www.fit4work-aal.eu

FIT4WORK

SELF-MANAGEMENT OF PHYSICAL AND MENTAL FITNESS OF OLDER WORKERS

CO-FUNDED BY

AAL PROGRAMME

The National Centre for Research and Development

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

ZonMw

UEFISCDI EXECUTIVE AGENCY FOR HIGHER EDUCATION, RESEARCH, DEVELOPMENT AND INNOVATION FUNDING

REPUBLIC OF SLOVENIA MINISTRY OF HIGHER EDUCATION, SCIENCE AND TECHNOLOGY

PARTNERS

PSNC

ANF

"Jožef Stefan" Institute

UNIE KBO

SGS

TEAMNET transforming technology

USM

Project coordinator: Poznań Supercomputing and Networking Center, ul. Jana Pawła II 10, 61-139 Poznań, Poland, email: fit4work@fit4work-aal.eu

Personal Wellness Record

Ambient Assisted Living Joint Programme project no. AAL-2013-6-060

Deliverable 5.5, version 1.0

Lead author: Aleksander Stroiński, Poznań Supercomputing and Networking Center

Co-authors: Michał Kosiedowski, Poznań Supercomputing and Networking Center

Maciej Bogdański, Poznań Supercomputing and Networking Center

Project coordinator: Poznań Supercomputing and Networking Center, ul. Jana Pawła II 10, 61-139 Poznań, Poland, email: fit4work@fit4work-aal.eu

Published on 28th of September, 2016

The Fit4Work project is co-financed though the AAL Joint Programme by:

- European Commission
- National Centre for Research and Development, Poland
- Ministry of Industry, Energy and Tourism, Spain
- Executive Agency for Higher Education, Research Development and Innovation Funding, Romania
- Ministry of Higher Education, Science and Technology, Slovenia
- The Netherlands Organisation for Health Research and Development (ZonMW), The Netherlands

Table of contents

Project coordinator: Poznań Supercomputing and Networking Center, ul. Jana Pawła II 10, 61-139 Poznań, Poland, email: fit4work@fit4work-aal.eu

1. Introduction

The Fit4Work system aims to deliver an innovative easy-to-use and unobtrusive product supporting older workers and the relevant stakeholders in reducing and managing physical and mental stress resulting from their occupation and daily duties. The system collects various data based on ambient monitoring, physical activity monitoring and mental stress monitoring at work. Smart algorithms are used to process collected data and provide context-sensitive personalized recommendations for adjusting the workplace and behavior at work. All this is done by continuous monitoring of the user, which requires storing the collected information in data repositories referred to as Personal Health Record (PHR) systems. The Fit4Work system has to be equipped with specialized components able to collect, store and manage the physical and mental fitness related information similar to PHR.

There are many different definitions of what the Personal Health Record is. One definition tells that PHR is an Internet-based set of tools that allows people to access and coordinate their lifelong health data and make appropriate parts of it available to those who need it, and offers an integrated and comprehensive view of health information [Markle]. Others define PHR as an electronic, universally available, lifelong resource of health information needed by individuals [AHIMA] or that PHR is a system that allows saving health related data from devices like: smart watches, wearable devices or fitness trackers and monitoring all aspects of health: from weight and fitness to life-threatening conditions such as diabetes, high blood pressure, heart, kidney and chronic lung disease [AppHealth].

The common thing that connects all these definitions is that PHR is a health record, where health data and information related to the given person are stored. The scope of data stored in such systems is strongly related to medical data, observations, symptoms, diseases, allergies, etc. It is true that some PHR systems allow to store fitness and wellbeing data, but this is not the leading functionality.

In case of the Fit4Work system, a record-keeping system is needed, which will store biomedical data and parameters related to fitness, wellbeing and ambient sensors. Fit4Work also needs to provide functionality allowing for easy and safe access to data for various components of the system, and for other external systems that the user will allow for an access. Implementation of such a component, within Fit4Work project, containing a database to store relevant data and mechanisms for accessing the data, will be called **Personal Wellness Record**.

2. General architecture and requirements

The Fit4Work system (shown in Figure 2.1) has been designed in such a way that there are only two components that can be accessed by many other components (or communicate with many other components): Cloud Integration Layer (CIL) and Personal Wellness Record. The former is a communication layer, a communication proxy, so by the definition CIL is the heart of the system and communication should take place through it. The latter is the Personal Wellness Record which can be accessed by all system components via CIL, or directly by all Fit4Work recommenders due to communication optimization [SysArch]. Personal Wellness Record acts as a central data repository within the Fit4Work system where processed data is stored.

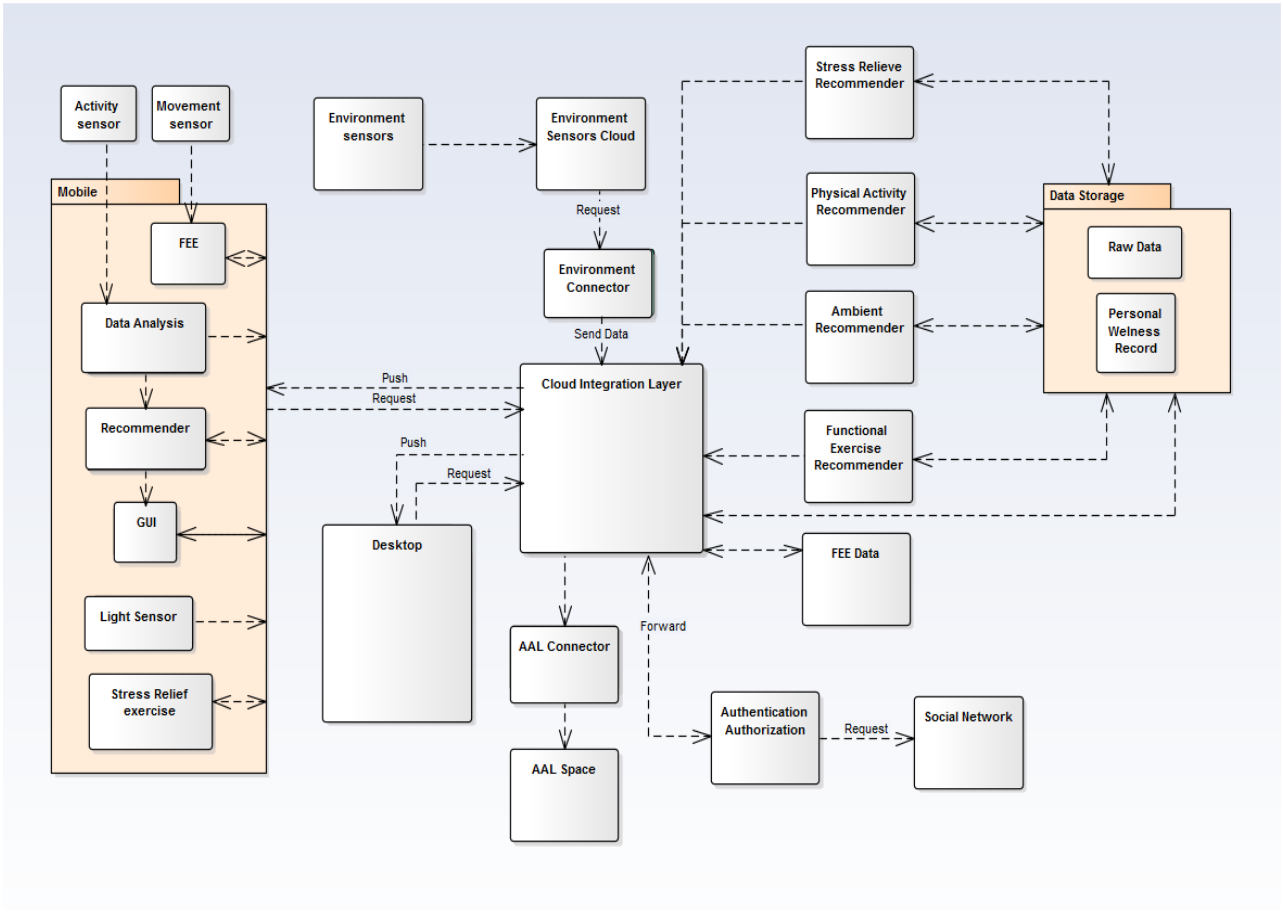

Figure 2.1 Fit4Work system architecture

Personal Wellness Record provides functionality related to the database (physical data storage, manages data access rights, assures transactions). On the other hand, Personal Wellness Record delivers a web service for simple and easy access and processing of information stored in the database.

Requirements for this data repository for the system perspective are the following:

- Archive of information related to the user
    - user account and personal data
    - user devices and sensor information
    - current state of user physical activity, mental stress and work environment
    - summary of current day, week and month of user physical activity, mental stress and work environment
    - history of the recommendation
    - history of activates
    - other historical data (e.g. weight history)
    - sensor data
- Services
    - Allowing the creating of data in the Record and reading of data collected in the Record
    - Providing access to advanced search

2.1. Common Data Model

In order to facilitate the communication between various Fit4Work components and devise format of stored data, a Common Data Model (CDM) has been developed. The model creates basic data structures used within Fit4Work and is developed from requirements posed by Fit4Work applications and other components.

Common Data Model is at the heart of the Personal Wellness Record, and directly relates to data gathered about the user's wellbeing (in terms of physical, mental and environmental wellbeing). Its development was also necessary for establishing remote interfaces between Fit4Work components, by creating a common language and terminology all components could agree upon. Data gathered by sensors, stored in databases or exchanged between Fit4Work components conform to CDM.

Common Data Model can be divided into two main parts:

- Raw Data Model - relating to raw data gathered by Fit4Work sensors. This type of data is stored on a separate server in order to be processed be Fit4Work recommenders. Raw Data is not part of Personal Wellness Record.
- Processed Data Model - relating to data aggregated, averaged or otherwise obtained by processing Raw Data by Fit4Work data analysis components. This data is stored in Personal Wellness Record.

In other words, structure of the Personal Wellness Record database (shown in Figure 2.2) is directly defined by Processed Data Model (the dominant part of Common Data Model). Processed Data Model describes a

wide scope of data related to a person's wellbeing and was built around the concept of a Wellbeing Factor. The Wellbeing Factor relates to a single user activity type monitored by the system (physical activity, mental stress or environmental conditions). The Wellbeing Factor is summarized in a form of a simple percentage score which determines how well the user is doing - i.e. determines the user "well-being" as far as a particular "factor" (physical, mental, environmental) is concerned. Based on this concept the Fit4work system works in a workflow consisting of following steps:

- archive all data related to the user and remember changes of this data in time
- gather sensor data form different user devices (smartphone, band, environmental sensor, etc.)
- based on archived and gathered data, make calculations and detect activates
- calculate Wellbeing Factors for a given period of time based on the detected activities and sensor data collected
- produce real-time recommendations based on Wellbeing Factors changes in time

Data in this workflow may be divided into following groups:

User related data:

- User – data related to user's name, age and email address
- Gender – indicates a gender of the person: female, male or unknown
- UserOrganization – data related to user's workplace
- OccupationType – indicates a type of occupation: sedentary or active
- UserParameter – data related to the user's height or weight (or potentially other parameters)
- UserParameterName – indicates type of user parameter: height, weight, etc.
- UserGoals – data related to the daily or weekly physical activity goals in minutes or kcal

Sensor related data:

- UserDevice – data related to the sensor devices used by the user
- SensorData – one single value based on raw data processing
- SensorDataGroup – group of values captured or semantically related sensor data
- SensorType – indicates a device which captured the data: smartphone, band, environment
- DataType – indicates type of data: heart rate, galvanic skin response, skin temperature, heart variability, humidity, CO2, noise, etc.

Activity related data:

- UserActivity – data related to the activity recognized by the algorithms, duration of the activity: start and end date, heart rate at the beginning and at the end of the activity, relaxation score at the beginning and at the end of the activity
- UserActivityType- indicates type of activity; cycling, walking, standing, running, etc.
- UserActivityGroup – indicates a group of activities: sports, inactive and mental stress relief exercises

Wellbeing Factor related data:

Project coordinator: Poznań Supercomputing and Networking Center, ul. Jana Pawła II 10, 61-139 Poznań, Poland, email: fit4work@fit4work-aal.eu

- WellbeingFactor – is a single value calculated by Fit4Work algorithms representing the state of physical activity, the level of relaxation or environmental conditions in a given period of time
- ScoreLevel – the level of wellbeing factor calculated for a given period of time
- WellbeingFactorType – indicates type of wellbeing factor: physical, mental or environmental
- WellbeingFactorGroup – groups all three types of wellbeing factors from the same period of time
- WellbeingFactorGroupType – indicates a type of grouped wellbeing factors: daily or weekly

Recommendation related data:

- Recommendation – data related to the type, name of recommendation, text and parameters of the given recommendtion
- RecommendationName – indicates the name type of recommendation: e.g. “open window”, “perform a stress relief exercise”, etc.
- RecommendationType – indicates if recommendation is short or long term
- State – indicates a change of recommendation state with timestamp
- StateType – indicates recommendation state: created, accepted, dismissed, finished

Based on these groups of data related to the user’s wellbeing, the Personal Wellness Record was designed.

Figure 2.2 Structure of Personal Wellness Record database

Project coordinator: Poznań Supercomputing and Networking Center, ul. Jana Pawła II 10, 61-139 Poznań, Poland, email: fit4work@fit4work-aal.eu

3. Implementation

From a technological point of view, the final implementation of the Personal Wellness Record can be summarized by the following statements.

Personal Wellness Record:

- is a web service providing a RESTful API
- is implemented using the Spring Boot framework
- uses the JPA standard for data manipulation
- is compatible with HAL standard which is valid with HATEOAS constraints
- uses Basic Authentication and HTTPS to secure communication

Implementation details, technical issues, programming nuances, explanations and motivation about technologies and standards behind Personal Wellness Record are described in the following chapters.

3.1. REST Service

There are two leading approach to creation of web services: SOAP and REST [RKS2016].

SOAP is an acronym from Simple Object Access Protocol. It assumes that a service has a set of operations that take certain arguments. SOAP is by definition very formal - each service should provide a WSDL [WSDL] file that describes each operation, what data it takes, what type of data it is, etc. Thanks to this, service clients have access to tools generating code based on a WSDL file, which makes using an external API easier from the implementation point of view. On the other hand, even simple queries require a lot of code around the transmitted message (Envelope), which makes it difficult to test manually, analyze queries or use them with very simple clients.

REST is an acronym for the REpresentational State Transfer and is built around completely different assumptions. In the case of REST, we have URLs that are some kind of identifiers. We send a request to these addresses, which can be JSON, XML, but also plain text or binary data. What should happen is determined by the HTTP method used – e.g. GET gets an element, DELETE removes it, etc. The web service response may be in JSON, XML or other, depending on the type of response the client wants.

SOAP is definitely the heavyweight choice for Web service access. It provides the following advantages when compared to REST [JohnMueller2013]:

- Language, platform, and transport independent (REST requires use of HTTP)
- Works well in distributed enterprise environments (REST assumes direct point-to-point communication)

- Standardized
- Provides significant pre-build extensibility in the form of the WS* standards
- Built-in error handling
- Automation when used with certain language products

REST is easier to use for the most part and is more flexible. It has the following advantages when compared to SOAP:

- No expensive tools required to interact with the Web service
- Smaller learning curve
- Efficient (SOAP uses XML for all messages, REST can use more compact message formats)
- Fast (no extensive processing required)
- Closer to other Web technologies in design philosophy

Considering that Personal Wellness Record should be a universal service which will be used not only by elements of the Fit4work system (physical activity recommender, mental stress recommender, environmental recommender, cloud integration layer) but also Personal Wellness Record could be accessed by external systems or services, a better and more flexible solution is the REST approach. From the point of view of the project, it is extremely important that the chosen technology be effective and easy to implement by third parties which again illustrates the advantages of REST.

Therefore, it was decided that the leading approach to designing Personal Wellness Record would be the REST approach.

REST implementation of Personal Wellness Records indicates that all data should be treated as a resource, which is a fundamental concept in any RESTful API [Geert2011]. A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it. It is similar to an object instance in an object-oriented programming language, with the important difference that only a few standard methods are defined for the resource (corresponding to the standard HTTP GET, POST, PUT and DELETE methods), while an object instance typically has many methods. Resources can be grouped into collections. Each collection is homogeneous so that it contains only one type of resource. According to the above, Personal Wellness Records is a resource-oriented web service and only basic operations like: add, update, remove or modify are allowed on a single resource at a time.

3.2. JPA

The main functionality of Personal Wellness Record is to archive data defined by the model (Common Data Model) in a relational database. Therefore, the JPA standard was used to support and accelerate this type of operation.

Java Persistence API is a collection of classes and methods to persistently store vast amounts of data into a database. JPA is an open source API. Using JPA reduces interacting with the database significantly from the development standpoint. JPA is a bridge between object models (service as a program using Common Data Model) and relational models (database). It describes how to map a POJO (Plain Old Java Object) as an entity and how to manage entities with relations. The mapping process is done by annotations. Every POJO class is an entity and is declared using the @Entity annotation (at the class level):

```
@Entity
public class User {
   @Id
   public Long id;
}
```

@Entity declares the class as an entity, @Id declares the identifier property of this entity. The other mapping declarations are implicit. The class User is mapped to the User table, using the column id as its primary key column. Using this concept whole model can be annotated. Common Data Model with annotations represents the structure of Personal Wellness Record database, in terms of the relational model. Each object is mapped to a table, and each field in the object is a separate column. Annotations can also describe relations between objects that are mapped in the database with foreign keys. An instance of an object is mapped in the database as a single row (entry).

3.3. Spring Boot

Creating web services using JPA requires a database and model with annotations that allows to map the object model to the relational model. The last element is access to this data and possibility to perform operations on it (e.g. get, create, delete, modify). In order to accomplish this, Spring Boot is used as a framework providing such functionality for web services compatible with the REST approach.

Spring Boot makes it easy to create stand-alone, production-grade web services. It contains an embedded web server (Tomcat, Jetty or Undertow), provides simple auto configuration and production-ready metrics and health checks. Additionally, creating web services using Spring boot minimizes the amount of code being created.

Spring Boot, based on the annotated model, connects to the database at startup, check the database structure and make necessary modifications (creating non-existent tables, adding fields to tables, etc.).

Spring Boot also generates repositories. Repositories are classes or interfaces that define operations that can be performed on the model (such as saving the object in the database, searching for objects etc.). Basic operations corresponding to the HTTP methods like: PUT, CREATE, DELETE, GET are generated by default.

3.4. HATEOAS Standard

Due to the fact that the Personal Wellness Record has been implemented in accordance with the REST paradigm, the basic requirement is the implementation of the Hypermedia as the Engine of Application State (HATEOAS) standard. HATEOAS is a constraint of the REST application architecture that keeps the RESTful style architecture unique from most other network application architectures. The term “hypermedia” refers to any content that contains links to other forms of media such as images, video or text [HATEOAS].

The Personal Wellness Record makes use of hypermedia links in the response content, so that the client can dynamically navigate to the appropriate resource by traversing the hypermedia links. This is conceptually the same as a user navigating through web pages by clicking the appropriate hyperlinks in order to achieve a final goal. A hypermedia-driven site provides information to navigate the site's REST interfaces dynamically by including hypermedia links with responses. This capability differs from that of SOA-based systems and WSDL-driven interfaces. With SOA, servers and clients usually must access a fixed specification that might be staged somewhere else on the website, on another website, or perhaps distributed by email [SpringDoc].

Similar to a human’s interaction with a website, a REST client hits an initial API URI and uses the Personal Wellness Record provided links to dynamically discover available actions and access to resources it needs. The client does not need prior knowledge of the Personal Wellness Record or the different steps involved in a workflow. Additionally, clients no longer have to hard code the URI structures for different resources. This allows the server to make URI changes as the API evolves without breaking the clients.

Examples

The following code represents a User object.

```
class User {
   String name;

   int id;
}
```

A simple JSON presentation is traditionally rendered as:

```
{
   "name" : "Alex",

   "id": "1"
}
```

The user data is there, but the data contains nothing about its relevant links.

A HATEOAS-based response would look like this:

```
{
   "name": "Alex",

   "id": "1",

 "links": [ {
      "rel": "self",
      "href": "http://localhost:8080/user/1"
   } ]
}
```

This response not only has the person's name and id, but includes the self-linking URL where that person is located.

- *rel* means relationship. In this case, it's a self-referencing hyperlink. More complex systems might include other relationships. For example, an address might have a "rel":"user" relationship, linking the address to its user.
- *href* is a complete URL that uniquely defines the resource.

In other words, HATEOAS provide semantic knowledge for the communication between client side and service side. HATEOS follows a simple rule: it is better when the data describes itself, rather than if the documentation had to describe it. Using HATEOAS API gain some semantic information [Adam2014]:

- client side immediately knows where to look for information about the element (link)
- owner of service can transfer the resource to a different address and on the API side, only links are changed. All clients using the API will continue to work.
- The resource address no longer depends on its identifier. They can be independent of each other
- the links to the resource can be hidden if the service owner does not want that clients know where to look for the details of the resource
- every resource address does not have to be written in the documentation. The API is self-documented.

The idea of semantic knowledge in Personal Wellbeing Record is to save data with the maximum possible information about their meaning. Based on the HATEOAS standard, an ideal service can have only one main address. The client acquires addresses of all other resources through semantic links. Personal Wellness Record was implemented to act as such “ideal service”. A client needs only a one main address to discover all data. The main address response of Personal Wellbeing Record looks like:

```
{
  "_links" : {
   "wellbeingFactor" : {
     "href" : "http://datastorage.fit4work-aal.eu/wellbeingFactor{?projection}",
```

```
  "templated" : true
},
"userPreference" : {
  "href" : "http://datastorage.fit4work-aal.eu/userPreference"
},
"userDevice" : {
  "href" : "http://datastorage.fit4work-aal.eu/userDevice{?projection}",
  "templated" : true
},
"user" : {
  "href" : "http://datastorage.fit4work-aal.eu/user{?projection}",
  "templated" : true
},
"userActivity" : {
  "href" : "http://datastorage.fit4work-aal.eu/userActivity{?projection}",
  "templated" : true
},
"userOrganization" : {
  "href" : "http://datastorage.fit4work-aal.eu/userOrganization"
},
"state" : {
  "href" : "http://datastorage.fit4work-aal.eu/state{?projection}",
  "templated" : true
},
"userGoals" : {
  "href" : "http://datastorage.fit4work-aal.eu/userGoals"
},
"recommendation" : {
  "href" : "http://datastorage.fit4work-aal.eu/recommendation{?projection}",
  "templated" : true
},
"userParameter" : {
  "href" : "http://datastorage.fit4work-aal.eu/userParameter{?projection}",
  "templated" : true
},
"sensorData" : {
  "href" : "http://datastorage.fit4work-aal.eu/sensorData{?page,size,sort,projection}",
  "templated" : true
},
"wellbeingFactorGroup" : {
  "href" : "http://datastorage.fit4work-aal.eu/wellbeingFactorGroup{?page,size,sort,projection}",
  "templated" : true
```

```
    },
    "sensorDataGroup" : {
      "href" : "http://datastorage.fit4work-aal.eu/sensorDataGroup{?page,size,sort,projection}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://datastorage.fit4work-aal.eu/profile"
    }
  }
}
```


3.5. JSON vs XML

According to the HATEOAS standard JSON and XML are accepted as a standard response format. HATEOAS doesn't impose the requirement of either format.

In order to select the appropriate format for Personal Wellness Record responses, a comparative analysis of JSON and XML was made [W3S]:

- Both JSON and XML are "self-describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest
- JSON does not use tags
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function – XML requires a lot more effort to parse than JSON. JSON is parsed into a ready-to-use JavaScript object. For AJAX applications, JSON is faster and easier than XML.

Due to the many advantages of JSON, this format has been chosen as the response format for Personal Wellness Record.

3.6. HAL format

Personal Wellness Record is a service compliant with the HATEOAS standard. This standard requires appropriate representation of service resources (data). It defines the way in which application clients interact with the server, by navigating hypermedia links they find inside resource models returned by the server. To implement HATEOAS, you need some standard way of representing resources, that will contain hypermedia information (links to related resources). HAL is one of such standard of data format. It defines a specific format of resource presentation, that can be used to implement HATEOAS.

HAL (Hypertext Application Language) [HAL] is a simple format that gives a consistent and easy way to link between different resources in service API. Thanks to HAL, the API is more explorable, self-documented and becomes easier to be implement by clients. Services that adopt HAL can be easily served and consumed using open source libraries available for most major programming languages.

HAL provides a set of conventions for expressing hyperlinks in either JSON or XML. The Conventions are very human-friendly, which means that dependencies described by HAL are represented in a simple and textual way understandable to a human. The HAL conventions revolve around representing two simple concepts: Resources and Links.

Resources have:

- Links (to URIs)
- Embedded Resources (i.e. other resources contained within them)

Links have:

- a target (a URI)
- a relation (the name of the link)
- optional properties to help with deprecation, content negotiation, etc.

Figure 3.1 illustrates how a HAL representation is structured.

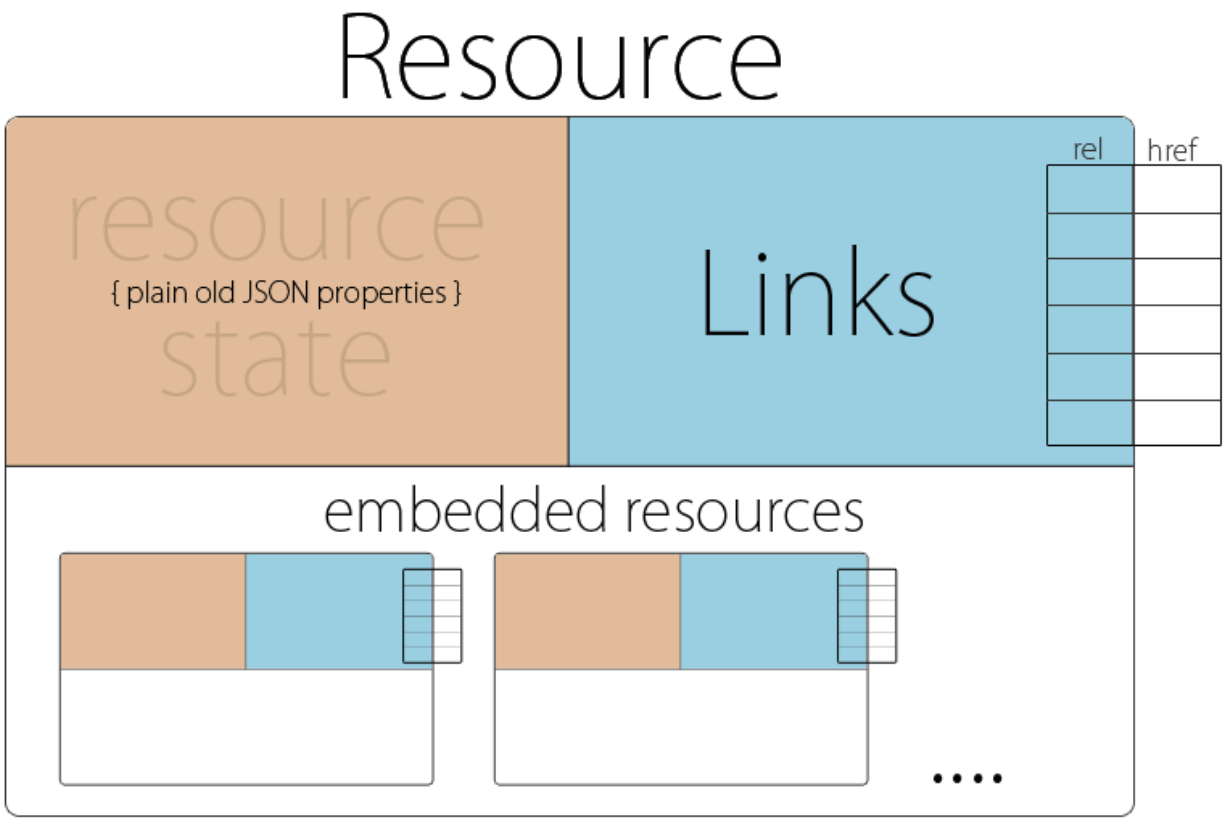

Figure 3.1 Structure of HAL representation.

HAL is designed for building APIs in which clients navigate around resources by following links. Links are identified by link relations. Link relations tell the client side about what resources are available and how they can be interacted with. Link relations are not just an identifying string but they are working URLs.

HAL has a media type for both the JSON and XML variants, whose names are application/hal+json and application/hal+xml, respectively. When serving HAL over HTTP, the Content-Type of the response should contain the relevant media type name.

For JSON resources, the minimum the service must do is provide a _links property containing a self-relational link.

User resource on Personal Wellness Record example:

```
{
   "id" : 1,
   "dateOfBirth" : "1984-12-14T15:22:03.207+0000",
   "gender" : "MALE",
   "_links" : {
                    "self" : {
                                "href" : "http://datastorage.fit4work-aal.eu/user/1"
      }
          }
}
```


If a resource is linked to other resources, in "_links" section, properties will appear with a names of linked resources and URLs to them.

A User resource link to weight and height resources example:


```
{
     "id" : 1,
     "dateOfBirth" : "1984-12-14T15:22:03.207+0000",
     "height" : {
       "timestamp" : "2017-05-24T15:22:29.995+0000",
       "name" : "HEIGHT",
       "value" : "185",
       "unit" : "cm"
     },
     "weight" : {
       "timestamp" : "2017-05-24T15:22:30.294+0000",
       "name" : "WEIGHT",
```

```
    "value" : "80",
    "unit" : "kg"
  },
  "gender" : "MALE",
  "_links" : {
    "self" : {
      "href" : "http://datastorage.fit4work-aal.eu/user/1"
    },
    "height" : {
      "href" : "http://datastorage.fit4work-aal.eu/user/1/height"
    },
    "weight" : {
      "href" : "http://datastorage.fit4work-aal.eu/user/1/weight"
    }
  }
}
```


3.7. Security: BasicAuth and HTTPS

The information in Personal Wellness Record is considered to be sensitive personal information, so many expectations of privacy, ethical and legal issues are implicated in its maintenance, such as third-party access and appropriate storage. The data stored in Personal Wellness Record should be well protected and secured. The first level of protection is based on BasicAuth. Basic access authentication is a method for an HTTP user agent to provide a user name and password when making a request. HTTP Basic authentication (BA) implementation is the simplest technique for enforcing access controls to web resources because it does not require cookies, session identifiers, or login pages; rather, HTTP Basic authentication uses standard fields in the HTTP header, removing the need for handshakes. The BA mechanism provides no confidentiality protection for the transmitted credentials. They are merely encoded with Base64 in transit, but not encrypted or hashed in any way. HTTPS is, therefore, typically used in conjunction with Basic Authentication. Because the BA field has to be sent in the header of each HTTP request, the client needs to

cache credentials for a reasonable period of time to avoid constantly prompting the user for their username and password.

Second level of protection is based on HTTPS. HTTP Secure is an adaptation of the Hypertext Transfer Protocol (HTTP) for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted by Transport Layer Security (TLS).

For communication purposes, the https protocol has been issued a certificate for the domain datastorage.fit4work-aal.eu

4. Bibliography

[AHIMA] - AHIMA e-HIM Personal Health Record Work Group (2005)

[Markle] Markle Foundation's Personal Health Working Group, Connecting for Health (2003)[5]:3

[AppHealth] Apple Health - https://patient.info/personal-health-record

[SysArch] Deliverable 5.1 System Architecture

[JohnMueller2013] Understanding SOAP and REST Basics And Differences, https://blog.smartbear.com/apis/understanding-soap-and-rest-basics/

[DrivenAPI] HATEOAS Driven REST APIs, https://restfulapi.net/hateoas/

[SpringDoc] Understanding HATEOAS, https://spring.io/understanding/HATEOAS

[Adam2014] RESTful API - how to do it right? ,http://adam.wroclaw.pl/2014/07/restful-api-jak-zrobic-je-dobrze/#more-52]

[W3S] JSON vs XML, https://www.w3schools.com/js/js_json_xml.asp

[HAL] HAL - Hypertext Application Language, http://stateless.co/hal_specification.html

[Geert2011] 2011, Geert Jansen RESTful api – Resource http://restful-api-design.readthedocs.io/en/latest/resources.html

[RKS2016] Roopesh Kevin Sungkur, 2016, Combining the best features of SOAP and REST for the implementation of web services

[WSLD] Web Services Description Language, https://www.w3schools.com/xml/xml_wsdl.asp