Deliverable 3.4b

Integrated Platform

Responsible Unit: CNR
Contributors: IDEABLE, Bartenbach

Document Technical Details:

Document Number	D3.4b
Document Title	Integrated Platform
Version	2.0
Status	Final
Work Package	WP3
Deliverable Type	Report
Contractual Date of delivery	30.09.2020
Actual Date of Delivery	18.12.2020
Responsible Unit	CNR
Contributors	IDE, BART
Keywords List	Personalisation, End-user Automation Creation, Architecture, Context-dependent systems
Dissemination Level	Public

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

Contents

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

1 INTRODUCTION

This deliverable reports on the work carried out to obtain the integrated PETAL platform for personalising support for older adults with mild cognitive impairments.
The personalisation support is obtained by rules that can be specified by caregivers and older adults, and allow them to control the behaviour of various types of appliances and applications depending on dynamic events that can occur in their context of use. The rules are able to connect triggers associated to events or conditions with the desired consequent actions. In order to execute such rules, it is necessary to have platform components able to detect the relevant events in the user behaviour, the surrounding environment, and the applications, and to send the actions to modify the state of the connected objects and the applications, and notify reminders and alarms in the indicated communication channels.
This deliverable is an update of D3.4a. since it provides an updated description of the architecture of the PETAL platform, and details on how various devices and applications have been integrated in the platform in order to obtain the version that has been deployed for the second round of the project trials, and has received some modifications based on the feedback received. Thus, the communication amongst the platform components and the external applications is described as well.
The platform has been designed and implemented considering various issues and problems that emerged with the prototype that was developed and deployed for the first round of trials. For this purpose, various PETAL platform components have been modified, and in addition, we have added new types of sensors and objects for integration with the platform in order to improve its possibilities and features.

In the deliverable, we first provide an overview of the software architecture of the platform in order to indicate its main components and how they communicate with each other. Then, we detail the new possibilities offered in terms of remote monitoring of the user of the platform, which is useful to be informed about what happens in the elderly homes, and how the features of the platform are actually used. This is also useful to better understand whether there are aspects that do not work as expected and decrease the need to go in person in the older adults home, which is problematic in particular in periods afflicted by the Coronavirus crisis. The next section describes how we have solved the problems with the GREAT luminaire, which were detected in the first round of trials, and we have successfully integrated with the platform for the second round of trials. This is

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

then followed by an updated description of the Kwido game and the tablet applications and their integration in the platform. Then, we detail the integration of new devices with respect to those considered in the first trial. In particular, we focus on the integration of the Alexa device and the results obtained in this perspective. Lastly, we draw some conclusions and indications for possible future evolutions.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

2 Architectural Overview

The architecture of the considered software platform (see Figure 1) includes a Tailoring Environment through which even people without programming background (e.g. domain experts, end users) can specify the desired personalization rules. The Tailoring Environment sends such rules to a Rule Manager, which receives information from the Context Manager when the triggers involved in the rules are verified: when this happens, the Rule Manager sends the actions to execute to relevant applications and appliances. The Context Manager is a software composed of one server and various delegates. The purpose of the Context Delegates is to communicate directly with the various sensors or other entities able to generate events, to be informed when the state of their associated variables changes. When this happens, the Context Delegates communicate such changes to the Context Server, according to a pre-defined vocabulary used to specify the triggers. There is also a Monitoring module which aims to provide support for analysing the use of the personalization platform, by showing relevant information to users interested in it (in Ambient Assisted Living scenarios they can be caregivers or platform managers). For gathering the relevant information, the Monitoring module receives data from the Rule Manager concerning the rules, their state and when they have been executed. This module is a key added value because it allows relevant stakeholders to focus on the personalization that has actually been put in place by users, thereby of actual interest for them. Applications can be integrated with the platform in order to: receive actions (which were included in relevant rules) indicating requests of modifications to make, and/or send events generated by the application itself (in this case the application acts as a Context Delegate, by sending the information associated with the occurred event(s) to the Context Server).

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

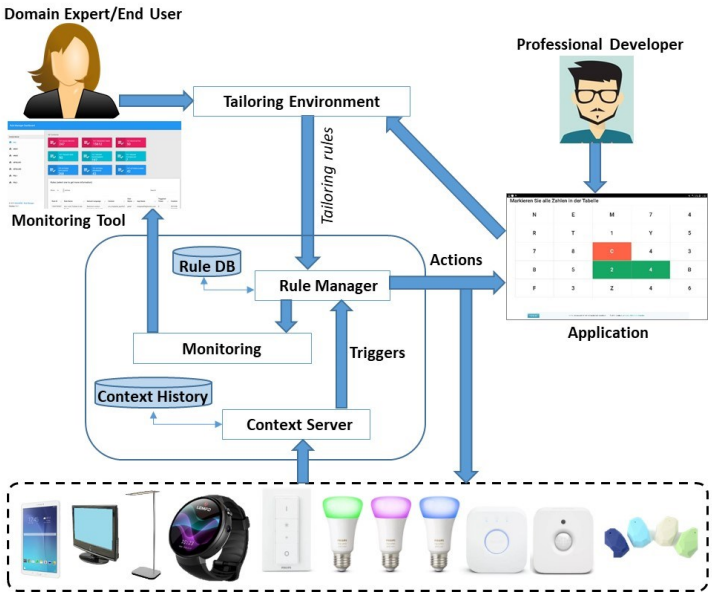

Figure 1: An Overview of the Main Components of the Platform

Figure 2 shows with more detail how the architecture of the Petal platform has been deployed in the houses of the users participating in the second round of the trials. The figure also shows the communications between the modules belonging to the platform.

Almost every sensor and appliance is provided with its own bridge/controller connected to the local network; the OpenHab service installed in the Geniatech gateway is able to communicate with each bridge/controller in order to retrieve the current status of the sensors and communicate it to the Petal Platform (Context Delegate Add-on) or to send the commands related to the actions that should be executed when a rule is triggered (Rule Manager Add-on). After Ideable's research on the most accurate devices for the project, we decided to use a Geniatech gateway for the trials in the PETAL project for the connection with the local IoT sensors. The selected gateway is GTW410 which is a high-end multi-function Smart Home Gateway, also used as industrial IoT gateway. It is reliable all day and capable of high real-time performance. OpenHab, an open-source automation software for the home is installed in the gateway. It integrates different home automation systems and technologies into one component, and its pluggable architecture supports more than 200 different technologies and systems and thousands of devices. Such a solution proved to be much more reliable than solutions based on Arduino or Raspberry Pi during the laboratory

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

tests. For the final pilots we decided to use a solution ready for production and reliable 24x7.
For a better support, we included a VPN connection to update and monitor the gateway and all the software installed remotely. This way we can give remote support during the pilots, on the fly, and without disturbing the end-users.
In order to integrate OpenHab with the Personalisation Platform we had to:
• develop an add-on, which subscribes to the Personalisation Platform at the start up and waits for the actions (using the MQTT protocol). This add-on is also able to interpret the actions and to apply them by changing the state of the supported appliances (lights). In addition to the Philips hue support provided by the OpenHab binding we also implemented the support for the actions involving the GREAT Luminaire (see next section).
• develop another add-on which acts as a Context Delegate. It monitors all the sensors supported by OpenHab (Philips sensors providing motion, temperature and light level; Xiaomi sensors for Humidity, pressure, window/door open, smoke and gas detection; Minew sensors attached to the medicine boxes; Open Weather Map binding providing information about the current weather and about the weather forecast) and sends the updated values to the context server.

The Context Server, the Rule Manager and the Rule Editor are installed in the CNR servers. In the CNR facilities, eight installations of the Context Server have been deployed: one for each house participating in the second round of trials. We needed eight installations of the Context Server because this module is in charge of maintaining the current state of the sensors and appliances installed in each field trials house; for "current state" we mean both the hierarchical structure of the context and the values sensed by the installed sensors. The Rule Manager is the same for all the installations; the only configuration that field trials users had to perform is to set the URL of the Context Server in charge of managing their specific target context. During the loading phase the Rule Editor receives the context model from the Context Server; the context model represents the current structure of the field trial context, such received structure allows the tool to present the relevant triggers to the end users. The Rule Manager is a middleware module between the Context Server and the OpenHab services; it receives the information from the Context Server when a rule is triggered, and it sends the actions defined within the triggered rule to OpenHab so that such rules can be applied. For the second round of trials, we also implemented a Monitoring Functionality that will be introduced in the next section.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

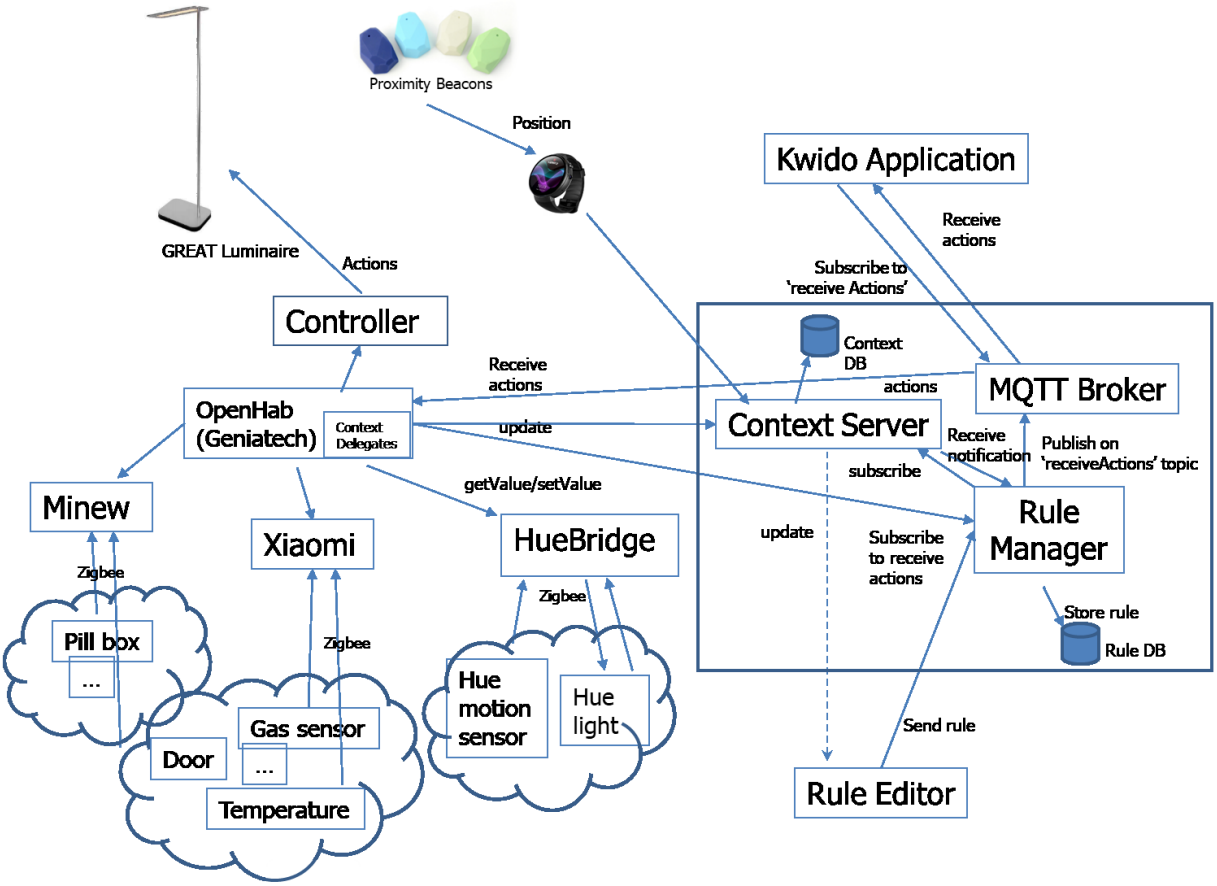

Figure 2: The Petal Platform deployed in the trials

3 Monitoring Functionalities

The main goal of the "Monitoring" module is to provide its users with information about what happens in the end-user context (e.g. seniors' homes), both in terms of activities done by the users (as detected by sensors), and in terms of personalisations that have been put in place through the specified rules. To this aim, this module receives input from the Rule Manager, specifically the personalisation rules that have been sent for execution, and the time when they have been actually triggered.

The information about triggered rules can represent valuable data to understand what is currently and actually going on in one or more end-user sites, to identify the personalisation aspects which users are focusing on most, the types of routines they have put in place and the frequency with which such automations occur. Indeed, users of the Tailoring Environment could create/add several rules

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

in their repositories over time. Thus, it can happen that some rules appear in the repository, but they are not currently exploited because the users did not want to have them active at the time. In general, the information about the triggered or active rules provides data that have been of actual interest for their users. The monitoring tool can also provide more detailed information, such as how many times a specific rule has been triggered. This value is highly dependent on the purpose of each particular rule.

In the following, we will better detail the information provided by the Monitoring Tool, which displays various types of structured information. Therefore, to facilitate its description, in Figure 3 we schematise its layout as structured into four main parts. The content of such four parts are further detailed in the following four sections (from Section 3.1 to Section 3.4).

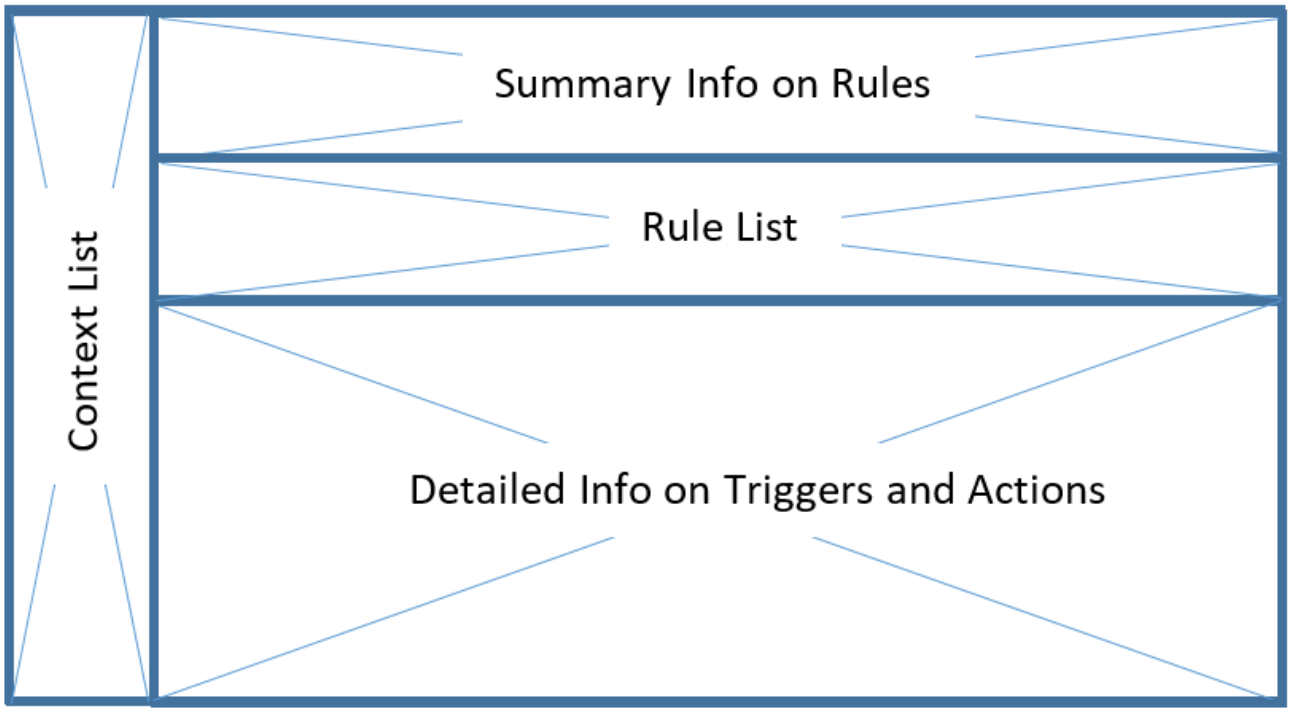

Figure 3: : The layout structure of the Rule Monitoring tool

3.1 Context List

When the Monitoring tool is accessed, it shows the list of the contexts associated with the various platform instances deployed (see the vertical left-hand panel shown in Figure), which in our project correspond to the various trials. In addition, there is a further option ("ALL", see Figure 3), which provides an overview of the

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

information associated with all such contexts. When a specific context is selected in the left-hand panel, the information about such context is shown in the central panel of the tool.

3.2 Summary Information on Rules

This part (see the main panel visualised in Figure 4) shows various pieces of information associated with all the contexts or just one context, depending on what is selected in the Context List. Starting from the top, there is a first row showing three pieces of information about rules:

- *Rules created*: the total number of rules created by the considered user(s);
- *Triggered times*: how many times the rules have been triggered;
- *Rules active*: the number of rules that are currently active and therefore, could be executed (they will be executed as soon as the involved triggers are verified).

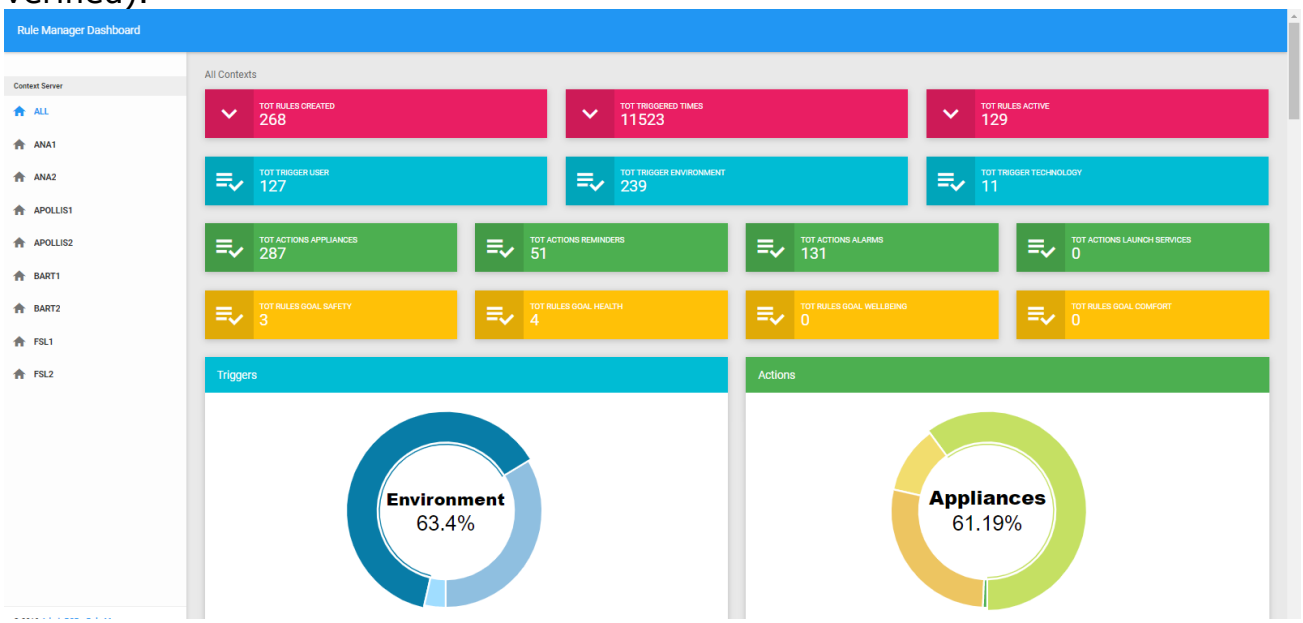

Figure 4: Visualising summary information on rules

Each of these pieces of information can be further detailed, by selecting the associated button. For instance, by selecting the one associated with "Tot Triggered Times" (top-center), it is possible to get further information about how many times the rules were triggered in the various trials (see Figure 5).

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

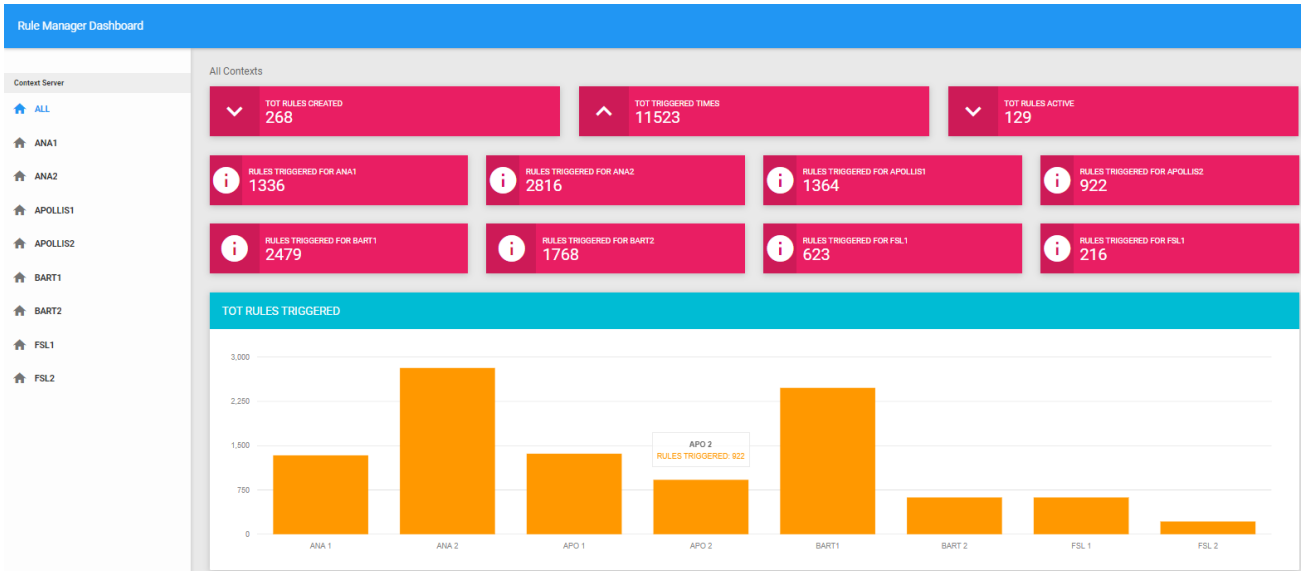

Figure 5: Visualising information about how many times rules have been triggered in the considered trials

The second row shows information about the triggers involved in the created rules, grouped according to the three main contextual dimensions (User, Environment, Technology). The last row displays information about the actions appearing in the created rules, grouped by their type (appliances, reminders and alarms). Besides such summary quantitative indicators, there are also two interactive donut charts (see the bottom part of Figure 4) visually presenting how the various types of triggers and actions are distributed. In particular, the first donut chart shows, for each contextual dimension (which can be interactively selected by clicking on the corresponding portion of the chart), the percentage of triggers of that type used in the created rules. Similar information is provided in the second chart, which is associated with actions. The percentage of each type of action used in the created rules is shown as a portion of the chart: when users interactively select one portion, further (textual and numerical) information is displayed accordingly. In the example in Figure 5, the "Environment" category was the most used contextual trigger category (counting for around 63% of occurrences as triggers), whereas the "Appliances" category was the most used action type (around 61% of occurrences as actions).

3.3 Rule List

In this part (which corresponds to the "Rule List" portion in Figure 3), a detailed description of all the rules created is shown. In particular, for each rule, the following information is provided (see Figure 6):

- Rule Name: the name of the rule, as given by its creator;

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

- Natural Language: the specification of the rule in natural language;
- *Context*: the context to which the rule refers;
- *App Name*: the application to which the rule refers;
- *User Name*: the creator/owner of the rule;
- *Triggered Times*: how many times the rule has been triggered, in total;
- *Creation Date*: when the rule has been created;
- *isActive?:* whether the rule is currently 'active' or not. "Active" means that a rule is currently included in the set of rules that the Rule Manager considers for possible execution.

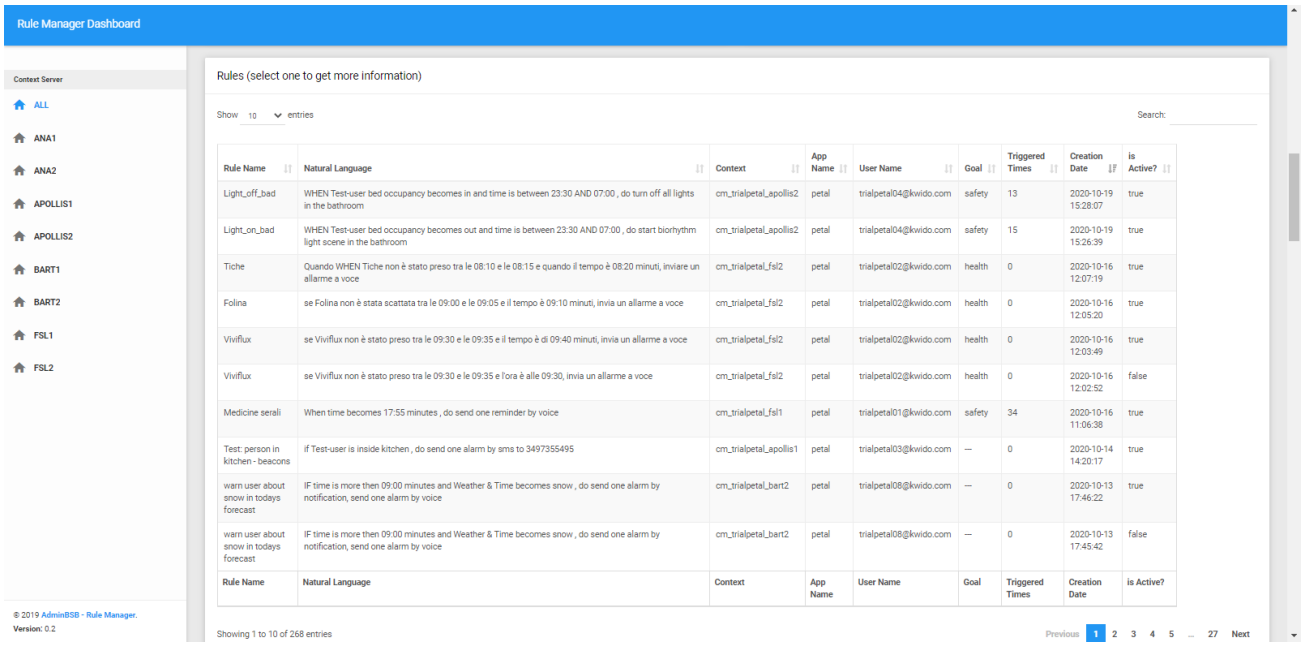

Figure 6: Showing Information about the Rules Executed

For instance, in the rule list shown in Figure 6, it is possible to see that the rule named "Light_off_bad" has been triggered 13 times in the past, and it is still active. This list of rules is also interactive: when users select a specific rule, they can obtain further information about it, in particular on the events, the conditions, and the actions involved.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

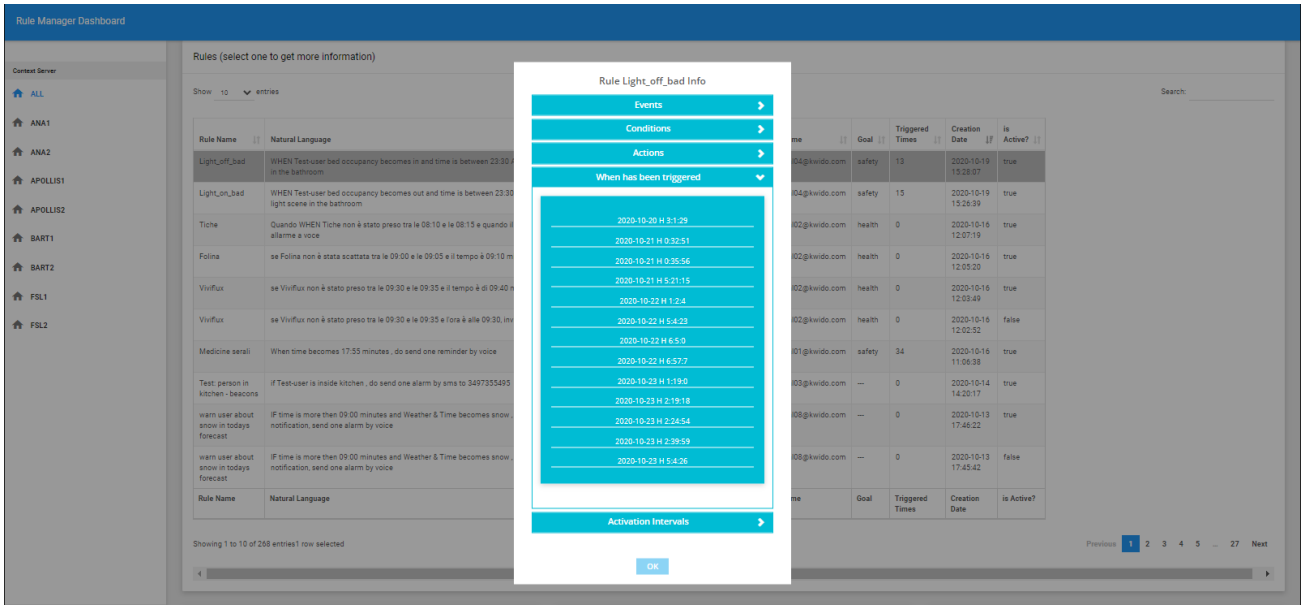

Figure 7: Showing information about the time when a rule has been triggered

In addition, when a specific rule has been triggered at least once, users can get further details about when each execution occurred (see Figure 7), by double-click on that rule. Finally, it is possible to order the visualisation of such rule list according to various included fields (e.g. according to the creation time). We also decided including additional information which shows how long a rule was active and the activation time intervals: an example is visualised in Figure 8, where it is possible to see when a specific rule has been active. However, it is also worth mentioning that, for some rules, not being executed within a specific (even long) time interval can be perfectly acceptable: for instance, this is the case of a rule that is expected to send an alarm after detecting a risky situation (e.g. when a gas leakage occurs).

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

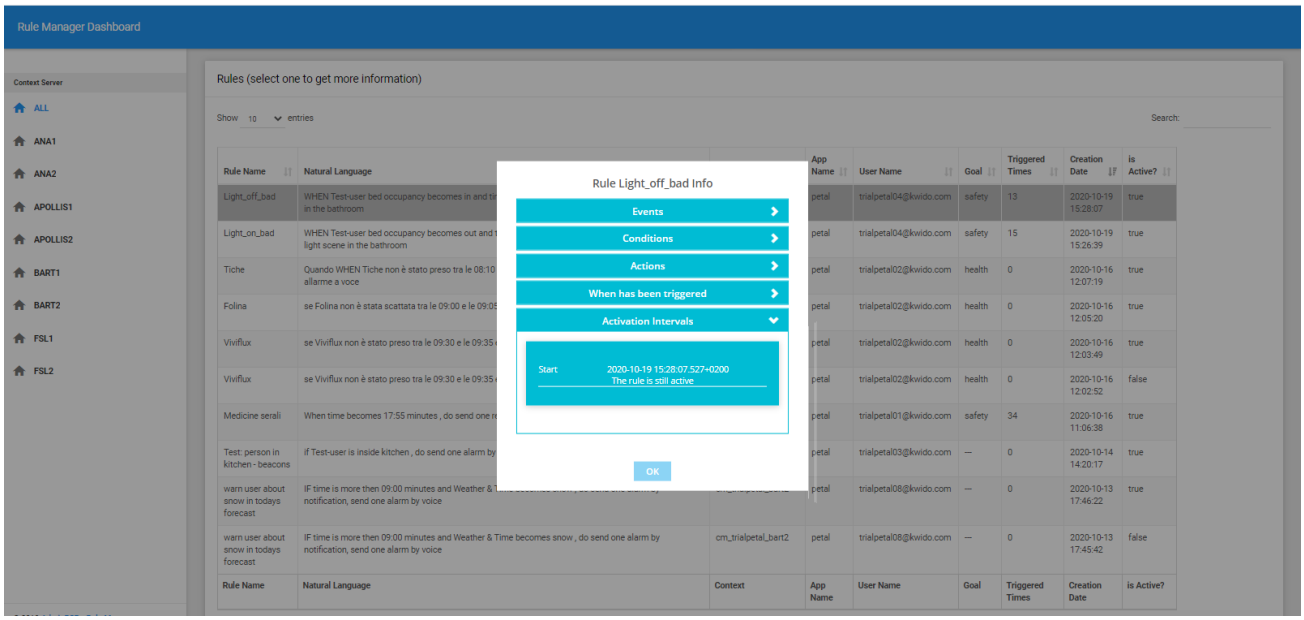

Figure 8: Showing information about when a rule has been active

3.4 Detailed Info on Triggers and Actions

Detailed information about the context dimensions involved in the created rules (in terms of event types and condition types) and the involved actions is shown in the dashboard visualised in Figure 9. This information is useful to understand which triggers and actions have been exploited by users for creating their rules. In particular, the dashboard shows three interactive donut charts visualising the frequency of use of the types of triggers and actions involved. By selecting a specific portion of the chart, the user gets more precise information (in percentage terms) about the frequency with which the associated event/condition/action type occurs in the created rules. For instance, Figure 9 (left part) shows that "Test-user" is the element of type "User" that was used, as an event, in about 40% of total cases, whereas "Weather-time" was used as a condition in around 59% of rules. For the actions, the "Voice" type was used in about 11% of cases.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

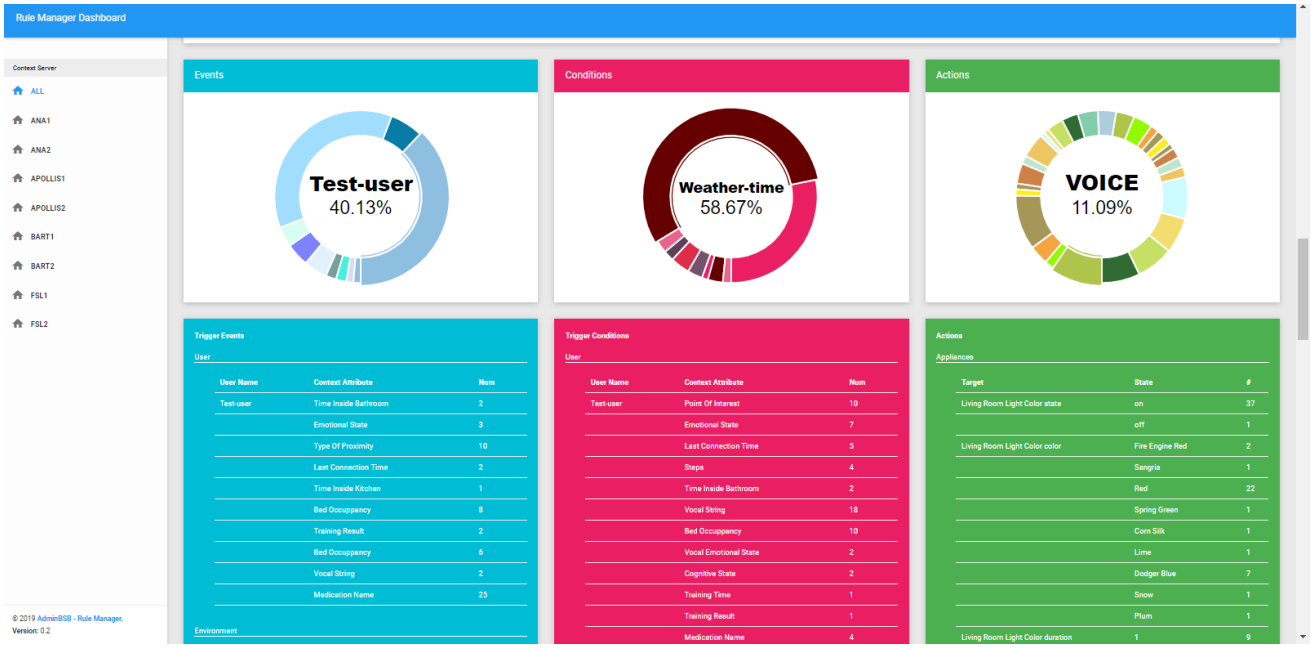

Figure 9: Showing Detailed Information about Triggers and Actions

While the charts visualise the event/condition/action occurrences only in percentage terms, more precise data about such occurrences are shown by the tool just below such charts, through some textual tables (see the bottom part of Figure 9). In such tables, the information is categorised according to the various trigger or action type, by providing a separate table for each of them. To better explain the meaning of the information contained in such tables, we will focus on the one dedicated to "Trigger Events" (i.e. the triggers of 'event' type that have been used in the created rules), visualised in the textual table shown in the left-most bottom part of Figure 7. However, it is worth pointing out that, in Figure 8, for the sake of legibility, some information provided in the tables at the bottom has been omitted. For instance, in the part dedicated to "Trigger Events" (see Figure 9, left) only the "User" category is visualised. The table dedicated to the "User" dimension presents the information according to three different fields: "User Name" (the name of the considered user), "Context Attribute" (the specific contextual trigger attribute considered, under the "User" dimension) and "Number" (the number of times the concerned trigger appears in the created rules). For instance, from that table, it is possible to see that the contextual attribute "Time Inside Bedroom" of the user named "Test-user" was used 2 times within the created rules, whereas e.g. the trigger named "Type of Proximity" was used 10 times. Along the same line, similar information is shown for the triggers of condition type, and the actions.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

5 Integration of GREAT Luminaire

In the PETAL project, the GREAT luminaire was used to support the elderly through different light scenes, which can be switched on by the end-user on demand. The GREAT luminaire, consisting of a floor lamp and an external controller, is a development from a previous AAL project, which did not implement the function of individual switchability as standard. The demand for a higher degree of usability by the end-user within the PETAL project, therefore required a system extension with additional control switches. To ensure an appropriate level of system acceptance, EnOcean-based wireless switches were used to control the GREAT luminaire. The wireless control of the luminaires is state of the art and thus fits in with the functionality of other system components in the PETAL project (see Philips Hue). The software had to be adapted accordingly for the use of the wireless pushbuttons. Since Bartenbach was primarily responsible for the development of the lighting concept in the preliminary project and the software and control-related hardware development was taken over by project partners (emt and intefox GmbH), these former project partners also took over the software adaptations in the PETAL project. In the course of the system extension by Bartenbachs partner companies, measures were implemented, as it were, to ensure functionality and above all stable communication of the GREAT luminaire in the PETAL system. While the GREAT luminaire had to fulfil the requirement of static processing of light scenes in the preliminary project, a higher dynamic is required with the extended operability by the wireless pushbuttons. In order to avoid command lists to be processed, a corresponding software adaptation was also necessary here. After detailed analysis, the following measures were implemented to ensure the functionality of the GREAT luminaire in the PETAL system:

- The software was adapted to the new requirements by interfox GmbH and emt. The adaptations were managed by Bartenbach. The adapted software was made available as an update. All luminaire controllers were checked for the software version and updated to the latest version.
- To ensure adequate communication, all hardware components of the lights were checked for proper connection. Besides, the EnOcean radio antennas were led outside to increase the range; this update significantly reduced the risk of data loss due to insufficient reception quality.
- Intensive system tests were also carried out in the laboratory to verify functionality. These tests included checking various settings at different distances. Based on the tests in the laboratory, it can be assumed that the

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

commands are transmitted in a stable manner and can be operated via the EnOcean-based wireless switches and the PETAL system.
In the course of the preparation for the second field test phase, problems occurred nevertheless at two lights, which were however corrected by Bartenbach before the start of the second field test phase.
At one Apollis luminaire the case occurred that after installation of the PETAL system the GREAT luminaire could be addressed via the wireless pushbutton but not via the PETAL rule-editor. After analysis and direct coordination between Bartenbach and Apollis, a faulty object ID of the luminaire scene was identified as the cause. This object ID is accessed via the admin user in the PETAL system. The error could be corrected via remote maintenance. For this purpose, the single object of the light scene had to be loaded separately from a working configuration via FoxConfigurator, the software for the controller of the GREAT luminaire. Finally, the functionality of the GREAT luminaire in the PETAL system was tested via the rule-editor.
At ANA ASLAN the same error behaviour occurred at one luminaire, but with different causes. The GREAT luminaire was also controllable via the wireless pushbutton, but not via the PETAL rule-editor. An error analysis via TeamViewer showed that a system update was incomplete. To solve the problem, a new image was downloaded to the controller via remote maintenance and the GREAT lamp and the wireless pushbutton were again taught and linked via the FoxConfigurator. Finally, an update of all system-relevant packages was carried out and the functionality of the GREAT-lamp was tested in the PETAL system.

6 Kwido Home

Ideable has included some changes in the platform for the second phase of the project, and especially for the final commercial product to be presented to the market.
As explained in the Business Plan deliverable, IDE is launching a commercial version based on the PETAL project, using some of the existing software already developed during these months.

6.1 PETAL Launcher

In order to facilitate the use of the platform at home in the end-users' tablets, an Android "launcher" application has been developed explicitly by Ideable that allows PETAL to offer a simplified interface to the main functionalities offered by the platform. The "launcher" type application overloads the tablet's standard user interface, in such a way that access to the applications is simplified and the reception of notifications and sound alerts can be personalised.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

The Petal Launcher Application is an Android application that can be configured as the main system launcher, taking control of the "root" screen shown to the user. The interface can be modified by the users, allowing them to adjust the size and colour of the text and background. Its main responsibilities and functionalities are:

- Notifications: hearing aids and voice control, with reminder/alarm notifications received from the Rule Manager corresponding to rules actions.
- Launch external Apps, at this moment Kwido Mementia, but it can be extended to launch other applications.

The communication -between this launcher and the rest of PETAL is done by MQTT. Using MQTT the launcher is able to retrieve the actions related to a triggered rule and transform them into notifications and sound alerts in the tablet.
The launcher also offers reminders to the end-users for playing kwido mementia, for example or for taking medicine in a precise time interval

6.2 Integration of Kwido Mementia

The Cognitive Stimulation Application, Ideable's Kwido Mementia, is integrated with PETAL platform via endpoints published in the Context Manager REST services. As Figure 10 shows, the actual communication is carried out by using a software developed to manage the synchronisation between Kwido Mementia and the Context Server. We call this software PETAL Mementia Bridge, or simply bridge. It implements a common pattern in software development. It is a translator from one part to another, in this case from Kwido Mementia to PETAL for communicating relevant events. By using this pattern, we avoid tightly coupling Kwido Mementia and PETAL. The bridge translates the data structure used in Kwido Mementia to the format understood by the PETAL platform.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

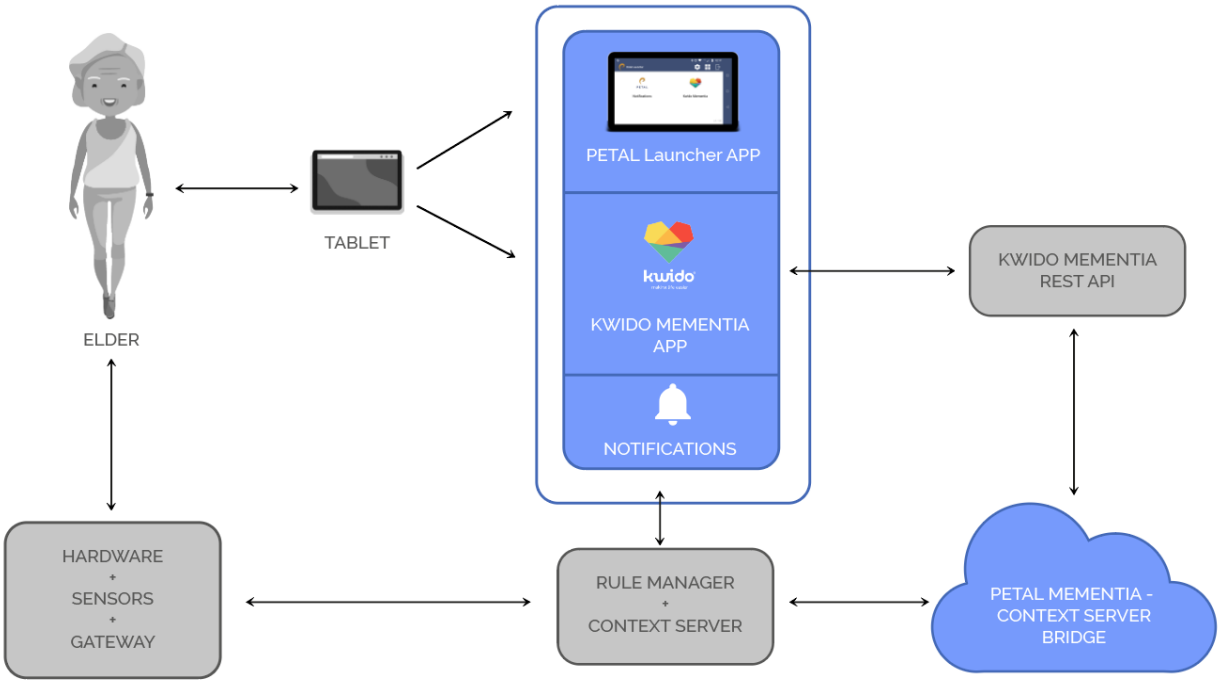

Figure 10: Comunication between Kwido and Petal Platform

During the project, new languages and games have been created manually for PETAL and apart from that, mood self assesment has also been included. In this case, new questions have been inserted during the games to infer the mood of the person and raise alerts related to low emotional status reported by the older adults.

The complete architecture created for Kwido Home is the one that follows.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

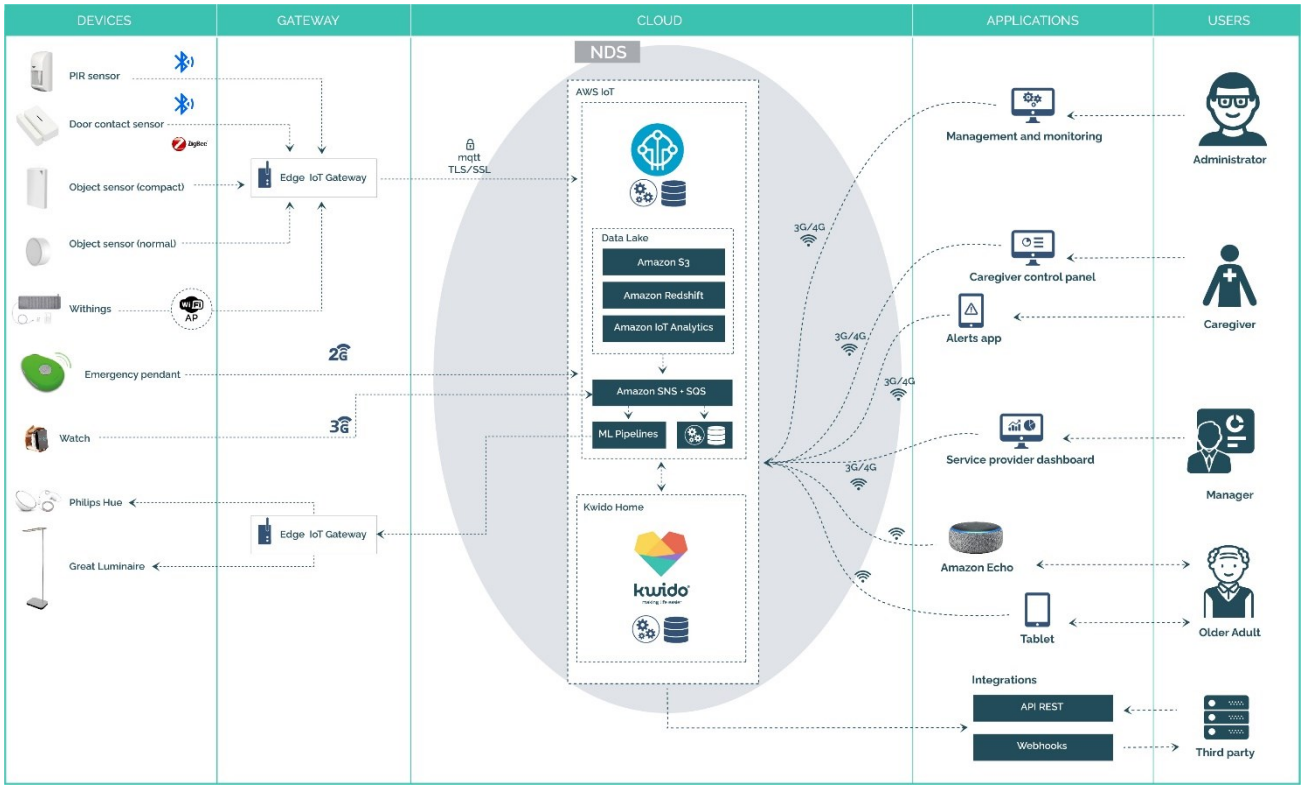

Figure 11: Deployment of Kwido in the Petal Platform

6.3 Rule engine for Kwido

One of the inputs received from the caregivers during the project, and by Ideable during the first tests with early adopters, was the necessity of defining more easily all the devices, rooms and rules involved.
For this purpose Ideable has integrated the rule engine into the Kwido engine.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

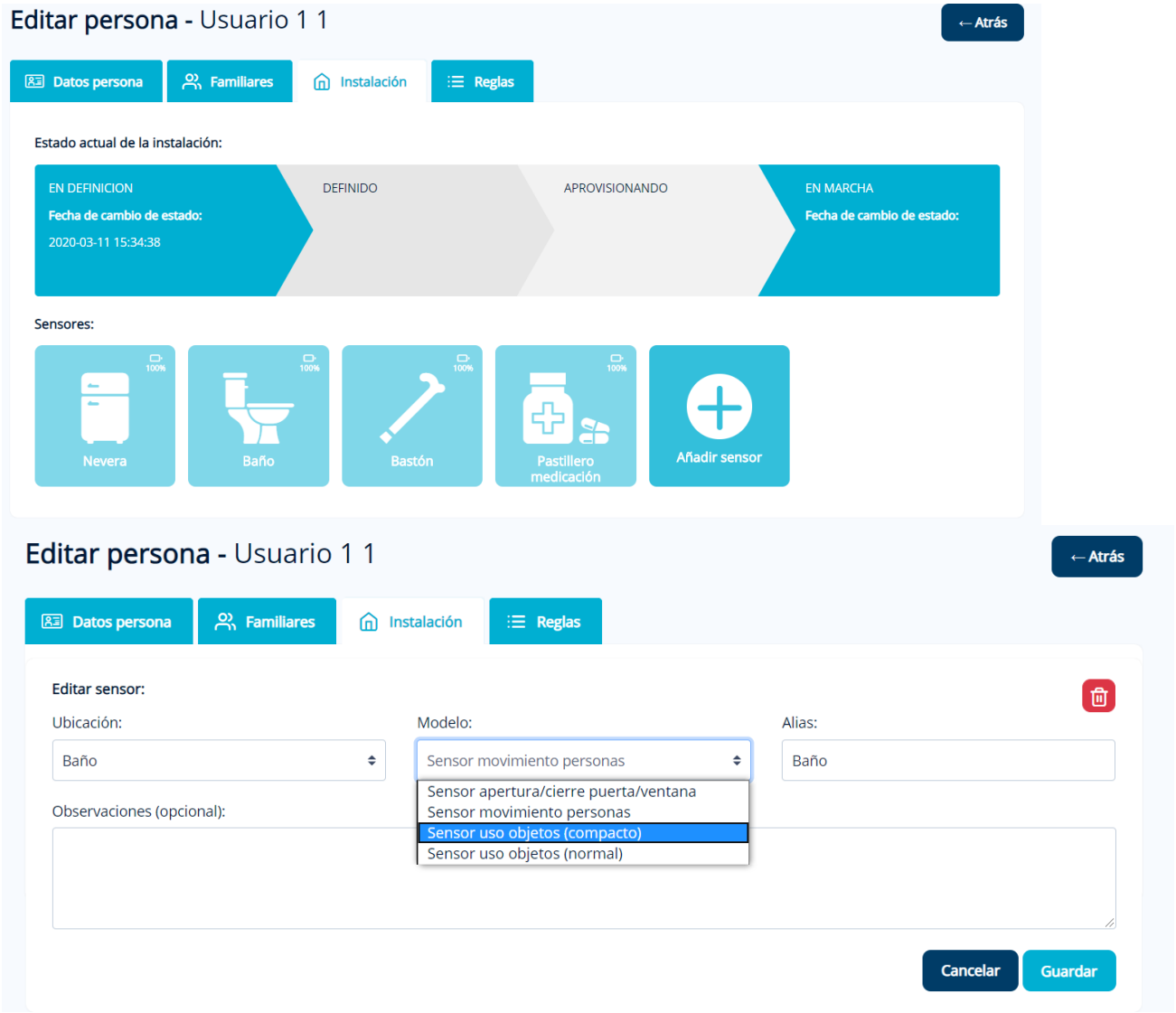

Figure 12: Device configuration

We can create devices and places where they have been located: movement sensors, presence sensors, etc. Moreover, we create the rules as combinations of conditions, for example: (No) movement between hours, (No) movement after hours, More than N times…, and basic triggers, for example send alerts, turn lights on, …
The most common rules are included in a catalogue to be easily reused among users.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

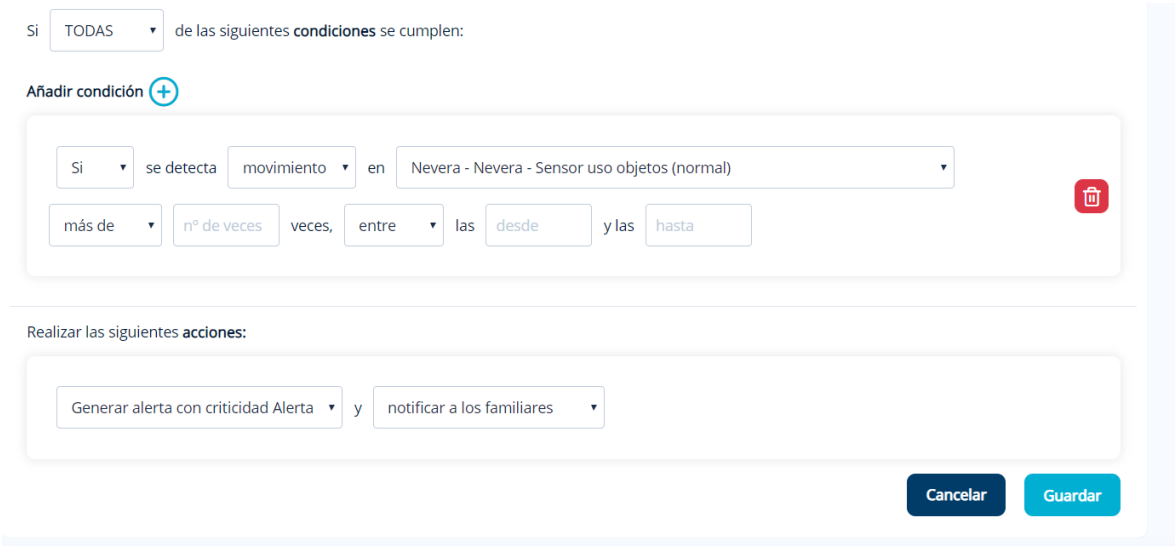

Figure 13: Examples of rule personalziation

Easy to read dashboards are provided to see if everything is progressing well and which is the activity of the older adults, see Figure 14.

Figure 14: Information on user activities

Some information is also available for access through the smartphone (see Figure 15), as many caregivers that already use the "Kwido for caregivers" app asked for this information to be integrated too.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

Figure 15: Information on user activities

7 New Devices Integrated for the Second Round of Trials

7.1 Bed sensor

After analysing Murata bed sensor to be included in the PETAL project, Murata finally decided to discontinue this device, so Ideable looked for another option and we chose Nokia Withings to offer information about bed occupancy.
The complete workflow implemented is shown in the next schema.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

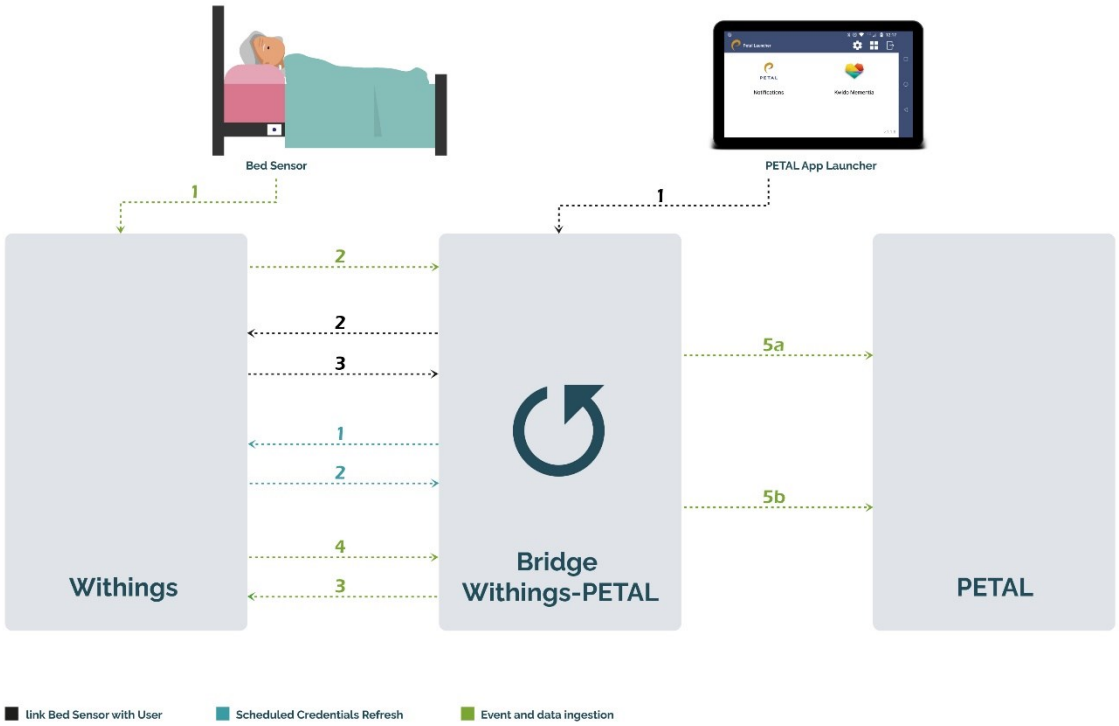

Figure 16: Integration of bed sensor in the platform.

- Link bed sensor with user
    1. Open link screen
    2. Generate link
    3. Receive callback
- Scheduled credentials refresh
    1. Every minute ask Withing new tokens
    2. Store updated credentials
- Event and data ingestion
    1. Bed sensor emits new events
    2. Withings relays this event to our bridge
    3. Our bridge asks for extended bed sensor data
    4. Recover and sync extended bed sensor
    5. Send event and data to Context Delegate

The Withings bed sensor is able to provide the Petal platform with the information related to:

- Bed Presence: the user is in bed or not;
- Sleep Duration: Time spent sleeping

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

- Wake up Count: Number of times the user woke up.
- Wake up Duration: Time spent awake
- Sleep Score: this measure is a very simple and intuitive way to understand how well the user slept

All such values have been mapped to the corresponding triggers in the context server so that they can be exploited in rules definition.

7.2 Amazon Echo

When we had to design the installation of the platform for the second round of trials, one of the new elements that were deemed interesting was the introduction of a vocal assistant for providing further support to older adults. The motivation was based on the fact that vocal assistants (with their associated speech recognition and synthesis services), can be potentially useful in providing support to older adults, and they seem a sufficiently mature technology to be introduced in household trials. Thus, it was decided to integrate Amazon Alexa in the platform. Such integration has given the possibility to extend the set of available triggers and actions. For the triggers, in the user-related contextual dimension, we added the features allowing users to vocally provide their current emotional state (i.e., happy, angry, fun, bored, serene, worried, enthusiast, sad), and to provide vocal commands (such as' turn on a relaxing scene in the living room'). Regarding the actions, we have introduced support for activating the available Alexa services (e.g., news or weather forecast skill) when a rule is triggered, and the possibility to communicate reminders or alarms through the vocal channel. This makes it possible to execute rules such as "when the user has not taken a medicine between 8 and 10 in the morning, then the Alexa speaker should remind them by saying, "remember to take your pill, it is important for your health!". Unfortunately, we found out that the Romanian language, among many others (e.g., Vietnamese, Dutch, Catalan, Russian), is not currently supported by Alexa, and neither by other voice assistants such as Google Home and it is unclear whether and when this support will be made available. Since we wanted to overcome this digital divide somehow, we started to investigate the possibility of finding solutions that, even if do not allow users to interact through such languages, they can allow rendering messages in unsupported languages and communicate notifications, reminders or alarms vocally through them.

7.2.1 The Integration of Amazon Echo Devices in the PETAL Platform

In this section, we describe how we have designed and integrated the support for Alexa in the PETAL personalisation platform. The PETAL platform uses OpenHab (OH) in its deployment, which is installed in Geniatech gateways placed

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

in the homes where the trials should be carried out. Within the personalisation platform OH acts both as an' application' (in that, it receives and applies the actions received by the Rule Manager), and as a' context delegate' since it can receive information from several associated sensors, and updates accordingly the Context Server. The integration of Alexa in the personalisation platform has a two-fold goal. On one side, Alexa vocal input can be used as a trigger, and on the other side, it can be seen as a target of an action, including playing vocal messages even for unsupported languages. Thus, we developed two different add-ons for OH as follows.
(1) *the Context Delegate Add-On* that gets the data from the connected sensors and appliances by exploiting an' OH binding'. OpenHab bindings are software packages that integrate physical hardware, external systems and web services in the OH system. Thus, the Context Delegate Add-On has a handler that bounds to various sensors and appliances managed by OH: every time they change their state it gets informed and notifies the Context Server accordingly.
(2) *the Rule Manager Add-On* subscribes to an external MQTT broker for a specific topic, in order to receive the actions coming from the Rule Manager. When the Rule Manager add-on receives an action, it can apply it in different manners, depending on the type of the involved action (this will be better detailed later on in this section).

For introducing the possibility of receiving new triggers involving Alexa (i.e., the vocal emotional state and the user vocal input) we extended the Context Server in order to receive and manage them. By exploiting such triggers, users can set up a rule, providing their emotional state as a trigger (e.g., by saying "I am nervous"). This can be done by activating an Alexa skill developed for this aim, in which Alexa asks the user about his feeling and the user will provide the corresponding answer. Subsequently, a specific action can be activated, e.g., either inviting the user to start an activity (e.g., "let's go outside for a walk!"), or automatically changing the state of smart object installed in the house (e.g., "turn on a light with a specific activating colour"). Moreover, users can also indicate and execute rules that change the home appliances' status (even those not recognisable by Alexa) such as lights, colour and scenes through vocal inputs. In this case, the vocal inputs are gathered through a skill able to recognise the inputs and send them to the Context Server, which will notify the occurrence of the event and thus trigger the corresponding rule.
The integration of the platform with Alexa is described in Figure 8. In particular, as soon as events are generated by the various triggers contained in the user's home (1), they are sent to the Context Server (2), which stores them and updates accordingly its repository. When a trigger contained in a rule is verified,

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

the Context Server notifies the Rule Manager (3), which then sends the actions to the OH Rule Manager add-on able to interpret and apply them (5).

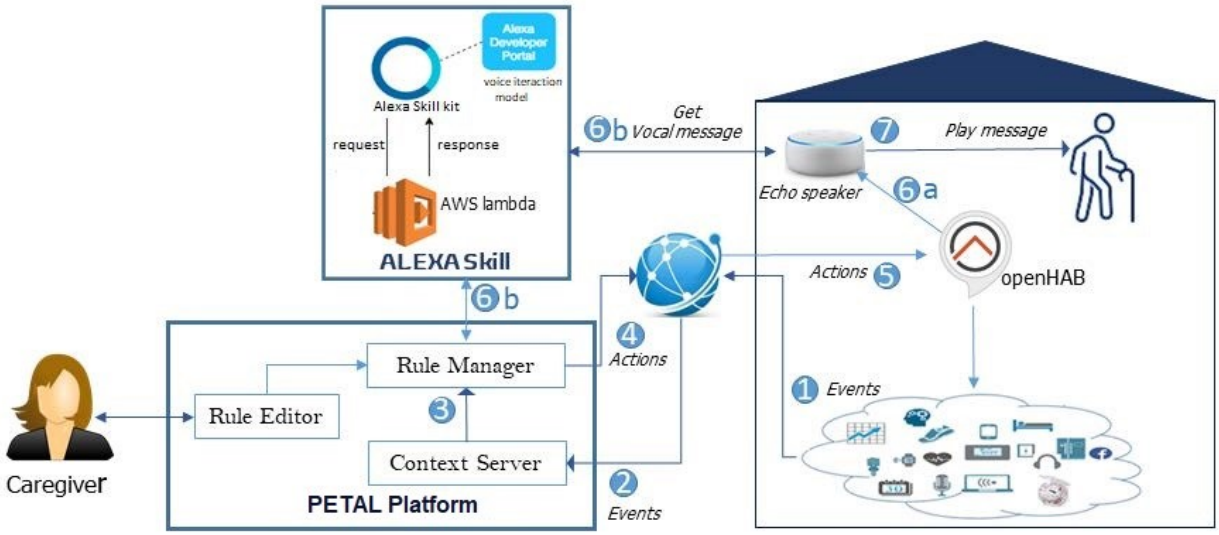

Figure 17: The personalisation platform integrated with Alexa

If the action involves an appliance (e.g., a light), the Rule Manager Add-On can exploit the OH binding mentioned before to interact with such light. If the action specifies a reminder or an alarm that should be vocally rendered, it acts differently, depending on the language of the message: if it is supported by the considered vocal assistant (Alexa), it exploits the OH binding for Amazon Echo devices[1]. This binding is able to exploit the echo device as a TextTo-Speech service, by sending a request containing the text of the alarm (or the reminder) defined in the received action, so that the smart speaker can vocally render it (6a and 7). However, this approach works only for messages defined in languages currently supported by Alexa (e.g., Italian, German, English). If the message is written in a language not yet supported, it is not possible to exploit the Amazon Echo Binding, because the message will be synthesised in a wrong manner.
To overcome such limitation, we propose a solution that first gets the user-defined text (used for specifying the concerned personal notification, alarm, or reminder) from the personalisation rule, then it generates the corresponding audio file and store it. This file will then be played through the vocal assistant, by exploiting SSML language (Speech Synthesis Markup Language) tags[2]. In this solution, we detect the language used in the alarm/reminder action by

[1] https://www.openhab.org/addons/bindings/amazonechocontrol/
[2] https://developer.amazon.com/it-IT/docs/alexa/custom-skills/speech-synthesis-markup-language-ssml-reference.html#audio

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

exploiting the Google Cloud Translation API[3]. If the language is not supported by Alexa, the Rule Manager converts the text of the message in an audio MP3 format, through an off-the-shelf text-tospeech synthesiser (provided by Google), and the resulting audio file will be stored in a server and will be made available through a REST service to be played by Alexa. Thus, when a rule concerning an unsupported language notification action will be triggered, it will be sent to the OH module. The OH module can start an Alexa automation routine (6a) whose purpose is to trigger the Alexa skill to play the audio file located in the concerned server (6b). More detail on the procedure of activation the Alexa skill through the OH is presented in the next sections.

7.2.2 Generation of Vocal Messages in Languages not Supported by Alexa.

The generation of vocal messages in languages not supported by Alexa has been obtained by exploiting Text-To-Speech (TTS) technology. In this regard, several solutions are available. Among all solutions, we used Google Cloud TTS[4], which provides a service that takes a text as input and outputs an MP3 file representing the synthesis of the provided text. Google Cloud TTS also offers an enterprise API for commercial scopes. However, for research purposes, we exploited the free online API provided by Google Translate Service. The service providing the audio file is composed of seven parameters: the character encoding (ie), the text to be synthesised (q), the target language (tl), the number of the messages that should be generated ($total$), the initial character index (idx), the length of the text ($textlen$), a randomly generated number (tk), and the service used to call the TTS functionality ("$client$"). The tk parameter changes for every query, and it must be instantiated through a matching $hash$: $tk = hash(q,TKK)$, where q is the text to be synthesised and TKK is a variable in the global scope when the API will be loaded. Below, is the function to generate the google TTS link using the $calcHash$ function.

```
function TTSLink(q, tl, tkk) {
        var tk = calcHash(q, tkk);
        return
                `https://translate.google.com/translate_tts?
                        ie=UTF-8&total=1&idx=0&client=t&ttsspeed=1&tl
                        =${tl}&tk=${tk}&q=${q}&textlen=${q.length}`;
}
```

[3] https://cloud.google.com/translate/docs/basic/detecting-language
[4] https://cloud.google.com/text-to-speech

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

The function call $TTSLink$ ($'salut!'$,$'ro'$,'410353.133636') is an example of generating the audio file for rendering in Romanian the' salut!' text. Finally, in order to use the free TTS API provided by Google Translate Service, the following steps have been carried out:

(1) Get the TTK private key from the Google Translate website;
(2) Calculate the value of the tk parameter by applying the hash function which takes in input the text and the TTK parameters;
(3) Call the API with the parameters ie, q, tl, $textlen$ and the calculated tk value;
(4) Store the received an audio file

As explained in the Alexa Skill Kit Developer documentation[5], in order to be rendered properly, the audio file must respect the following constraints: it cannot be longer than 240 seconds, the bit rate must be 48 kbps, and the sample rate must be 22050 Hz, 24000 Hz, or 16000 Hz. Since the file provided by the Google Translate service is an MP3 file with a 32 kbps bitrate, it cannot be played through the SSML audio tag. In order to make the generated file compliant with the needed requirements, the Rule Manager automatically modifies the bit rate of the downloaded file by using FFmpeg tool[6]. Later, the Rule Manager associates an ID to each personalised vocal message generated and exposes three different REST services that will be used by the Alexa Skill: one service is used for getting the ID/s of messages that have not been played yet, the second one takes as input an ID and returns the associated audio file, and the last one takes an ID and deletes the associated message.

7.2.3 The "Petal Notification" Alexa Skill

Through the Alexa Skill Kit, developers can define new vocal functionalities by accessing external services. To demonstrate the effectiveness of the proposed solution, we developed a proof of concept Alexa Skill, *Petal Notification*, which is able to connect to the Rule Manager, get the vocal message(s) and play it (them). The skill has also been certified. We defined the skill's logic as an AWS Lambda[7] Function in which, first, we identify the user who is interacting with the Echo device by allocating an ID to each user in the system. Next, a request will be sent to the Rule Manager to get the IDs of vocal messages related to the user.

[5] https://developer.amazon.com/it-IT/docs/alexa/custom-skills/speech-synthesis-markup-language-ssml-reference.html#audio
[6] https://ffmpeg.org/
[7] https://aws.amazon.com/it/lambda/

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

Later, a request for each retrieved ID will be sent to receive the corresponding audio file. As soon as an audio file is played, the vocal message will be deleted from the server. When the skill is activated, it queries the Rule Manager to discover whether there are vocal messages that are not yet played. If at least one message exists, it will be retrieved and played by the Echo device. This will be done by exploiting the audio tags specified in the Speech Synthesis Markup Language (SSML).

7.2.3 The "Petal Assistant" Alexa Skill

IDEable created a skill for Amazon Alexa to for receiving voice commands that are forwarded to the rule engine in the form of vocal inputs to exploit the vocal input as a trigger; in this way the end-users can define the desired related actions such as turn on / off and set the colour of the lights, and send the mood, put music on, etc.

7.2.3.1 Skill Activation

The Amazon Alexa Skills are in general passive: users should explicitly issue voice commands, by uttering a so-called wake word like "Alexa" followed by the invocation word associated to the skill (e.g., "Alexa, open Petal Notification"). This, for the users such as those who live with a disability, illness, or ageing could be difficult. In our project the target users are older adults, therefore we deemed useful to introduce additional ways to listen to the vocal reminders/alarms.
In the integrated platform exploiting trigger-action rules, a vocal notification can be an action contained in a rule whose trigger(s) are activated when specific events occur in the user contest: in this case, the notification is played automatically, without any user explicitly asking for it.
In order to solve this problem, we exploited the OH Echo binding, which gives the possibility to start an Alexa Routine programmatically. An Alexa routine allows users to bundle together and automate several actions (e.g., activate a skill) using a single trigger or voice command (e.g., voice, time, location). Thus, through a routine, it is possible to launch a Skill whenever an event recognised by the Alexa ecosystem occurs. For instance, it is possible to set a routine so that when the user says, "I'm leaving", Alexa responds, "Have a good day" and then turns off all the lights. Thus, we defined a routine that launches the Petal Notification Skill, but, instead of expecting the user to say a specific voice utterance, it activates the routine in OH whenever the Rule Manager Add-On receives an action involving a reminder/alarm that is in a language not currently

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

supported by Alexa. In this way, the skill is activated as soon as the rule is triggered by any relevant contextual event (for example, the user has not taken a pill in a certain period of the day), and the user can immediately listen the content of the reminder/alarm action defined in the triggered rule.

7.3 Philips Switch

The Philips switch is a wireless switch provided along with the Philips Hue ecosystem. This switch became very useful during the trials since the Philips hue lamps cannot be controlled by standard physical switches because they should be always on in order to be reached by the Philips hue bridge; thus the switch allows users to turn on/off the coloured lights and even to change their colour.
This switch has been manufactured by Philips, and obviously it allows only to control the hue lights; however, some elderlies who were participating to the trials asked whether it was possible to extend the functionalities of this small and lightweight switch in order to control other devices.
This need is due to the particular shape and the weight of the switch provided by Bartenbach, indeed, some of the elderlies found difficulties in clicking the button of the GREAT luminaire switch. Thus we decided to exploit such Philips Switch within the platform as an additional trigger. In order to integrate this switch in the Petal platform, we had to modify the context model within the context server so that we can define the events generated by the new trigger. The Philips Switch has been modelled as a technology, and it has only one attribute representing its state (ON/OFF).
After modelling the trigger in the context server, we had to update the Context Delegate Addon in OpenHab; the add-on is able to receive the events when a user clicks the switch and then it sends the value to the context server.
By adding the switch as a new trigger in the Petal platform, we allow the end-users to define the actions they want that will be executed when the user clicks on the switch: first of all, the new trigger has been exploited to control the GREAT luminaire, but it can also be useful to send an alarm (by SMS or mail) to a caregiver for example when a user falls and he/she needs help.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.

Conclusions

We have presented the updates and the new integrations made to the Petal Platform for the second round of trials.
We have taken into account the feedback received during the first round, and we have improved the platform with new functionalities, sensors and appliances in order to satisfy the user needs.
The platform is now able to monitor also the sleep quality of the users, is better integrated with the Kwido mementia game, and the GREAT luminaire.
We also exploited the Alexa device with a two-fold purpose, as an input device to allow users to self assess their mood and provide vocal commands related to the lights; and as a text to speech to synthesise reminders and alarms vocally.
We also provided the end-users with a Monitoring functionality in order to continually see and check the defined rules and the additional information related to when they have been triggered and how long they have been active. This has been particularly useful in a period characterised bythe Covid virus crisis since we could have information about what was happening in the older adults' homes without having to go there in person.
One possible evolution of the platform is to integrate it with the possibility of creating new personalization rules through the smartphone by exploiting augmented reality techniques in order to enable more immediate creation of rules, for those users who have familiarity with smartphones.

The project PETAL is cofunded by the Active and Assisted Living Programme (AAL-2016) and the following National Authorities and R&D programs in Italy, Spain, Austria and Romania.