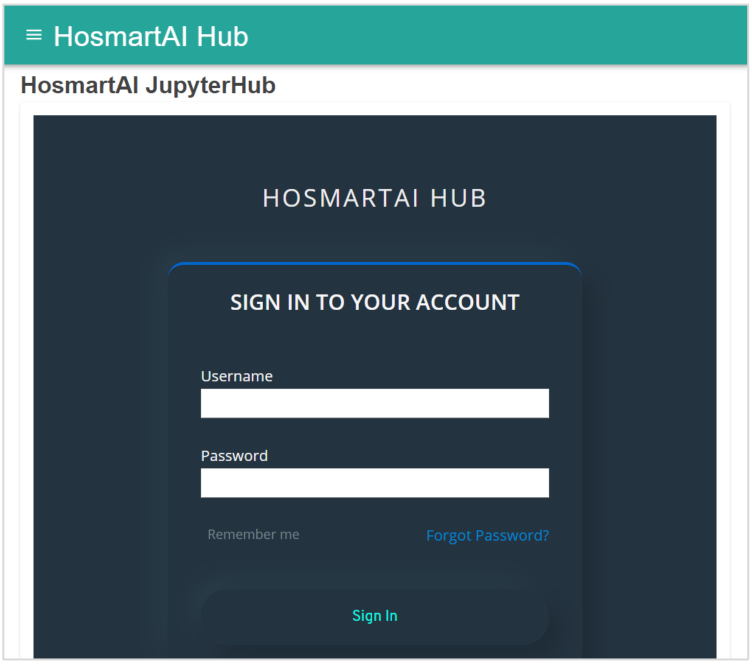


Figure 16: HHub login.

2.1.9.2 Keycloak in Pilot 3

VIMAR uses Keycloak as an access and authorization management provider. It emits an access token/refresh token, and it validates the emitted tokens. The identification is carried out through an external provider, namely the MyVIMAR portal, where the user can access with its own credentials.

In Pilot 3, Keycloak allows communication with the KNX IoT 3rd party client, providing the authorization to access the various resources.

2.1.9.3 Keycloak in Pilot 5

Regarding data access and security, UM implements Keycloak. Keycloak adds authentication to applications and secure services. It provides user federation, strong authentication, user management and fine-grained authorization. UM Keycloak protects the UM FHIR server by periodically generating new JWT tokens that are needed to access the UM FHIR API and connect to the FHIR server to handle the FHIR resources. Keycloak is also deployed as a layer of protection for access to UM’s backend services, such as speech recognition, speech synthesis, natural language generation services and MRAST framework.

2.1.10 Sentry

Sentry [REF-11] is defined as a system focused on application monitoring that works through a fault tracker that monitors and responds to faults that may occur in the application in real-time.

In addition to this, Sentry offers easy management of applications through an intuitive interface that allows you to perform your monitoring activities in a user-friendly manner.

Among the main features and properties of the Sentry monitoring system, we highlight its ability to perform context monitoring of user applications to report errors or failures that occur in the servers or services. This property also allows developers to have immediate visibility into the impact or effects of production code on real users.

Once Sentry has identified application failures, it then triages, classifies and ultimately fixes these problems as part of its workflow.

So, this system is characterized by enabling the identification of performance bugs, before they become uptime that would affect the application's operation.

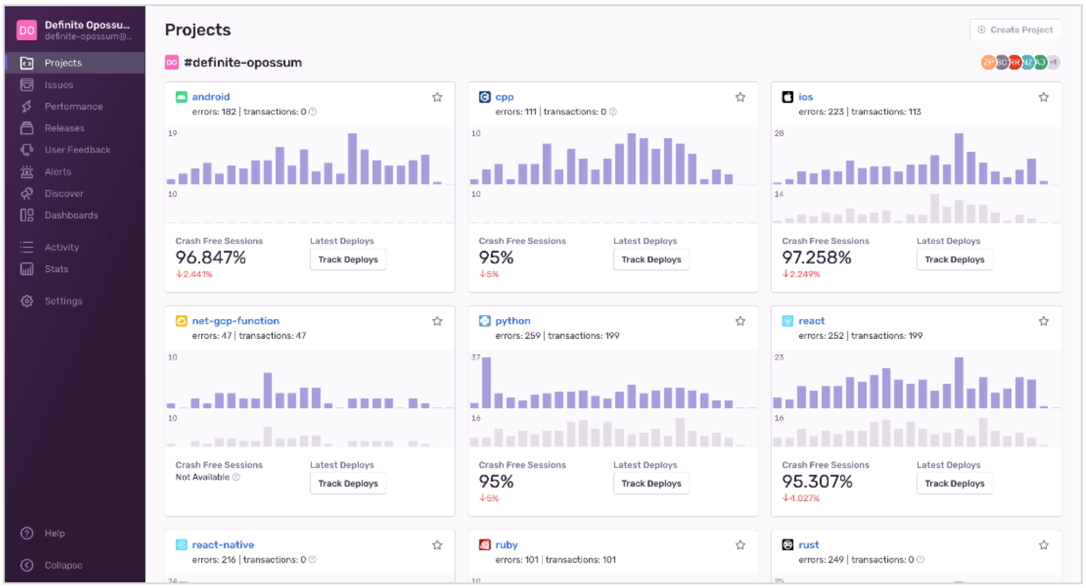


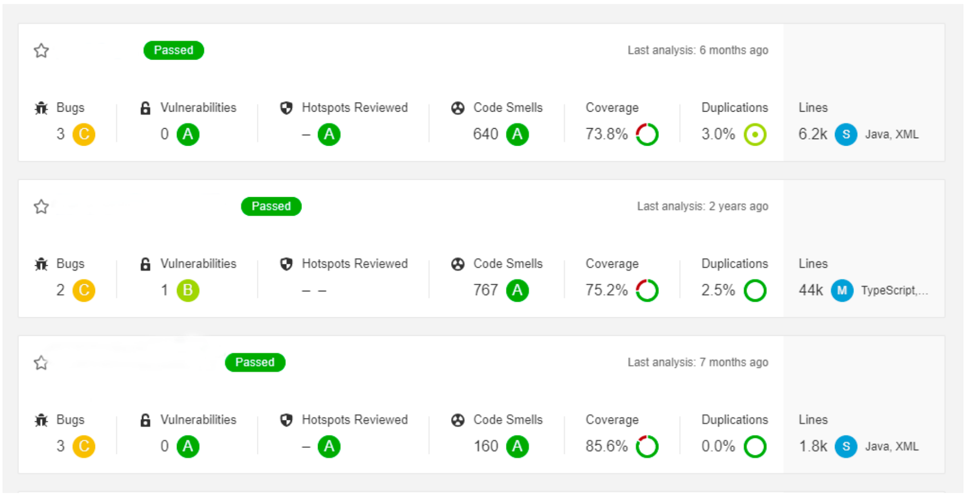
Figure 17: Sentry dashboard.

UM uses Sentry Dashboard to track and monitor logs, errors, and performance of the deployed applications. Sentry monitor multiple applications developed for the HOSMARTAI project over one Dashboard that's accessible over a web browser. To monitor and log python-based applications we exploit Sentry's Python SDK. It includes powerful hooks that enable automatic reporting of errors and exceptions as well as identify performance issues in our python-based services (e.g., MRAST framework). The Sentry's Java SDK enables capturing sessions for Release health as well as reporting messages and errors (i.e., from Camel Spring Boot). At its core, Sentry for Java provides a raw client for sending events to Sentry.

2.1.11 SonarQube

SonarQube is an open-source platform for continuous inspection of code quality through different static source code analysis tools. It provides metrics that help improve the quality of a program's code by allowing development teams to track and detect bugs and security vulnerabilities to keep the code clean.

It is an essential tool for the testing and code auditing phase of the application development cycle and is considered perfect for guiding development teams during code reviews. It supports a continuous inspection stage.



Passed — Last analysis: 6 months ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
3 C	0 A	– A	640 A	73.8%	3.0%	6.2k S Java, XML

Passed — Last analysis: 2 years ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
2 C	1 B	– –	767 A	75.2%	2.5%	44k M TypeScript,...

Passed — Last analysis: 7 months ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
3 C	0 A	– A	160 A	85.6%	0.0%	1.8k S Java, XML

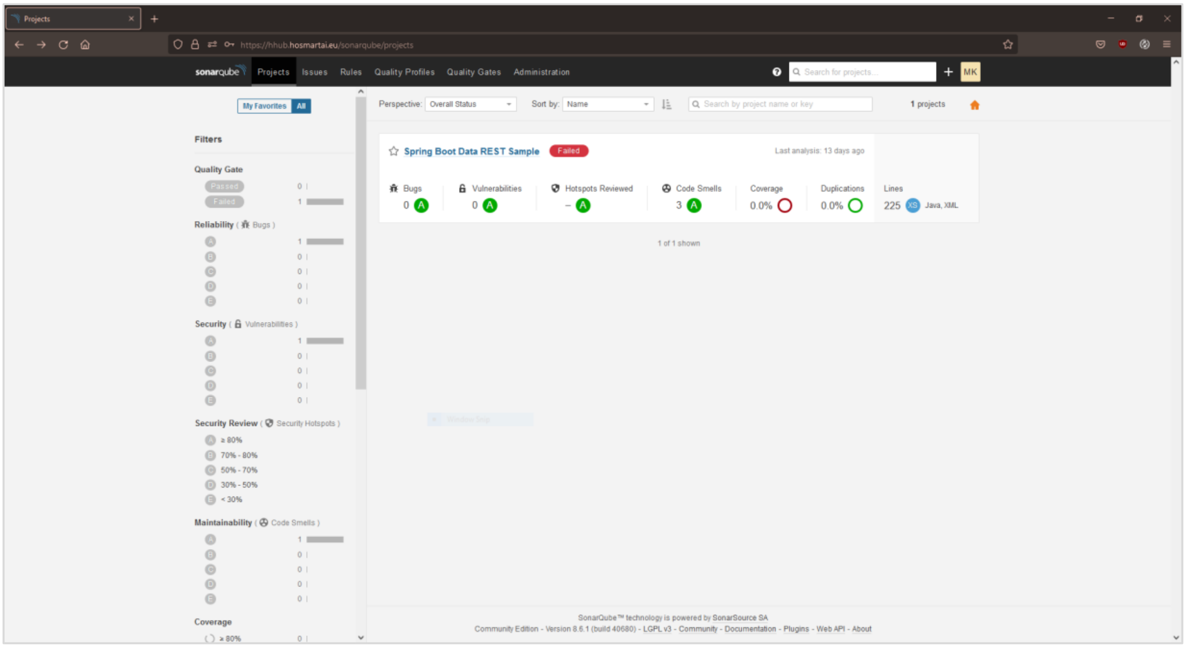
Figure 18: Issues navigator.

In the HosmartAI Platform, SonarQube is integrated with Keycloak therefore you need to follow the link https://hhub.hosmartai.eu/sonarqube and provide the Keycloak credentials.

During the development phase a step on the Jenkinsfile should be added. INTRA for demo purposes has delivered the projects CICD-Demo found on the project's Gitlab. This project is based on a Spring Boot microservice with Maven and the step on the Jenkinsfile is as follows:

```
stage('SonarQube analysis') {
steps {
withSonarQubeEnv(credentialsId:'sonarqube-integration', installationName: 'sonarqube') {sh 'mvn sonar:sonar'}
            }
 }
```

This results the SonarQube analysis to be performed during the integration with Jenkins. The result is by selecting SonarQube from the menu of the HosmartAI HUB.


Spring Boot Data REST Sample — Failed — Last analysis: 13 days ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	0 A	– A	3 A	0.0%	0.0%	225 XS Java, XML

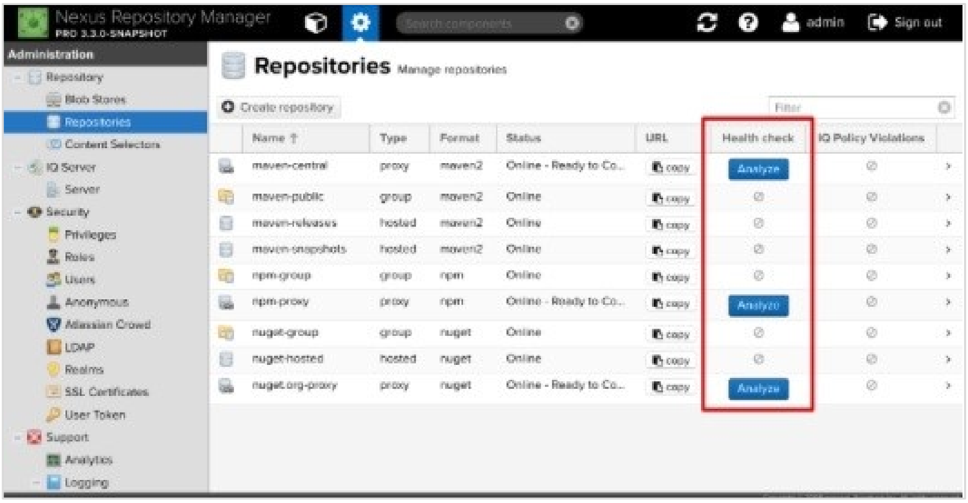
1 of 1 shown

Figure 19: SonarQube UI with project analysis.

2.1.12 Sonatype Nexus OSS

Sonatype Nexus Repository is a repository manager for binary software components (artifacts, packages, ...). A binary repository manager is a software tool designed to optimize the download and storage of binary files used and produced in software development.

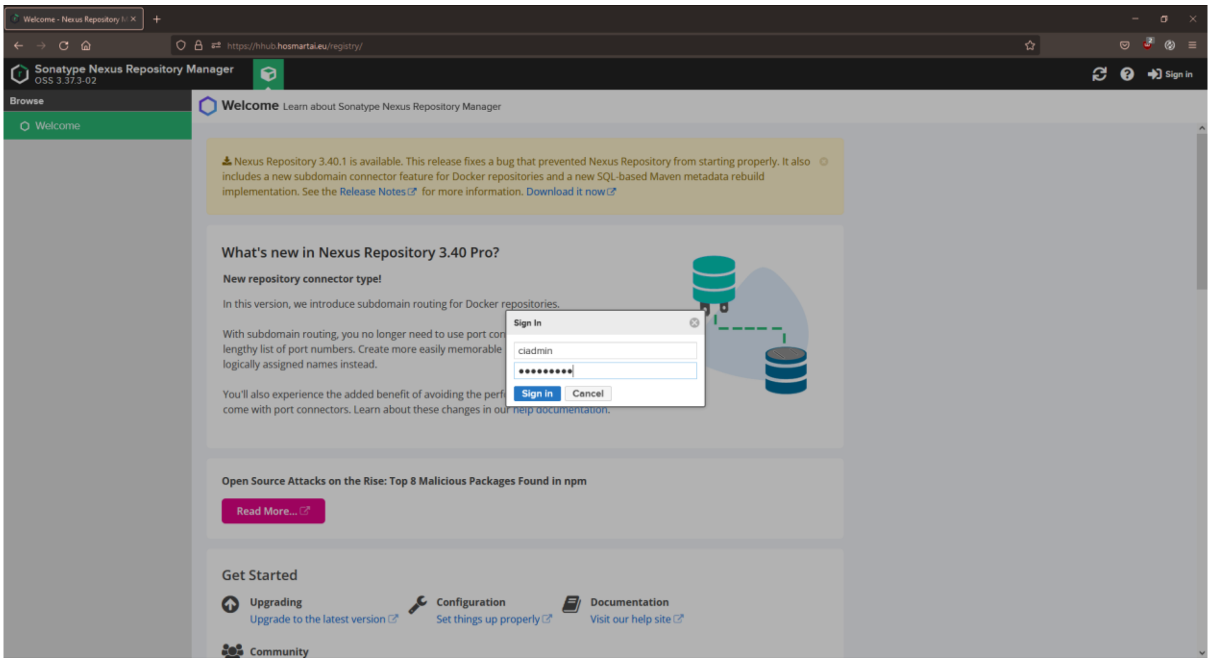
A binary repository manager is an essential part of a continuous software integration and delivery architecture.


Repositories Manage repositories

Name	Type	Format	Status	URL	Health check
maven-central	proxy	maven2	Online - Ready to Co...	copy	Analyze
maven-public	group	maven2	Online	copy	
maven-releases	hosted	maven2	Online	copy	
maven-snapshots	hosted	maven2	Online	copy	
npm-group	group	npm	Online	copy	
npm-proxy	proxy	npm	Online - Ready to Co...	copy	Analyze
nuget-group	group	nuget	Online	copy	
nuget-hosted	hosted	nuget	Online	copy	
nuget.org-proxy	proxy	nuget	Online - Ready to Co...	copy	Analyze

Figure 20: Sonar repository list.

In the HosmartAI Platform, Sonatype Nexus is integrated with Keycloak therefore someone needs to provide Keycloak credentials after visiting https://hhub.hosmartai.eu/registry.



Welcome - Nexus Repository M

https://hhub.hosmartai.eu/registry/

Sonatype Nexus Repository Manager
OSS 3.37.3-02

Sign in

Browse

Welcome

Welcome Learn about Sonatype Nexus Repository Manager

Nexus Repository 3.40.1 is available. This release fixes a bug that prevented Nexus Repository from starting properly. It also includes a new subdomain connector feature for Docker repositories and a new SQL-based Maven metadata rebuild implementation. See the Release Notes for more information. Download it now

What's new in Nexus Repository 3.40 Pro?

New repository connector type!

In this version, we introduce subdomain routing for Docker repositories.

With subdomain routing, you no longer need to use port con lengthy list of port numbers. Create more easily memorable logically assigned names instead.

You'll also experience the added benefit of avoiding the perf come with port connectors. Learn about these changes in our help documentation.

Sign In

ciadmin

••••••••••

Sign in | Cancel

Open Source Attacks on the Rise: Top 8 Malicious Packages Found in npm

Read More...

Get Started

Upgrading — Upgrade to the latest version

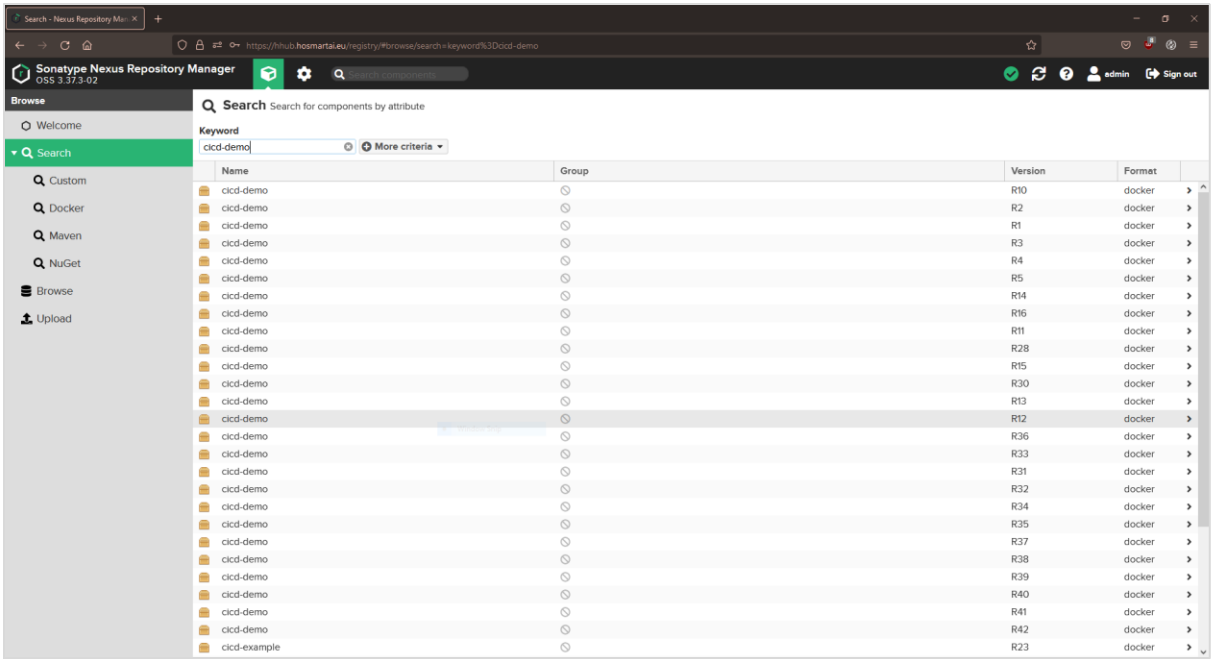
Configuration — Set things up properly

Documentation — Visit our help site

Community

Figure 21: Nexus integration to Keycloak.

It is used mostly as a Docker image repository. By visiting Nexus, the user can view the images that are stored there and the different versions.



Search - Nexus Repository Man

https://hhub.hosmartai.eu/registry/#browse/search=keyword%3Dcicd-demo

Sonatype Nexus Repository Manager
OSS 3.37.3-02

Search components

admin | Sign out

Browse

Welcome

Search

Custom

Docker

Maven

NuGet

Browse

Upload

Search Search for components by attribute

Keyword

cicd-demo

More criteria

Name	Group	Version	Format
cicd-demo		R10	docker
cicd-demo		R2	docker
cicd-demo		R1	docker
cicd-demo		R3	docker
cicd-demo		R4	docker
cicd-demo		R5	docker
cicd-demo		R14	docker
cicd-demo		R16	docker
cicd-demo		R11	docker
cicd-demo		R28	docker
cicd-demo		R15	docker
cicd-demo		R30	docker
cicd-demo		R13	docker
cicd-demo		R12	docker
cicd-demo		R36	docker
cicd-demo		R33	docker
cicd-demo		R31	docker
cicd-demo		R32	docker
cicd-demo		R34	docker
cicd-demo		R35	docker
cicd-demo		R37	docker
cicd-demo		R38	docker
cicd-demo		R39	docker
cicd-demo		R40	docker
cicd-demo		R41	docker
cicd-demo		R42	docker
cicd-example		R23	docker

Figure 22: Nexus Repository Manager UI.

2.1.13 Spark

Spark is an ultra-fast engine for storing, processing and analysing large volumes of data and is specially designed for implementation in big data and machine learning. Its processing power speeds up the detection of patterns in data, the organized classification of information, the execution of intensive computation on data and parallel processing in clusters.

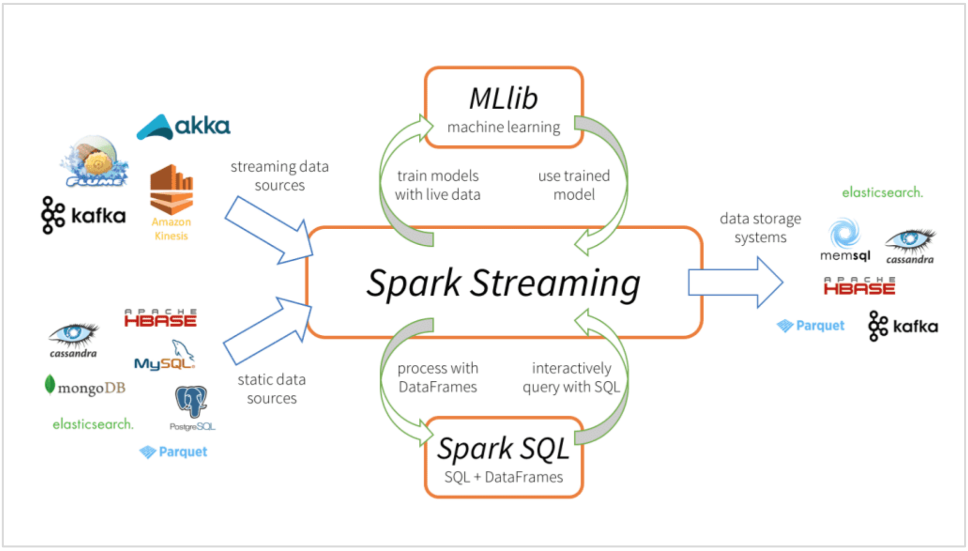
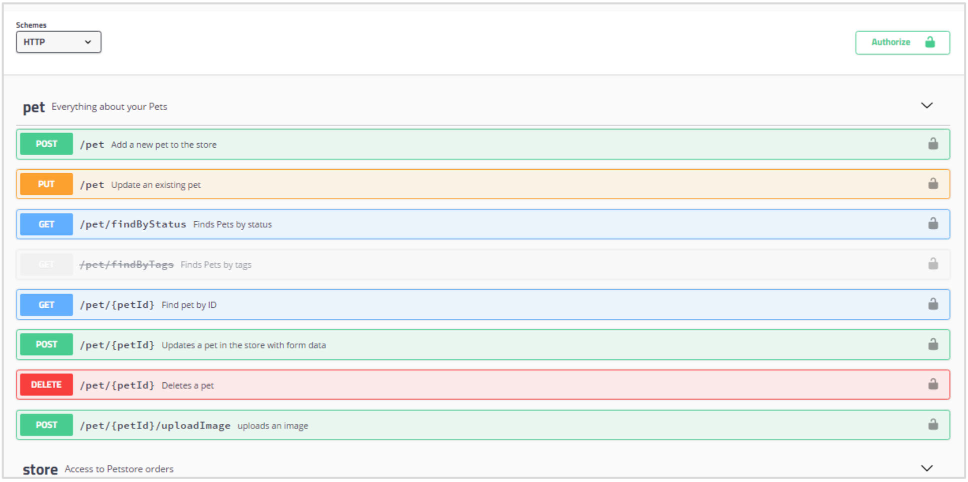


Figure 23: Spark architecture.

In the HosmartAI Platform, there is a PySpark kernel that has been installed in JupyterHub, which allows users to directly use the local Apache Spark deployment. Except for Jupyter notebooks that use the PySpark kernel, PySpark can be used from the JupyterHub terminal as well.

2.1.14 Swagger

Swagger is a suite of tools for API developers and a former specification upon which the OpenAPI Specification is based.



Schemes
HTTP

Authorize

pet Everything about your Pets

POST `/pet` Add a new pet to the store

PUT `/pet` Update an existing pet

GET `/pet/findByStatus` Finds Pets by status

GET ~~`/pet/findByTags`~~ Finds Pets by tags

GET `/pet/{petId}` Find pet by ID

POST `/pet/{petId}` Updates a pet in the store with form data

DELETE `/pet/{petId}` Deletes a pet

POST `/pet/{petId}/uploadImage` uploads an image

store Access to Petstore orders

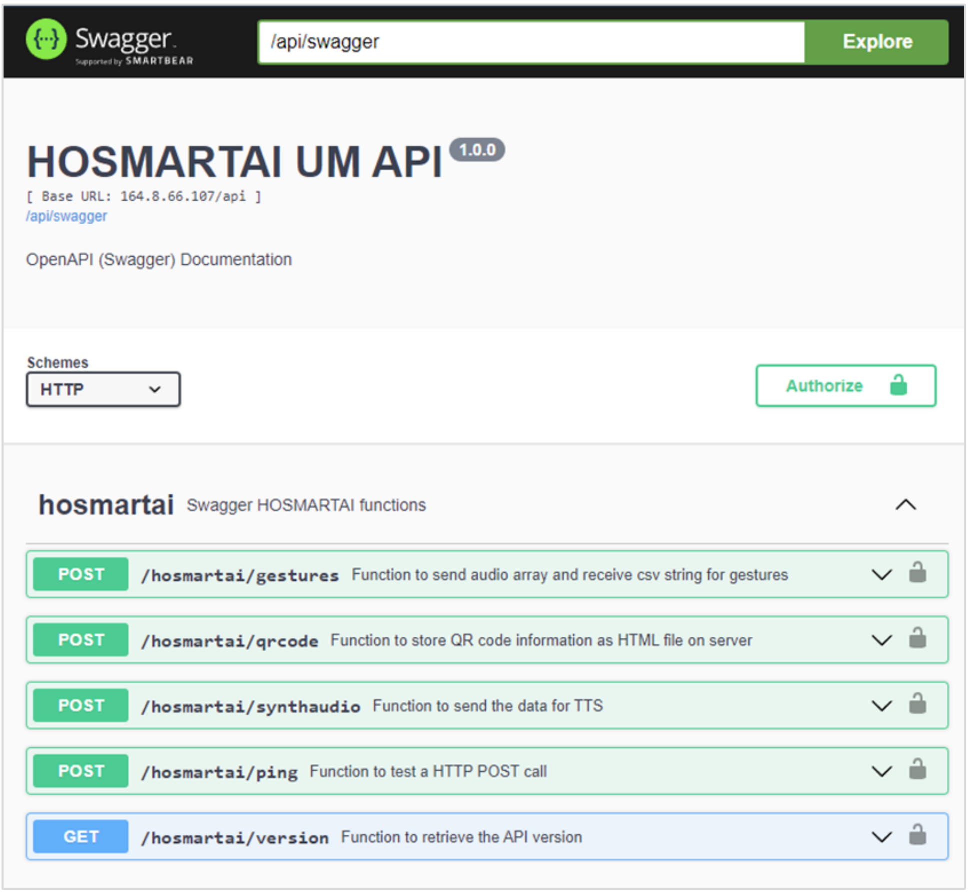
Figure 24: Swagger example API.

2.1.14.1 Swagger in Pilot 3

In Pilot 3, Swagger is used to defining the APIs for communication with the cloud and to handle the documentation. VIMAR provided the KNX Swagger as documentation to the KNX IoT 3rd party integrator, which can use the documentation as reference in their implementation.

2.1.14.2 Swagger in Pilot 5

Swagger UI allows one to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from OpenAPI Specification, with the visual documentation making it easy for back-end implementation and client-side consumption. UM Swagger documentation consists of multiple HTTP endpoints each with a specific method that on user request runs the Java code in the backend of the Spring Boot application. Swagger UI is protected with the API KEY and uses HTTPS SSL encryption.



HOSMARTAI UM API 1.0.0

[Base URL: 164.8.66.107/api]
/api/swagger

OpenAPI (Swagger) Documentation

hosmartai Swagger HOSMARTAI functions

POST /hosmartai/gestures Function to send audio array and receive csv string for gestures
POST /hosmartai/qrcode Function to store QR code information as HTML file on server
POST /hosmartai/synthaudio Function to send the data for TTS
POST /hosmartai/ping Function to test a HTTP POST call
GET /hosmartai/version Function to retrieve the API version

Figure 25: Pilot 5 API.

2.2 Physical Platforms

Physical platforms are hardware elements that interact with the environment where they are deployed, like sensors, specialized tools, or robots.

2.2.1 Arduino

Arduino is an open-source electronics creation platform, which is based on free hardware and software, flexible and easy to use for creators and developers. This platform allows the creation of different types of single-board microcomputers that can be put to different types of use by the maker community.

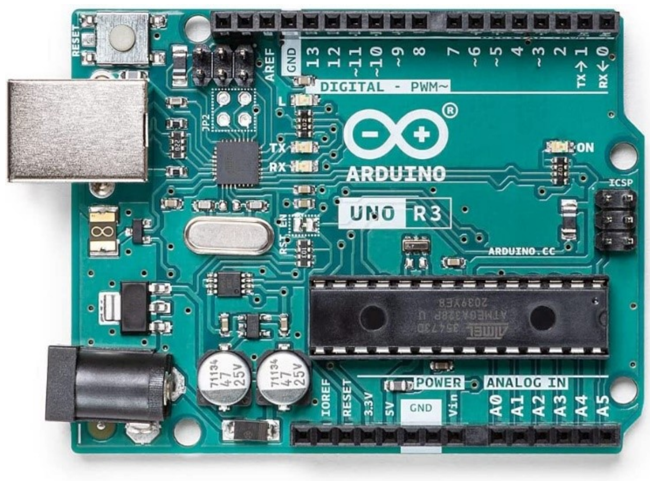


Figure 26: Arduino board.

2.2.2 Capsule Endoscopy System

The Capsule Endoscopy System is a minimally invasive ingestible camera in a pill-shaped capsule that allows visualization of the small bowel, which is not accessible through upper and lower endoscopy. The patient is equipped with a small recording device that wirelessly receives and stores the images captured continuously by the camera as it travels through the gastrointestinal tract. The resulting video sequence can be examined by a gastroenterologist to identify abnormalities or lesions and subsequently diagnose conditions. In the video capsule endoscopy scenario of Pilot 1, a capsule endoscopy system will be used to acquire capsule endoscopy videos that will be analysed with the pilot's AI-based tool for automatic detection and classification of small bowel abnormalities, to evaluate the tool's capacity to accelerate and improve the accuracy of the examination procedure and, ultimately, to improve small bowel condition diagnosis via AI-assisted capsule endoscopy.

2.2.3 Clarius PA HD Scanner

The Clarius PA HD Scanner is a handheld wireless ultrasound scanner that provides high-definition imaging for a variety of medical applications including cardiac, abdominal, lung, and vascular imaging. The portability and ease of use offered by the PA HD scanner is of high value for cardiologists performing echocardiographic examinations under time pressure. To this end, both a regular bedside ultrasound scanner and the PA HD scanner will be used in the echocardiography scenario of Pilot 1 for the acquisition of echocardiograms that will be analysed with the pilot's AI-based tool for automatic estimation of left ventricular (LV) ejection fraction (EF) and global longitudinal strain (GLS). Evaluating the tool with scans from both devices will yield more complete evidence on its estimation capacity and will reveal whether a fast workflow for accurate LV function diagnosis based on portable ultrasonics and AI-based LVEF and LVGLS estimation is possible.

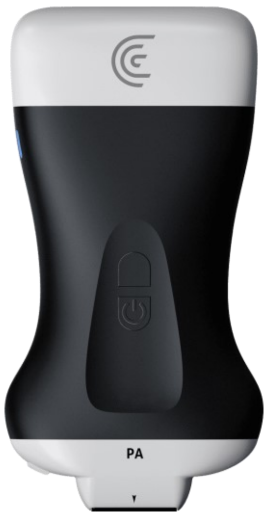


Figure 27: Clarius hardware.

2.2.4 MagnoFlush

The MagnoFlush and MagnoBlate Catheters are designed to be used during a wide variety of robotic cardiac ablation procedures.

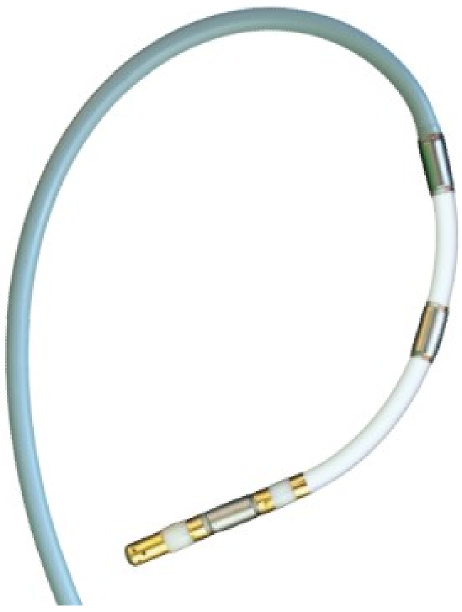


Figure 28: MagnoFlush hardware.

2.2.5 Maxim 32660

In the DARWIN family, the MAX32660 is an ultra-low-power, cost-effective, highly integrated 32-bit microcontroller designed for battery-powered devices and wireless sensors. It combines a flexible and versatile power management unit with the powerful Arm® Cortex®-

M4 processor with a floating-point unit (FPU) in the industry's smallest form factor: 1.6mm x 1.6mm, 16-bump WLP or 4mm x 4mm, 20-pin TQFN-EP, or 3mm x 3mm, 24-pin TQFN-EP.

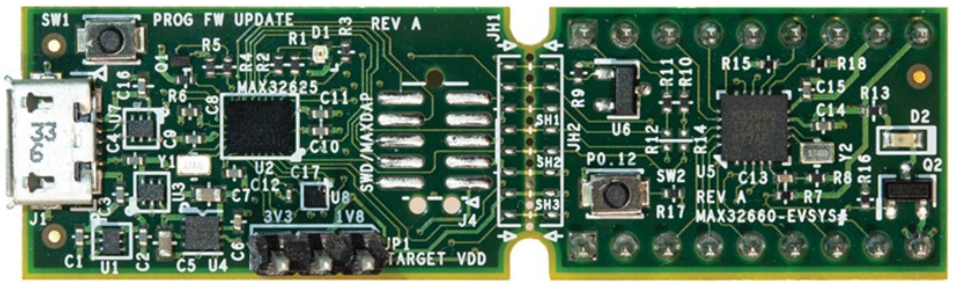


Figure 29: Maxim board.

2.2.6 Navion (Magnebotix)

The Magnebotix-Navion is a magnetic field generator that enables field-guidance studies in much larger volumes. Its patented coil system can produce electrically steerable fields of up to 50mT at 20cm from the face of the generator, thus allowing extended studies at a medically relevant scale. The Magnebotix-Navion has the same characteristics as its medically certified counterpart and is ideally suited for the development of surgical procedures in animals and further human-scale biological and engineering applications.

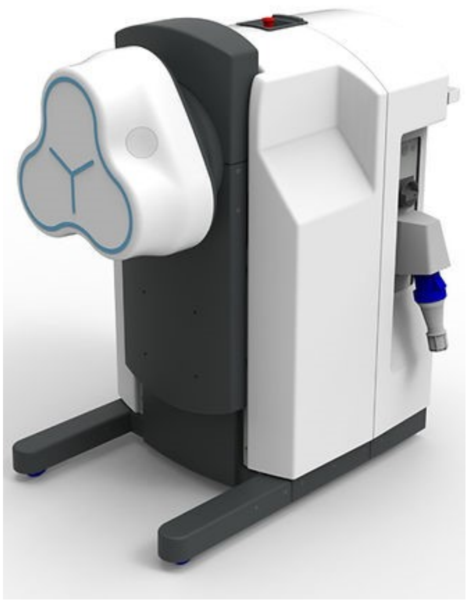
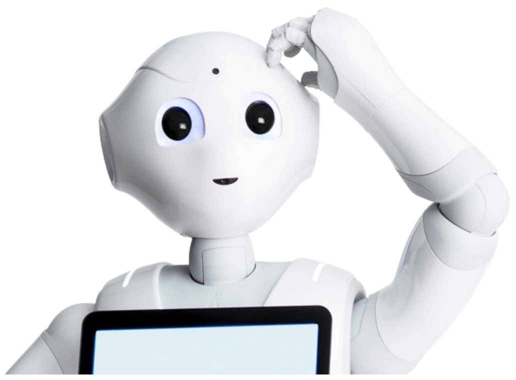


Figure 30: Navion hardware.

2.2.7 Pepper Robot

Pepper is a semi-humanoid robot manufactured by SoftBank Robotics designed with the ability to read emotions. Pepper can chat autonomously with prospective clients.



Pepper is not a functional robot for domestic use. Instead, Pepper is intended "to make people enjoy life", enhance people's lives, facilitate relationships, have fun with people and connect people with the outside world.

2.2.7.1 Pepper Robot in Pilot 5

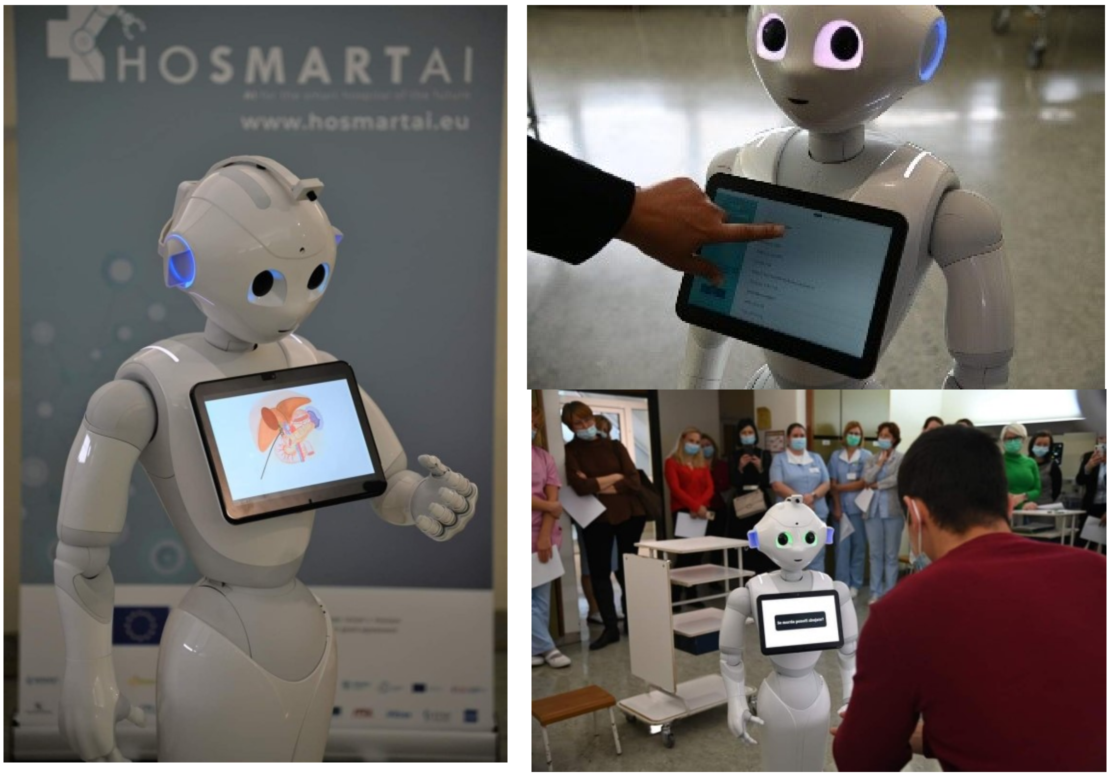


Figure 31: Pepper use cases.

The Pepper robot, developed by SoftBank Robotics, is a 120 cm tall social humanoid robot optimized for human interaction and engaging with people through conversation and a touch screen. It is capable of natural movement, navigation, speech recognition, and dialogue, and is equipped with perception modules and various sensors for multimodal interactions (e.g., microphones, infrared sensors, cameras, and sonars). Thus, Pepper in pilot 5 represents an integrated solution comprised of a computerized clinical decision support system and a service robot enabling the intuitive collection and display of health/clinical data, to support nurses by taking over activities that represent no added value to them (e.g., administrative and routine tasks) and to entertain, inform and motivate patients.

These functions allow for smooth interaction that does not require any specific training. The Pepper robot also has several safety mechanisms, such as bumper sensors, that prevent it from physically harming participants. For the purposes of HosmartAI, the robot was taught to understand and express gestures, facial expressions, and speech in the Slovenian cultural context and language. It was also taught to perform exercises and scenarios that are part of the intervention.

The overall system, however, enables not just telemonitoring but also working with qualitative and self-reported data from patients regarding pain, issues, and psychological well-being. The robot can be integrated into general administrative data collection and digitalization processes before or during patient admission. It also provides support during grand-round routines by displaying relevant data, and it assists the distribution of medicines by identifying alternatives if the prescribed drugs are not available. Basically, the robot collects whatever needs to be done with the patient and it prevents patients' inputs need to be retrieved manually by nurses or doctors themselves. The humanoid features facilitate more natural, trustworthy, and engaging interaction while computer vision, language, and acoustic processing help to understand and motivate patients to exercise or move. Thanks to integrated blockchain technology, the robot's accountability is guaranteed as well. This makes it possible to know what the robot is doing at any given time, where it is, and whether it is the cause of eventual mistakes

In pilot 5 the robot is connected to a proprietary edge platform capable engage with robot and digital services and to record and trace data and activates via a blockchain service. We deliver our own graphical interface and a control API to control the robot developed in the python programming language. With this interface, we can move the robot, add gestures and speech, and enable multiple levels of autonomy, including localization and task or care-workflow execution.

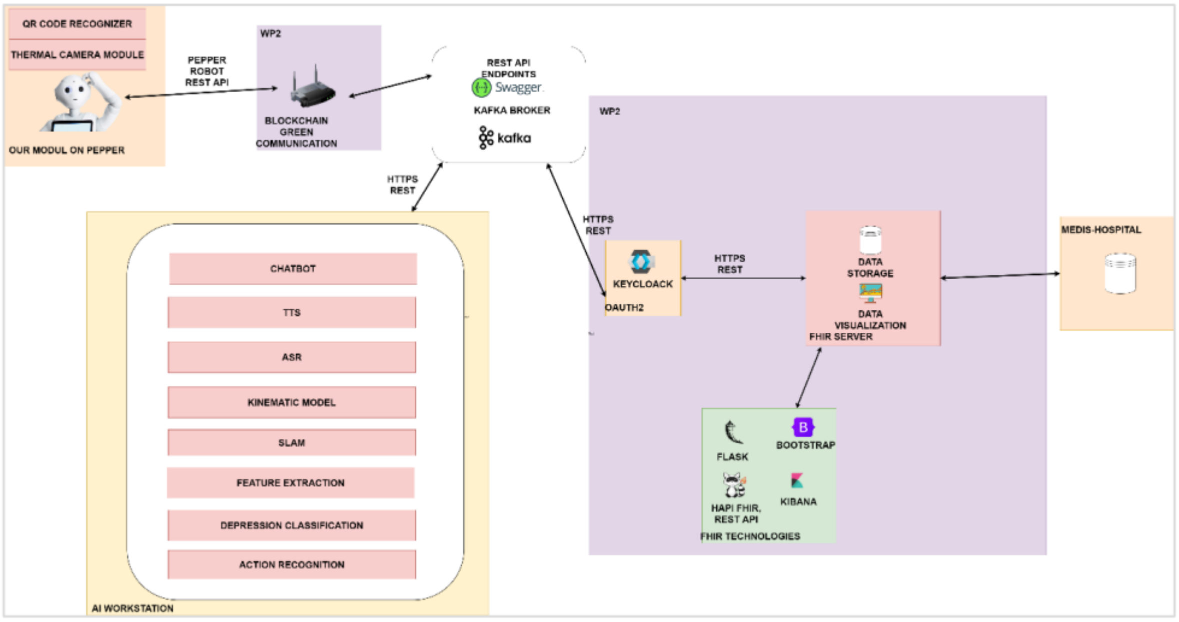


Figure 32: Pepper integration.

2.2.7.2 Pepper Robot in Pilot 6

In pilot 6, the robot carries out social activities in the clinical care center with users in group and individual formats.

- In the individual format, emotion recognition actions are carried out through the functions that pepper integrates into the development SDK.
- In the group format, Pepper will be a new working tool the therapists will count on for carrying out physical stimulation activities (e.g., psychomotricity sessions) and/or active aging workshops.

The activities that are proposed to the user through Pepper are designed in the Activity Plan Editor and the connection with the platform and the registration of the actions are carried out through a blockchain service. The connectivity herein mentioned is detailed in D3.2 "First set of AI-based Solutions and Autonomous Smart Components", Sections 6.1.2 and 6.2.2, and D5.4 "HosmartAI Pilots – First version", Chapter 8.

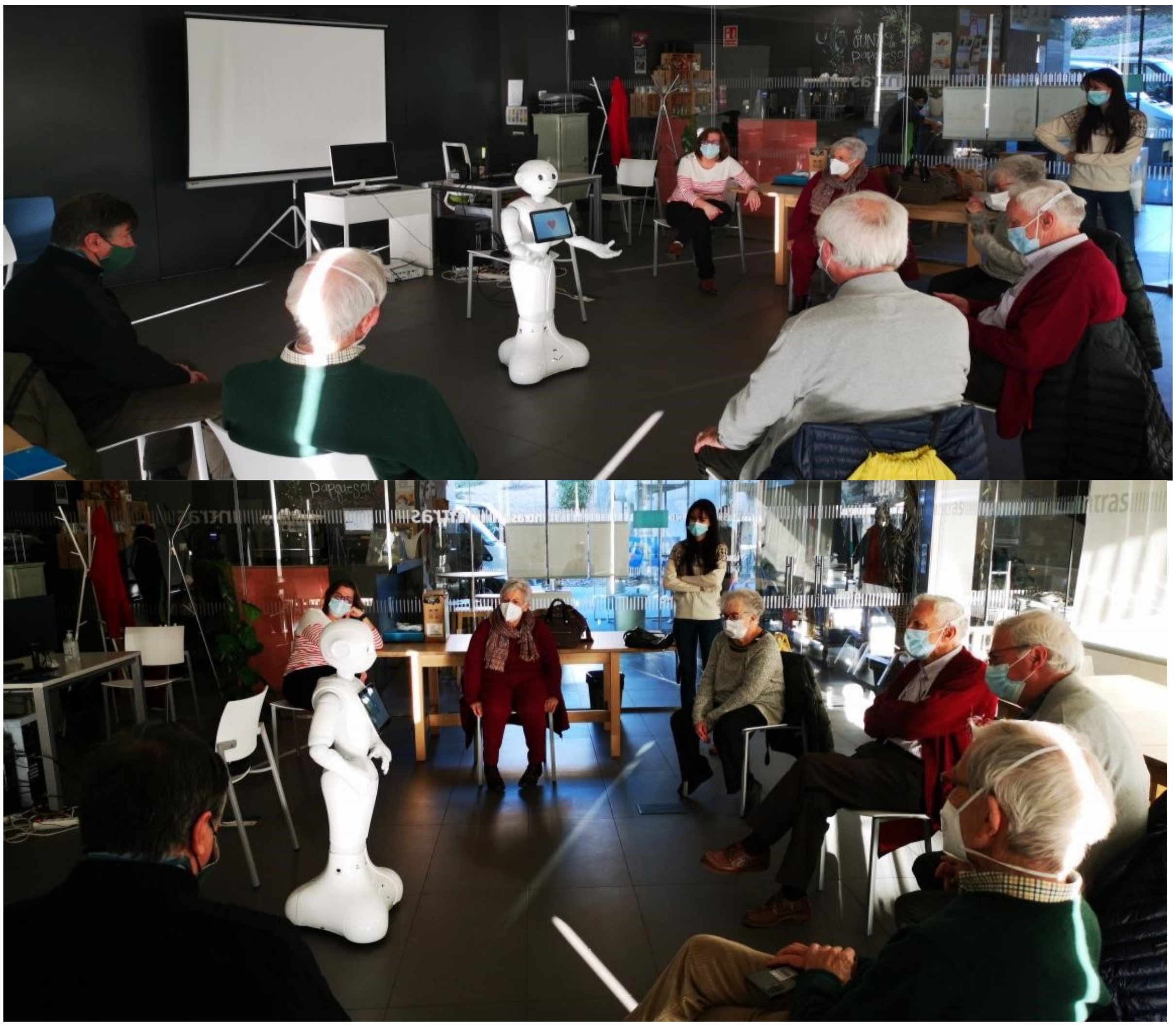


Figure 33: Pepper use case.

2.2.8 Smart Digital Monitors

The BeneVision Patient Monitor with N1 module is intended for monitoring, displaying, reviewing, storing, alarming, and transferring multiple physiological parameters including ECG (3-lead, 5-lead, 6-lead or 12-lead selectable, Arrhythmia Detection, ST Segment Analysis, QT/QTc Analysis, and Heart Rate (HR)), Respiration (Resp), Temperature (Temp), Pulse Oxygen Saturation (SpO2), Pulse Rate (PR), Non-invasive Blood Pressure (NIBP), Invasive Blood Pressure (IBP), Pulmonary Artery Wedge Pressure (PAWP), Carbon Dioxide (CO2), Oxygen (O2), and Continuous Cardiac Output (CCO). The monitor also provides an interpretation of resting 12-lead ECG.

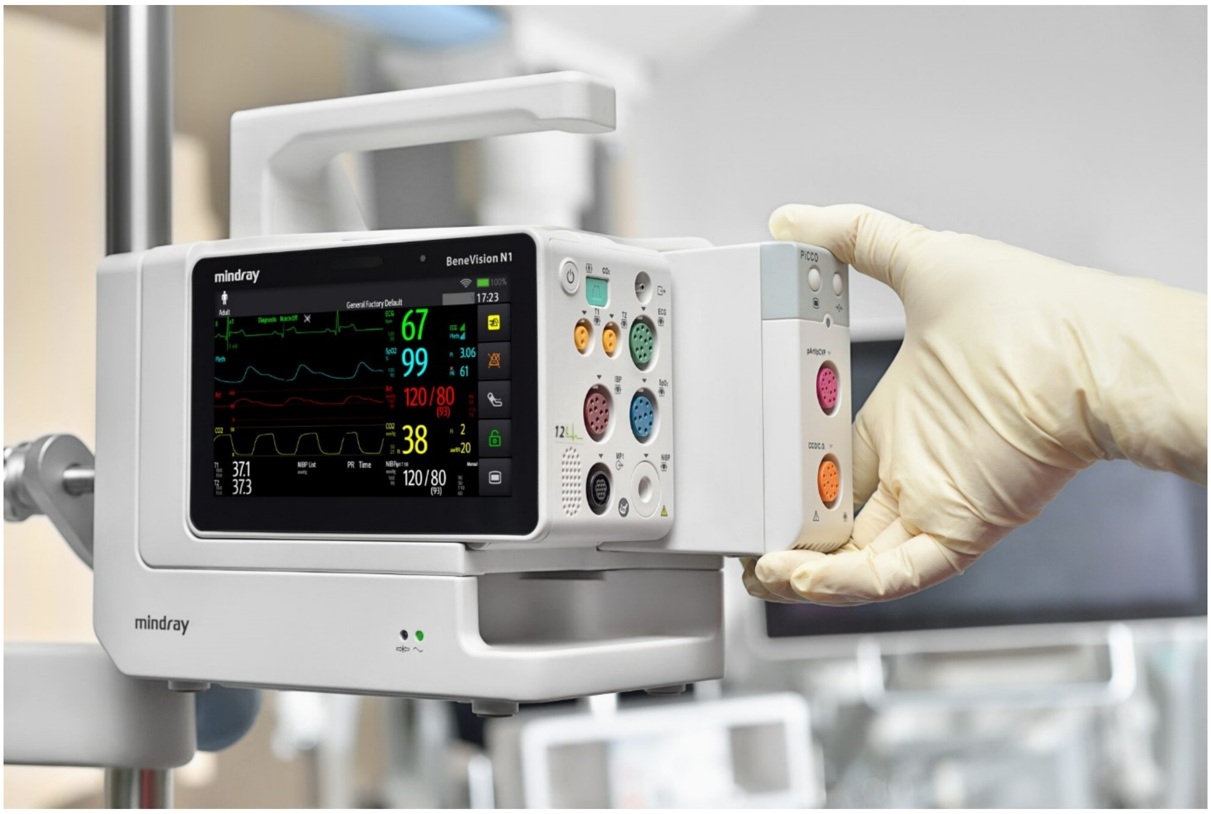


Figure 34: Smart monitor hardware.

The applied parts of the monitor are:

- ECG electrode and lead wire
- SpO2 sensor
- Temp probe
- NIBP cuff
- IBP transducer
- PiCCO sensor
- CO2 sampling line/nasal sampling cannula, water trap, and mask

The monitor comes with a modular parameter module (N1) that allows patients to freely move even while connected to the monitor. When the N1 is connected to the host monitor, the N1 enters the module mode. The N1 monitor has the following features when it enters the module mode:

- The patient information, parameter setup, and alarm setup of the N1 and the host monitor will be synchronized. For data transfer strategy, see the operator's manual of the host monitor.
- The N1 can still store the parameter data and the alarm events.
- The N1 receives and stores the parameter trends data from the host monitor.
- All audible sounds of the N1 are off.
- Wired and wireless network of the N1 is not available.

- The alarm indications of the battery-related alarms of the N1 are given by the host monitor.
- Turning on or off the host monitor simultaneously powers on or off the N1.
- The main screen of the N1 is off when it is connected to the host monitor through the SMR or the module rack of the host monitor.
- The N1 resumes monitoring even when it is disconnected from the host monitor

In HosmartAI the monitors are used to collect health quality measures and structure them in an interoperable HL7-FHIR format. We also exploit the MU-Connect IT solution which provides a universal central monitoring platform and fully integrates bedside medical devices using a standard interface to connect a 3rd-party information system (i.e., UM HAPI FHIR). The real-time access to health data during the grand round routine is ensured with speech-enabled user interfaces delivered via a tablet attached to the socially assistive robot Pepper.

2.2.9 Gradior Management System and APP Gradior

Gradior Cognitive is a neuropsychological assessment and rehabilitation system for the implementation of training and recovery programmes of higher cognitive functions in people with cognitive deficits and/or impairment. This program is accessible through electronic devices such as a tablet or a computer.

The program consists of the application of a series of cognitive modalities, in the form of audio-visual exercises (attention, memory, orientation, perception, calculation...). Each of them contains several sub-modalities (e.g., sustained auditory attention, iconic short-term memory, etc.) and each sub-modality can have from 2 to 11 levels of difficulty. In total, the Gradior program contains 45 different types of exercises. Data collection is done automatically by a statistical score tool called Gradior score. This score is a quantitative variable that calculates the average of results obtained in the tests of each cognitive sub-modality providing a monthly score since the beginning of the cognitive treatment, summarizing performance in each sub-modality and overall performance.

In a standard therapeutic routine (commonly called treatment), the predetermined time for each exercise is approximately 1 minute, and the total duration of the session is 30 minutes. Participants finish the 30-minute session with a certain number of exercises performed. The next treatment session begins with the exercises not performed in the previous session. This prevents the participant from performing the same exercise more than once in a session.

Treatment for participants includes exercises adapted to each treatment group, complemented with a baseline evaluation to define starting level of difficulty for each participant. The intervention routine is reviewed for each participant monthly.

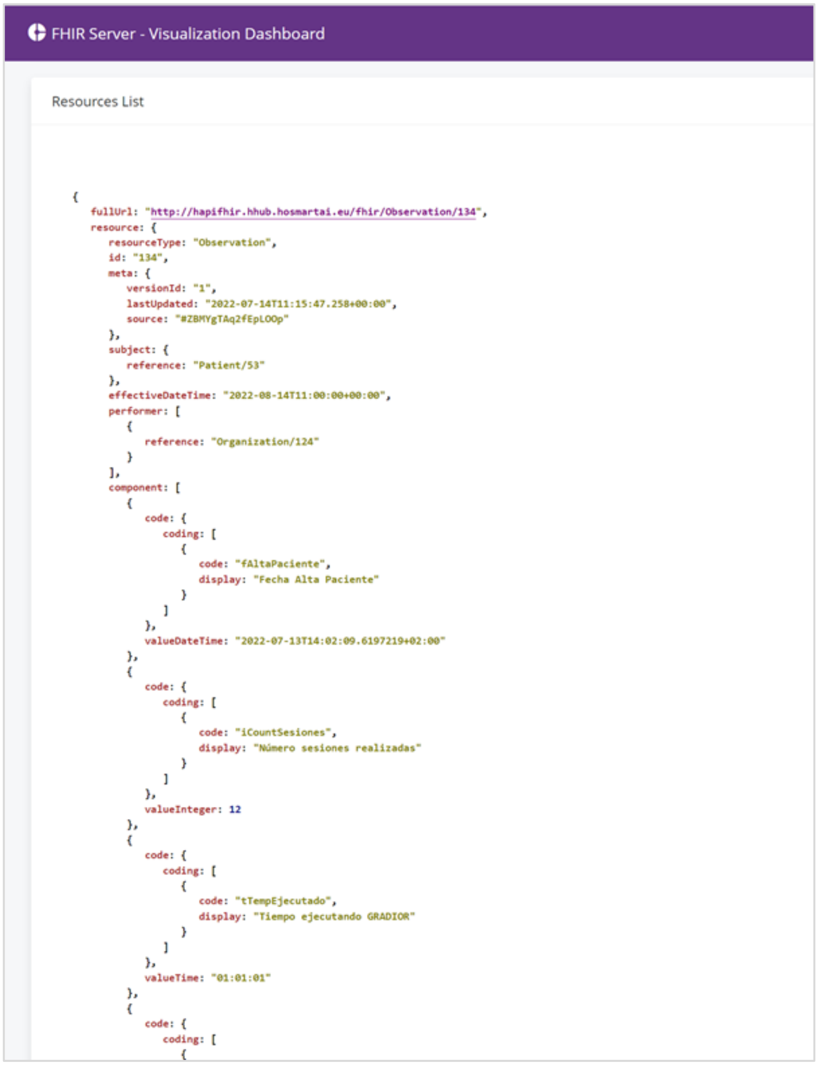
Requirements:

- WEB Manager: No specific requirements are necessary. Any device (Tablet, mobile, PC) with internet connection and internet connection using any browser will be enough.

- GRADIOR APP: Depending on the device where the GRADIOR APP is installed, the requirements are the following:
  - PC:
    - Processor: Intel Core i3 (6th generation or better)
    - Memory: RAM 4GB minimum
    - Minimum Operating Systems: Windows 10 Fall Creators Update OS build 16299. iOS 13.
    - Ethernet or WIFI internet connection
  - Tablet**:**
    - Processor frequency: 1.6 GHz
    - RAM: 2Gb LPDDR4X minimum
    - Memory (ROM): 32 Gb (requires 2 Gb free for installation)
    - Screen: 10'' (for usability reasons)
    - Minimum operating system: Android 9.0
    - 3G or WIFI internet connection

The professional's web management and the patient intervention app offer an **Intuitive monitoring dashboard** summarizing results to facilitate the clinical assessment. Further description of the Gradior component can be found in D3.1 "Design of AI-based Solutions and Autonomous Smart Components" in pages 70 to 72. Information regarding the connection to other tools and the deployment in the clinical setting can be consulted in D5.4 "HosmartAI Pilots – First version", in Chapter 8.

The results collected during the Gradior sessions are sent to the HosmartAI platform in an interoperable HL7-FHIR format according to the corresponding regulations.



```
{
   fullUrl: "http://hapifhir.hhub.hosmartai.eu/fhir/Observation/134",
   resource: {
      resourceType: "Observation",
      id: "134",
      meta: {
         versionId: "1",
         lastUpdated: "2022-07-14T11:15:47.258+00:00",
         source: "#ZBMYgTAq2fEpLOOp"
      },
      subject: {
         reference: "Patient/53"
      },
      effectiveDateTime: "2022-08-14T11:00:00+00:00",
      performer: [
         {
            reference: "Organization/124"
         }
      ],
      component: [
         {
            code: {
               coding: [
                  {
                     code: "fAltaPaciente",
                     display: "Fecha Alta Paciente"
                  }
               ]
            },
            valueDateTime: "2022-07-13T14:02:09.6197219+02:00"
         },
         {
            code: {
               coding: [
                  {
                     code: "iCountSesiones",
                     display: "Número sesiones realizadas"
                  }
               ]
            },
            valueInteger: 12
         },
         {
            code: {
               coding: [
                  {
                     code: "tTempEjecutado",
                     display: "Tiempo ejecutando GRADIOR"
                  }
               ]
            },
            valueTime: "01:01:01"
         },
         {
            code: {
               coding: [
                  {
```

Figure 35: Gradior configuration.

2.2.10 VIMAR View Wireless system

The Vimar View Wireless system is designed to manage lighting in environments, roller shutters or motorized curtains, and monitor energy consumption. The users can control and interact with many devices in their home with an app on the smartphone, through which it is also possible to create personalized scenarios. The installation is simple and does not require masonry, thus it is ideal for renovations or to boost the functions of an existing system. The possibility of interacting with different devices using exclusively a smartphone makes this solution a useful means of support for the elderly and people with restricted mobility.

The installation of the connected framework is easy and does not require invasive intervention. It is possible to create a connected system with recessed devices, suitable for any architectural context, thanks to the completely matching styling of the digital products and their easy functional expandability. The wiring of connected devices requires a power supply (L, N) and connection to the related loads and/or electro-mechanical control devices (2-way switches, 1-way switches, push buttons) to replicate control points or activate scenarios. The battery-free and wireless controls based on energy harvesting technology by

EnOcean make it possible to add control points in complete freedom at any time. It is sufficient to substitute the traditional modules with the new connected version.

A gateway module needs to be installed, it allows the connection and communication with the cloud.

Each device must be installed and configured through View Wireless App.

2.2.10.1 Network and Communication

The devices are pre-configured by default with the Bluetooth® technology 5.0 standard and this is the communication protocol used in the installation in San Camillo hospital. The devices can also operate with the Zigbee technology standard.

The Bluetooth technology standard is designed to use devices in a mesh network, in which the gateway (Bluetooth® and Wi-Fi technology) is designed to control the system from the user View App both locally and remotely, and to control the system with voice assistants. The system is compatible with IFTTT, also integrating IFTTT-compatible third-party devices.

The system is configured in Bluetooth Mesh technology mode and all the parameters are set via the configuration APP.

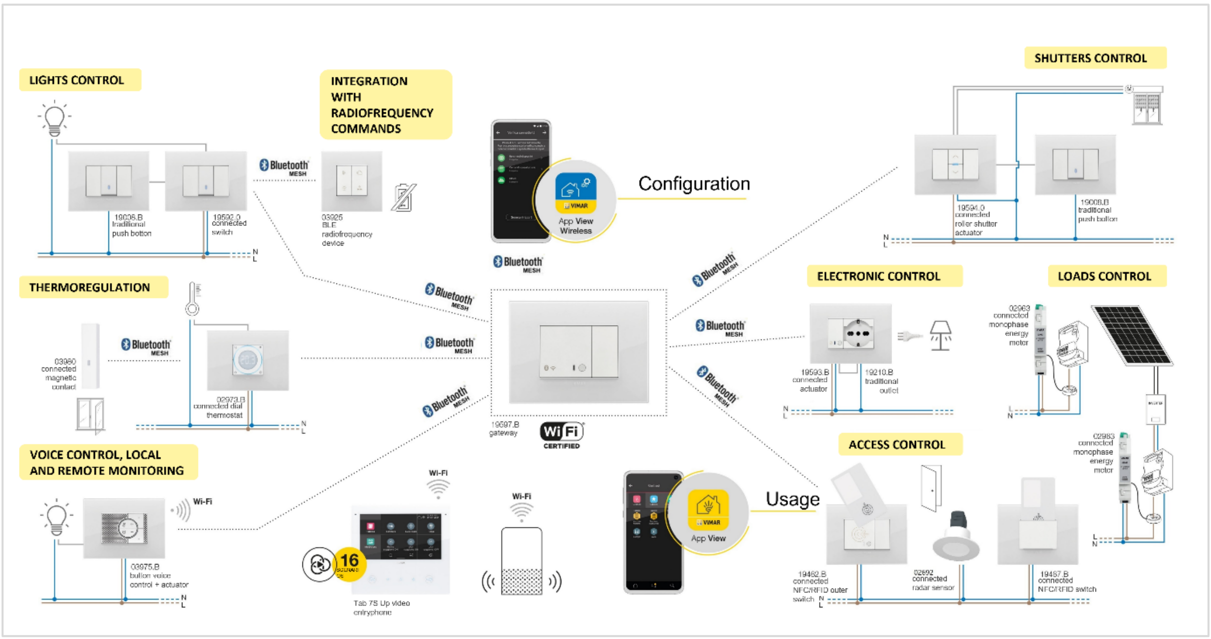

LIGHTS CONTROL
19006.B traditional push botton
19592.0 connected switch
INTEGRATION WITH RADIOFREQUENCY COMMANDS
03925 BLE radiofrequency device
Bluetooth MESH
App View Wireless
Configuration
SHUTTERS CONTROL
19594.0 connected roller shutter actuator
19008.B traditional push button
THERMOREGULATION
03980 connected magnetic contact
02973.B connected dial thermostat
19597.B gateway
Wi Fi CERTIFIED
ELECTRONIC CONTROL
19593.B connected actuator
19210.B traditional outlet
LOADS CONTROL
02963 connected monophase energy meter
VOICE CONTROL, LOCAL AND REMOTE MONITORING
03975.B button voice control + actuator
Wi-Fi
Tab 7S Up video entryphone
16 SCENARIOS
ACCESS CONTROL
App View
Usage
19462.B connected NFC/RFID outer switch
02692 connected radar sensor
19467.B connected NFC/RFID switch
N
L

Figure 36: Architecture of the system.

2.2.10.2 Available Devices

Table 2: Catalogue of available devices.

Type of device	Code	Description	Functions
Connected gateway	20597 19597	Bluetooth technology Wi-Fi device designed to allow dialogue with wireless	

Type of device	Code	Description	Functions
	16497 14597	devices to permit the configuration, supervision, system diagnostics and its integration with voice assistants. Main device that manages the Bluetooth technology Mesh network. Via the View Wireless App, it receives the system configuration through Bluetooth technology. The presence of Wi-Fi connectivity is required to allow the connection to the cloud for supervision and for integrations with Alexa, Google Assistant, and Siri voice assistants.	
Connected 2-way switch 	20592.0 19592 19592.0 16492 14592.0 14592 03981	The electronic switch mechanism connected is designed to operate a load via an onboard push button, through a wireless connection, and from a traditional remote push button. The device is equipped with 2 interlocked relay outputs to accomplish the switch function and a front key to control the connected load. It performs the automatic opening of the relay for thermal protection. Switching on zero crossing. The electronic switch can be connected to existing wired multi-way/two-way switches to make the load function “connected”.	• Toggle on/off • One-position stable activation time
Connected rolling shutter mechanism 	20540 19594 19594.0 16494 14594 14594.0 03982	The device makes it possible to control the roller shutter/slat using the onboard keys and via a wireless connection. It is equipped with mutually exclusive activation of the relays with a minimum interlocking time. The front keys of the device control the onboard roller shutter actuator, starting or stopping the slat movement or the rotation. It allows also the recall of a favourite position.	• Slat orientation • Roller shutter activation • Preferred position • Movement check • Scenario activation • Status check
Connected actuator 	20593 19593 16493 14593	The actuator is equipped with a relay output with a current meter and a front push button with which to reset the load and perform configuration/reset. Its function is to protect against overcurrent by cutting off the load when the threshold value set via the View Wireless App is exceeded. Load reactivation, aside from the front push button, can also be done via the View App. The View App also makes it	• Load cut-off threshold function • Consumption threshold for load cut-off • Load status when the power supply is restored • Relay operation: two-position stable or one-position stable

Type of device	Code	Description	Functions
		possible to View the instant power consumed.	• One-position stable activation time
Monophase IoT energy meter	02963	The device is designed to measure the consumption/production of instantaneous electricity and consumption logs with an hourly, daily, monthly and annual resolution. It should be connected to the single-phase line using the current probe provided. Only one meter for total consumption can be installed in a system.	• Energy consumption/production • Monitoring of instant power consumption/ production • Monitoring of instant energy consumption/ production
NFC/RFID smart card landing reader	19462 20462 14462	Access control is achieved with the combined usage of an NFC/RFID smart card landing reader and NFC/RFID smart card reader pocket, both controlled and configured by using View Wireless App. The smart card landing reader device is designed to be installed outdoors and near an entrance and it grants access only if the smart card associated with it is read and recognized.	• Recognition of the smart card (that triggers the door opening) • Anomaly detection on the reader • Do Not Disturb signaling • "Crossover relay" option for combined operation with card reader pocket
NFC/RFID smart card reader pocket	19467 20467 14467	NFC/RFID smart card reader pocket allows the activation of utilities only if the wireless smart card associated with it is read and recognized. The two devices are designed to communicate (if associated during configuration) to manage accesses to the same room and ensure greater safety via the "Crossover relay" option.	• Recognition of the smart card (with toggle off if card removed) • "Crossover relay" option for combined operation with card landing reader
Ultra-Wide Band (UWB)	14179 16629 19179 20179 30179 02692	This sensor can detect human movement/presence without using Fresnel lenses. It employs a military-based radar UWB technology capable of detecting centimetres-wide human movements. It has been conceived a recessed version and one to be installed in the ceiling. The sensor has been developed during the HosmartAI project.	• People Presence/absence • Micro movements detection/Breath detection • Load activation • Area/volume of detection parametrization

2.2.11 Windows and Android Tablets, Smartphones, for View VIMAR APP and iMat.

2.2.11.1 App View and View Wireless

The connected environment provided by VIMAR can be managed and controlled through App View Wireless and App View, both available on main stores. The first is used by the installer to set up and configure the system. It allows the creation of environments and the association of all the devices with the respective environments. For every device the installer can set the function, the parameters, and any accessory devices.

The installer, via the View Wireless App, delivers the configured system to the Administrator. The Administrator user, via the View App, can now manage the system functions and associate other users assigning rights and permissions. App View allows the user to:

- customize up to 16 scenarios.
- check the status of presence, lights, roller shutters or curtains and of the loads connected to the socket outlets.
- view the consumption throughout the home.
- receive notifications if the contractual power level is exceeded.
- integrate the app with the IFTTT platform to integrate with third-party connected devices.
- check the presence and access of users.

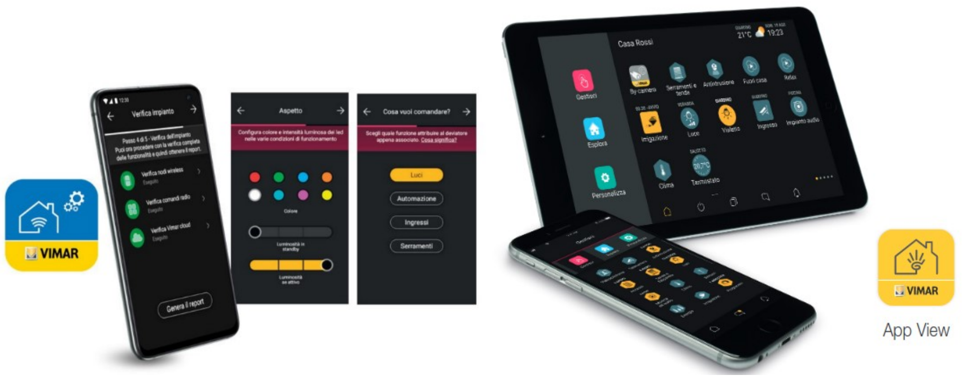
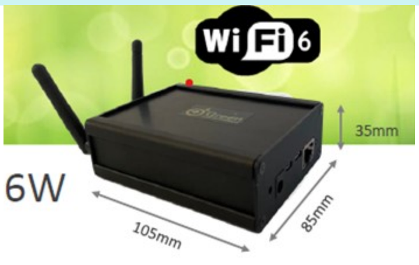


Figure 37: Examples of the VIMAR App.

2.2.12 YOI-6 router (embedded blockchain node) - SBC with Wi-Fi or 4G interfaces



Green Communications provides ready-to-use edge platforms for the easy setup of an edge-based solution with connectivity, edge cloud and services among which the blockchain.

YOI is an embedded Linux router equipped with Green Communications' software (GreenSoft). Each router comes with one dual-band Wi-Fi 6 interface that creates a network with other YOI, and provides access to smartphones, tablets, laptops, or any other Wi-Fi device. YOI features a web server, so one can provide local content, services, and applications to the network. YOI can also be configured as a gateway. In this case, local traffic stays local and global traffic is sent across the gateways to other networks.

YOI router specifications are listed in the following table:

Table 3: YOI router specifications.

Ethernet:	1 Ethernet port (10/100/1000 Mbit/s)
Wi-Fi:	1 dual-band Wi-Fi card (a, b, g, n, ac, ax), providing 2 Wi-Fi interfaces to the system, one assigned to the 2.4 GHz band, and one assigned to the 5 GHz band. Each interface can operate simultaneously with the other on its dedicated band
Suggested operation:	Wi-Fi no. 1: Backhaul Wi-Fi no. 2: Access Ethernet: Internet when available
Frequency:	2.4 GHz and 5 GHz
Antenna:	2 dual-band Wi-Fi (RP-SMA, 5 dBi max)
Wireless rates:	Up to 1.2 Gbit/s
Encryption:	WPA2/WPA3 (access), SAE (backhaul)
Operating system:	Custom Linux system (based on Buildroot)
CPU:	Cortex-A9 800 MHz Dual Core
RAM:	512 MB
Other interfaces:	1 USB OTG, 1 Micro-HDMI
Environmental features:	-40ºC +85ºC
Dimensions:	Approx. 85 × 105 × 35 mm (casing without antenna)
Weight:	280g (enclosure and antennas included)
Power supply:	power through Ethernet (passive PoE) or Barrel Jack DC from 8 to 60 V (AC adapter included)
Power consumption:	≃ 6W
Software:	Each YOI comes with a GreenSoft license

In the context of the HosmartAI project, YOI routers will be displayed on Pilot #5 and Pilot #6 premises to connect the PEPPER robots and collect and log the robot's activities in the blockchain service embedded in the YOI router.

3 OpenAPI Specification

In this section we present the main elements of the architecture that require OpenAPI communication to perform their function.

3.1 Tools for managing OpenAPIs

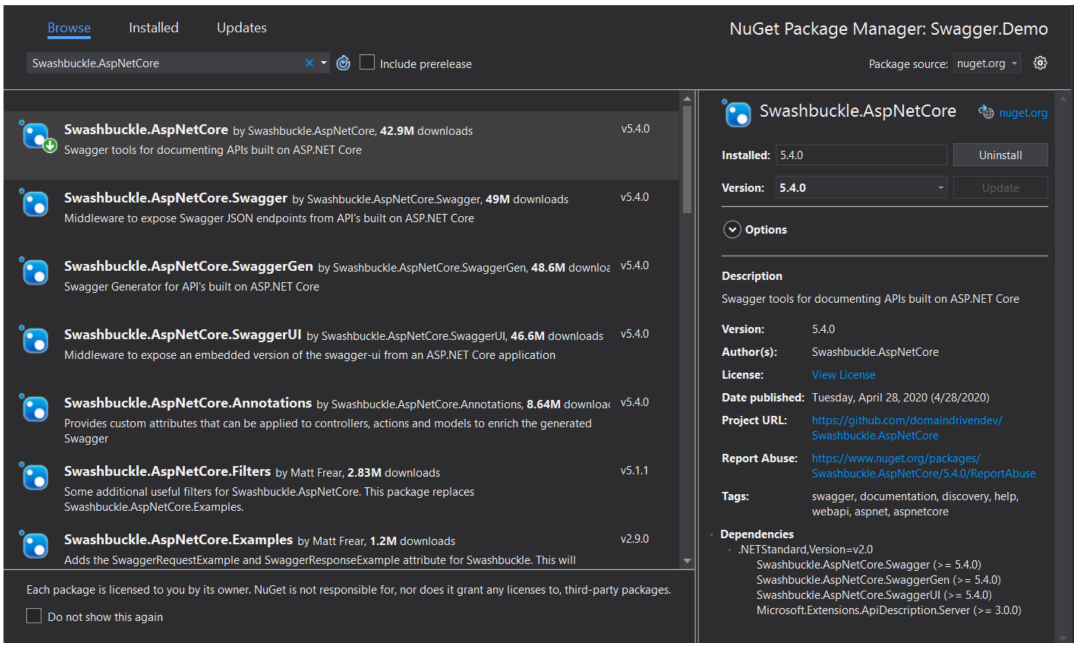
The OpenAPI can be generated and edited in different forms. There is a long list of such tools for this task. The following are the tools used to perform this task.

3.1.1 Generators

3.1.1.1 Swashbuckle (NET Core)

This tool consists of a swagger tooling for APIs built with ASP.NET Core. Generates API documentation, including a UI to explore and test operations, directly from the routes, controllers, and models.

In addition to its Swagger 2.0 and OpenAPI 3.0 generator, Swashbuckle also provides an embedded version of the swagger-UI that's powered by the generated Swagger JSON. This means that can complement your API with living documentation that's always in sync with the latest code. It requires minimal coding and maintenance, allowing you to focus on building an awesome API.



NuGet Package Manager: Swagger.Demo

Swashbuckle.AspNetCore

Package source: nuget.org

Swashbuckle.AspNetCore by Swashbuckle.AspNetCore, **42.9M** downloads v5.4.0
Swagger tools for documenting APIs built on ASP.NET Core

Swashbuckle.AspNetCore.Swagger by Swashbuckle.AspNetCore.Swagger, **49M** downloads v5.4.0
Middleware to expose Swagger JSON endpoints from API's built on ASP.NET Core

Swashbuckle.AspNetCore.SwaggerGen by Swashbuckle.AspNetCore.SwaggerGen, **48.6M** downloa v5.4.0
Swagger Generator for API's built on ASP.NET Core

Swashbuckle.AspNetCore.SwaggerUI by Swashbuckle.AspNetCore.SwaggerUI, **46.6M** downloads v5.4.0
Middleware to expose an embedded version of the swagger-ui from an ASP.NET Core application

Swashbuckle.AspNetCore.Annotations by Swashbuckle.AspNetCore.Annotations, **8.64M** downloa v5.4.0
Provides custom attributes that can be applied to controllers, actions and models to enrich the generated Swagger

Swashbuckle.AspNetCore.Filters by Matt Frear, **2.83M** downloads v5.1.1
Some additional useful filters for Swashbuckle.AspNetCore. This package replaces Swashbuckle.AspNetCore.Examples.

Swashbuckle.AspNetCore.Examples by Matt Frear, **1.2M** downloads v2.9.0
Adds the SwaggerRequestExample and SwaggerResponseExample attribute for Swashbuckle. This will

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Swashbuckle.AspNetCore nuget.org

Installed: 5.4.0
Version: 5.4.0

Description
Swagger tools for documenting APIs built on ASP.NET Core

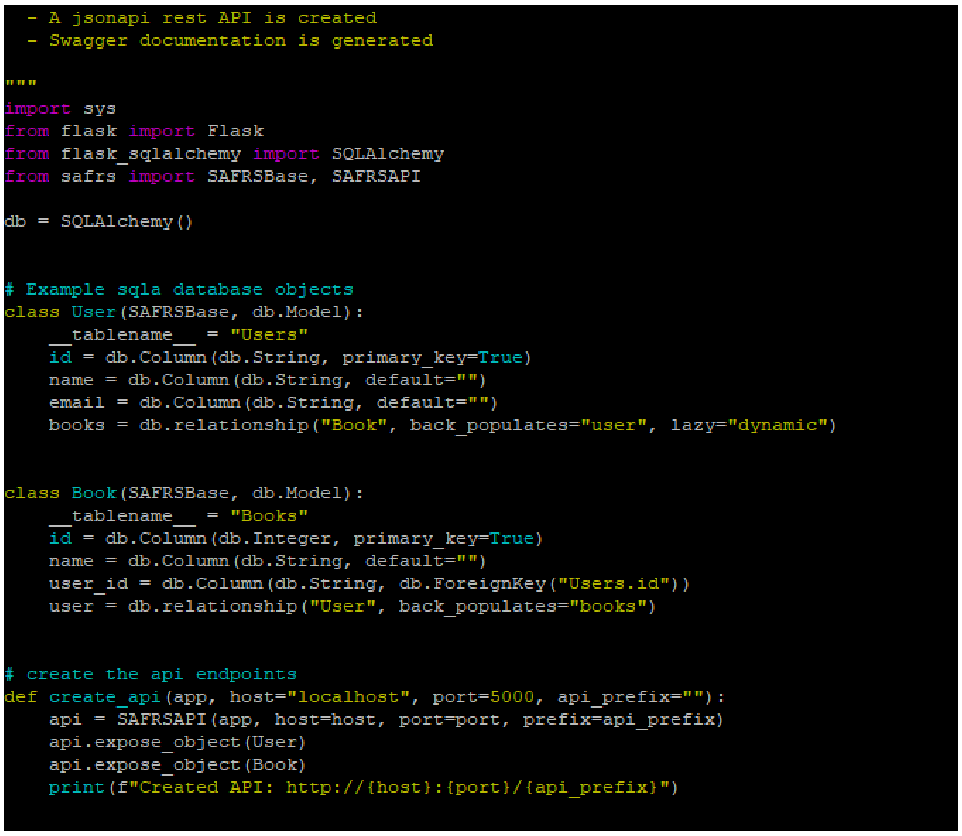
Version:	5.4.0
Author(s):	Swashbuckle.AspNetCore
License:	View License
Date published:	Tuesday, April 28, 2020 (4/28/2020)
Project URL:	https://github.com/domaindrivendev/Swashbuckle.AspNetCore
Report Abuse:	https://www.nuget.org/packages/Swashbuckle.AspNetCore/5.4.0/ReportAbuse
Tags:	swagger, documentation, discovery, help, webapi, aspnet, aspnetcore

Dependencies
.NETStandard,Version=v2.0
Swashbuckle.AspNetCore.Swagger (>= 5.4.0)
Swashbuckle.AspNetCore.SwaggerGen (>= 5.4.0)
Swashbuckle.AspNetCore.SwaggerUI (>= 5.4.0)
Microsoft.Extensions.ApiDescription.Server (>= 3.0.0)

Figure 38: Tool installation from IDE.

3.1.1.2 safrs (Python)

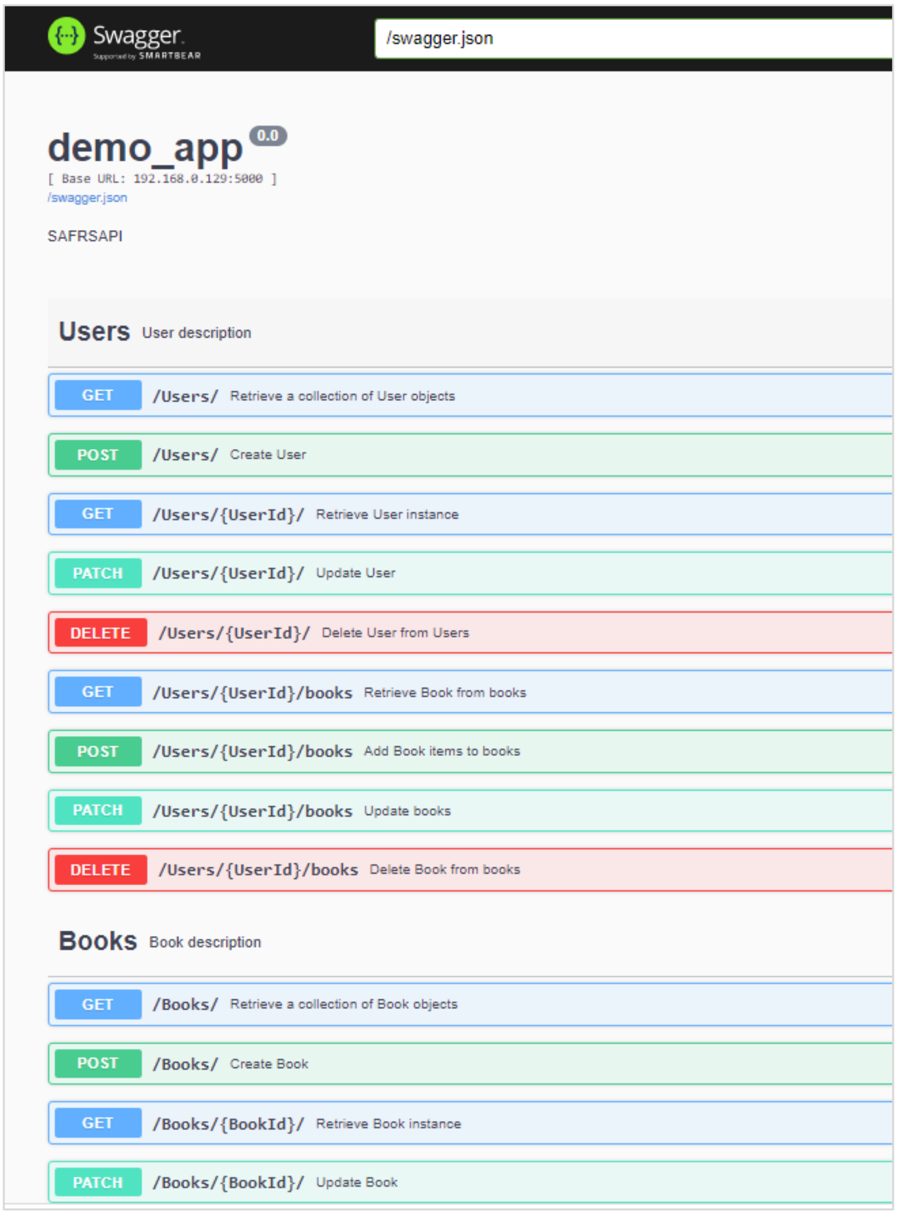
SAFRS [REF-12] is a Python library that exposes a database defined with the framework SQLAlchemy as a JSON:API web service and generates the corresponding OpenAPI specification for a swagger service.



```
  - A jsonapi rest API is created
  - Swagger documentation is generated

"""
import sys
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from safrs import SAFRSBase, SAFRSAPI

db = SQLAlchemy()


# Example sqla database objects
class User(SAFRSBase, db.Model):
    __tablename__ = "Users"
    id = db.Column(db.String, primary_key=True)
    name = db.Column(db.String, default="")
    email = db.Column(db.String, default="")
    books = db.relationship("Book", back_populates="user", lazy="dynamic")


class Book(SAFRSBase, db.Model):
    __tablename__ = "Books"
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String, default="")
    user_id = db.Column(db.String, db.ForeignKey("Users.id"))
    user = db.relationship("User", back_populates="books")


# create the api endpoints
def create_api(app, host="localhost", port=5000, api_prefix=""):
    api = SAFRSAPI(app, host=host, port=port, prefix=api_prefix)
    api.expose_object(User)
    api.expose_object(Book)
    print(f"Created API: http://{host}:{port}/{api_prefix}")
```

Figure 39: Example of safrs code.

By default, it generates GET (retrieve an object), POST (Create an object), DELETE (Remove an object) and PATCH (Update an object) methods for a selected object and the objects it's related to.



demo_app 0.0
[Base URL: 192.168.0.129:5000]
/swagger.json

SAFRSAPI

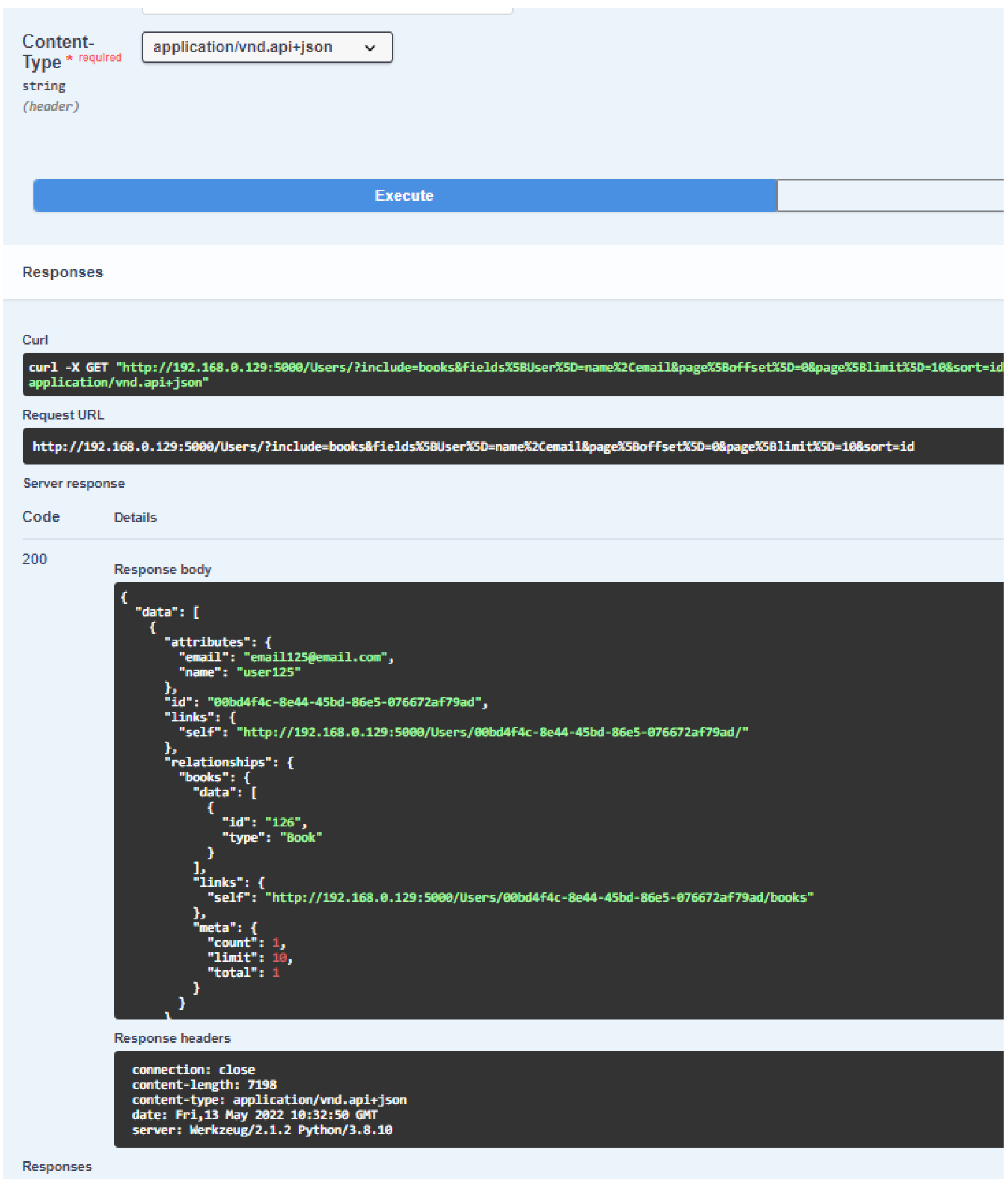
Users User description

GET /Users/ Retrieve a collection of User objects
POST /Users/ Create User
GET /Users/{UserId}/ Retrieve User instance
PATCH /Users/{UserId}/ Update User
DELETE /Users/{UserId}/ Delete User from Users
GET /Users/{UserId}/books Retrieve Book from books
POST /Users/{UserId}/books Add Book items to books
PATCH /Users/{UserId}/books Update books
DELETE /Users/{UserId}/books Delete Book from books

Books Book description

GET /Books/ Retrieve a collection of Book objects
POST /Books/ Create Book
GET /Books/{BookId}/ Retrieve Book instance
PATCH /Books/{BookId}/ Update Book

Figure 40: Safrs autogenerated Swagger.



Curl

```
curl -X GET "http://192.168.0.129:5000/Users/?include=books&fields%5BUser%5D=name%2Cemail&page%5Boffset%5D=0&page%5Blimit%5D=10&sort=id
application/vnd.api+json"
```

Request URL

```
http://192.168.0.129:5000/Users/?include=books&fields%5BUser%5D=name%2Cemail&page%5Boffset%5D=0&page%5Blimit%5D=10&sort=id
```

Server response

Code Details

200

Response body

```
{
  "data": [
    {
      "attributes": {
        "email": "email125@email.com",
        "name": "user125"
      },
      "id": "00bd4f4c-8e44-45bd-86e5-076672af79ad",
      "links": {
        "self": "http://192.168.0.129:5000/Users/00bd4f4c-8e44-45bd-86e5-076672af79ad/"
      },
      "relationships": {
        "books": {
          "data": [
            {
              "id": "126",
              "type": "Book"
            }
          ],
          "links": {
            "self": "http://192.168.0.129:5000/Users/00bd4f4c-8e44-45bd-86e5-076672af79ad/books"
          },
          "meta": {
            "count": 1,
            "limit": 10,
            "total": 1
          }
        }
```

Response headers

```
connection: close
content-length: 7198
content-type: application/vnd.api+json
date: Fri,13 May 2022 10:32:50 GMT
server: Werkzeug/2.1.2 Python/3.8.10
```

Responses

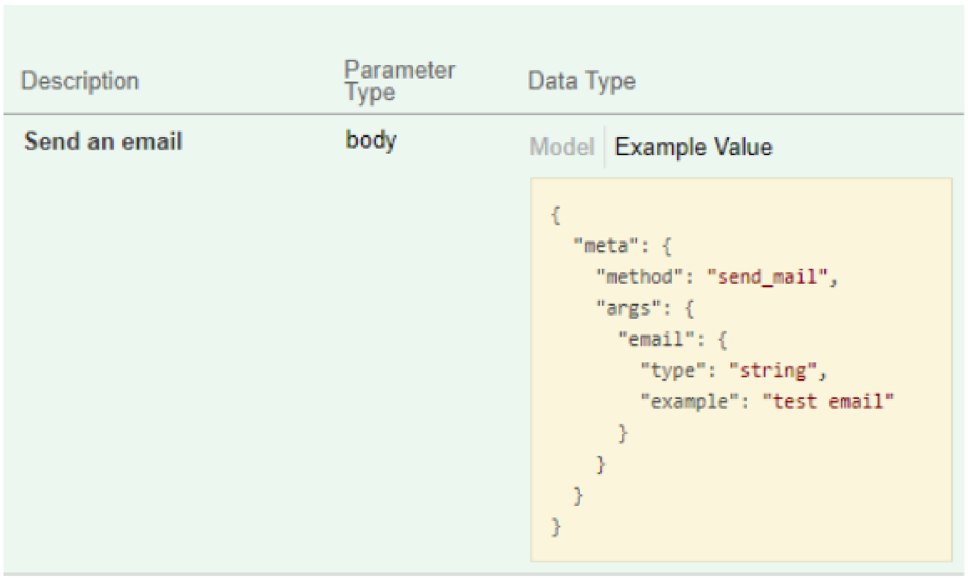
Figure 41: Safrs Swagger call example.

OpenAPI descriptions can be added by using comments to classes and functions, mainly by adding a triple double-quote comment below the class or function definition.

The “description” comment describes the parameter, and the “args” comment can give more information about the field's names, types, and examples of use.

For example:

```
def send_mail(self, **args):
"""
    description: Send an email
    args:
      email:
        type: string
        example: test email
"""
```



Description	Parameter Type	Data Type
Send an email	body	Model Example Value

```
{
  "meta": {
    "method": "send_mail",
    "args": {
      "email": {
        "type": "string",
        "example": "test email"
      }
    }
  }
}
```

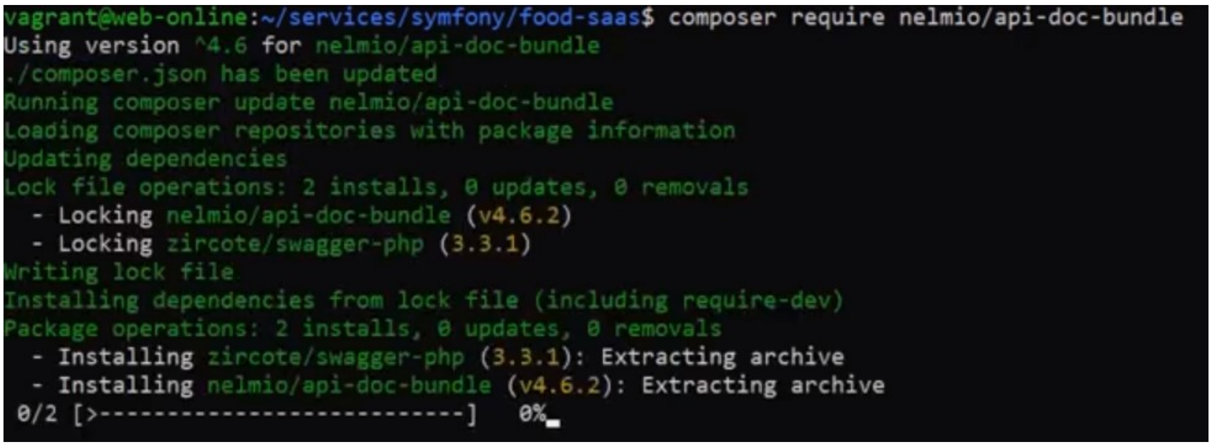
Figure 42: Safrs description swagger example.

3.1.1.3 NelmioApiDocBundle (PHP-symfony)

NelmioApiDocBundle is a framework that adds OpenAPI and swagger generation to PHP-Symfony.

To install it, Symfony and PHP must be installed and configured. Then, run the following command in the terminal:

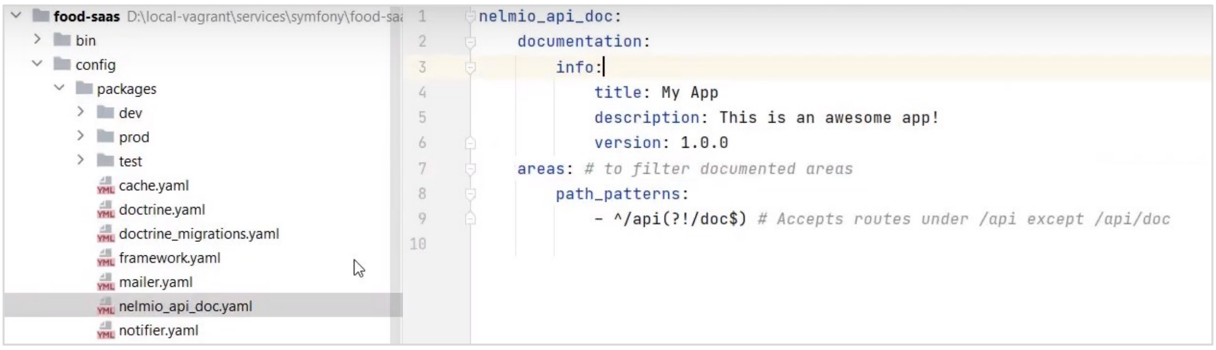
composer require nelmio/api-doc-bundle



```
vagrant@web-online:~/services/symfony/food-saas$ composer require nelmio/api-doc-bundle
Using version ^4.6 for nelmio/api-doc-bundle
./composer.json has been updated
Running composer update nelmio/api-doc-bundle
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
  - Locking nelmio/api-doc-bundle (v4.6.2)
  - Locking zircote/swagger-php (3.3.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
  - Installing zircote/swagger-php (3.3.1): Extracting archive
  - Installing nelmio/api-doc-bundle (v4.6.2): Extracting archive
 0/2 [>---------------------------]   0%
```

Figure 43: Nelmio api-doc-bundle installation.

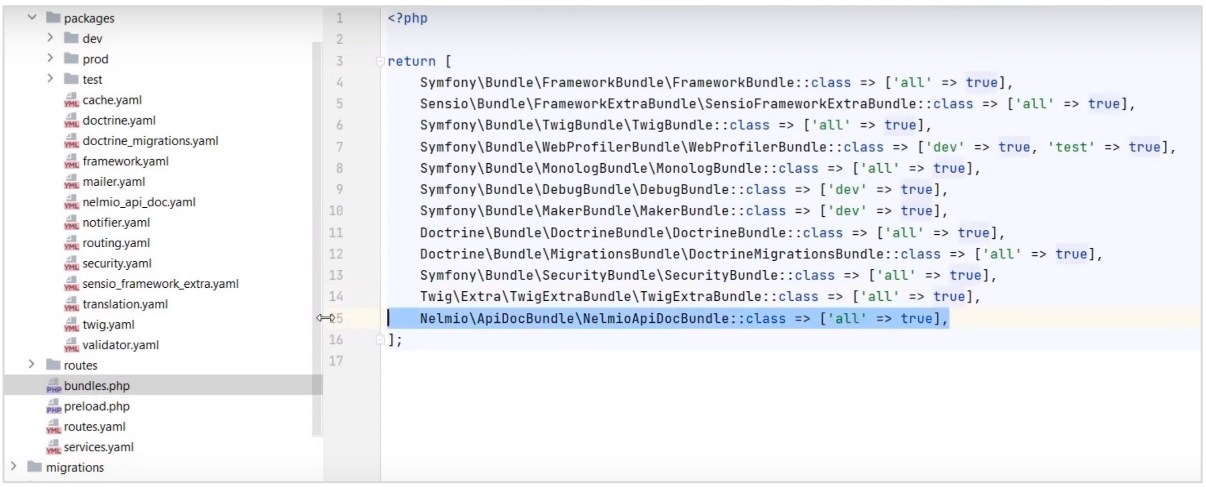
The project needs the file: nelmio_api_doc.yaml, where the main documentation for the API can be added, and a regular expression for only showing paths to API URLs in the openAPI.



```
nelmio_api_doc:
    documentation:
        info:
            title: My App
            description: This is an awesome app!
            version: 1.0.0
    areas: # to filter documented areas
        path_patterns:
            - ^/api(?!/doc$) # Accepts routes under /api except /api/doc
```

Figure 44: Nelmio api doc yaml.

In the file bundles.php, there must be a line including nelmio \ apiDocBundle \ NelbioApiDocBundle.

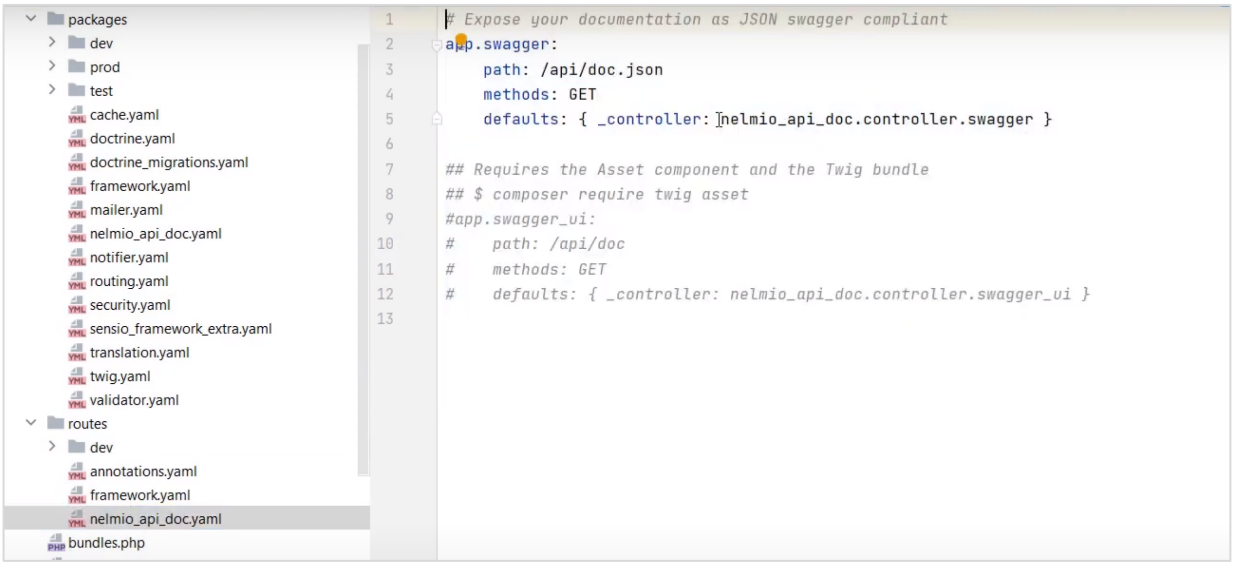


packages
dev
prod
test
cache.yaml
doctrine.yaml
doctrine_migrations.yaml
framework.yaml
mailer.yaml
nelmio_api_doc.yaml
notifier.yaml
routing.yaml
security.yaml
sensio_framework_extra.yaml
translation.yaml
twig.yaml
validator.yaml
routes
bundles.php
preload.php
routes.yaml
services.yaml
migrations

```php
<?php

return [
    Symfony\Bundle\FrameworkBundle\FrameworkBundle::class => ['all' => true],
    Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle::class => ['all' => true],
    Symfony\Bundle\TwigBundle\TwigBundle::class => ['all' => true],
    Symfony\Bundle\WebProfilerBundle\WebProfilerBundle::class => ['dev' => true, 'test' => true],
    Symfony\Bundle\MonologBundle\MonologBundle::class => ['all' => true],
    Symfony\Bundle\DebugBundle\DebugBundle::class => ['dev' => true],
    Symfony\Bundle\MakerBundle\MakerBundle::class => ['dev' => true],
    Doctrine\Bundle\DoctrineBundle\DoctrineBundle::class => ['all' => true],
    Doctrine\Bundle\MigrationsBundle\DoctrineMigrationsBundle::class => ['all' => true],
    Symfony\Bundle\SecurityBundle\SecurityBundle::class => ['all' => true],
    Twig\Extra\TwigExtraBundle\TwigExtraBundle::class => ['all' => true],
    Nelmio\ApiDocBundle\NelmioApiDocBundle::class => ['all' => true],
];
```

Figure 45: bundles.php nelmioApiDocBundle.

And the last thing to configure the OpenAPI must be a file “nalmio_api_doc.yaml”, in the routes folder, with the path for showing the swagger interface.



packages
dev
prod
test
cache.yaml
doctrine.yaml
doctrine_migrations.yaml
framework.yaml
mailer.yaml
nelmio_api_doc.yaml
notifier.yaml
routing.yaml
security.yaml
sensio_framework_extra.yaml
translation.yaml
twig.yaml
validator.yaml
routes
dev
annotations.yaml
framework.yaml
nelmio_api_doc.yaml
bundles.php

```yaml
# Expose your documentation as JSON swagger compliant
app.swagger:
    path: /api/doc.json
    methods: GET
    defaults: { _controller: nelmio_api_doc.controller.swagger }

## Requires the Asset component and the Twig bundle
## $ composer require twig asset
#app.swagger_ui:
#    path: /api/doc
#    methods: GET
#    defaults: { _controller: nelmio_api_doc.controller.swagger_ui }

```

By doing this, the /api/doc URL can be accessed, with the swagger interface, and the /api/doc.json can be accessed with the source code for the OpenAPI.

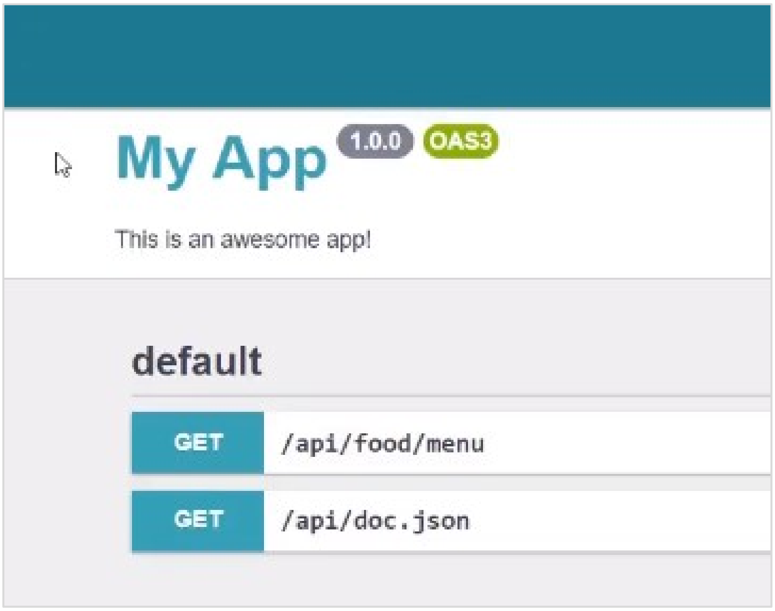
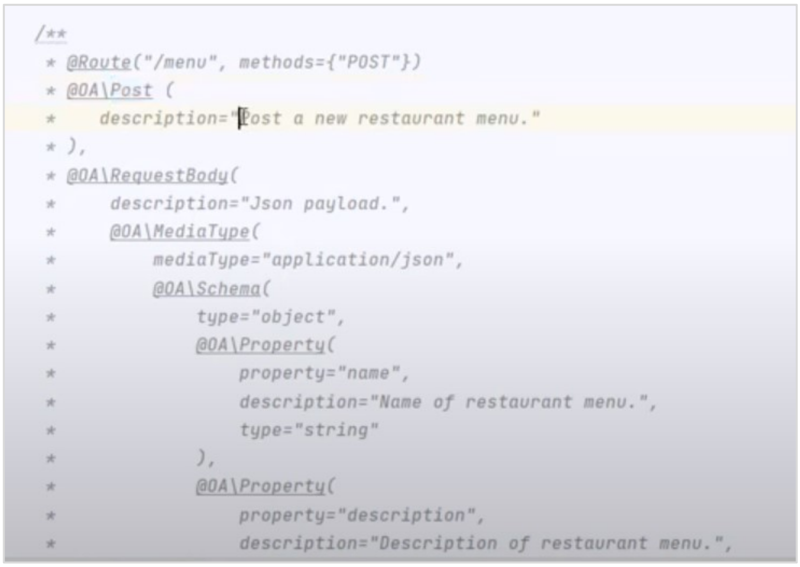


Figure 46: Swagger interface for NelmioApiDocBundle.



```
/**
 * @Route("/menu", methods={"POST"})
 * @OA\Post (
 *     description="Post a new restaurant menu."
 * ),
 * @OA\RequestBody(
 *      description="Json payload.",
 *      @OA\MediaType(
 *          mediaType="application/json",
 *          @OA\Schema(
 *              type="object",
 *              @OA\Property(
 *                  property="name",
 *                  description="Name of restaurant menu.",
 *                  type="string"
 *              ),
 *              @OA\Property(
 *                  property="description",
 *                  description="Description of restaurant menu.",
```

Figure 47: OpenAPI source code for NelmioApiDocBundle.

3.1.1.4 Swagger Maven Plugin

Swagger Maven Plugin is a plugin for java projects using maven to generate swagger API documentation while building with maven.

This plugin does not serve the online documentation after building but only generates the spec docs to be used later.

For using it, it’s only necessary to add it in the plugins block by writing the plugin definition:

```
<plugin>
        <groupId>com.github.kongchen</groupId>
        <artifactId>swagger-maven-plugin</artifactId>
        <version>${swagger-maven-plugin-version}</version>
        <configuration>
                <apiSources>
                        <apiSource>
                        ...
                        </apiSource>
                </apiSources>
        </configuration>
        <executions>
                <execution>
                        <phase>compile</phase>
                        <goals>
                                <goal>generate</goal>
                        </goals>
                </execution>
        </executions>
</plugin>
```

And edit the pom.xml file to add the dependency:

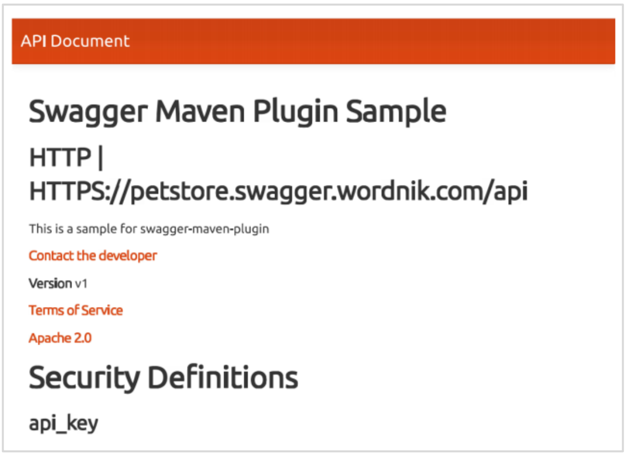
```
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.1</version>
</dependency>
```

Installation is complete at this point and just run the `mvn compile` command to generate the swagger.

As this plugin does not serve the web with generated documentation, it is necessary to expose it somehow. Nginx or Apache can be used, for example. The following command allows to get it up with docker:

docker run -it --rm -d -p 8080:80 --name web -v /home/adrian/swagger-maven-example/generated/:/usr/share/nginx/html nginx

Exposed documentation would be like this:


API Document

Swagger Maven Plugin Sample

HTTP | HTTPS://petstore.swagger.wordnik.com/api

This is a sample for swagger-maven-plugin

Contact the developer

Version v1

Terms of Service

Apache 2.0

Security Definitions

api_key

Figure 48: Swagger maven plugin main documentation web.

3.1.1.5 drf-yasg (Yet another Swagger generator)

It's a real **swagger/openAPI** specifications generator for **Django REST APIs**. Provides the option to choose between swagger-UI and redoc or both for documentation generation but it's only working with OpenAPI 2.0 and has no support for OpenAPI 3.0 and it's unplanned to give support for 3.0 in short term.

This tool is installed using python-pip and requires python to be installed in one of the "3.6,3.7,3.8,3.9" versions. Once installed, only need to run the following command to install drf-yasg:

pip install -U drf-yasg

Once installed, we should add it to the requirements.txt file by adding the following two lines*:*

```
djangorestframework==3.11
drf-yasg==1.20
```

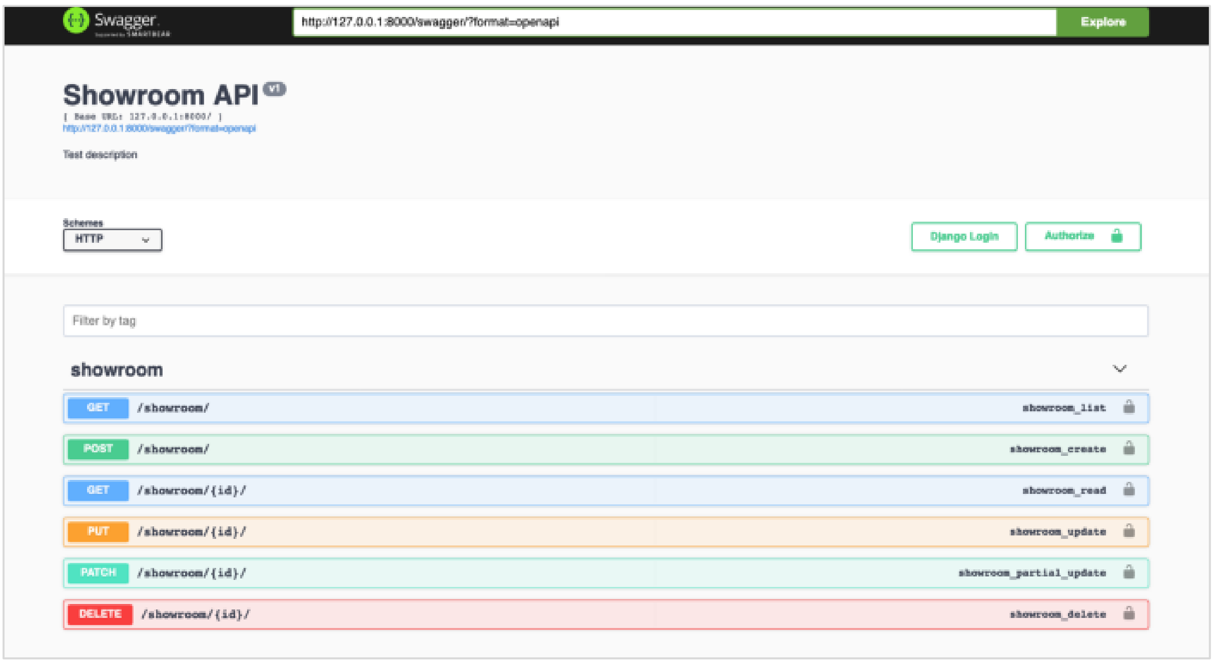
The next step is to add the following to the INSTALLED_APPS variable in settings.py:

```
INSTALLED_APPS=[
        ...
        'rest_framework',
        'django.contrib.staticfiles',
        'drf_yasg',
        ...
]
```

In the **urls.py** file, the following lines must be added to enable the WebUI for swagger. In urlpatterns, we added two different views, redoc and swagger to make them available as examples. Two exposed endpoints show a *JSON* and a *YAML* representation of the API specification.

```
schema_view = get_schema_view(
   openapi.Info(
      title="Snippets API",
      default_version='v1',
      description="Test description",
      terms_of_service="https://www.google.com/policies/terms/",
      contact=openapi.Contact(email="contact@snippets.local"),
      license=openapi.License(name="BSD License"),
   ),
   public=True,
   permission_classes=[permissions.AllowAny],
)

urlpatterns = [
   re_path(r'^swagger(?P<format>\.json|\.yaml)$',
schema_view.without_ui(cache_timeout=0), name='schema-json'),
   re_path(r'^swagger/$', schema_view.with_ui('swagger', cache_timeout=0),
name='schema-swagger-ui'),
   re_path(r'^redoc/$', schema_view.with_ui('redoc', cache_timeout=0), name='schema-
redoc'),
    ...
]
```

With these changes, the API is exposed in /swagger/ as shown:



http://127.0.0.1:8000/swagger/?format=openapi

Showroom API

[Base URL: 127.0.0.1:8000/]
http://127.0.0.1:8000/swagger/?format=openapi

Test description

Schemes: HTTP

showroom

```
GET     /showroom/            showroom_list
POST    /showroom/            showroom_create
GET     /showroom/{id}/       showroom_read
PUT     /showroom/{id}/       showroom_update
PATCH   /showroom/{id}/       showroom_partial_update
DELETE  /showroom/{id}/       showroom_delete
```

Figure 49: Swagger endpoint in drf-yasg.

3.1.1.6 HAPI swagger

OpenAPI generator plugin for HAPI to self-document the API interface for JavaScript objects.

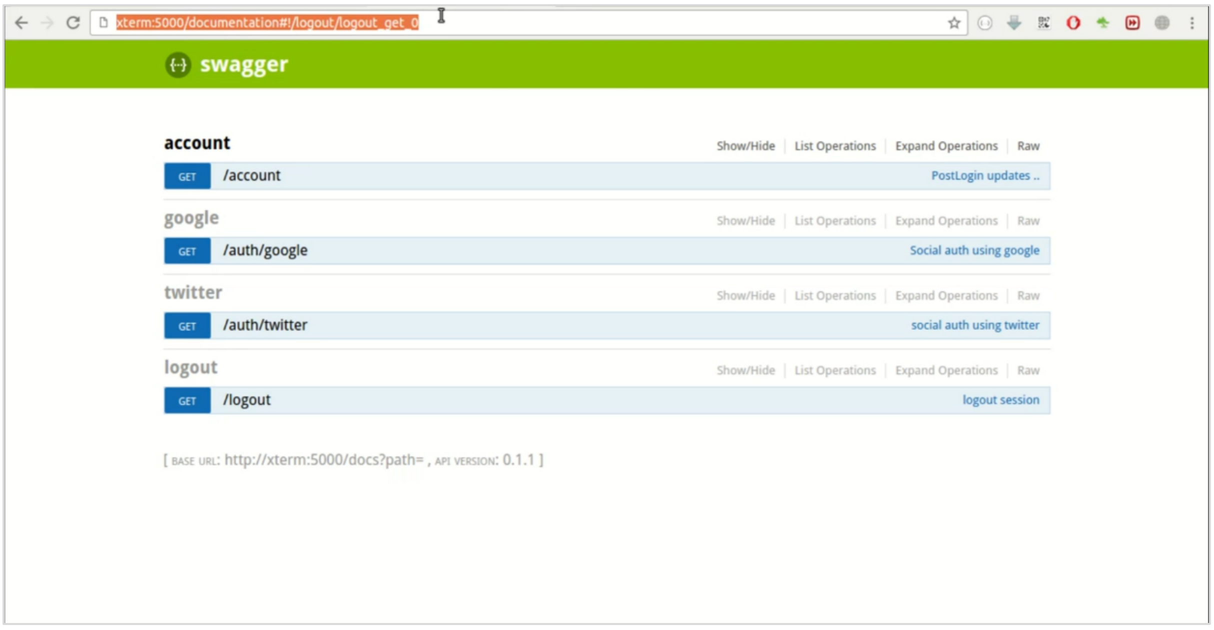
nodeJS or npm must be installed to install this tool. Once installed, only need to run the following command to install hapi-swagger:

npm install hapi-swagger --save

npx install-peerdeps hapi-swagger

After installation, modules must be imported into the JavaScript code and endpoints tagged by ***tags: ['api'],*** in the *options* section of the code.

After running the application again, API documentation is generated in /documentation/ route.



xterm:5000/documentation#!/logout/logout_get_0

swagger

account — Show/Hide | List Operations | Expand Operations | Raw
GET /account — PostLogin updates ..

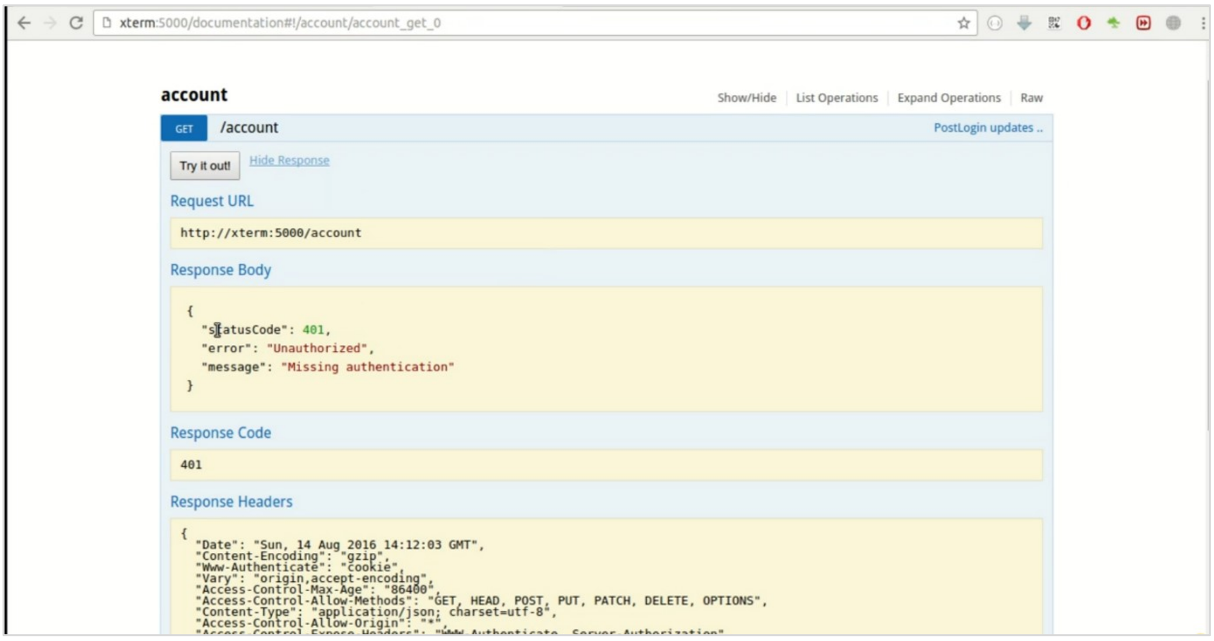
google — Show/Hide | List Operations | Expand Operations | Raw
GET /auth/google — Social auth using google

twitter — Show/Hide | List Operations | Expand Operations | Raw
GET /auth/twitter — social auth using twitter

logout — Show/Hide | List Operations | Expand Operations | Raw
GET /logout — logout session

[BASE URL: http://xterm:5000/docs?path= , API VERSION: 0.1.1]

Figure 50: Swagger endpoint in hapi-swagger.



xterm:5000/documentation#!/account/account_get_0

account — Show/Hide | List Operations | Expand Operations | Raw
GET /account — PostLogin updates ..

Try it out! Hide Response

Request URL

```
http://xterm:5000/account
```

Response Body

```
{
  "statusCode": 401,
  "error": "Unauthorized",
  "message": "Missing authentication"
}
```

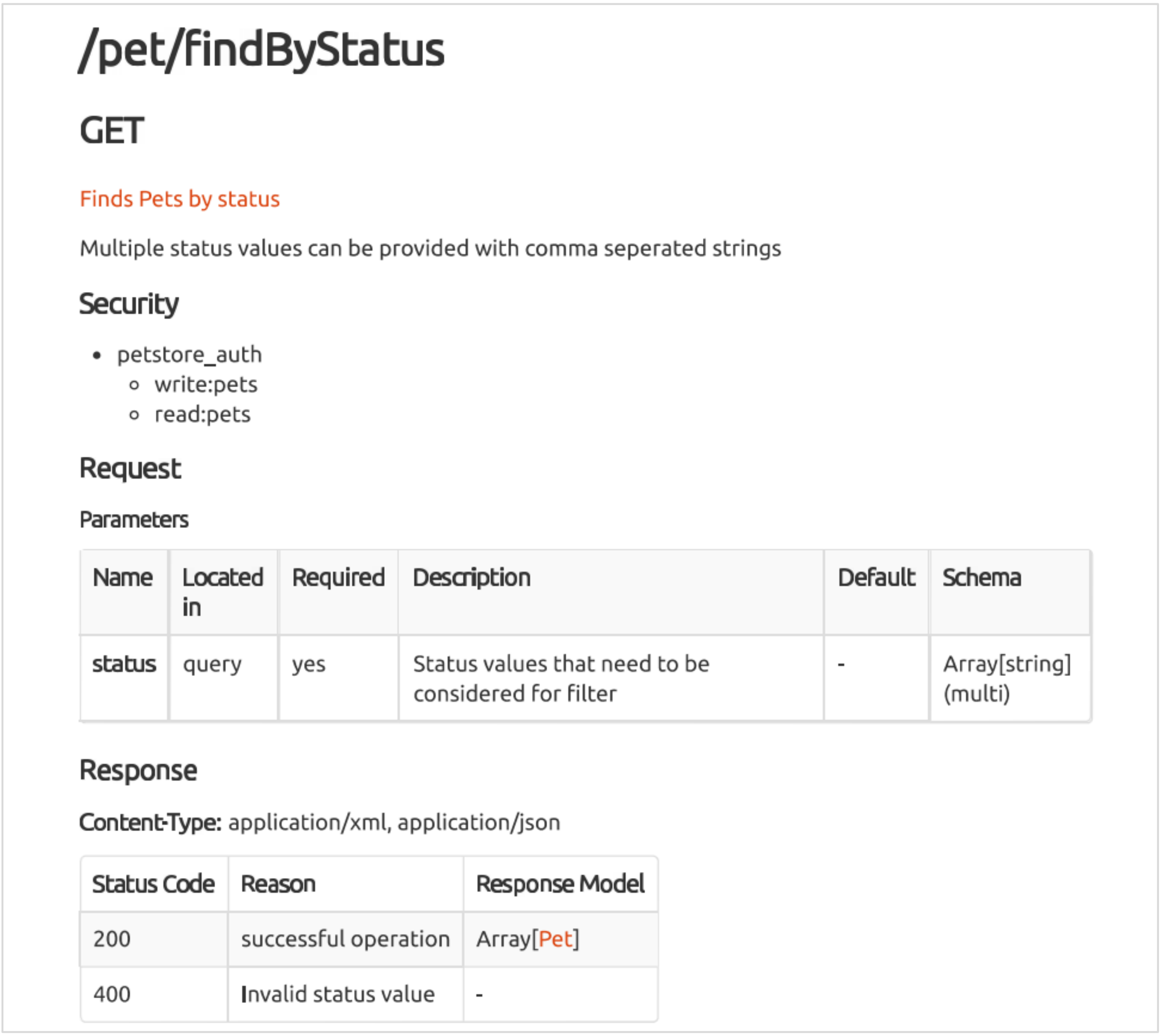
Response Code

```
401
```

Response Headers

```
{
  "Date": "Sun, 14 Aug 2016 14:12:03 GMT",
  "Content-Encoding": "gzip",
  "Www-Authenticate": "cookie",
  "Vary": "origin,accept-encoding",
  "Access-Control-Max-Age": "86400",
  "Access-Control-Allow-Methods": "GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS",
  "Content-Type": "application/json; charset=utf-8",
  "Access-Control-Allow-Origin": "*",
  "Access-Control-Expose-Headers": "WWW-Authenticate, Server-Authorization"
```

Figure 51: Swagger endpoint details for the ‘account’ object in hapi-swagger.



/pet/findByStatus

GET

Finds Pets by status

Multiple status values can be provided with comma seperated strings

Security

- petstore_auth
  - write:pets
  - read:pets

Request

Parameters

Name	Located in	Required	Description	Default	Schema
status	query	yes	Status values that need to be considered for filter	-	Array[string] (multi)

Response

Content-Type: application/xml, application/json

Status Code	Reason	Response Model
200	successful operation	Array[Pet]
400	Invalid status value	-

Figure 52: Swagger maven plugin API call example.

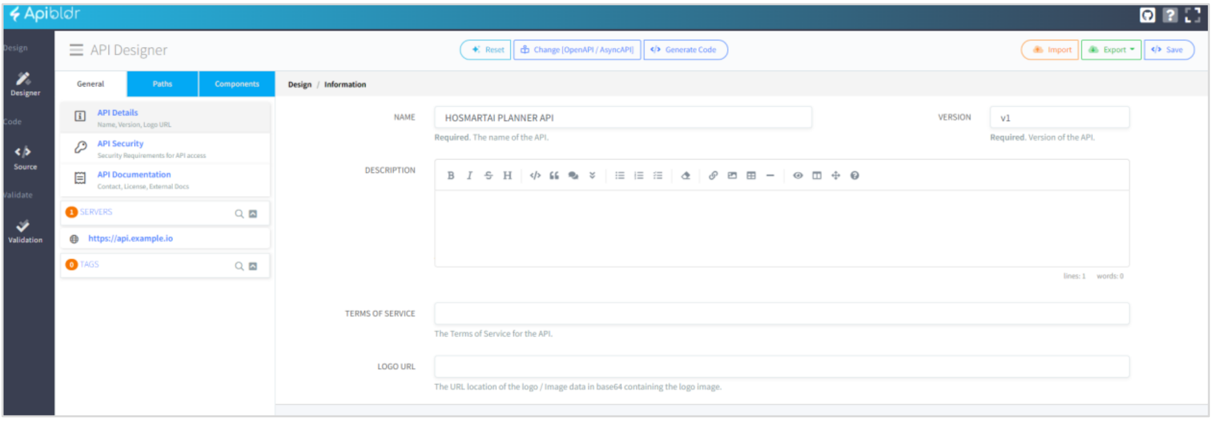
3.1.2 GUI Editors

Certain generator tools only generate a basic definition of its OpenAPI, so it's needed to add more information to clarify all the methods, parameters, schemas and all the other elements that an OpenAPI needs.

3.1.2.1 ApiBldr

ApiBldr is an application for API design and modelling. A primary goal of ApiBldr is to enable the API-First design approach for both developers and non-developers and to save development time.

With this tool, any OpenAPI specification can be edited, reviewed, and validated.



NAME: HOSMARTAI PLANNER API
Required. The name of the API.

VERSION: v1
Required. Version of the API.

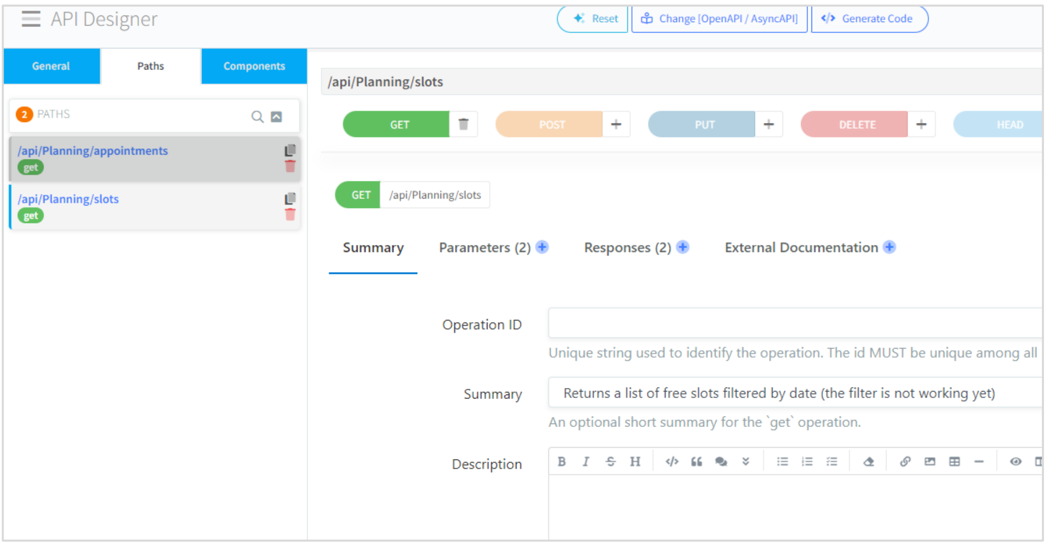
DESCRIPTION

TERMS OF SERVICE
The Terms of Service for the API.

LOGO URL
The URL location of the logo / image data in base64 containing the logo image.

Figure 53: Main screen of Apibldr.

Form-based design means you don't need to be an OpenAPI expert to operate with this tool. ApiBldr has a write mode with full OpenAPI autocomplete, and a read mode for visualizing HTTP operations and models.



Operation ID
Unique string used to identify the operation. The id MUST be unique among all

Summary: Returns a list of free slots filtered by date (the filter is not working yet)
An optional short summary for the `get` operation.

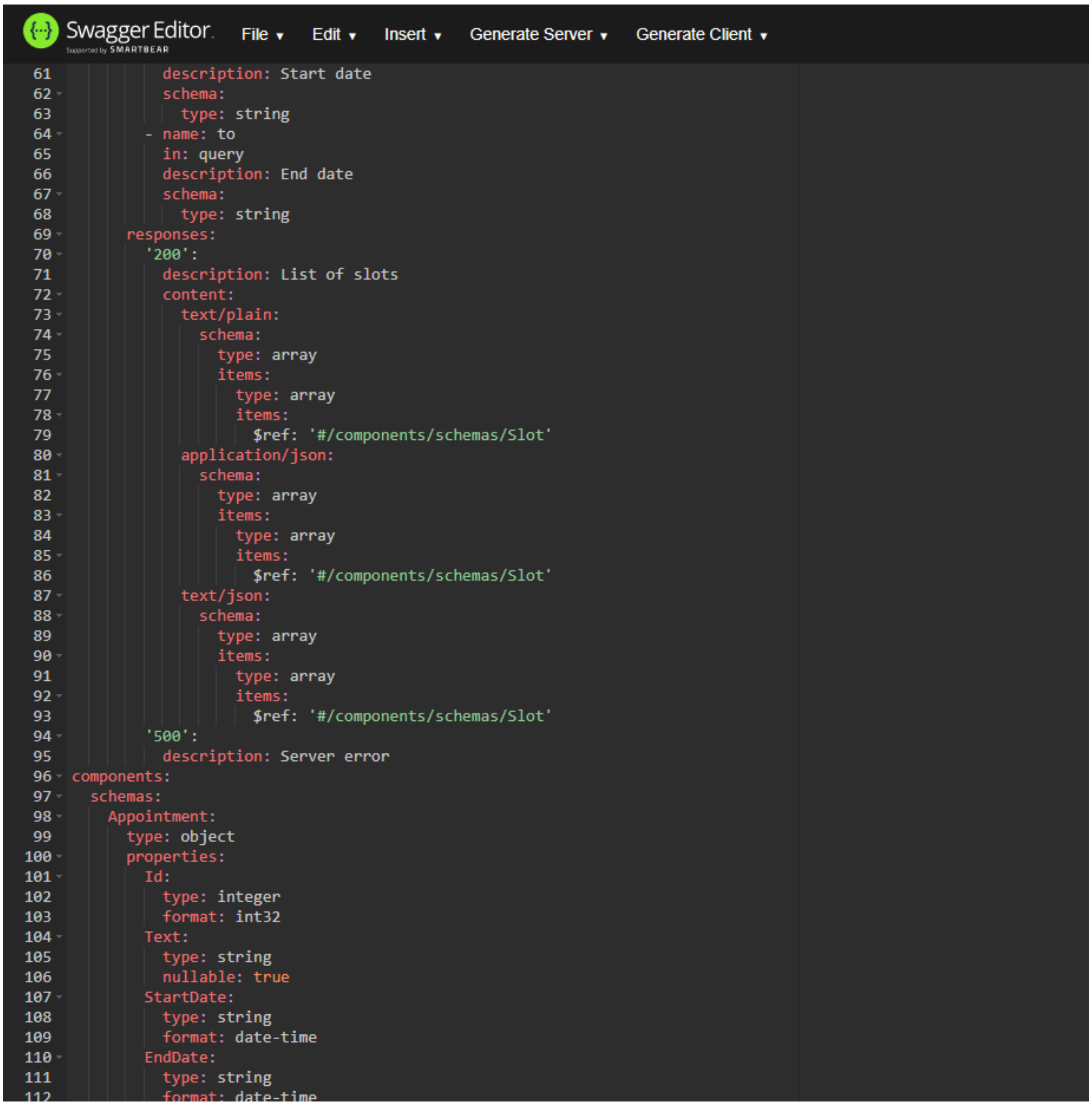
Description

Figure 54: Method description.

3.1.2.2 Swagger Editor

With Swagger editor you can perform a visual check of the elements that make up the API. At the same time, it allows you to identify code errors, indicating the line where the error is located. This editor requires specific knowledge of the OpenAPI standard.

This tool also has a wide range of possibilities in the field of testing. It allows the server part to be generated to make the calls defined in the API. At the same time, we can generate a client from a wide variety of languages to include it in the tool being developed.



```yaml
          description: Start date
          schema:
            type: string
        - name: to
          in: query
          description: End date
          schema:
            type: string
      responses:
        '200':
          description: List of slots
          content:
            text/plain:
              schema:
                type: array
                items:
                  type: array
                  items:
                    $ref: '#/components/schemas/Slot'
            application/json:
              schema:
                type: array
                items:
                  type: array
                  items:
                    $ref: '#/components/schemas/Slot'
            text/json:
              schema:
                type: array
                items:
                  type: array
                  items:
                    $ref: '#/components/schemas/Slot'
        '500':
          description: Server error
components:
  schemas:
    Appointment:
      type: object
      properties:
        Id:
          type: integer
          format: int32
        Text:
          type: string
          nullable: true
        StartDate:
          type: string
          format: date-time
        EndDate:
          type: string
          format: date-time
```

Figure 55: OpenAPI source code.

4 Updated HosmartAI Architecture Design

To understand the new version of the architecture, we present below the resulting version of D4.1.

4.1 Previous version

We present the old version of the diagram, to give visibility to the changes made during this period.

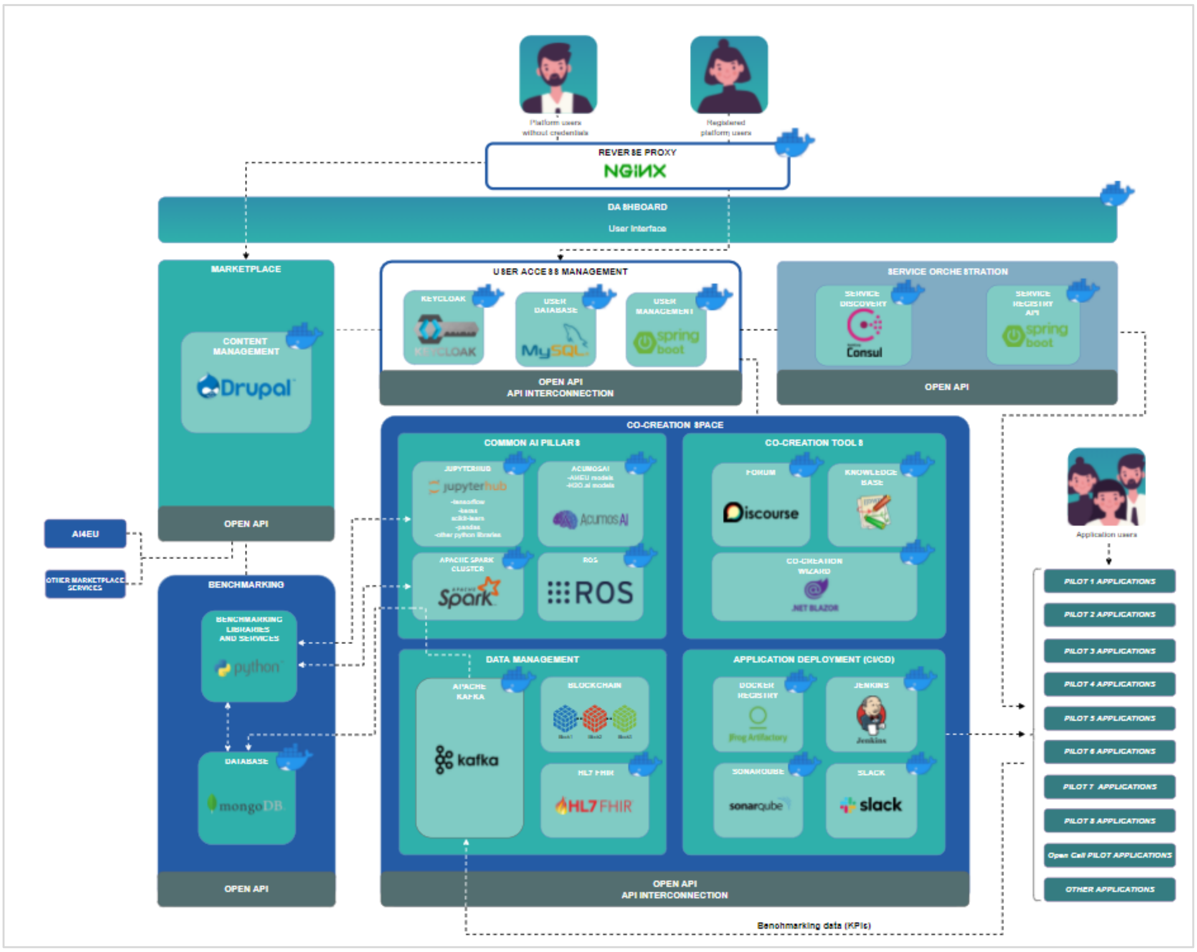


Figure 56: Old architecture diagram.

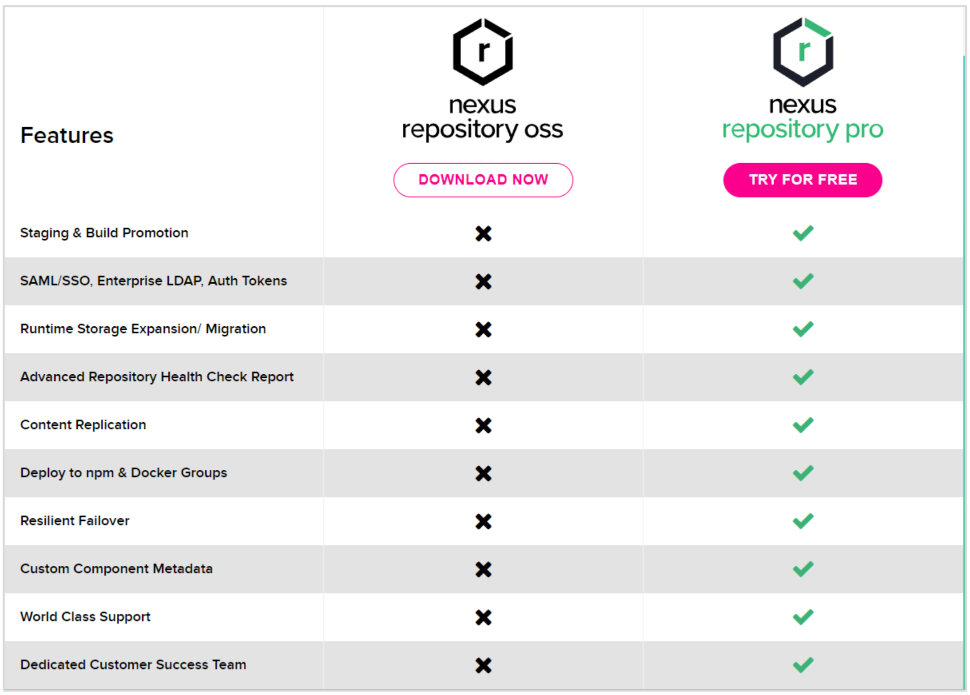
4.2 New elements

This section describes all the components that have been added to the platform. As in the previous deliverable, it details the requirements and general functioning of each component.

4.2.1 Nexus repository

Nexus is a software that offers artifact repository functionality. It allows an organization to privately share artifacts between different projects. It supports different types of repositories depending on the type of artifacts, jar libraries for Java, npm packages for JavaScript, container images for Docker, Python and Go packages.

There are two versions of the Nexus repository, the open-source version, and a commercial version that offers support and a few extra features.



Features	nexus repository oss DOWNLOAD NOW	nexus repository pro TRY FOR FREE
Staging & Build Promotion	✗	✓
SAML/SSO, Enterprise LDAP, Auth Tokens	✗	✓
Runtime Storage Expansion/ Migration	✗	✓
Advanced Repository Health Check Report	✗	✓
Content Replication	✗	✓
Deploy to npm & Docker Groups	✗	✓
Resilient Failover	✗	✓
Custom Component Metadata	✗	✓
World Class Support	✗	✓
Dedicated Customer Success Team	✗	✓

Figure 57: Nexus repository versions.

Hardware elements
1. 64-bit processor 1. > = 4 cores 2. 8GB RAM 3. BIOS hardware virtualization support
Software elements
1. Linux OS 2. MacOS 3. Windows

4.2.2 Security VM

To security monitoring of all the selected components inside the HosmartAI platform, a central virtual machine has been deployed.

This VM exposes (to the other virtual machines of the platform) a Logstash endpoint where all application logs of the selected components are sent, parsed, transformed into a structured form and finally stored in Elasticsearch, a NoSQL database suitable for ingesting and searching large amounts of text data.

Platform components to be monitored have been selected based on their criticality and exposure to the internet, and are the following:

1. Keycloak: all authentication/authorization is centralized here, so it is crucial to monitor successful and failed authentication attempts, IP and the country origin of the attempt, access token errors and other Keycloak-specific event types
2. NGINX: this represents the main "gateway" from the internet to the various platform components. By logging every request passing through NGINX, it is possible to observe attack patterns, determine which platform component is being targeted, and act upon this knowledge (implement firewall rules, harden applications running behind NGINX, etc...)
3. Kafka: Kafka will be another important "gateway" for data exchange between the HosmartAI platform and Pilot applications, hence the importance to monitor accesses and statistics of data passing through it
4. Jenkins: as a CI/CD platform, some of the platform components will be automatically deployed using Jenkins. Over the last few years, supply-chain attacks targeting CI/CD platforms have proven to be successful and difficult to detect without proper monitoring. Inside HosmartAI, user authentication, configuration changes and pipeline actions are all monitored.

Some other components are new or not yet in their final deployment state, but will be monitored too once possible:

1. JupyterHub: enables code execution and collaboration by researchers and developers. Code execution is a powerful capability for attackers too, so proper monitoring and traceability are critical to implement
2. Marketplace: as a web application that will be exposed to the internet, it will inevitably be targeted by automated vulnerability scanners and fuzzers

A set of dashboards, along with read-only access for project partners who requested it, have been implemented in Kibana, the graphical user interface for Elasticsearch. Keycloak, NGINX, Jenkins and Kafka are all regularly monitored for unusual activity.

Future improvements include automated alerting using custom rules implemented for HosmartAI.

4.3 Updated elements

In this section, we will present the elements that are maintained from one iteration to the next and have been updated, either in operation or implementation.

4.3.1 KeyCloak implementation

Keycloak has been configured to handle the login requests from various platform components and can be configured for external components as well. For example, when someone tries to access JupyterHub using HosmartAI credentials, they are presented with a form to provide their credentials, which is generated by Keycloak. Upon successful login, the user is redirected back to the component they initially requested to access, e.g., JupyterHub.

4.4 New version

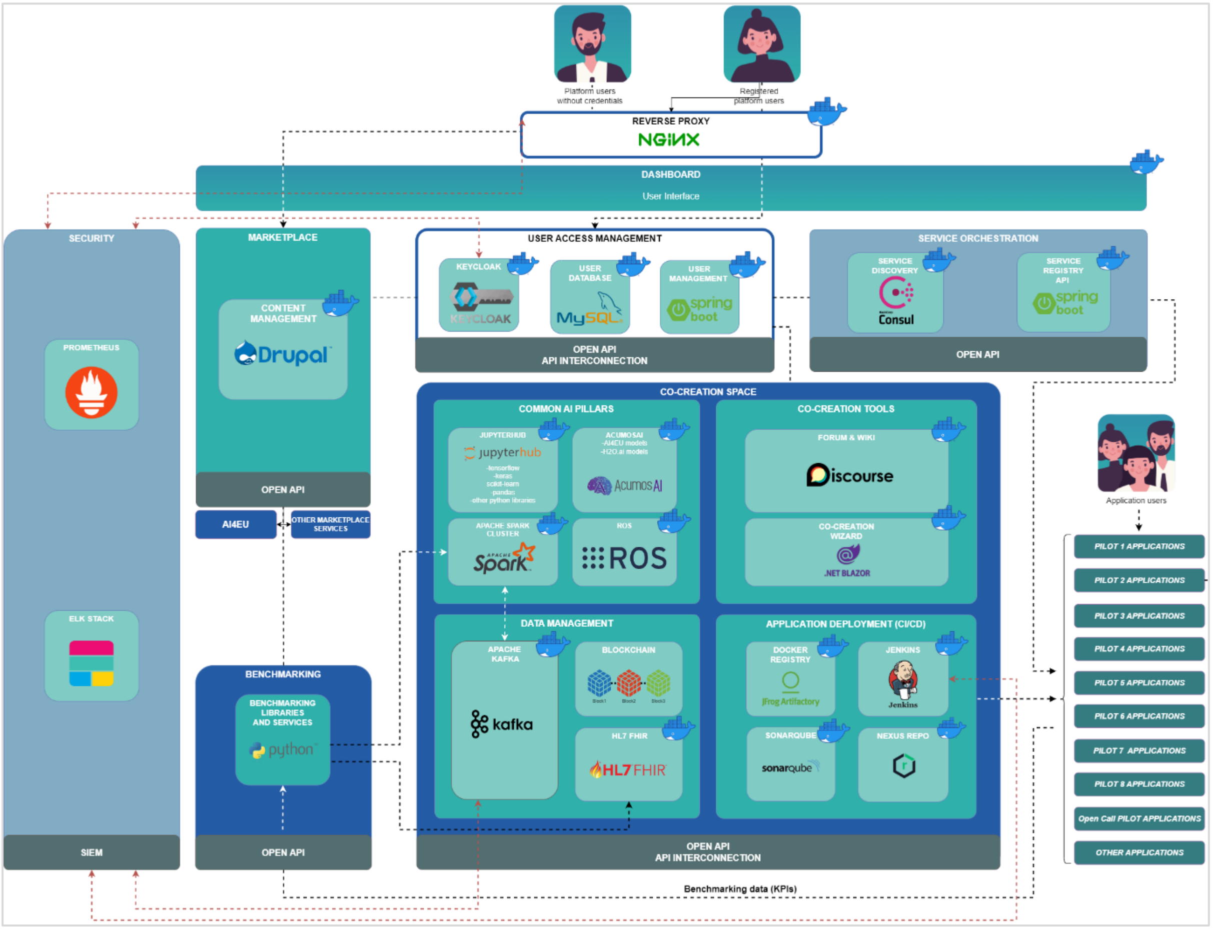


Figure 58: New architecture diagram.

5 Conclusion

As a result of time and research into different components of the platform, the first version of the architecture has undergone changes to improve the end-user experience.

To this end, some elements that made no sense within the architecture have been eliminated. At the same time, others have been updated, changing how they are integrated, or the version originally used. New elements have also been added since it has been detected that they were missing to cover certain needs of the system.

At the same time, the different platforms involved in the architecture have been analysed and explained in depth, dividing them into digital and physical platforms.

6 References

[REF-01]	https://github.com/domaindrivendev/Swashbuckle.AspNetCore
[REF-02]	https://www.apibldr.com/
[REF-03]	https://gaia-x.eu/
[REF-04]	https://www.docker.com/
[REF-05]	https://github.com/docker/compose
[REF-06]	https://maven.apache.org/maven-features.html
[REF-07]	https://camel.apache.org/camel-core/
[REF-08]	https://spring.io/projects/spring-framework
[REF-09]	https://camel.apache.org/camel-spring-boot/3.20.x/
[REF-10]	https://hapifhir.io/hapi-fhir/docs/server_jpa/architecture.html
[REF-11]	https://sentry.io/
[REF-12]	https://github.com/thomaxxl/safrs