AAL PROGRAMME	**PersonAAL**	

Deliverable 1.5.b

Integrated Platform

Responsible Unit: **Reply**

Contributors: **CNR, FCID, PLUX, USI, IBM**

Document Technical Details:

Document Number	D1.5.b
Document Title	Integrated Platform
Version	1.0
Status	Final
Work Package	WP1
Deliverable Type	Prototype
Contractual Date of delivery	30/9/2018
Actual Date of Delivery	5/10/2018
Responsible Unit	REPLY
Contributors	CNR, FCID, PLUX, USI, IBM
Reviewers	Fabio Paternò (CNR) Ivan Elhart (USI)
Dissemination Level	Public

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Document Change Log:

Version	Date	Status	Author	Description
0.1	15/5/2018	Draft	C.Chesta, REPLY	Initial Structure from D1.5a
0.2	9/8/2018	Draft	C.Duarte, FCID G.Telo, PLUX M.Manca, CNR	Updated sections 3.3, 3.6, 3.7, 3.9, 3.10
0.3	18/8/2018	Draft	R.Mitchley, IBM	Updated section 3.8
0.4	30/8/2018	Draft	C.Chesta, REPLY	Updated section 4
0.5	12/9/2018	Draft	C.Chesta, REPLY	Revised section 4.3
0.6	24/9/2018	Draft	I. Elhart	Updated section 3.5
0.7	24/9/2018	Review	I. Elhart	First Review
0.8	24/9/2018	Review	F.Paternò	Second Review
0.9	2/10/2018	Draft	C.Chesta, REPLY	Integrated Reviewer's feedback
1.0	5/10/2018	Final	C.Chesta, REPLY	Final Version

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Contents

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

1 INTRODUCTION

The PersonAAL project aims at extending the time older people can live in their home environment, by utilizing a technological platform (the "PersonAAL Platform") and intelligent and easy-to-use web applications for receiving personalized and context-dependent assistance based on contextual and health information obtained through sensors.

The platform, its subcomponents and web applications follow a distributed web-based architecture and are implemented in different technologies and running on different platforms, supporting flexibility of the development process. The components communicate over the Internet using common communication protocols assuring easy integration and easy extension of their functionality.

The Work Package 1 focuses on design, development and integration of the functional components of the platform.

This deliverable contributes to the Task 1.5 and is an accompanying document to the final Integrated Platform. It aims to collect and document technical information about the platform, its components, web applications, and sensors developed, describe the design of the interfaces for the system components and report the integration activities and decisions.

The document is organized as follows:

- Section 2 provides an overview of the platform and of communication between modules.
- Section 3 includes a description of the functionalities provided by each component and the interfaces between the components.
- Section 4 reports the technical details to realize the final integrated platform tested in the field trials.
- Section 5 summarizes the conclusions and lessons learned.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

2 PLATFORM OVERVIEW

Figure 1 shows the architecture of the personalization platform along with the communications between its modules. More details are provided in D1.1b.

The platform is composed of a number of software modules. Through the Personalization Rule Editor, users can define suitable rules following the trigger-action paradigm, which connects dynamic events and/or conditions occurring in the current context (triggers) with expected reactions (actions). The events and/or conditions are associated with some contextual aspects, namely user, environment, technology, and social relationships. The Personalization Rule Editor provides intuitive panels allowing even non-professional users to intuitively select relevant contextual situations and consequent actions.

After creation, rules are sent in JSON format to the Adaptation Engine, the architectural module aimed at handling the rules at run-time. The Adaptation Engine subscribes to the Context Manager, the module that monitors the current context, which informs the Adaptation Engine when a change in the context activates the execution of a rule.

The Context Manager provides the functionalities to collect and store contextual data from external sources (e.g. sensors, devices). To this goal, it is composed of a centralised unit (the Context Server) and a set of software modules (Context Delegates).

The Context Delegates collect contextual data from the associated sensors and then continuously provide data according to which the Context Server updates its current view of the context.
For each subscribed event and condition that occurs in the current context, the Context Manager notifies the Adaptation Engine, which extracts the list of actions from the verified rules (i.e., the rules having the 'trigger' part verified) and sends them to the application (and then to the relevant appliances, when necessary) for execution. The actions supported by the platform range from UI modifications, to sending messages, to the possibility of changing the state of appliances and devices available in the surrounding context.

The platform also includes an Authentication server (based on the implementation of the OAuth 2.0 authentication protocol), which is used by platform components, applications, and users.

The platform has evolved during the project, in several ways:

- Full integration with the Authentication Server with single-signon feature for all applications.
- Refinements of Authoring Tool and Context Manager functionalities to better support the applications developed.
- Further development of advanced modules (Persuasion Module, Social Module, Behaviour Analysis).
- Improved and extended sensors integration, with development of user-friendly context delegate app for the chestband and inclusion of additional 3rd party sensors (Fitbit, Arduino, Estimote).

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

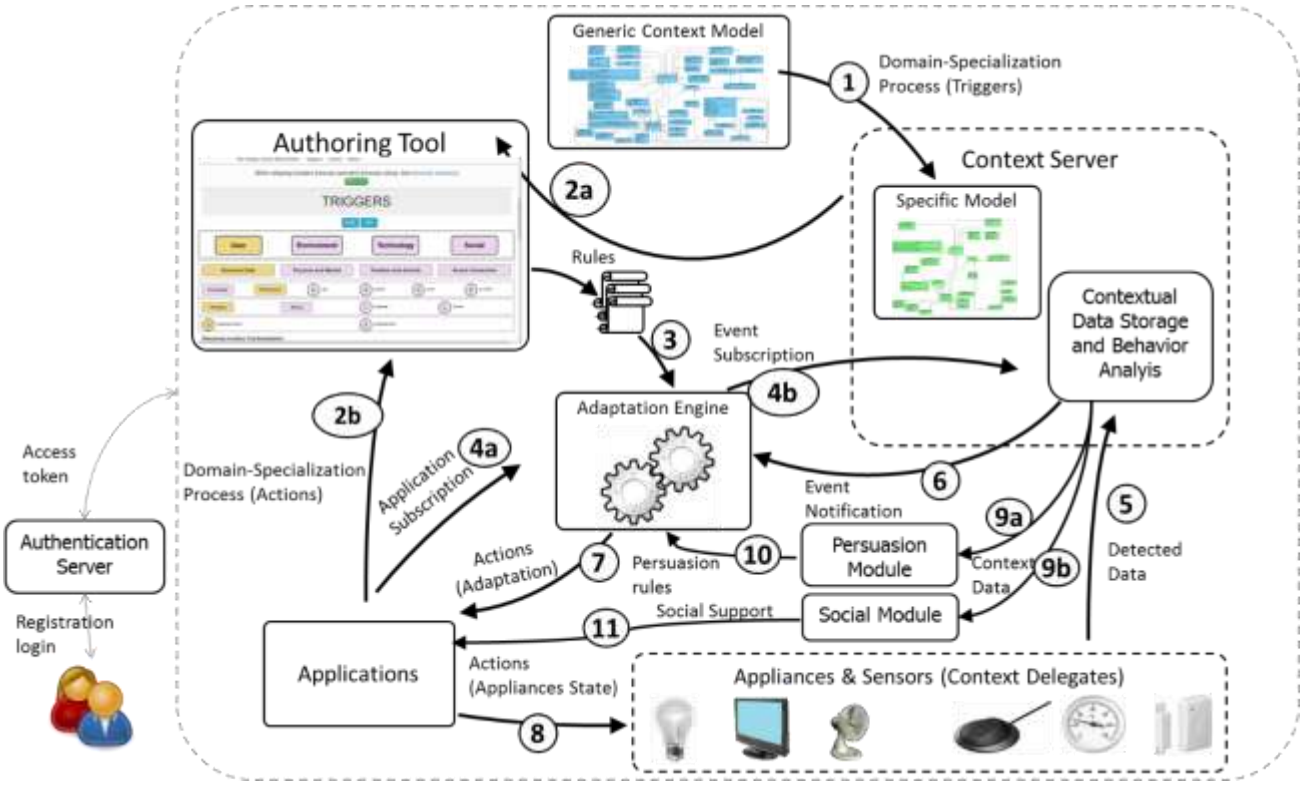

Figure 1 – Overview of the PersonAAL platform architecture

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

3 COMPONENTS AND INTERFACES

In the following for each platform component, a brief description of the functionalities and of the interfaces exposed is provided. The Applications are described in D3.1c, D3.2c and D3.3c respectively.

3.1 Personalization Rule Editor

To enable the personalization of both applications and persuasion mechanisms, we provide caregivers (and technologically expert elderly) with an intuitive Web authoring environment where they can define and refine rules (which will be provided to the Adaptation module to manage the adaptations in the platform).

To create and refine rules, users can start either from triggers or actions. Regarding the former, selection is performed by navigating in the hierarchy of concepts associated with each contextual dimension until a basic element is reached. When this element is chosen, the tool shows the possible attributes and relevant values to build the trigger of the concerned rule. In a similar way, when selecting an action, the tool shows the corresponding supported options. Triggers refer to elements identified in the contextual domain-specific model and at the highest level consider: i) user characteristics, ii) environment aspects, iii) technology, and iv) social aspects. Actions involve appliances, UI modifications, UI distribution, functionalities, alarms and reminders.

The rules are expressed through an ECA-based (Event, Condition, Action) format; where events are changes of context state, conditions are Boolean predicates referring to context state (they are optional) and actions are changes in the interactive application, in the state of the appliance or they may activate some functionalities. The language used to describe rules is presented in D1.1b (Architecture Specification Final), togheter with a detailed description of the Authoring Tool.

3.1.1 Authoring Tool REST Services

The Authoring Tool offers REST Services to save and delete rules. The rule can be saved in the personal repository (and then sent via HTTP(S) POST to the Adaptation Engine (3) in XML format).

Save Rule

Method	POST
Endpoint	https://giove.isti.cnr.it:8443/NewAdaptationEngine/rest/saveRule/appName/{appName}/userName/{userName}
Parameters	appName =[alphanumeric]: the application name associated to the rule userName =[alphanumeric]: the user name associated to the rule
Response	JSON object with two fields: • status (SAVED, NOT_SAVED) • msg

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Delete Rule

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/NewAdaptationEngine/rest/deleteRule/appName/{appName}/userName/{userName}/ruleName/{ruleName}
Parameters	appName =[alphanumeric]: the application name associated to the rule userName =[alphanumeric]: the user name associated to the rule
Response	JSON object with two fields: • status (DELETED, NOT_DELETED, ERROR) • msg

3.2 Adaptation Engine

The Adaptation Engine enables Web applications (and the Internet of Things appliances) to have adaptive behavior (changing in accordance to relevant events occurring in the elderly's context, needs, requirements, (dis)abilities, etc.). It is responsible for deciding the best combination of modalities to render messages to the user. It receives rules specified by both previous modules and the caregivers' (through the personalization rule editor) and communicates with the applications and the context manager.

The application on the user device, when loaded, subscribes for updates by sending a request to the Adaptation Engine via HTTP(S) POST. This is done by the scripts previously injected into the HTML. The Adaptation Engine extracts the rules content and subscribes to the Context Manager for the specified events and conditions.

3.2.1 Application Subscription

The application, when is loaded, subscribes for updates by sending a request to the Adaptation Engine (4a) via HTTP(S) POST; the subscription parameters are: application name, user name and action format (JSON or XML). This can be done in two ways: Web Socket communication or REST communication. More details on the communication mechanisms are reported in D1.1b.

Method	POST
Endpoint	https://giove.isti.cnr.it:8443/NewAdaptationEngine/rest/subscribe
Parameters	• userName • appName • sessionID • actionFormat (JSON or XML).
Response	JSON or XML including: • errorMessage • subscribedRules array

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

3.2.2 Event Subscription

The Adaptation Engine retrieves the rules associated to that user and application and subscribes to the Context Manager for the specified events and conditions (4b).

3.2.3 Event Notification

When one or more events are verified, and the rule conditions are met, the Context Manager notifies the Adaptation Engine (6) via HTTP(S) POST.

3.2.4 Application Adaptation

The Adaptation Engine forwards the list of actions to be applied to the UI (7). This is done via Web Socket or via a REST service defined in the target application. The client scripts, used to subscribe to the Adaptation Engine, will receive the action part of the rules and should be able to interpret them in order to apply the adaptation.

When the application receives the actions list from the Adaptation Engine, a function called applyRules is invoked; this function is defined in the adaptation-script.js file and for each action, depending on the action type (e.g. update, create, delete, etc.), invokes the corresponding function which will apply the customization.

3.3 Context Manager

The Context Manager is the module that gathers and manages contextual data. It is composed of a server and several delegates installed in various devices (e.g., a smartphone can host software detecting environment noise through the device's microphone). These delegates collect data and pass them to the server. Data is gathered from sensors (physical activity, physiological attributes, temperature, humidity, motion, etc.) or external services (e.g. weather forecasts, planned activities or medications).

The Context Manager offers REST Services to update the context sending a value sensed from a sensor and to get the current context values.

All communications to the Context Manager are protected through HTTPS protocol and requires the users to be authenticated through the Auth0 server.
Most of the REST services access the information through the user id.
For some REST services, related to sensitive information, we implemented the verification of the token released by Auth0. In those services we check if the received token is valid (every token has an expiration date) and then we extract from the token the correspondent user id. Here is the list of parameters that require the token:

- FitbitDailySummary
- CompletedActivity
- PlannedActivity
- physiological
- relativePosition

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

- medication_planned
- medication_occurred

The complete list of the services currently available are listed below.

3.3.1 Environment

The sensors used in the project provide the possibility to gather values of the data related to the aepcts indicated in the table below.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/environment/{environmentId}/{attributeName}
Parameters	environmentId =[alphanumeric]: the environment identifier (={userId} + "Env") attributeName =[alphanumeric]: the desired environment attribute. This is the list of the possible environment attributes: • temperature; • gas-sensor; • humidity; • light_level; • motion; • noise_level; • smoke-sensor;
Response	JSON object with fields: • status (ERROR or OK)[a typical error is the wrong environment id); • message; • value

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/environment/{environmentId}/{attributeName}/{attribute_value}
Parameters	environmentId =[alphanumeric]: the environment identifier (={userId} + "Env") attributeName =[alphanumeric]: the desired environment attribute. This is the list of the possible environment attributes: • temperature; • gas-sensor; • humidity; • light_level; • motion; • noise_level; • smoke-sensor; attribute_value =[alphanumeric]: the new value
Response	JSON object with fields:

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

	• entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED) • msg

3.3.2 Physiological attributes

The physiological attributes include: heart rate, body position, respiration rate, number of daily steps and body temperature, detected using BITalino chestband.

It is possible to get and update the value of each single attribute, or to get all these attributes through a single invocation calling the `physiological` attribute.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/{attributeName}
Parameters	token =[alphanumeric]: the Auth0 token attributeName =[alphanumeric]: the desired physiological attribute. This is the list of the possible physiological attributes: • heartRate • body_position • respirationRate • steps • bodyTemperature • physiological
Response	JSON object with fields: • status (ERROR or OK)[a typical error is the wrong environment id); • message; • value

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/{attributeName}/{attribute_value}
Parameters	token =[alphanumeric]: the Auth0 token attributeName =[alphanumeric]: the desired physiological attribute. This is the list of the possible physiological attributes: • heartRate • body_position • respirationRate • steps • bodyTemperature • physiological attribute_value =[alphanumeric]: the new value
Response	JSON object with fields: • entityName;

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

	• value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED) • msg

3.3.3 Devices

In order to update an attribute related to a device (it could a PC or a tablet or a smartphone, ..) it is necessary first to register the device to the context and then update a device attribute.

Method	Add the device to a user context
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/addDevice/{deviceName}
Parameters	userId =[alphanumeric]: the user identifier deviceName =[alphanumeric]: the device name
Response	JSON object with fields: • deviceId; • status [ADDED, NOT_ADDED]; • msg;

Method	Add the device to an environment
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/environment/{environmentId}/addDevice/{deviceName}
Parameters	environmentId =[alphanumeric]: the environment identifier deviceName =[alphanumeric]: the device name
Response	JSON object with fields: • deviceId; • status [ADDED, NOT_ADDED]; • msg;

The deviceId value is used to invoke the services below:

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/devices/{deviceId}/attribute/{attributeName}
Parameters	deviceId =[alphanumeric]: the device identifier attributeName =[alphanumeric]: the desired device attribute. This are some of the possible device attributes: • name • room • state • color • ...
Response	JSON object with fields: • status (ERROR or OK)[a typical error is the wrong device id);

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

	• msg (error message); • value

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/devices/{deviceId}/attribute/{attributeName}}/{attributeValue}
Parameters	deviceId =[alphanumeric]: the device identifier attributeName =[alphanumeric]: the desired device attribute. This are some of the possible device attributes: • name • room • state • color • ... attribute_value =[alphanumeric]: the new value
Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED); • msg (error message)

3.3.4 Physical Objects

Physical Objects are very similar to devices: before updating the value of an attribute you have first to register the physical object to the context.

Method	Add the physical object to a user context
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/addPhysicalObject/{objectName}
Parameters	userId =[alphanumeric]: the user identifier objectName =[alphanumeric]: the physical object name
Response	JSON object with fields: • physicalObjectId; • status [ADDED, NOT_ADDED]; • msg;

Method	Add the physical object to an environment
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/environment/{environmentId}/ addPhysicalObject/{objectName}
Parameters	environmentId =[alphanumeric]: the environment identifier objectName =[alphanumeric]: the physical object name
Response	JSON object with fields: • physicalObjectId; • status [ADDED, NOT_ADDED];

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

	• msg;

physicalObjectId value is used to invoke the services below:

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/physicalObject/{physicalObjectId}/attribute/{attributeName}
Parameters	physicalObjectId =[alphanumeric]: the physical object identifier attributeName =[alphanumeric]: state
Response	JSON object with fields: • status (ERROR or OK)[a typical error is the wrong physical object id); • msg (error message); • value

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/physicalObject/{physicalObjectId}/attribute/{attributeName}/{attribute_value}
Parameters	physicalObjectId =[alphanumeric]: the physical object identifier attributeName =[alphanumeric]: state attribute_value =[alphanumeric]: the new value (on or off)
Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED); • msg (error message)

3.3.5 Weather

Here we consider information gathered by an external service about local weather conditions.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/weather/{attributeName}
Parameters	userId =[alphanumeric]: the user identifier attributeName =[alphanumeric]: the desired weather attribute. This is the list of the possible weather attributes: • temperature; • humidity; • wind;
Response	JSON object with fields: • status (ERROR or OK); • message; • value

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/weather/{attributeName}/{attribute_value}
Parameters	userId =[alphanumeric]: the user identifier attributeName =[alphanumeric]: the desired weather attribute. This is the list of the possible weather attributes: • temperature; • humidity; • wind; attribute_value =[alphanumeric]: the new value
Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED) • msg

3.3.6 Personal Data

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/personalData/{attributeName}
Parameters	userId =[alphanumeric]: the user identifier attributeName =[alphanumeric]: the desired personal data attribute. This is the list of the possible personal data attributes: • age; • gender; • name; • surname; • education
Response	JSON object with fields: • status (ERROR or OK); • message; • value

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/personalData/{attributeName}/{attribute_value}
Parameters	userId =[alphanumeric]: the user identifier attributeName =[alphanumeric]: the desired personal data attribute. This is the list of the possible personal data attributes: • age; • gender; • name; • surname; • education attribute_value =[alphanumeric]: the new value
Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED) • msg

3.3.7 Motivation

The motivation parameter is used by the Reply Remote Assistant application to tailor physical activity interventions and presentation to the individual preferences.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{user_id}/motivation/
Parameters	userId =[alphanumeric]: the user identifier
Response	JSON object with fields: • status (ERROR or OK); • message; • value

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{user_id}/motivation/{value}
Parameters	userId =[alphanumeric]: the user identifier value =[alphanumeric]: the new value. The possible values are: "wellness", "health", "social" e "fitness"
Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED) • msg

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.3.8 User Weight

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{user_id}/weight/
Parameters	userId =[alphanumeric]: the user identifier
Response	JSON object with fields: • status (ERROR or OK); • message; • value

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{user_id}/weight/{value}
Parameters	userId =[alphanumeric]: the user identifier value =[alphanumeric]: the new value.
Response	JSON object with fields: • status (ERROR or OK); • message; • value

Method	HISTORY
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/weight/history/getNlastValues/{numValues} https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/weight/history/getValuesOnDate/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/weight/history/getValuesBetweenDates/date1/{yyyy-mm-dd}/date2/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/weight/history/getValuesFromDateToNow/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/weight/history/getValuesBeforeDate/{yyyy-mm-dd}
Parameters	userId =[alphanumeric]: the user identifier numValues =[alphanumeric]: the number of values required yyyy-mm-dd =[date]: date
Response	JSON object with fields: • historyUserWeight - timestamp - userId - userWeightId - weight • msg • status

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.3.9 User Height

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{user_id}/height/
Parameters	userId =[alphanumeric]: the user identifier
Response	JSON object with fields: • status (ERROR or OK); • message; • value

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{user_id}/height/{value}
Parameters	userId =[alphanumeric]: the user identifier value =[alphanumeric]: the new value.
Response	JSON object with fields: • status (ERROR or OK); • message; • value

3.3.10 User Profile

These services are used to GET/SET the User Profile from Remote Assistant Profile page.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/profile
Parameters	userId =[alphanumeric]: the user identifier
Response	JSON object with fields: • name; • surname; • birth_date; • gender; • state; • city; • postal_code; • address

Method	POST
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/profile
Parameters	userId =[alphanumeric]: the user identifier JSON body example: { "name": "John", "surname": "Personaal",

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

	"birth_date": "1950-06-15", "gender": "Male", "state": "Belgium", "city": "Brussels", "postal_code": "1000", "address": "3 Rue du Luxembourg" }
Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED); • msg;

3.3.11 User Interest List

These services are used to GET/SET the User Interest List from Remote Assistant Profile page.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/interest_list
Parameters	userId =[alphanumeric]: the user identifier
Response	JSON object with fields: • interest_list - interest_name - interest_category

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/interest_list

Parameters	userId =[alphanumeric]: the user identifier
	JSON_body example:
	{
	"interest_list": [
	{
	"interest_name": "sport",
	"interest_category": "Swim"
	},
	{
	"interest_name": "television",
	"interest_category": "Documentary"
	}
	]
	}

Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED); • msg;

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.3.12 User Contact List

These services are used to GET/SET the Contact List from Remote Assistant Contacts page.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/contact_list
Parameters	userId =[alphanumeric]: the user identifier
Response	JSON object with fields: • contacts_list - name - surname - phone_number - email - relationship_type

Method	POST
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/contact_list

Parameters	userId =[alphanumeric]: the user identifier
	JSON_body example:
	{
	"contacts_list": [
	{
	"name": "Mary",
	"surname": "Project",
	"phone_number": "123456789",
	"email": "mailto:mary@gmail.com",
	"relationship_type": "friend"
	},
	{
	"name": "Peter",
	"surname": "Personaal",
	"phone_number": "987654321",
	"email": "mailto:peter@outlook.com",
	"relationship_type": "close family"
	}
	]
	}

Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED); • msg;

The following services are used respectively to add and remove contacts from the list:

Method	POST

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/addContact

Parameters	userId =[alphanumeric]: the user identifier
	JSON_body example:
	{
	"contacts_list": [
	{
	"name": "Mary",
	"surname": "Project",
	"phone_number": "123456789",
	"email": "mailto:mary@gmail.com",
	"relationship_type": "friend"
	}
	]
	}

Response	JSON object with fields: • contacts_list - name - surname - phone_number - email - relationship_type

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/removeContact/{idx}
Parameters	userId =[alphanumeric]: the user identifier idx : the index of the contact to be removed
Response	JSON object with fields: • contacts_list - name - surname - phone_number - email - relationship_type

3.3.13 Chat Messages

These services are used by the Social Module to send messages to a recipient and to get the history of messages sent.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Method	SET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/message/{recipientId}

Parameters	userId =[alphanumeric]: the user identifier
	recipientId =[alphanumeric]: the recipient user identifier.
	JSON body example
	
	{
	"message": "How are you?",
	"timestamp": "2012-04-24T18:25:43"
	}

Response	JSON object with fields: • entityName • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED); • message; • value

Method	HISTORY
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/message/{recipientId}/history/getNlastValues/{numValues} https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/message/{recipientId}/history/getValuesOnDate/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/message/{recipientId}/history/getValuesBetweenDates/date1/{yyyy-mm-dd}/date2/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/message/{recipientId}/history/getValuesFromDateToNow/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/message/{recipientId}/history/getValuesBeforeDate/{yyyy-mm-dd}
Parameters	userId =[alphanumeric]: the user identifier recipientId =[alphanumeric]: the recipient user identifier. numValues =[alphanumeric]: the number of values required yyyy-mm-dd =[date]: date
Response	JSON object with fields: • historyMessages - message - messageId - recipientId - userId - timestamp • msg • status

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

3.3.14 Medication Planned and Medication Occurred

These parameters are used to model the medication data used by the IBM Medication monitoring application.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/medication_planned https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/medication_occured
Parameters	userId =[alphanumeric]: the user identifier
Response	JSON object with fields: • status (ERROR or OK); • message; • value

Method	POST
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/medication_planned https://giove.isti.cnr.it:8443/cm/rest/user/{userId}/medication_occured

Parameters	userId =[alphanumeric]: the user identifier
	
	data sent to the context manager should be a JSON object having the following format:
	for medication_planned attributes:
	{
	"notification_timestamp": "Sat, 28 Apr 2017 10:00:00 GMT",
	"notification_time": "10:00",
	"medication": "EPCLUSA",
	"dosage": "600MG"
	}
	
	for medication_occurred attributes:
	
	{
	"registration_timestamp": "Sat, 28 Apr 2017 10:05:00 GMT",
	"registration_time": "10:05",
	"medication": "EPCLUSA",
	"dosage": "600MG"
	}

Response	JSON object with fields: • entityName; • value; • status (ENTITY_NOT_UPDATED or ENTITY_UPDATED) • msg

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.3.15 Activity Planned and Activity Completed

These parameters model the user's activities and are used by both the Remote Assistant Plan page and the Phisical Rehabilitation application.

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/activity/PlannedActivity
Parameters	token =[alphanumeric]: the user token
Response	JSON object with fields: • notification_timestamp • notification_time • activity_name • activity_type • activity_intensity • activity_duration

Method	POST
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/activity/PlannedActivity

Parameters	token =[alphanumeric]: the user token
	JSON body example:
	{
	"notification_timestamp": "2018-03-07T12:37:43.048+01:00",
	"notification_time": "19:00",
	"activity_name": "Styrketrening",
	"activity_type": "2",
	"activity_intensity": "1",
	"activity_duration": "01:00"
	}

Response	JSON object with fields: • status (ERROR or OK); • message; • value

Method	HISTORY
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/PlannedActivity/history/getNlastValues/{numValues} https://giove.isti.cnr.it:8443/cm/rest/user/{token}/PlannedActivity/history/getValuesOnDate/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{token}/PlannedActivity/history/getValuesBetweenDates/date1/{yyyy-mm-dd}/date2/{yyyy-mm-dd}

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/PlannedActivity/history/getValuesFromDateToNow/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{token}/PlannedActivity/history/getValuesBeforeDate/{yyyy-mm-dd}
Parameters	token=[alphanumeric]: the user token numValues =[alphanumeric]: the number of values required yyyy-mm-dd =[date]: date
Response	JSON object with fields: • historyPlannedActivity - notification_timestamp - notification_time - activity_name - activity_type - activity_intensity - activity_duration • msg • status

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/activity/CompletedActivity
Parameters	token =[alphanumeric]: the user token
Response	JSON object with fields: • completed_timestamp • completed_time • activity_name • activity_type • activity_intensity • completed_duration

Method	POST
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/activity/CompletedActivity

Parameters	token =[alphanumeric]: the user token
	JSON body example:
	{
	"completed_timestamp": "2017-02-06T12:30:28+01:00",
	"completed_time": "12.30",
	"activity_name": "ABC",
	"activity_type": "Social Activity",
	"activity_intensity": "high",
	"completed_duration": ""
	}

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Response	JSON object with fields: • status (ERROR or OK); • message; • value

Method	HISTORY
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/CompletedActivity/history/getNlastValues/{numValues} https://giove.isti.cnr.it:8443/cm/rest/user/{token}/CompletedActivity/history/getValuesOnDate/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{token}/CompletedActivity/history/getValuesBetweenDates/date1/{yyyy-mm-dd}/date2/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{token}/CompletedActivity/history/getValuesFromDateToNow/{yyyy-mm-dd} https://giove.isti.cnr.it:8443/cm/rest/user/{token}/CompletedActivity/history/getValuesBeforeDate/{yyyy-mm-dd}
Parameters	token =[alphanumeric]: the user token numValues =[alphanumeric]: the number of values required yyyy-mm-dd =[date]: date
Response	JSON object with fields: • historyPlannedActivity - completed_timestamp - completed_time - activity_name - activity_type - activity_intensity - completed_duration • msg • status

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.3.16 Fitbit Daily Summary

Method	GET
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/activity/FitbitDailySummary
Parameters	token =[alphanumeric]: the user token
Response	JSON object with fields: • lightlyActiveMinutes • veryActiveMinutes • fairlyActiveMinute • restingHeartRate • heartRateZones - max - min - minutes - name • sedentaryMinutes • steps • floors • distance • elevation

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Method	POST
Endpoint	https://giove.isti.cnr.it:8443/cm/rest/user/{token}/activity/FitbitDailySummary

Parameters	token =[alphanumeric]: the user token
	JSON body example:
	{
	"lightlyActiveMinutes": 222,
	"veryActiveMinutes": 16
	"fairlyActiveMinutes": 22,
	"restingHeartRate": 54,
	"heartRateZones": [
	{
	"max": 144,
	"min": 103,
	"minutes": 50,
	"name": "Fat Burn"
	},
	{
	"max": 175,
	"min": 144,
	"minutes": 0,
	"name": "Cardio"
	},
	{
	"max": 220,
	"min": 175,
	"minutes": 0,
	"name": "Peak"
	}
	],
	"sedentaryMinutes": 1059,
	"steps": 8979,
	"floors": 28,
	"distance": 6.16,
	"elevation": 85.34
	}

Response	JSON object with fields: • status (ERROR or OK); • message; • value

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.4 Behavior Analysis Module

The Behavior Analysis Module is expected to analyze the data collected in the context manager, model the elderly's behavior (activity levels, application interaction, etc.) and detect deviations from standard behavior showing that the individual behavior is deteriorating, or situations of no progress towards elderly's goals. The output of this analysis is then passed to: i) the Persuasion module, to identify what and how necessary persuasions are going to be applied, and ii) the Adaptation module, to adapt the outcome of user behavior analysis before being delivered to the elderly.

3.5 Authentication Server

The PersonAAL authentication server provides the main security and data protection mechanisms to all platform components and associated web applications. The server safeguards information collected within the platform against unauthorized access and ensures data confidentiality and data integrity. The server allows only authenticated users, web applications, and platform components to upload, access, and process collected and stored data within the platform. For example, it allows only authenticated sensors to upload user data to the platform eliminating any possibility of an intentional injection of false data through malicious sensors and components. Also, the server provides a single sign-on mechanism allowing a single user account, i.e., a single set of log-in credentials to be used across all applications and platform components. This distributed authentication architecture allows for storing personally identifiable information on the secure authentication server (e.g., user name, email, password, or user id) and separating it from actual personal user data (e.g., activity data or health information) collected and stored within different elements of the platform or web applications. The collected personal user data is then identifiable by pseudonymous user identifier or authentication token issued by the authentication server after a successful authentication procedure.

The server supports two types of authentication procedures that allow active users to request PersonAAL platform services (users login into the platform or Web applications to request a service) and background platform processes to securely exchange information by authenticating each other automatically through the authentication server (to provide a secure API access). In order to secure the communication, the authentication server issues two communication tokens, an "access" and a "refresh" token. In the first authentication procedure, the authentication server issues the tokens after a user explicitly logs-in into a supported Web application or platform component. With a valid token, a Web application can then access requested resources within the platform on behalf of the user or refresh expiring access token during the active user session. In the second procedure, there is no active user involvement. Instead, the background platform processes can obtain access and refresh tokens from the authentication server for accessing communication APIs available within the platform to establish background communication and data exchange. All components accessing the platform API calls have to be registered within the authentication server before they can obtain and utilize access tokens.

The authentication server is based on the state-of-art "OAuth 2.0" protocol implementation that can authenticate all components within the system. The server is hosted on a professional and secure hosting platform that provides uninterrupted and continuous operation and access. The

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

server provides a set of API calls available to the web applications and platform components. The API calls are encapsulated into classes and objects that can be directly integrated into the application or platform code. Within the project, we provide a set of authentication components implemented using popular development technologies such as Java, JavaScript, PHP, NodeJS, and Android. For example, the JavaScript authentication component can be directly integrated into HTML file as a DOM object providing a custom login page that handles all communication to the authentication server. After a successful authentication, the authentication component receives a unique access token that identifies personal user data and can be used for subsequent calls to the platform components. The set of API calls allows users and components to signup, change password, login, logout, get access token, refresh access token, revoke access token, and get user account information. The API calls are described below.

3.5.1 Authentication API: *signup*

The *signup* API call allows users to create online accounts through the web applications and platform components. The call requires "client_id", i.e., the id of the web application or platform component through which the user wants to create an account. The users account contains "email" and "password" information. Additional user information is contained within "user_meta" field. The authentication server automatically sends a verification email before allowing users to log in.

```
POST https://personaal.eu.auth0.com/dbconnections/signup
Content-Type: 'application/json'{
       "client_id": "meVGye3Pdf6fy7zAmc1JpNJ4TSr34xQZ",
       "email": "EMAIL",
       "password": "PASSWORD",
       "connection": "CONNECTION",
       "user_metadata": "{name: 'Jon', colour: 'blue'}"}
```

An example response to the *signup* API call:

```
       {"_id": "58457fe6b27...", "email_verified": true, "email":
       "test.account@signup.com"}
```

3.5.2 Authentication API: *change_password*

After creating the account users can change the password using *change_password* API call. After the API call, the server sends an email with instructions and a separate link where users can change the password.

```
POST https://personaal.eu.auth0.com/dbconnections/change_password
Content-Type: 'application/json'{
       "client_id": "meVGye3Pdf6fy7zAmc1JpNJ4TSr34xQZ",
       "email": "EMAIL",
       "password": "",
       "connection": "CONNECTION",}
```

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

An example response to the *change_password* API call:

```
{"We've just sent you an email to reset your password."}
```

3.5.3 Authentication API: *login*

Users that have created the account can login into the server to obtain access or refresh token as well as to retrieve user account information.

```
GET https://personaal.eu.auth0.com/authorize?
       response_type=code|token&
       client_id=meVGye3Pdf6fy7zAmc1JpNJ4TSr34xQZ&
       connection=CONNECTION&
       redirect_uri=https://uc-dev.inf.usi.ch/personaal&
       state=STATE&
       additional-parameter=ADDITIONAL_PARAMETERS
```

3.5.4 Authentication API: *logout*

The logout API call allows users to logout from the authentication server.

```
GET https://personaal.eu.auth0.com/v2/logout?
       client_id=meVGye3Pdf6fy7zAmc1JpNJ4TSr34xQZ&
       returnTo=LOGOUT_URL
```

3.5.5 Authentication API: *get_access_token*

Users and platform components can obtain the access and refresh tokens. The access token is used to authenticate users and platform components without using user credentials within the system. In subsequent calls to other web applications or platform components that provide PersonAAL services, only a valid access token is needed.

```
POST https://personaal.eu.auth0.com/oauth/token
Content-Type: 'application/json'{
       "grant_type": "authorization_code",
       "client_id": "meVGye3Pdf6fy7zAmc1JpNJ4TSr34xQZ",
       "client_secret":"zIpBb2ROMTq2-zVqik_d3-WV2Zh-
KG1YkA6JHRTagnCqH841tbp5KOlWgDmnoQJs",
       "code": "AUTHORIZATION_CODE",
       "redirect_uri": https://uc-dev.inf.usi.ch/personaal}
```

An example response to the *get_access_token* API call:

```
HTTP/1.1 200 OK
Content-Type: application/json{
```

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

```
        "access_token":"eyJz93a...k4laUWw",
        "refresh_token":"GEbRxBN...edjnXbL",
        "id_token":"eyJ0XAi...4faeEoQ",
        "token_type":"Bearer",
        "expires_in":86400}
```

3.5.6 Authentication API: *refresh_access_token*

The access tokens expire after a predefined time interval (e.g., 3600 seconds). After the timeout interval, the users or components do not need to login again. They can simply refresh the access token using the token API call with the "grant_type" field set to "refresh_token".

```
POST https://personaal.eu.auth0.com/oauth/token
Content-Type: 'application/json'{
        "grant_type": "refresh_token",
        "client_id": "meVGye3Pdf6fy7zAmc1JpNJ4TSr34xQZ",
        "client_secret": "zIpBb2ROMTq2-zVqik_d3-WV2Zh-
KG1YkA6JHRTagnCqH841tbp5KOlWgDmnoQJs",
        "refresh_token": "YOUR_REFRESH_TOKEN"}
```

An example response to the *refresh_access_token* API call:

```
HTTP/1.1 200 OK
Content-Type: application/json {
        "access_token": "eyJ...MoQ",
        "expires_in": 86400,
        "scope": "openid offline_access",
        "id_token": "eyJ...0NE",
        "token_type": "Bearer"}
```

3.5.7 Authentication API: *revoke_access_token*

In a case of a suspicious activity, the user or platform component can revoke access token and prevent any further use of the token within the system.

```
POST https://personaal.eu.auth0.com/oauth/revoke
Content-Type: 'application/json'{
        "client_id": "meVGye3Pdf6fy7zAmc1JpNJ4TSr34xQZ",
        "client_secret": "zIpBb2ROMTq2-zVqik_d3-WV2Zh-
        KG1YkA6JHRTagnCqH841tbp5KOlWgDmnoQJs",
        "token": "YOUR_REFRESH_TOKEN"}
```

An example response to the *revoke_access_token* API call

```
HTTP/1.1 200 OK
(empty-response-body)
```

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.5.8 Authentication API: *get_user_info*

Authenticated and login users can access personal login credentials and information stored on the authentication server using *user_info* API call.

```
GET https://personaal.eu.auth0.com/userinfo
Authorization: 'Bearer {ACCESS_TOKEN}'
```

An example response to the *get_user_info* API:

```
{"email_verified": false,
 "email": "test.account@userinfo.com",
 "clientID": "q2hnj2iu...",
 "updated_at": "2016-12-05T15:15:40.545Z",
 "name": "test.account@userinfo.com",
 "picture": "https://s.gravatar.com/avatar/dummy.png",
 "user_id": "auth0|58454...",
 "nickname": "test.account",
 "created_at": "2016-12-05T11:16:59.640Z",
 "sub": "auth0|58454..."}
```

3.6 Sensor Chestband

The chestband is comprised of two elements that are attached to each other: a **wearable textile band** and a **hub**, in order to acquire and gather measurements from the different sensors. The textile band has a temperature transducer embedded, along with two ECG electrodes and a displacement transducer (that is used to measure the respiration) as it can be seen in the figure below (*Figure **2***).

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Figure 2 - Chestband embedded sensors

These transducers are connected to an acquisition unit that converts analogue into digital values and then streams them via Bluetooth to the Chestband's context delegate. The acquisition unit connects to the textile band through snaps and both of them together form the chestband. This includes the following sensors:

- **Respiration** – this sensor measures the thorax displacement and it not only allows the extraction of the breath rate, but also the volume of the chest.
- **Electrocardiogram** – this is a two leaded ECG placed in the V3 and V4 positions that enables a bipolar measurement between these two points. The signal is equivalent to a lead I measurement.
- **Accelerometer** – Triaxial, with 3.6g range accelerometer, used to obtain the body position and the motion. The accelerometer is included in the acquisition unit pcb and is not embedded in the textile band as the other transducers.
- **Temperature** – 0-50ºC body temperature sensor, positioned in the textile band such as it stays under the armpit, for a more accurate reading.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Figure 3 - Snap connectors that allow the textile band to attach to the acquisition unit

This architecture can be better understood with the help of the visual diagram below.

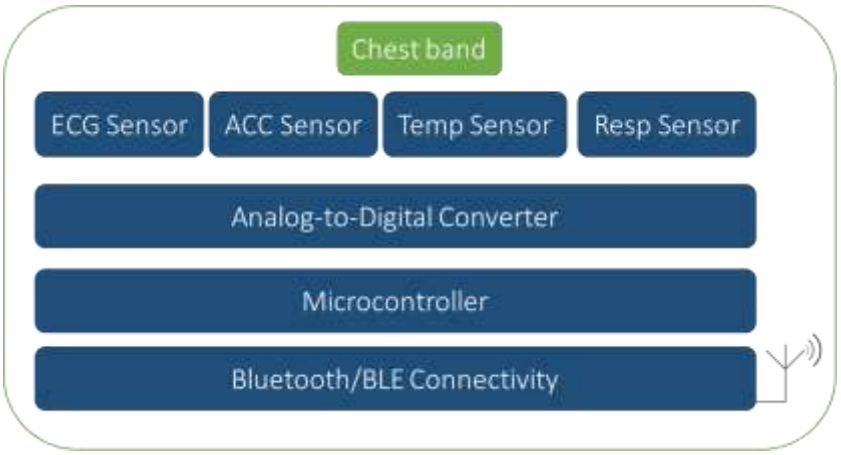

Figure 4 - Chestband high level architecture diagram

The communication protocol used to transmit the sensor signals of the chestband to the context delegate is described in full detail in the next section.
The chestband has a rechargeable battery that lasts up to 18 hours continuously streaming and is powered at 3.3V. The system is then rechargeable via micro-USB and it takes 1h30 until being fully charged. The system has three LEDs:

- **Status LED** – fades once per second if the system is on standby and blinks twice per second if the system is in acquisition mode.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

- **Low Battery LED** – turns on solid orange if the battery is below 20%.
- **Charging LED** – is solid orange if the system is correctly charging; is solid purple if the system is fully charged; is solid red if the system is not charging either because the device is on or the battery has problems.

Since the sensors of the chestband are directly connected to the body, for safety reasons, the charging module is galvanically separated from the rest of the system, only allowing to charge if the system is manually turned off.
The MAC address of the BTH module is placed under the acquisition unit to allow an easy assignement of this number to the patient profile.
Alongside the features described above, the chestband also has the following characteristics:

- Includes a shoulder strap, to prevent the band from slipping;
- Has lycra electrodes to ensure the maximum comfort, without compromising the ECG measurement;
- Is washable and reusable;
- Features a system on the back that allows the band to be adjusted to various chest sizes;
- Elastic band that allows the accurate measurement of the respiration while ensuring maximum comfort;

3.7 Chestband Context Delegate

The chestband's context delegate is an application responsible for all the exchange of queries and data between the chestband and the mobile device, as it is also responsible for the extraction of features of the raw data and sending them to the Context Manager. This application can be described taking into consideration 2 modules: the front-end module (User Interface) and the data module. The front-end module comprehends all the displays and information that is shared with the user, as for the data acquisition module it comprehends a background service that can be described with three workflows: start service workflow, connection workflow and acquisition workflow.

3.7.1 User Interface

The Context Delegate's User Interface has two main views: the Authentication View and the Data Visualization View. The authentication view (*Figure 5*) is where the user will log in using his own account, so that the Data Acquisition Module can upload the extracted features to the Context Manager. This only needs the user's interaction once, to set the account, after this first action the application will manage the authentication workflow throw the Data Acquisition Module. Therefore, the next time that the user accesses the application this view will be skipped, as long as there is a PersonAAL account available on the tablet.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

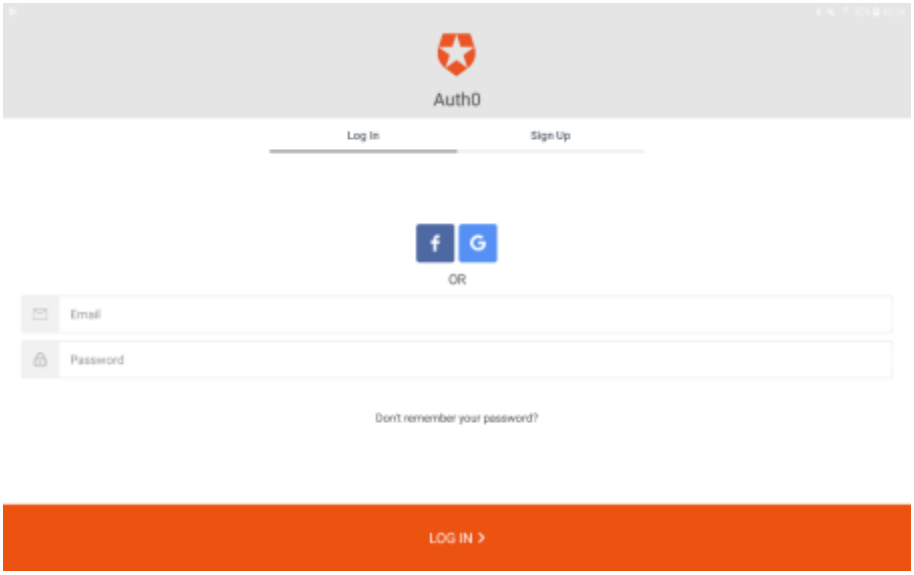

Figure 5 - Authentication view

The Data Acquisition View (*Figure 6*) is responsible for displaying the features extracted from the received data. In this view the user will have access to the Respiration and Electrocadiogram charts displayed in real time, along with all 5 features extracted from the data received from the different sensors and the status of the device connection and data upload.

As for the application being on the foreground, it is optional, since the changes on the device connection and data upload states are also provided to the user via notifications.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

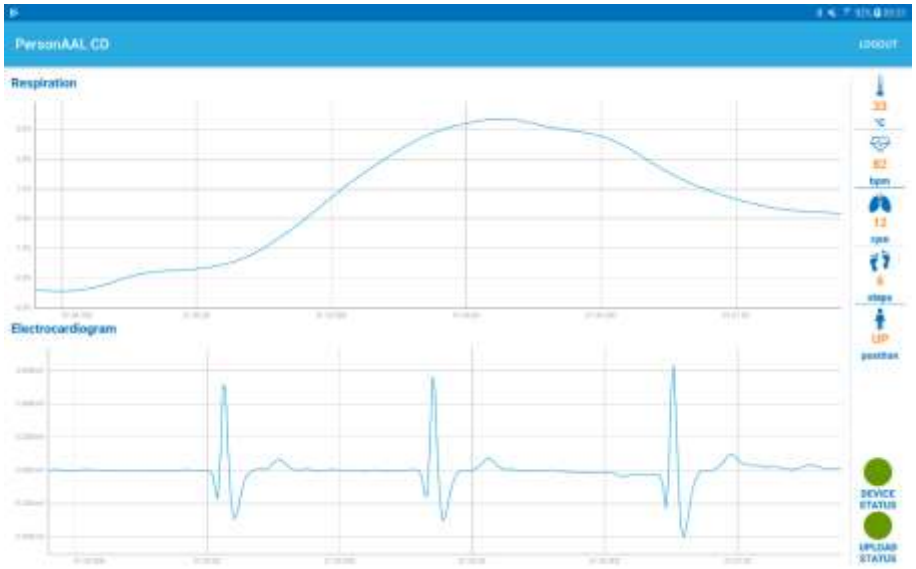

Figure 6 – Data acquisition view

3.7.2 Data acquisition background service

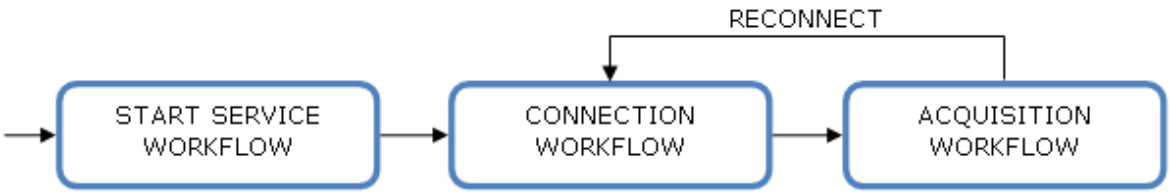

Figure 7 - Context delegate workflow

The Start Service Workflow is the implementation that allows our service to be launched when the device's boot is complete. This behaviour guarantees that our Context Delegate Service is always running on background, when the mobile device is turned on. When this procedure is completed, the Connection Workflow starts.

The Connection Workflow is the segment responsible for the scan and connection to the chestband. The scan job will always be running until a suitable device is found. As soon as the device is found and it is available to establish a connection, the scan job stops and the connection procedure starts. After a successful connection between the devices, the Acquisition Workflow starts, otherwise it keeps trying to establish the connection. If for any unknown reason the connection is lost after it is established, the Context Delegate will return to this stage and will keep trying to reconnect.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

The data acquisition is set for the chestband's six channels with a sampling frequency of 100 Hz. Once this is set, the context delegate will receive the data asynchronously. As data becomes available, it is run through the processing algorithms in order to extract the features of interest.

The features extracted from each sensor are:

- **Respiration** – after filtering and smoothing the signal, the respiration rate is extracted;
- **Electrocardiogram** – after filtering the signal, the heart rate is extracted;
- **Accelerometer** – the body position is extracted. For this feature there are 5 possible positions: UP, SUPINE, PRONE, RIGHT and LEFT. From this sensor we also run a pedometer algorithm that enables the estimation of the number of steps taken;
- **Temperature** – in this case, a transfer function is used to convert the raw data in temperature value in degrees celsius.

To avoid big data issues, all the extracted features are summarized into 5 minutes object data by: the average value of the respiration rate, heart rate and temperature during the period of time, the number of steps taken during the period of time and all the positions during this time. To enable position tracking, this value is associated with the timestamp whenever the position changes.

After getting this package of data, it needs to be sent to the Context Manager. To avoid any loss of data, a simple local database was structured and implemented to save all data until it is sent to the Context Manager. Once we receive the information that this data was sent with success, it is deleted from the database. To send data to the Context Manager a job scheduler was implemented, running a task every 5 minutes, if a wi-fi connection is available, accessing our database and sending the information to the server.

3.8 Fitbit integration and Notification App

3.8.1 Fitbit integration

To show how the platform can be used to integrate 3rd party applications and devices we have considered the Activity Tracker application integrated with fitbit data from the FitBit platform API's (see *Figure* ***8***) The users can enter the user profile in Activity Tracker, connect their Fitbit account to PersonAAL platform to allow the PersonAAL platform to subscribe to their Fitbit data. When the Fitbit device synchronizes data to the platform the PersonAAL platform is notified and requests the latest data. This is stored in the Activity Tracker database and is also passed on to the Context Manager. This allows the user to create rules in the rule editor based on their fitbit data, such as number of steps, floors, intensity level etc, and have the adaptation engine send triggers based on the fitbit data. Applications connected to the PersonAAL platform can also use Activity Tracker API to get fitbit data from the Activity Tracker database.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

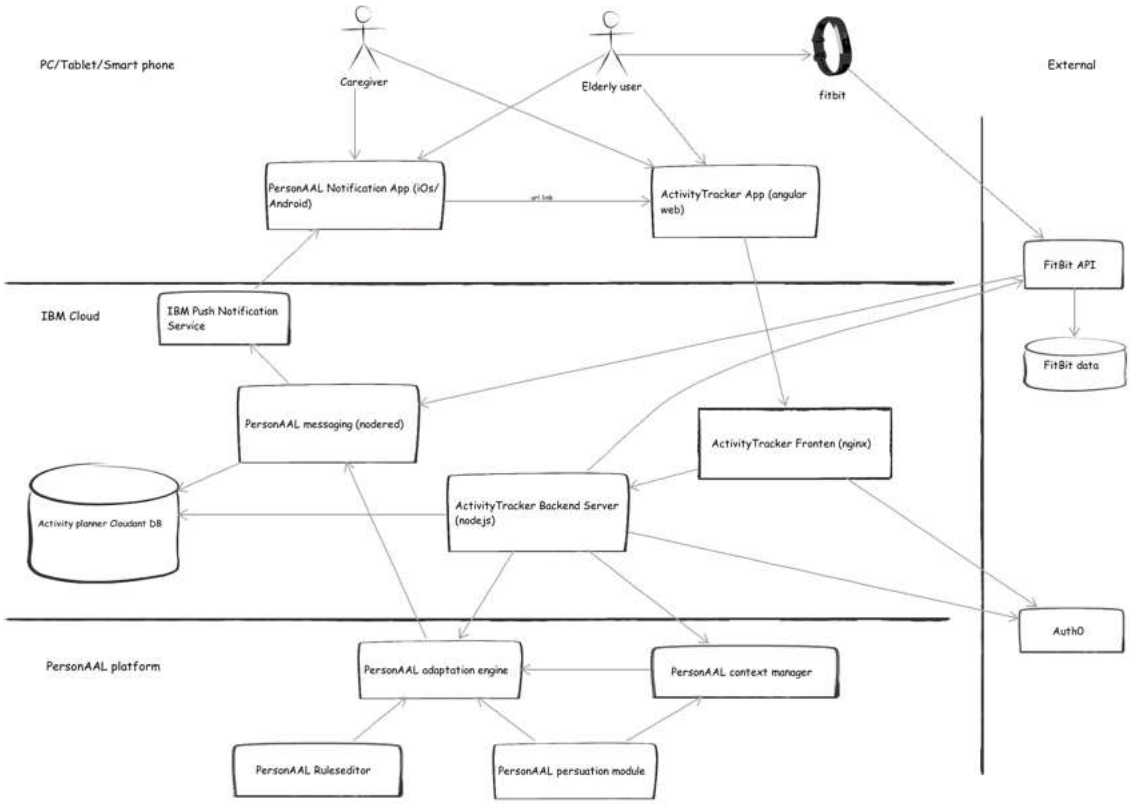

Figure 8 - Activity Tracker

3.8.2 Notification App

The Notification App has two main purposes. (1.) To receive push notifications and present these to the user and (2.) Have a single-entry point for all applications integrated in the PersonAAL platform.

1. Push notifications. This allows the PersonAAL platform to push notifications to the user without the user having to be logged on to the system. The notifications are presented on the screen of the device also when locked. All applications can send messages to a user using the API:
   https://messaging-personaal.eu-de.mybluemix.net/sendPushMessage
   Taking the request payload: {title: , message: , auth0_id: , url:}
   The PersonAAL Messaging component is also subscribing to Adaptation trigger messages. Messages with type notification and alarm are processes and sent to the Push Notification service

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

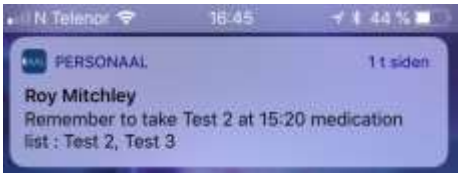

Figure 9 - Push Notification

2. Entry point to all applications. The start screen in the app contains buttons to open each of the integrated applications. Remote Assistant, Medication Monitor, Activity Tracker and Rule Editor. The application is opened in the device default browser when the user presses the button.

Figure 10 - Entry point to applications

3.9 Persuasion Module

The PersonAAL Persuasion Module aims to motivate older adults to adopt behaviours that improve their well-being and, ultimately increase the time they can live independently at home. The Persuasion Module uses the data collected by the PersonAAL framework to reason about the current behaviour of the user and decide about the motivation techniques that can be applied to try to reinforce or change the behaviour.

The concepts grounding the Persuasion Module can be found in Deliverable D1.3a. In that deliverable, the implementation details of the five modules comprising the Persuasion Module are also described. *Figure **11*** presents the architecture of the Persuasion Module, showing the five modules that process the data and generate the persuasive events, the other PersonAAL components that feed and receive data from the Persuasion Module, and an alternative data

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

source for specific activity sensing software or hardware (which can be used when the required information is not available from the Context Server).

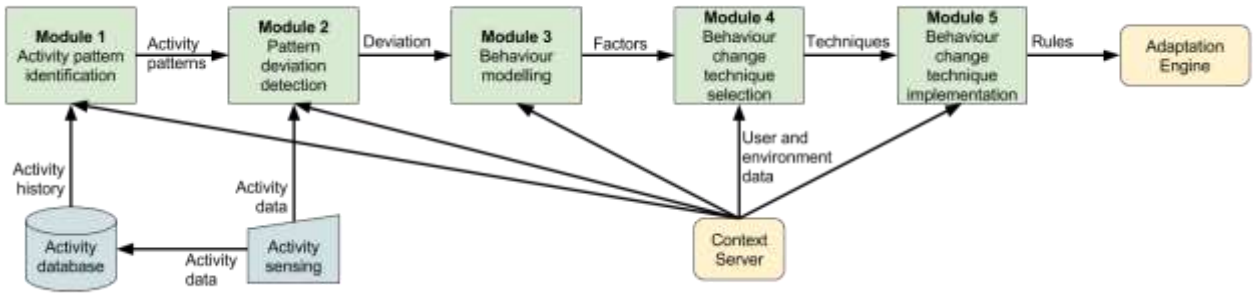

Figure 11 – Architecture of the Persuasion Module

The Persuasion Module primarily exchanges information with two other components of the PersonAAL framework: the Context Server and the Adaptation Engine. The Context Server provides the data that is required for the Persuasion Module operation. This data is required to identify the user's behavioural patterns, detect deviations from the patterns and implement behavioural change techniques. The Persuasion Module stores the context variables that are required to run the behaviour models stored and to generate the messages or events associated with behavioural change techniques. Requests to the Context Server are made to collect the most recent values of these context variables. These requests are made through the REST API made available by the Context Server.

The outcome of the Persuasion Module is a message to be presented to the user. To have these messages reach the user in a manner that is independent from existing applications, the Persuasion Module creates rules directly in the Adaptation Engine. Given the different types of behaviours and variables at play, the Persuasion Module executes its modules with a specified periodicity. When this process leads to the creation of a persuasive message, the Persuasion Module creates a rule that triggers at a specified time and executes an action that corresponds to the presentation of a reminder (whose text was created in the Persuasion Module) to the user. For example, the Persuasion Module will run at 16:00 to check the physical activity levels of a user in a given day (as observed in the user's step count for that day). If the module finds a deviation from the user's activity pattern it will generate a motivational message (to try to get the user to walk more if the deviation was negative, or to congratulate the user if the deviation was positive). This will lead to the creation of a rule in the Adaptation Engine to be triggered in the next minutes and the generated message to be shown to the user. These rules are described in JSON and sent to the Adaptation Engine through the available REST API. The Persuasion Module is also responsible for deleting the created rules from the Adaptation Engine, to prevent them from, incorrectly, triggering in the next day. This is also done through the Adaptation Engine's REST API.

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

3.10 Social Module

The PersonAAL Social Module is responsible for providing a set of functionalities or techniques focused on social support. It extends the operation of the Persuasion Module to consider also opportunities to improve the social activity level of the users of the PersonaAAL framework. Information regarding the concepts grounding the Social Module can be found in Deliverable D1.4b., including the definition of the different techniques employed by the Social Module.

The Social Module provides opportunities for social activities to be activated by the user. For this to happen several PersonAAL components are involved. Initiation can be defined by users (such as caregivers) or by the Persuasion Module through the creation of rules and correspondent rule activation. Every time a trigger is activated the rule is fired and a related user interface procedure is activated (through the adaptation module) with the respective social opportunity being initiated by the social module. Additionally, the Social Module might also identify social opportunities or lack of social activities through a set of applicational sensors (social listeners) in the users' smartphone (number of calls, number of contacts, contact information, etc.). This social information is then sent to the Context Manager, where both the applications and the Persuasion Module can pick it up for both identifying social behaviour deviations or the aforementioned social persuasion ends.

The Social Module main source of information is the Remote Assistant Application. The Plan page of this application offers to the users the ability to plan events. Social events are one of the type of events that can be scheduled via the application. The user also has the opportunity to specify which type of social event is being planned. This information is made available (through a REST API that provides access to planned events) to the Persuasion Module. The Persuasion Module combines this information with information on completed events made avaible by the Context Server, to analyse the social activity of the user. The analysis can trigger rules that are related to the social activity of the user (e.g. by triggering rules when there is an event planned for the current day, or when the user hasn't planned events for a long period of time). It is also used to personalise the messages sent to the user (e.g. "You have planned to go to the restaurant today. Why don't you invite someone else to go with you?").

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

4 INTEGRATION OF THE FIELD TRIAL ENVIRONMENT

In this section we report the scenario description and steps performed for the field-trial set-up, in order to highlight the integration aspects involved and the decision taken.

4.1 Hardware Set-up

Each of the end-users has been equipped with the following sensor-kit, as illustrated in Figure 12:

- Tablet (Lenovo Yoga Tablet2 or Samsung S5 Neo smartphone)
- Plux Bitalino Chestband
- Fitbit Charge 2
- Arduino Hub for environmental parameters (motion, temperature, gas)
- Estimote proximity beacons
- Philips Hue Lamps

Figure 12 – PersonAAL Field trial sensor kit

The Android-based tablet run Firefox or Chrome browser and are used to run the web-based applications (Remote Assistant, Medication Monitoring, Phisical Rehabilitation, Authoring Tool).

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

The Plux Bitalino chestband (see section 3.6) and Fitbit Charge 2 fitness wristband (see section 3.8) are personal devices acquiring biosignal data and described in detail in previous sections. The associated context delegate are installed on the mobile device.

The Arduino hub is used to collect environmentals parameters as described in Figure 13.

The configuration of the sensor network is composed of an Arduino Master Uno (enhanced by an Ethernet shield to provide the Arduino board with Internet access) and multiple Arduino Micro (Nodes) connected to it. Each Arduino unit is equipped with a NRF24L01 module, a transceiver that allows to make a wireless communication between two Arduino elements.

As for the Nodes, they include an Arduino-compatible sensor for gas (MQ5), another one for detecting temperature and relative humidity (DHT11), and one for detecting motion.

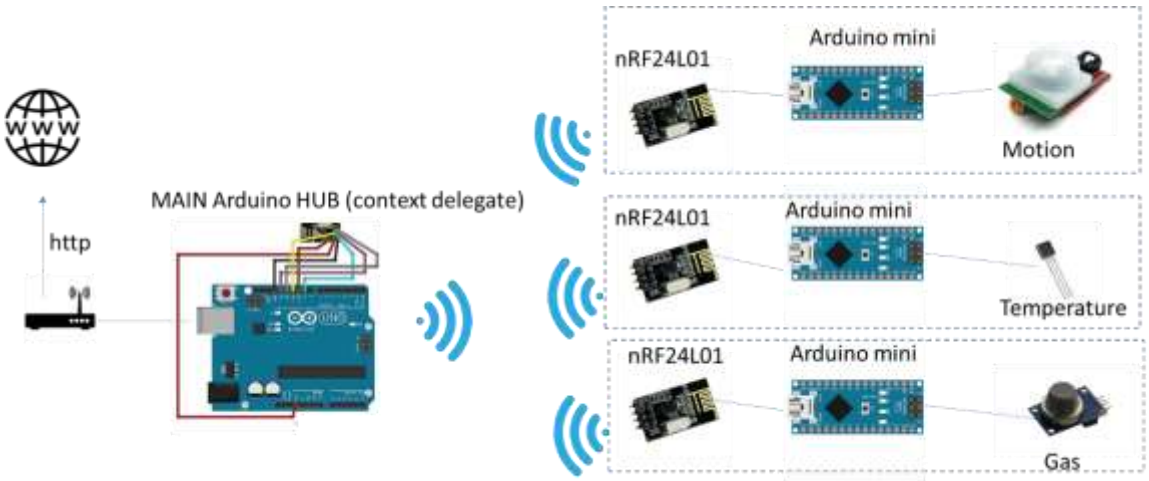

Figure 13 - Arduino Hub and sensors

Furthermore, we exploit some Estimote Proximity beacons to derive the position of the user. The beacons provide three proximity zones: immediate, near, far, according to the strength of the signal received (from a few centimetres to some meters).

Besides the sensors, the kit also includes as actuators Philips Hue white and color ambiance lamps equipped with Philips Hue Bridge 1.0. The bridge allows the light system to connect to a wifi router, in order to remotely control the lamps (turn on/off or change color).

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

4.2 Implementation technical details

This section collects the technical information about the implementation of all components, including the development and execution environment, information about storage of data, and information about communication protocols.

The Authentication Server provides Single Sign On endpoints for all platform components and apps.

	Execution Environment	Development Environment	Data Storage	Communication	Domain
Authoring Tool	Windows server 2016, Apache Tomcat 8.0.44	Java Servlet Pages, Spring 4.3.7.RELEASE , spring security 4.2.2.RELEASE	Mysql 5.6 (for user settings) Rules are saved on a private folder on the server at CNR. Logs of user interactions (timestamp + event type) are saved in a CNR server.	RESTful API (to communicate with the server side part and to send the rules to the adaptation engine), Https	https://giove.isti.cnr.it:8443/AuthoringTool/login
Adaptation Engine	Windows server 2016, Apache Tomcat 8.0.44	Java Servlet Pages, Spring 4.3.7.RELEASE	no data base, Rules are saved on a private directory on the CNR server,associated with username and application	RESTful API over https, WSS and STOMP (http://www.concretepage.com/spring-4/spring-4-websocket-sockjs-stomp-tomcat-example) for sending the actions to the application when a rule is triggered	https://giove.isti.cnr.it:8443/NewAdaptationEngine/rest/subscribe (POST call. User have to provide user name and app name) The adaptation engine sends the actions through secure web socket or to rest endpoint
Context Manager	Windows server 2016, Apache Tomcat 8.0.44	Java Servlet Pages, java servlets	mysql database for history information. Current context entities are stored in memory and then saved in a xml file representing all the context. Saved information is in CNR server	RESTful API to update a context entity, Https POST for the subscription from the adaptation engine and to send to the adaptation engine updates when a rule is triggered	https://giove.isti.cnr.it:8443/cm/
A1 Remote Assistant	Windows Server 2008 R2, Microsoft Information Server IIS 7	PHP 5.3.228	MySQL Workbench 6.3	RESTful API, Websocket, Https	https://personaal.cloud.reply.eu/login.php

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

A2 Medication Monitoring	Bluemix using Cloud Foundry with node.js runtime	node.js, npm, express, cordova (android/iOs) app	cloudant (nosql), dashdb (sql), stored encrypted in Bluemix datacentre	Https, RESTful API	medication-personaal.eu-gb.mybluemix.net
A3 Physical rehabilitation	Bluemix using Cloud Foundry with node.js runtime	node.js, npm, express, cordova (android/iOs) app	cloudant (nosql), dashdb (sql), stored encrypted in Bluemix datacentre	Https, RESTful API	activitytracker-personaal.eu-gb.mybluemix.net
Chestband Context Delegate	Android Platform, min 2.1	Android Native Code, Java	No local storage of data	Rest service over https	Not Applicable – context delegates connect to Context Manager
Persuasion Module	Ubuntu 14.04.5 LTS	Java 7	MySQL 5.6	RESTful API, HTTP	http://accessible-serv.lasige.di.fc.ul.pt/~personaal/
Authentication Server	Cloud Sandbox with Secure Storage	Linux	MySQL	RESTful API, HTTPS	https://personaal.eu.auth0.com

4.3 Integrated Scenario Realized at M36

Scenario description	Component Involved
User registers to the Authentication Server	Authentication Server
User logs-in and accesses the Personalization Rules Editor. User shows the context dimensions that compose the triggers and the action structure	Authentication Server Personalization Rules Editor
User defines the following rules: 1. WHEN motivation is wellness AND time is 16:00 AND daily steps are less than 1000, DO send a reminder with the text: “Your daily steps are below the goal, please go out for a walk”; 2. WHEN respiration rate (or heart rate) is more than 80 breaths/min (or 180 bpm) AND body temperature is more than 38°, DO send an ALARM with the text: “Your body temperature is too high”. 3. IF Time is after 22:00 AND WHEN Motion becomes true, DO Turn On and set color light Living Room color to White for 2 minutes The defined rules are sent to the adaptation engine.	Personalization Rules Editor Adaptation Engine

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

User accesses the Remote Assistance Application and shows all the implemented functionalities (health, plan, profiles, contacts). In the health section show the ECG and temperature values received live from the sensor chestband.	Authentication Server Remote Assistant application Context Manager Context Delegates Sensor Chestband
The user adds to the plan a “walk” activity and he is given the possibility to invite a contact.	Social Module Remote Assistant application
Trigger of rule 1: Modify manually through the context manager REST Service the value of steps and time, modify the motivation through the application and highlight the effects of the actions on the application Trigger of rule 2: Modify manually the values of respiration rate and body temperature and highlight the effects of the actions on the application; Trigger of rule 3: the user moves near the sensor and manually another partner change the time through the REST service and the lights turn on.	Context Manager Adaptation Engine Remote Assistant application
User accesses the IBM Medication application and show the implemented functionalities	Authentication Server Medication application
User shows that the medication information entered through the Medication Application are sent to the Context Server.	Medication application Context Manager
If user has not complied to medication as planned over time a persuasion rule is activated to remind the user to take his medication. This rule sends messages to the Notification app on smart phone / tablet.	Medication application Persuasion module Adaptation engine
User accesses the IBM Physical application and shows the implemented functionalities. Creating an activity plan, being notified about planned activity, record completing activity. Receiving persuasion messages to the Notification app when not performing the planned activities.	Physical Rehabilitation Context Manager Adaptation Engine
If a deviation from the expected behavior is detected (medications taken in the wrong order) an alert is sent to the user.	Behaviour Analysis
Persuasion Module generates a rule and then sends it to the adaptation engine. Load the adaptation engine page that shows all the stored rules (https://giove.isti.cnr.it:8443/NewAdaptationEngine)	Persuasion Module Adaptation Engine

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

5 CONCLUSIONS

The document presented the description of the final integrated PersonAAL platform, as it evolved in M36, detailing the interfaces exposed to the applications and the hardware setup.

The presented platform supports personalisation of context-dependent applications in AAL scenarios. It has been integrated with various sensors, appliances and applications.

The platform has been used in field trials to test its effectiveness in real settings as documented in D3.4.

The project was very productive delivering significant functionality towards the project goals and also providing a number of lessons learned, best practices and recommendations.

We summarize below the main lessons learned during the integration process:

- We adopted a loosely coupled design for the PersonAAL platform, supported by REST APIs. This approach offers optimal flexibility and reusability. It allow adding, replacing, or modifying components in a distributed way, and showed to be very convenient and productive for the consortium team. The use of protocols more oriented to sensors and mobile devices (such as MQTT) could be exploited in the future to obtain more realible and efficient communication.

- We ensured a seamless experience to the end-users, by allowing them to access all applications and platform components using the same login credentials and keeping their user profile personalized across the platform.

- In the trials different personalization needs where expressed by the users involved, thus confirming the assumption of the project that a platform to facilitate such personalization can be useful and effective.

- We dedicated considerable effort to implement security and data protection mechanisms to all platform components and associated web applications. This aspect is very important for the acceptance of the web-based service, both during the field trials and for further exploitation.

- Despite our best efforts, some system upgrades and configurations were not completely straightforward, causing some unexpected instability. During the third year of the project we improved the process adopting an Issue Log to plan releases and an Installation Manual to document and replicate configuration of hardware and software.

- Another good practice we recommend is keeping updated documentation of each component for both the end users and developers willing to try the solution, as we started to do in the Technical Information page on the project web site (http://www.personaal-project.eu/).

The project PersonAAL is cofunded by the AAL Joint Programme (AAL-2014) and the following National Authorities and R&D programs in Italy, Portugal, Norway and Switzerland.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra