Acronym:	vINCI
Name:	Clinically-validated INtegrated Support for Assistive Care and Lifestyle Improvement: the Human Link
Call:	AAL 2017 “AAL Packages / Integrated Solutions”
Contract nr:	AAL-2017-
Start date:	01 June 2018
Duration:	36 months

D4.2. Kits deployment and validation

Nature[1]:	R
Dissemination level [2]:	CO
Due date: Month:	Month 6
Date of delivery:	Month 6
Partners involved (leader in bold):	**ICI**, NIGG, UNRF, NIT, SAL, CTR, CMD

Project Co-Funded by:

Project Partners:

[1] L = legal agreement, O = other, P = plan, PR = prototype, R = report, U = user scenario
[2] PU = Public, PP = Restricted to other programme participants (including the Commission Services), RE = Restricted to a group specified by the consortium (including the Commission Services), CO = Confidential, only for members of the consortium (including the Commission Services)

Partner list:

No.	Partner name	Short name	Org. type	Country
1	National Institute for Research and Development in Informatics	ICI	R&D	Romania
2	Marche Polytechnic University	MPU	R&D	Italy
3	University of Nicosia Research Foundation	UNRF	R&D	Cyprus
4	National Institute of Telecommunications	NIT	R&D	Poland
5	Connected Medical Devices	CMD	SME	Romania
6	Automa Srl	AUT	SME	Italy
7	Optima Molliter (f. Salvatelli) Srl	SAL	SME	Italy
8	National Institute of Gerontology and Geriatrics “Ana Aslan”	NIGG	R&D	Romania
9	Comtrade Digital Services	CTR	Large enterprise	Slovenia
10	Orange Polska S.A.	OPR	Large enterprise	Poland

Revision History

Rev.	Date	Partner	Description	Name
1	05.05.2019	ICI	Created the template, added the sections and draft content	Ciprian Dobre
2				
3				

Contents

1. Introduction

vINCI project proposes a novel approach for providing personalized assistance services for patients in an IoT-based ecosystem. In this context, the challenge remains to develop technologies that meet the needs of older adults, accommodate their cognitive and perceptual declines, capitalize on their intact abilities, support them in performing everyday activities, and protect their privacy, independence and security. In this Deliverable, we describe these exact technologies, at the heart of the project. This follows Deliverable D4.1, where we described the environment in which vINCI will be implemented and the behaviours / biomarkers to be monitored.

vINCI aims, thus, to develop an integrated and validated framework using the Internet of Things (IoT) to provide non-intrusive monitoring and assistance services to older people in order to increase the level of quality medical care. By integrating open data analysis technology with user-centric IoT devices in four standardized kits as well as by implementing business models, vINCI aims to provide support to people providing intelligent care and assistance to the elderly who are treated in ambulatory clinics or doing outdoor activities. To verify, test (clinically validate) and identify added value, two controlled multidisciplinary pilots will be launched (in Romania and Cyprus) and two open-call validations (in Poland and Slovenia) with implementation in controlled environments real-life case studies with the involvement of older people in Europe. The ultimate goal is to demonstrate a systematic approach to ensuring the highest level of quality control, automatic monitoring and data management.

The main objective of the vINCI project is to create an integrated technology platform, through which the elderly is discreetly monitored (mixed technology) through sets of extensible technologies, personal records being safely stored (through confidentiality techniques) and analyzed to retrieve information to be delivered to carers (via a single dashboard) to detect early symptoms of age-related deficiencies (proactive loop) and trigger alerts related to possible incidents (such as falling , reactive loop).

The vINCI platform has four inputs: a static profile of the patient; the results of a Quality of Life Questionnaire (QoL) as perceived by the elderly person monitored; data from monitors that are integrated into the platform (smartwatch, smart shoes, depth camera); and questionnaire data on the level of activity and the psychosocial level of the monitored person. All of these compose a model of the data monitored for a vINCI person, a model we will validate under clinical conditions using clinical data provided by medical staff in Romania (the NIGG pilot).

The **smart shoes** are equipped with Force Sensing Resistors (FSRs), which provide variable resistance according to the force applied to the active area of the sensor. An algorithm implemented and executed within the shoe electronics collects sensor resistance values and identifies various conditions and activities: sitting, walking, running, or if shoes are not worn. Each such state is identified by a different numeric tag that is transmitted to a remote server via a wireless communication interface (the transmission is triggered by a state change). When shoes are not worn by the subject, an initially selected time interval is used to set a proper status tag; from that moment, whenever the condition changes (e.g., from state to rewind or back), a packet is sent to the remote server. By collecting the packets of data transmitted over time and by checking server time-outs, it is possible to plot a timeline of the physical activities performed by the subject, such as the derivation of the information related to how long it stood in each state considering the entire sub- observation.

The smart shoes (SS) feature a wireless communication interface (BLE or LoRA) that can be used in indoor scenes. A transceiver is using in-shoe electronics to convert sensor-generated data into packets that are transmitted to a BLE receiver, such as a smartphone, or to a WiFi router, from where they are forwarded to a remote server. For each SS pair, a shoe is equipped with an onboard sensing device (which includes the FSR). The electronic board is equipped with a wireless transceiver that includes low power capabilities (so the shoe can be operated for a long time). These details can be found in the Figure below. Each transmitted message contains a numeric value representing the status / activity recognized by the device, such as running, walking, etc.

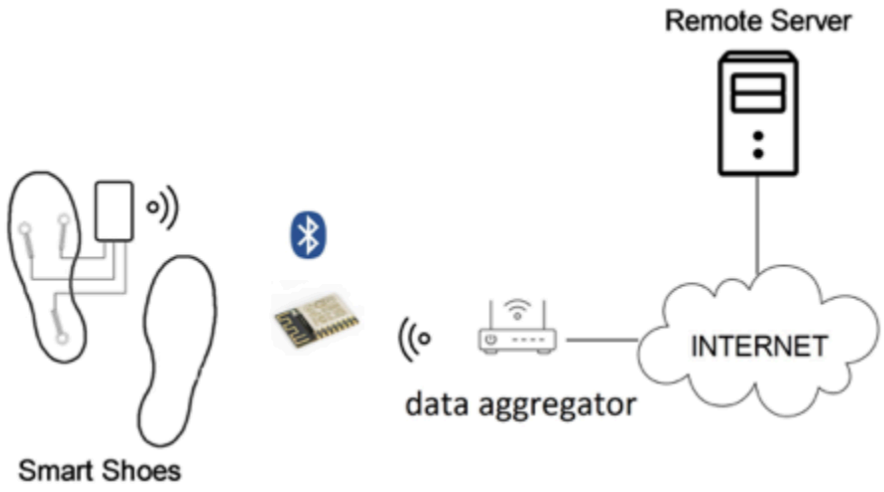


Fig. 1. Data transmission for the smart shoes.

The data provided by the **depth sensor** depends on the configuration of the function the sensor is used for. In a top-view configuration (which fits best with the project), with the sensor located physically on the ceiling, we can recognize and track the subject by processing the frames generated by the sensor.

Once the *tracking mode* is active, the depth sensing system sends the list of spatial coordinates according to the top-view plan to the remote server. In such a configuration, we can identify the patient's "blob" and locate it in an observable environment (eg, we can detect how long the subject spends in a certain area, such as how long it stayed on the couch).

On the server, the information about how long the subject spent in a particular area is extracted - we have the necessary technology, but the depth sensor will most likely have to be coupled with a video camera to detect a particular person in a group (suppose the tracking action takes place in a family, and we want to track the in-house actions of a particular member of that family, in particular). Research in this regard is innovative and has been launched within the project.

If the depth sensor is used in front of the subject, it is possible to extract the so-called *body joints' coordinates* relative to the subject, that is the spatial coordinates of a number of specific points associated with the human body's landmarks. By collecting these coordinates and processing them, it is possible to obtain a representation of the movements made by the subject - this is also an innovative aspect of the project.

The depth camera is equipped with a USB interface (USB3.0 for Kinect v2, or USB 2.0 for Orbbec Astra Mini) and requires a connection with a computer - we may use a mini computer (such as Intel NUC i3-7100U) to interface with the sensor, get the signals and process it locally. Data sent to the vINCI platform will not be raw data but information data (such as already recognized junction points, or even detected activity). Figure 2 schematically shows how the deep sensor connects to the remote server.

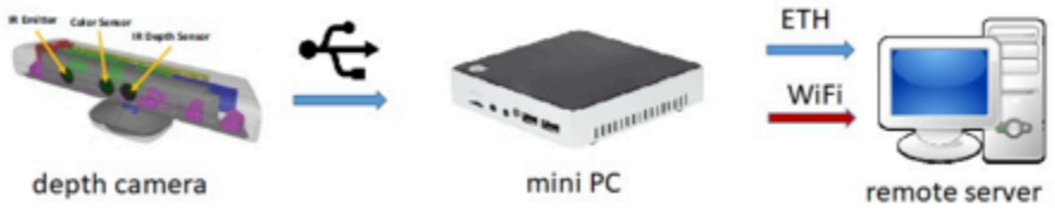


Fig. 2. Connection between the depth sensor and the backend server.

The **smartwatch** transmits information to the *Connected Medical Devices platform*, where it is exported in JSON format and sent to the vINCI platform (see also Figure 3). The data format is as follows: GPS location / time; Number of steps; Battery level; Nr. times when the clock came out of a defined area; time intervals when the clock was removed from the hand.

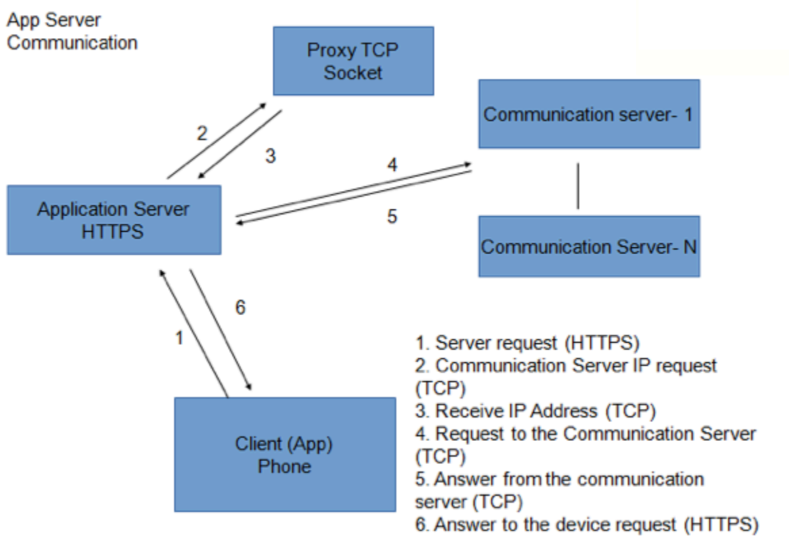

App Server
Communication
Proxy TCP
Socket
Communication server- 1
Communication Server- N
Application Server
HTTPS
Client (App)
Phone
1
2
3
4
5
6
1. Server request (HTTPS)
2. Communication Server IP request (TCP)
3. Receive IP Address (TCP)
4. Request to the Communication Server (TCP)
5. Answer from the communication server (TCP)
6. Answer to the device request (HTTPS)

Fig. 3. Connection between the CMD smartwatch and the platform.

The CMD platform stores and interprets information such as the phone number associated with the SIM inserted in the clock, the phone number of the device where the CMD mobile application is running, GPS points in time (tracking information), or "safe zones / geo fences" currently associated with a clock.

The fourth component is represented by **vINCI Digital Caregiver application** - an application running on the mobile tablet that the Subject has at home. The connectivity between the application and the vINCI modules is provided using conventional broadband communications technologies (WiFi or LTE, if there is a Wireless Router in the Subject's home, or Ethernet or 3G / 4G, depending on the specific conditions in the home). This Caregiver is integrated directly with the vINCI platform, without the need for interfacing through edge nodes.

All information provided to the vINCI platform will be sent using cryptography solutions (for personal data protection)

1.1. An updated work scenario

After analyzing the technological changes compared to the time of the project proposal (in Deliverable D2.1), we came to the conclusion that the working scenario described (originally) in the vINCI proposal requires a reconsideration. Thus, for the purpose of monitoring a topic, we use the following: DHL One watch, intelligent shoes, depth camera (the exact proposal is reflected here), and a series of Quality of Life Questionnaires (newly introduced).

The analysis carried out together by technology experts, partners with expertise in geriatrics and psychosocial science has led to the following scenario of vINCI:

Step 1 - The static profile of the patient is determined through a set of clinical questionnaires. Once enrolled in vINCI, the subject (or its family, if required) receives a Tablet that runs the VINCI Digital Caregiver application. The subject registers and creates an account / profile in the system (a Web alternative will also be available for this step).

Step 2 - Next, the subject completes a WHOQOL questionnaire to determine the perceived level related to the Quality of Life Index. An example of how we want to construct this questionnaire (and others) can be found in Figure 3. As we can see, we want an easy way for the Topic, most likely based on Smileys.

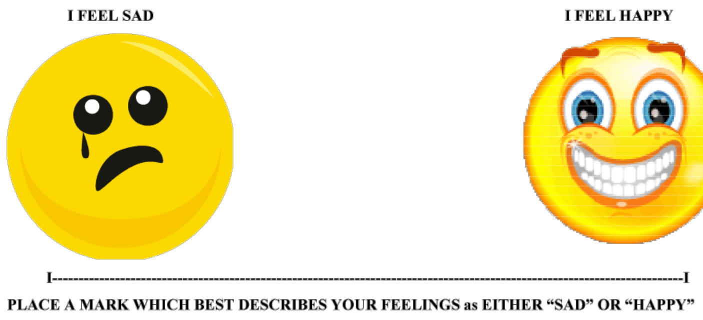

I FEEL SAD
I FEEL HAPPY
I--I
PLACE A MARK WHICH BEST DESCRIBES YOUR FEELINGS as EITHER "SAD" OR "HAPPY"

Fig. 3. Examples of questions to be answered by the Subject.

The questionnaire will be provided with as few details as possible (the absolutely indispensable text being provided in the native language of the Subject) and as many accessible input methods as possible. The tablet online form will be repeated periodically to continually re-evaluate the patient's clinical status (and adjust the scales used to monitor vINCI accordingly).

Step 3 - From this point on, the Subject wears the CMD watch. In the house, the patient is monitored by the depth camera coupled with a video camera. First, we train the algorithm to track the person in the house, with the goal of continuously detecting the level of activity associated with it. Next, we aim to extend the project to detect fragility issues by monitoring the person while doing some exercises developed by us and tracking how the subject performs them (we are interested in mobility in the hands and feet, for example).

Periodically, the Subject is also asked to complete the D-VAMS and IPAQ questionnaires to get the levels he perceives related to activity and social issues.

Step 4 – Room kits, watch, shoes, questionnaires, they all serve to obtain subject-relevant data. However, to correctly interpret these data, we need properly validated clinical models. In the clinical pilots, we will use alternative medical methods to get factual issues related to the subject's health. These aspects will then be mapped to data captured by vINCI technology (based on sensors), with the goal of building supervised learning models.

1.2. vINCI's Architecture

The architecture of the project (Figure 4) is based on requirements analysis and a study of the specialized literature, where we started from consulting similar previously developed projects.

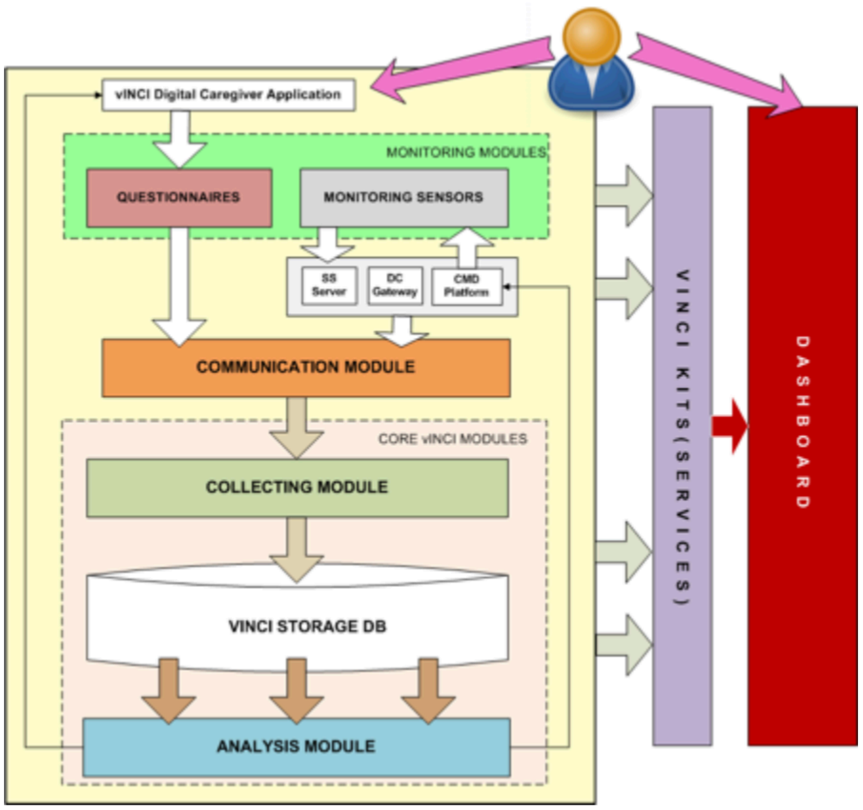


Fig. 4. Architecture behind the vINCI platform.

Conceptually, our project falls within the Internet of Things scope of - more precisely, the vINCI platform can be assimilated to a service exposure solution across a data collection platform (which in the literature is called *Platform as a Service*). There are platforms from which we could have started, such as Amazon AWS IoT[1], Microsoft Azure[2], Google Cloud Platform[3], or IBM Watson Internet of Things[4]. Each one follows a similar architectural principle, including a broker, security engine and identity modules, or modules for managing the status of sensors or connected devices. However, such platforms do not help to create applications (kits in our case) tightly enough to support the processing of data captured under the most diverse conditions.

Which is why, in the project we decided to go to an architecture based on **micro-services**. The architectural extensions for this are shown in Figure 5 below. Hardware devices (such as smartwatches or shoes) are recorded in vINCI (with a *unique ID*). The data sent by the sensors are received either via an API over HTTPS or through the MQTT (messaging queue) protocol by a *gateway*. Since vINCI provides more services / kits, we will assign a list of sensors that the platform receives to each monitored subject, and we will select which services to receive what data. Everything is dynamic, in the sense that a user can decide not to use the shoes for example, just the watch, or she can decide that she does not need the home monitoring kit, etc. In addition, this architecture allows a service adaptability according to

[1] https://aws.amazon.com/iot
[2] https://azure.microsoft.com/en-us/product-categories/iot/
[3] https://cloud.google.com/gcp
[4] https://www.ibm.com/internet-of-things

the communication protocols we use - intelligent shoes, for example, can send data over Bluetooth Low Energy (BLE) to the smartwatrch, from which a single message is sent at a time which contains the concatenation of the information with those provided and the clock, or the shoes can send data over LoRa or other WiFi protocols. The smartwatch, on the other hand, uses broadband transmission technology.

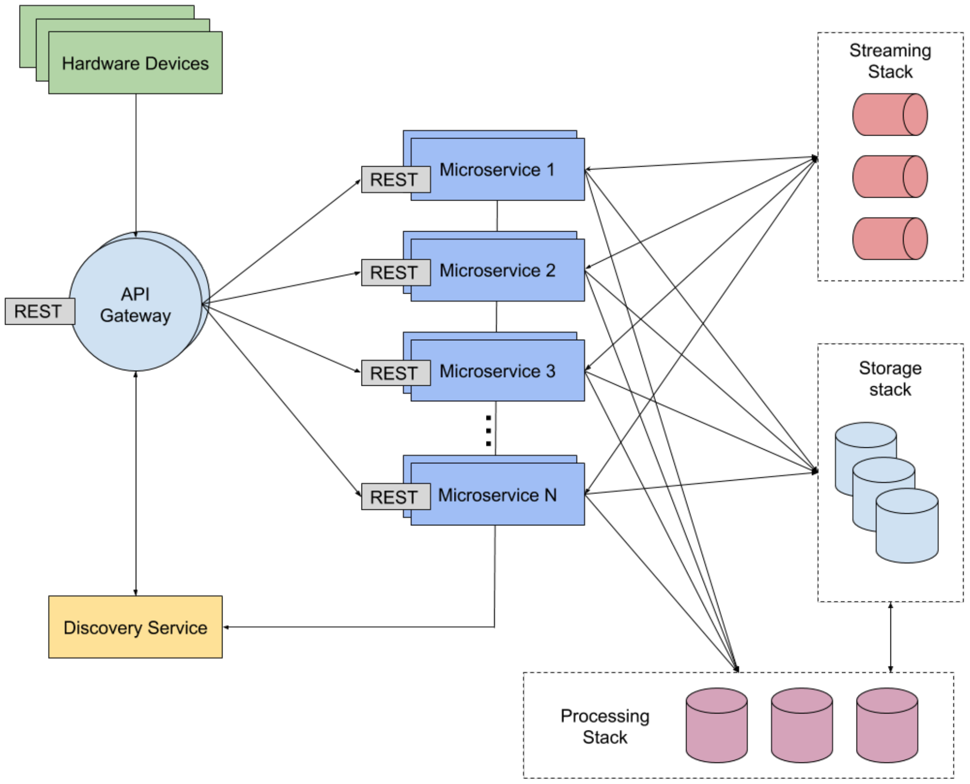


Fig. 5. The micro-services based architecture of the vINCI platform.

Such a micro-services based solution has several advantages: each microservice can be developed independently of each other; microservices are faster and require lower development and operating costs compared to monolithic services; each component can be coupled with its own database, depending on the services developed; a microservice-based architecture provides security through encapsulation, high availability and scalability given by the orchestration component.

In this architecture, *hardware device* refers to the sensors used to collect data sent to the vINCI platform. A network server receives this data and performs a first set of checks (eg, no duplicate data, entries recorded are valid, etc.) and sent to a decoding service (from sensor-specific data formats to internal work format, JSON, vINCI platform). An example of data sent to the gateway and stored in the vINCI database is shown below - a JSON array containing information from multiple devices (eg, shoes and watch together). *lat* and *lon* are the geographic coordinates of the device that sent data, *timestamp* is the time when data is received by the network server (which can also be deployed on a smartphone), *devAddr* and *devEUI* are the device addresses, and *payload* is the information sent by the device.

```
{
 "devices" : [
 {
    "lat" : "45.862",
    "lon" : "26.642",
    "timestamp" : "2018-02-13 T 10:30 UTC",
    "devAddr" : "DD15AF9E",
    "devEUI" : "0004A30B001C1C79",
    "payload" : "00FF00FF00FF"
  },
  {
    "lat" : "45.861",
    "lon" : "26.342",
    "timestamp" : "2018-02-13 T 10:31 UTC",
    "devAddr" : "DD15AF0E",
    "devEUI" : "0004A30B011C1C79",
    "payload" : "FF00FF00FF00FFFF"
  }]
}
```

In particular, this information is interpreted by a decoding service in vINCI, because sensor data can come in any type of owner. To illustrate this, we present the data format as received from CMD THL One Smartwatches:

```
[
  {
    "_id": "5b19c84e807dd234ff805f4f",
    "ei": "352413080006397",
    "si": "9226103000013506",
    "dt": "2018-06-08T00:05:34.590Z",
    "s": "::ffff:109.166.135.51",
    "c":                          "#@H10@#;352413080006397;9226103000013506;862182;2018-06-
08;03:05:33;heart;\u0005;\u0001",
    "y": 2018,
    "m": 6,
    "d": 8,
    "p": "H10"
  }, {
    "_id": "5b1b1496807dd234ff8064df",
    "ei": "352413080006397",
    "si": "9226103000013506",
      "dt": "2018-06-08T23:43:18.150Z",

      "s": "::ffff:109.166.135.133",

      "c":                              "#@H11@#;352413080006397;9226103000013506;862182;2018-06-
  09;02:43:17;Shutdown;3;?;\u0001",

      "y": 2018,

      "m": 6,

      "d": 8,

      "p": "H11"

    }
  ]
```

These data can be of two types: $H02-GPS$ and $H14\WiFi$ (of course, we will have a JSON translation in vINCI for each type). The watch sends the current position in the $H02$ format if it has an active GPS connection, or in the $H14$ format otherwise. Information sent is IMEI, device ID, transmission date, etc.

Over this architecture, a service / kit is a set of microservices that use data collected from different sensors to perform various computations on them, to display them, etc. Each kit includes, among other things, general parameters (e.g., telephone number where an alert is sent when predefined conditions are met), application parameter parameters defined per sensor (eg, a touch of a temperature threshold, if in the future we will use such a sensor, trigger an alert), other application dependencies, or a data view format.

Thus, in vINCI we have three architectural levels: *storage*, *processing*, and *streaming / viewing*. The storage level contains various databases that are used to store various types of information, from sensor data to application dependent information. The processing part contains components that facilitate data processing, and the streaming part contains the tools that use the data and information obtained.

1.3. Mapping vINCI's architecture on partners and technologies

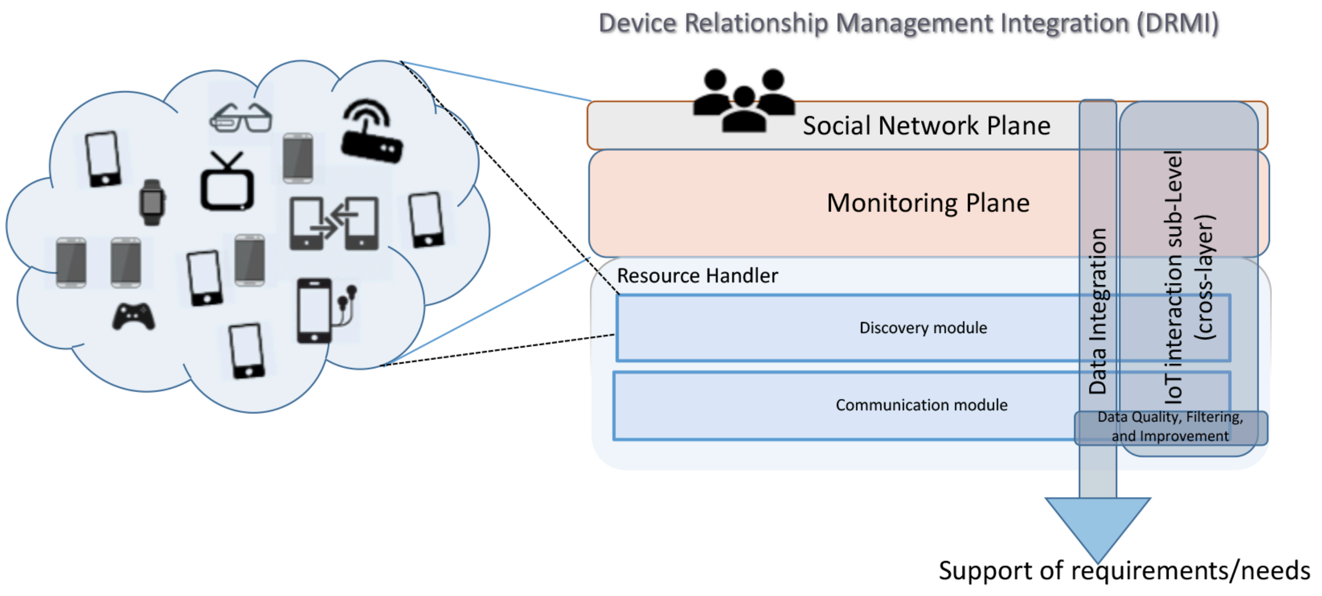

Device Relationship Management Integration (DRMI)
Social Network Plane
Monitoring Plane
Resource Handler
Discovery module
Communication module
Data Integration
IoT interaction sub-Level (cross-layer)
Data Quality, Filtering, and Improvement
Support of requirements/needs

Fig. 6. Device Relationship Management Integration (DRMI).

The proposed architecture will integrate the various devices produced by the partners into the project using Device Relationship Management Integration (DRMI) (see Figure 6). DRMI is a user-centered end-to-end architecture that integrates (1) wearable monitoring systems that include wearable devices and context dependent contexts describing a Subject, and (2) network-dependent content components for linking service devices monitoring.

The user is equipped with wearable devices that monitor a subject in a capable communications infrastructure (e.g., using BLE).

Contrary to existing literature where solutions generally integrate various devices using a “black box”, vINCI architecture provides a close link between the various entities that provide relationships and interactions between components (here, “links”' are explicitly represented in the diagram) resource discovery monitors all available resources in terms of wireless devices connected to the platform (and we also include Edge resources in architecture because an initial processing part will occur in the premises of the Subject, such that sensitive personal data not to leave her home. The communication module allows data integration between IoT devices (*data fusion*) as a result of the interaction between the Discovery, Communication and Monitoring Plane modules.

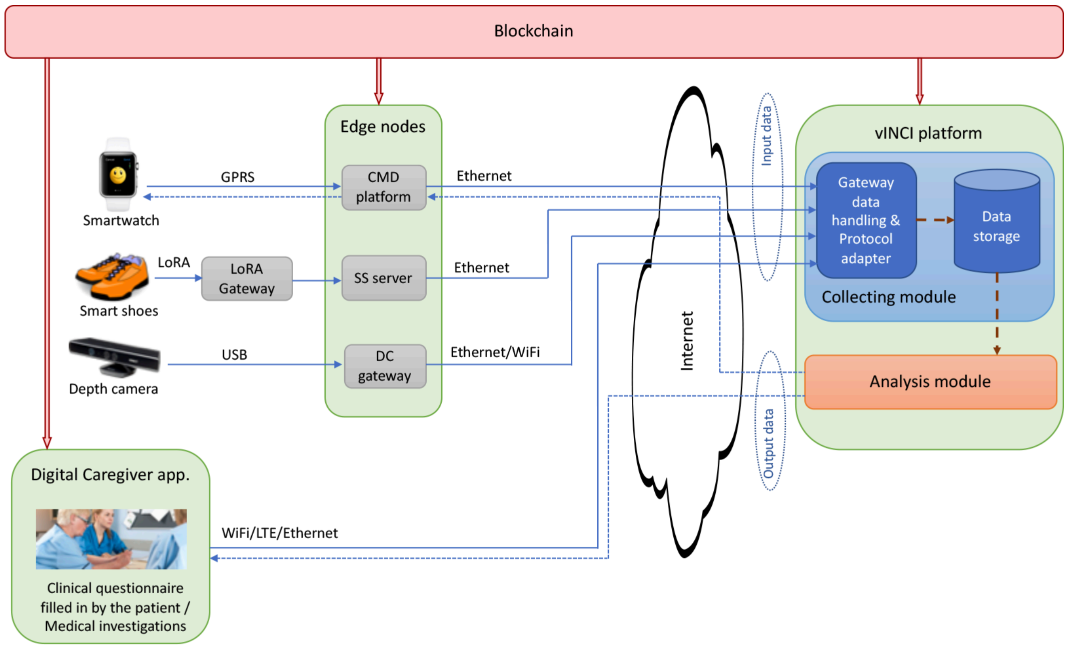


Fig. 7. Communication principles underlying the vINCI platform.

The communication architecture is shown in Figure 7. This assumes that the vINCI platform will be accessible through a public IP address, and connectivity with incoming data providers and data consumers is via the Internet. In the case of sensor data sources, the platform will interact with edge nodes: the CMD platform, which will be intermediate for clock data, where raw data from clocks is processed at a first level, ie. a DC gateway, where Deep chamber data is initially processed to extract features relevant to the vINCI platform. These edge nodes implement the communication and discovery functionality of DRMI.

2. vINCI Kits

2.1. Description

vINCI develops a platform where an older adult is monitored through a set of extensible technologies. Each individual recording is stored and analysed to automatic extraction of features (information) to be used in the detection of deterioration of symptoms associated with old age.

A set of monitoring devices (the *kits*) are being used:

- The CMD watch. This is a product being developed in Romania and sold in several countries in Europe, but in constant improvements in new versions to integrate additional features.

Fig. 8. The smart watch being used in vINCI (photo example).

- Smart shoes: the design started first with the idea of developing the shoes that integrate in the sole a set of pressure sensors. But, business-driven, we now migrated towards the development of smart *insole* that integrate these sensors (it makes sense, as an insole pair can then be used with *any* shoes already belonging to the older adult).
- On-premises camera. Here we first started the development with Kinect cameras (as mentioned in the proposal). Some example of our findings in this process are illustrated below.

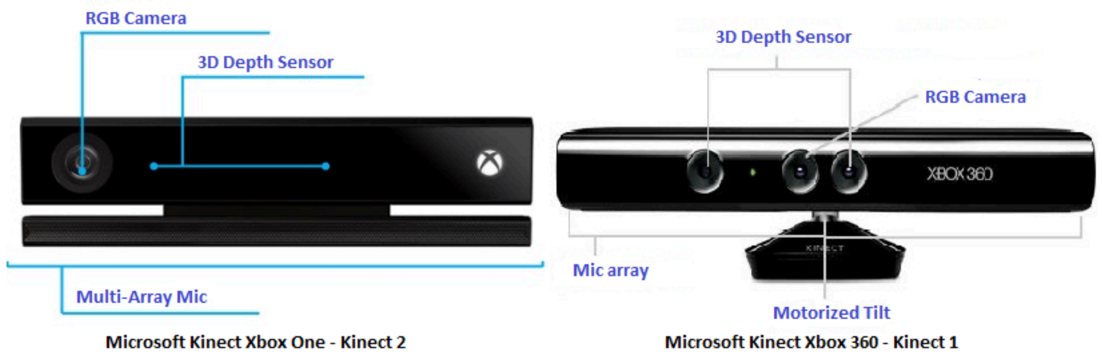


Fig. 9. We used for the development kits both the Microsoft Kinect Xbox One (left) and Microsoft Kinect Xbox 360 (right).

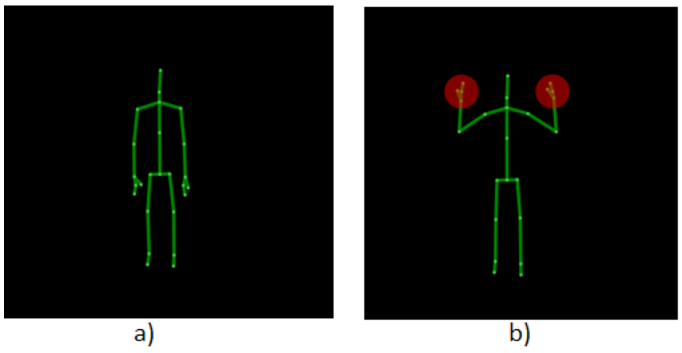
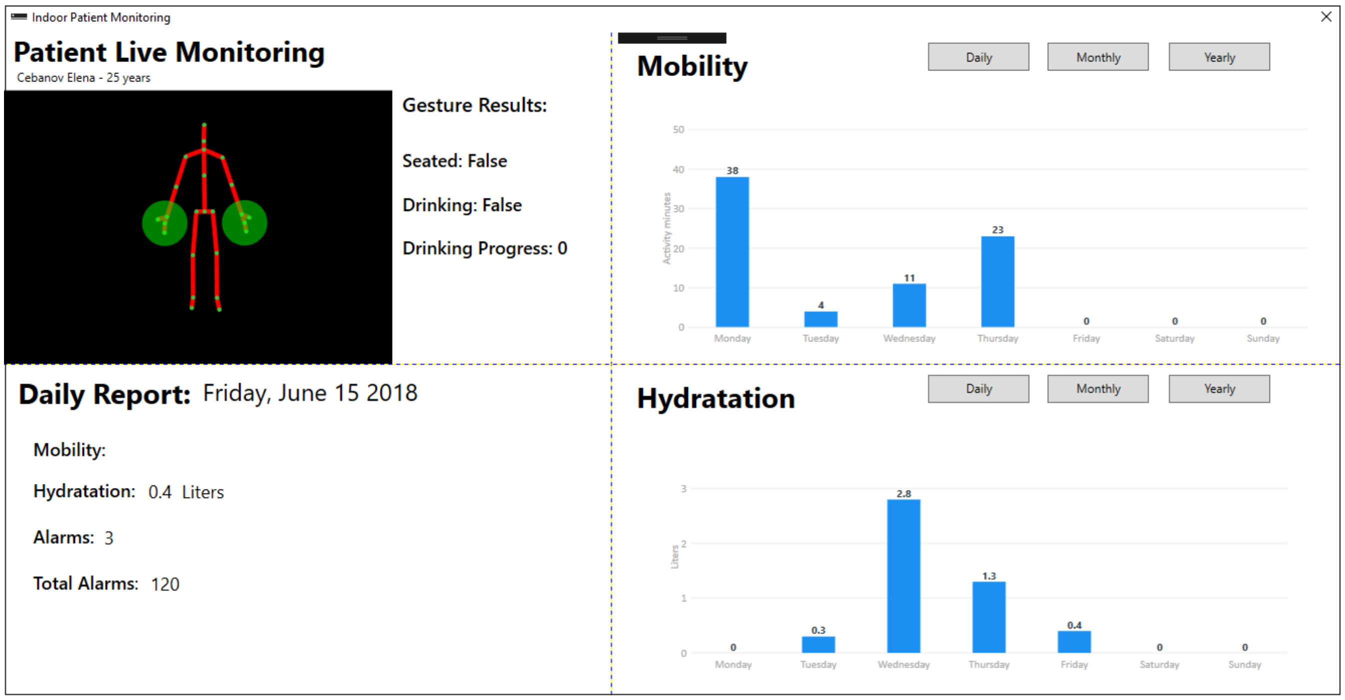


Fig. 10. Tracking the skeleton of a person using a Kinect sensor a) the person stands in front of sensor; b) the person has raised his hands with closed fists.


Indoor Patient Monitoring

Patient Live Monitoring

Cebanov Elena - 25 years

Gesture Results:

Seated: False

Drinking: False

Drinking Progress: 0

Daily Report: Friday, June 15 2018

Mobility:

Hydratation: 0.4 Liters

Alarms: 3

Total Alarms: 120

Fig. 11. Example of a first Kit example (testing with Kinect).

However, Microsoft discontinued its support for Kinect[5], which rendered our development efforts in this direction useless (although technically we can still develop the streaming-based Kit with Kinect, the fact that Microsoft no longer provides updates for Kinect and connected tools, would raise considerable security problems later on).

Because of this, in vINCI we started looking for alternatives. Out of the many technologies surfacing in the area of depth sensors, our analysis show that the Orbbec Persee[6] camera to be one of the most advances and interesting alternatives. This camera, integrating depth and video cameras, is being developed by an SME, Orbbec, founded in 2013 in US.

[5] https://www.theverge.com/circuitbreaker/2018/1/2/16842738/microsoft-kinect-adapter-xbox-one-x-s-discontinued

[6] https://orbbec3d.com/product-persee/

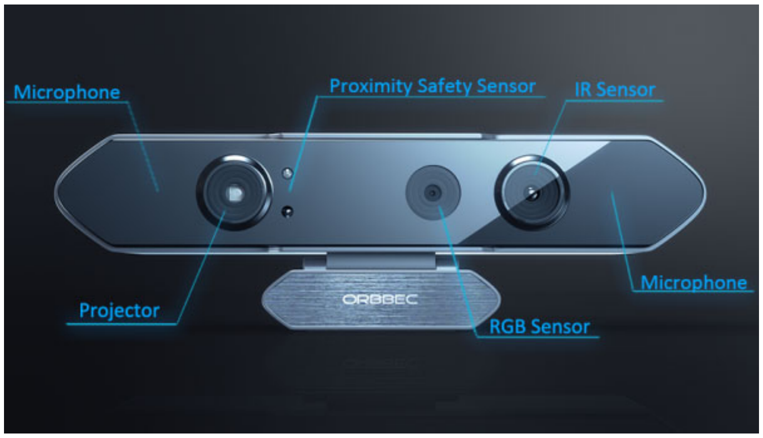


Fig. 11. The Orbbec Persee camera used in vINCI.

As such, we bought development Persee cameras (in Italy and Romania) and currently develop the kit with this technology. In the *clinical pilots*, we are creating dedicated rooms where the camera and monitors are being installed, and where the patients participating in the validation periodically visit and perform a set of monitored exercises.

- The fourth vINCI Kit is the represented by the questionnaires (for QoL, physical activity, social aspects).

At this point, we recall the flow of data in vINCI (already discussed in Deliverable D2.1, based on the analysis of requirements):

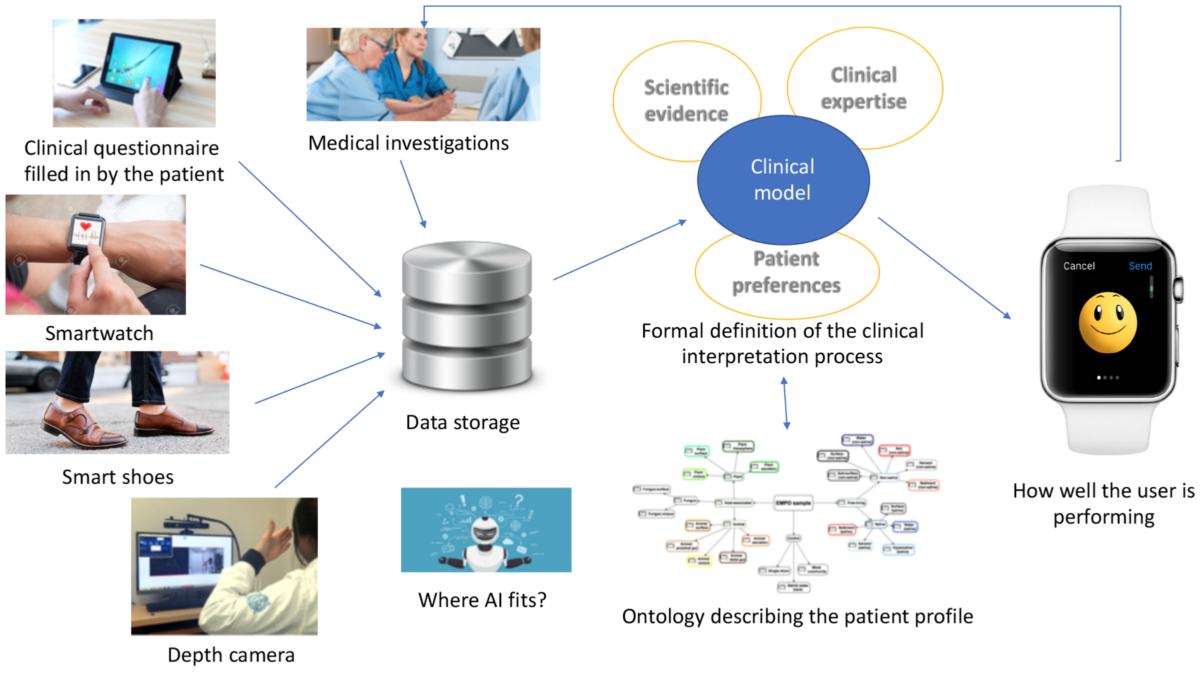

Clinical questionnaire filled in by the patient
Medical investigations
Scientific evidence
Clinical expertise
Clinical model
Patient preferences
Smartwatch
Data storage
Formal definition of the clinical interpretation process
Cancel
Send
Smart shoes
How well the user is performing
Where AI fits?
Ontology describing the patient profile
Depth camera

Fig. 12. The flow of data, starting from the devices, derived into information based on an evidence model, towards possible alerts based on a sensed deterioration in old age conditions.

The way these Kits work is the following (with links to technologies):

Step 1 – the static profile of the patient is detected by means of a set of clinical questionnaires. With the enrolment in vINCI, the subject (or her family) receives a smart tablet which runs the **vINCI Digital Caregiver**. In the business case, the smart tablet can be included with the Kit, or we can use a smart tablet already own by the Subject (we offer choices). The subject registers and creates an account / profile in the vINCI system. A Web-based alternative is also offered for this step (the Digital Caregiver is a mobile-enabled Web application).

Step 2 – Next, the Subject fills in the **WHOQOL questionnaire**, for determining her perceived Quality of Life level. We want to ease the interaction of the Subject with all questionnaires, and for this reason we try to adjust the paper-based forms to an electronic one where some questions could use as input visual helpers (like smiley faces, showing possible alternatives in answers, from very sad, which would be represented as *sad face*, to very happy, represented as *happy face*).

In the clinical pilots, the answers will be filled in under the supervision of a caregiver. For real-world, the Subject will either be helped by some member of the family, or she could enter the answers herself.

The questionnaire will be periodically re-run, for a continuous evaluation of potential deterioration of the old age-associated conditions. If the QoL level is low, then we go to...

Step 3 – From this point on, the Subject wears the nice CMD smartwatch and smart insoles / shoes. Within the house premises, the Subject is monitored periodically with the Orbbec Persee camera.

At first, we train a machine learning model for tracking the person within the house (the purpose is to have a continuous monitoring of activities being done by that person). The idea is to use multiple sensors, like the depth sensors for detecting the posture, and the stereo video camera for face detection (you can imagine that in the house there are several habitants, and we want to be sure who is the tracked person). The Orbbec Persee camera is equipped with a standalone processing unit (running Android), and on this unit all the detection takes place. This means that no personal data goes outside the house, we only send features (like the posture of the person, not its identity of any aspects related to its surroundings) to the Cloud backend.

Next, we monitor, using the same camera, frailty conditions again associated with old age. For this, we develop a separate application for exercise tracking: the Subject is placed in front of the TV/monitor, and a virtual avatar asks her to perform (showing how) a set of exercises. The Subject performs them, and she is being monitored in terms of how accurate and according to instructions the exercises are performed: on premises, we extract the skeleton features, which are sent to the Cloud (as time frames) along with the joints of the avatar (for comparison in terms of how far away from an ideal posture).

Step 3.1 – Periodically, the Subject is notified to fill in the D-VAMS and IPAQ questionnaires (using either the tablet or the browser), to get the perceived levels of physical and social activities.

Step 4 – The smartwatch, smart insoles, questionnaire, all serve to get relevant data features about the Subject. Such data is used to train a model of her profile (matching the data also against the Patient Profile model being constructed in vINCI). However, to interpret and construct such a model, we need clinically-validated facts (what it means to experience a deterioration in the conditions associated with old age, what parameters are affected and how...). Within the Romanian clinical pilot (with the Ana Aslan institute), we use alternative medical tools to get such facts: those will be mapped on the data captured through the **vINCI technology**. Thus, the “smartness” in vINCI will be given from a supervised machine learning algorithmic construction (in the clinical tests), that will next be used to extract set of clusters to advance towards an unsupervised (or reinforced, if possible) construction (in the real-world cases).

2.2. Infrastructure

After discussing the decisions taken towards the construction, we now present considerations linked to what we use for the development of these kits.

First, to cope with quality guarantees (reliability, scalability, technological adaptation, and separation of concerns), we decide to construct our backend using a microservice approach (using Docker containers). The functionality and orchestration over the production serverless virtual infrastructure is, as such, decoupled. On top, Kubernetes[7] (and Rancher[8] for setting up the environment) provides the orchestration service.

The set of considered non-functional requirements (that limits in the end the decisions we took to only a set of potential technologies to employ for the vINCI development) are:

- *Security* (any access to the API is to be verified against a role-based policy). We use here OAuth 2.0.
- (Site) *Reliability* (all microservices run stand-alone but considering that at high peak loads we need the replication of its service). Docker + Kubernetes ensure this layer.
- All commits are atomic / transactional. We will have parallel reads and writes. Consider for example that, periodically, the ML algorithms runs against all data collected within the last time interval, but simultaneously some devices might want to still update or insert new data into the same tables. Concurrent transactions are to be inserted.

Considering past experience of partners in the project, the technological alternatives were:

- For the backend, use either Python (3.x), Node.JS (preferable over vers. 4), or Java Spring. These are the *mature* technologies, working great over a containerized setup.
- For the frontend, there is point in doing anything less than working with React or AngularJS (Typescript).

In the end, we decided and started the development process using JHipster[9]. This is a free and open-source application generator used to quickly develop modern web applications and Microservices using Angular or React (JavaScript library) and the Spring Framework. So it maps well with our development requirements.

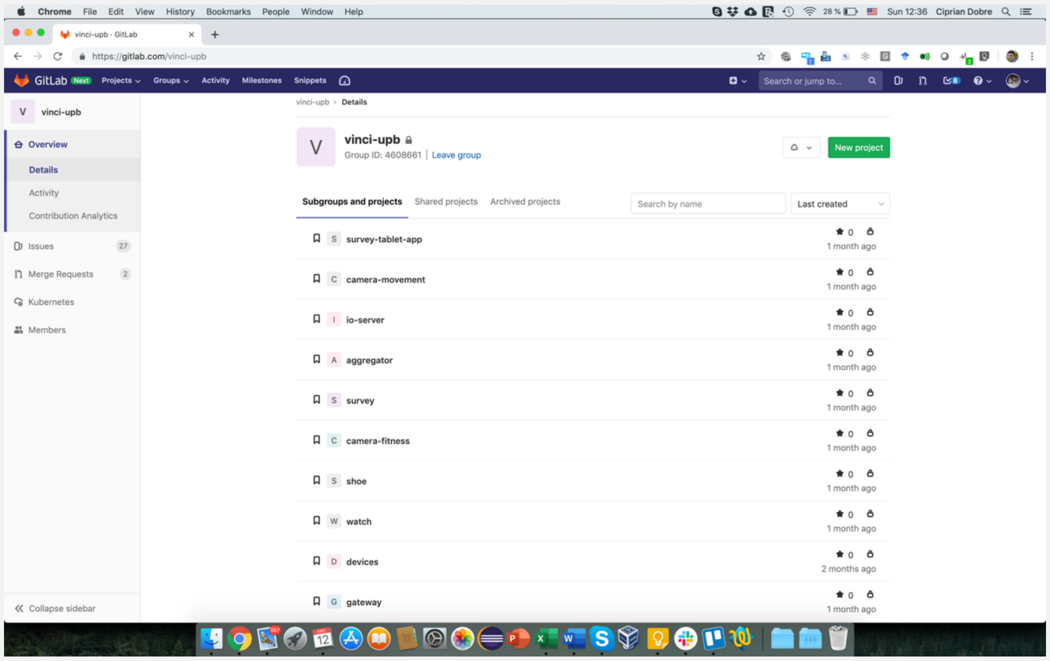
In addition, JHipster provides tools to generate a project with a Java stack on the server side (using Spring Boot) and a responsive Web front-end on the client side (with Angular and Bootstrap). It can also create microservice stack with support for Netflix OSS, Docker and Kubernetes.

The following apply to the source code:

- The source code is accompanied by a Docker file used to build the runtime for each individual vINCI Kit[10].
- The image is built such that to also ensure the bootstrap (initial startup of the service), without requiring external intervention. When this is not possible, we provide sample setups – but they need to be tested against the deployment.
- The preferred interface for connecting all services is Web JSON Rest API. If possible, no service should communicate externally other than through such ways.
- Where it makes sense, microservices should support horizontal scaling, by container replication. A special scalability case is the when web services run as *daemons*, where the classic approach is to start (using *gunicorn* or *wusgi*) multiple workers/threads for each microservice, in each container. For latter ones, we avoid as much as possible the use of *global variables* (as they will have, most likely, a local interpretation alone, on that particular worker). If one needs to store the application state, we use in-memory databases (i.e., Redis).

[7] https://kubernetes.io/
[8] https://rancher.com/kubernetes/
[9] https://www.jhipster.tech/
[10] In constructing the runtime, we consider avoiding debugging interfacing methods of development frameworks (eg, flask run, which starts the development server). The runtime includes software ready for production (served by the web server).

The source code repository can be consulted at: https://gitlab.com/vinci-aal (private repository, to protect the IPR around the project).

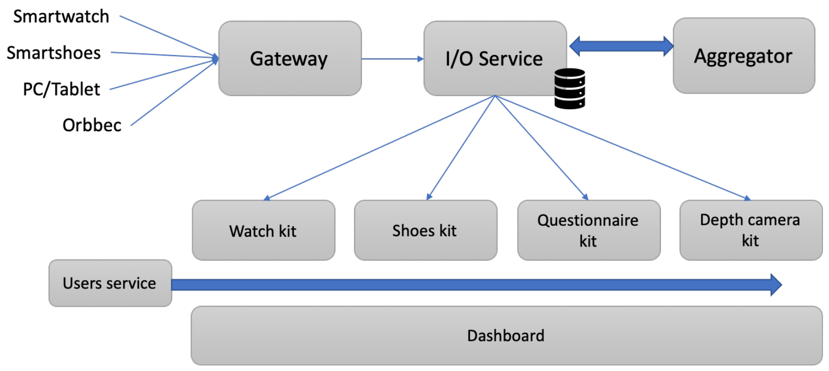


vinci-upb
Group ID: 4608661 | Leave group

Project	Stars	Updated
survey-tablet-app	0	1 month ago
camera-movement	0	1 month ago
io-server	0	1 month ago
aggregator	0	1 month ago
survey	0	1 month ago
camera-fitness	0	1 month ago
shoe	0	1 month ago
watch	0	1 month ago
devices	0	2 months ago
gateway	0	1 month ago

When the service renders impossible (or it is inneficient to) running over Docker, we include additional deployment options (using a keep it simple principle).

The *operation mode* is described below:



We present each component:

1. The **Gateway** is the entry component into the vINCI platform (everything from the gateway back runs into the Cloud, every rectangle in the Figure above being a separated Docker). The Gateway has a public IP and runs a REST API through which the data from devices is received (we use a Push model, devices sending the data).

2. The **I/O Service** is in charge with data persistency. Once the data is received from devices, the I/O Service checks whether or not that particular device was already registered in the platform. Data is being

saved into the personal data vault of the user to which the device belongs (or in behalf of which the data was sent, as the tablet can be shared by multiple user). Due to privacy considerations, no personal data is being receives (and, consequently, stored). Still, we regard highly the right to privacy, so the data belonging to each participant is done with guarantees regarding the verification and authorization (no other user can see or use data coming from other participants, unless explicit consent is being granted).

3. On top, the **Dashboard** offers several interfaces:

- Logging/register the user (with the verification that the email is valid)
- Device registration (after login, a user can register devices belonging to her). This means also the possibility of reading a QR code (in the future, as this is not implemented as of this writing).

Following the registration, the Dashboard calls (REST API) the **user service**.

- Now the user can see a dashboard where she can select from a number of screens (each corresponding to one particular device registered). The user can, at any time, access the function to register/unregister devices.

So, for each device we have a separate Docker service (the **vINCI Kit**). Its role is to mediate the data routes between the database and the Web interface of each particular service.

For each kits, we have:

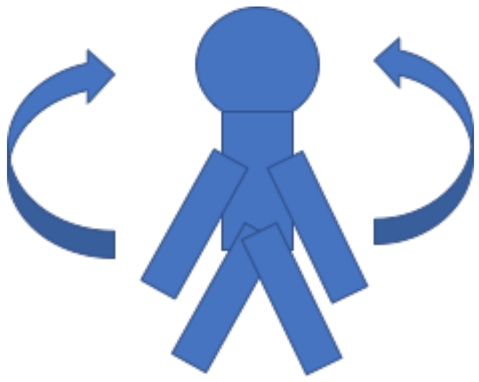
1. Smart watch

The user registers her watch on the platform (using the IMEI or the barcode scanning for the code on the back of the watch). Next, the Subject starts wearing the watch, which in turn periodically sends data to the CMD's servers, which checks and validates them (and applies anonymization) and further sends it (bulk) to the vINCI Gateway (only for the watches registered in vINCI – participants to our studies). The Gateway further calls the I/O service, which in turn authorizes the information and multiplexes it per registered users (remember the CMD server sends bulk data, periodically).

From the Dashboard, the user visually accesses the data such as tracks, number of steps, geofences, alerts. For all these, the service calls the Docker container responsible for retrieving data for the currently logged user.

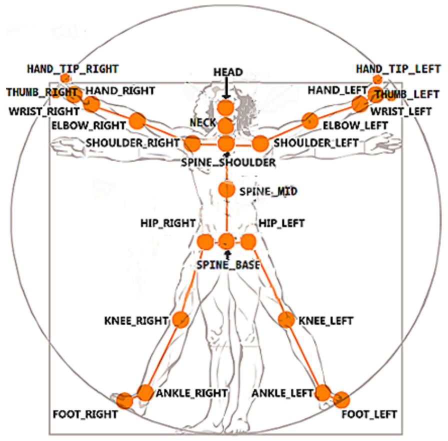
2. Orbbec kit for activity monitoring in front of the TV

Here we assume the setup where an Orbbec camera is connected to a TV. In the clinical pilot, a special room will be dedicated to this purpose. We have an Android app that runs on the Orbbec camera. This app first requires the user to log in – the login verification is done by consulting the Cloud backend. On the future, we investigate the possibility to employ face recognition for login at this step.



The application present on the TV an avatar (a virtual character) in an exercise-like stage. This avatar instructs the user to perform and repeat certain exercise, like raising the arms to a certain level (with visual and verbal instructions provided real-time), or the legs, or moving the posture in a certain position.

In other words, the Subject has to repeat whatever the Avatar instructs her to. Using the Orbbec camera, we scan and extract the skeleton (posture) of the Subject in real-time, while repeating / performing the exercises. The Skeleton extraction is done only locally on the camera. We sample the positions (at time intervals) and we send from the camera into the Cloud only a sequence of skeleton movements (joint positions, adnotated with timestamp), of both the Subject and the Avatar:



In the Cloud, we compare for each sample how far the position of the Subject's joint was from the "perfect" exercise. In theory, when doing this in-time over several times, we could remark (employing an Anomaly Detection algorithm) when the Subject could suffer from a decline in physical conditions.

In the Web interface, the user is represented by lines and dots (the Skeleton), composing a simple animation of how she was doing.

3. The Orrbec Kit for detecting social isolation

This one assumes we have an Orbbec camera mounted somewhere up on the ceiling (or some higher grounds, such that the Camera has a full view of the camera). In the vINCI Pilot, this will still stay on the TV, as we assume this to be enough to have a good impression on what's happening in the room.

On the Orbbec camera, an Android app is running. This app takes the data (depth video data and stereo visual data) and process it locally. We are interesting in detecting how much time the Subject spends in isolation – in front of the TV or somewhere without interacting with others. This involves two things: 1) recognition the user (inside a home there are multiple inhabitants, out of which we monitor and track one particular Subject); and 2) recognition the posture of the Subject (skeleton/pose detection). Also, the sound is a good indicator whether or not the user speaks on the phone for example. We are interesting in: how much activity the user performs (in-home movement) and how much he talks (or sits with no human interactions). The data is captured only local to the camera, and from there periodically we send snapshots (the derived information, as of how much time he was sitting reported to the whole-time interval from the last snapshot) into the Cloud. There, we analyze when the user's performance degrades (anomaly detection, such as when the Subject is depressed the first sign is she tends to self-isolate, spending more time sitting instead of walking, or having less human interactions).

In the development, we had problems in adapting the ML models into the camera, so probably the ultima approach for this kit will be to detect/track the pose on the camera, send the skeleton posture into the Cloud and run there the activity detection algorithm (to detect more accurate whether the

Subject is sitting or standing). We believe this to still be an approach which saves privacy in data, as no personal information about the user is used in this detection.

In the Dashboard, we show an animation of how the human Subject moved at various moments, and also activity graphs.

4. Questionnaires Kit

We assume that the user poses a PC tablet (no matter the other Kits, the tablet or a PC is a must to exist – otherwise, the WHO questionnaire alone cannot be applied and we lose the sense of QoL impression of the user).

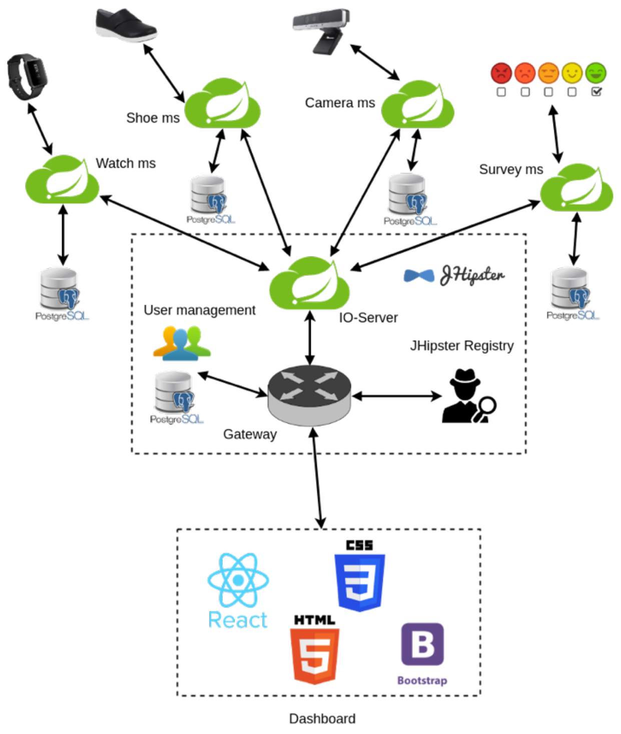
Periodically, the tablet app places alerts, asking the Subject to re-fill in the questionnaire(s). The data is sent to the Gateway, where they are saved and further sent for visualization to the Web-based Dashboard frontend (here we use happy / sad faces to "illustrate" the result).

5. Aggregator

We have another service that, periodically (where the period is a configuration parameter, to be set based on some analytical tests) takes all the data from the database (from the last processing onwards), and extracts a ML model (for anomaly detection). This outputs the probability, given the data collected by kits, that the Subject suffers a clinically-significant deterioration in his condition(s) associated with old age (and how these conditions reflect over her subjective perception of QoL).

The dashboard includes for this another screen (independent of other kits) where we present an evolution in data (as recorded in vINCI) and throw a potential forecasting alarm to ask the Subject to consult the doctor when we detect anomalies in her functions (like symptoms of depressed behavior, solitary life, severe deterioration in capabilities to perform physical activities, etc.).

Putting it all together, this is how the technology combines in the vINCI platform:



3. The vINCI Dashboard - UI

The VINCI UI can be separated into 2 domains: public and private.

Technical Specs for UI

A. Public domain

This refers to any pages that are accessible without requiring any login.

Such pages are: login screen and register screen.

1. Login screen

This includes a form with fields for username and password and buttons for login, register and recover password.

The form has validation for empty username and password fields.

If login is unsuccessful, the user is presented with a corresponding error message.

Upon successful login, the user is redirected to its homepage.

2. Register screen

This includes a form with fields for first name, last name, email, password and password confirmation. The form has validation for required fields, valid email, valid password and password matching.

Upon registering the user is sent an email with a confirmation link and is redirected to the login page with a message to check his email.

The email link will take the user to the login page and automatically send a message to the server to confirm the account. The user will be presented with a confirmation message, as well.

3. Recover password

From the login screen, the user can fill in just the username and request a password recovery. He will be sent an email with a reset password link.

The link will take the user to a password reset page with a form with 2 fields: password and password confirmation. The form has validation, similar to the registration form.

Once the password is reset, the user will be taken to the login screen and shown a confirmation message.

B. Private domain

These are pages that require a login. The pages can be different according to the user type/role.

Available user roles:

- Patient
- Family
- Organization

- System administrator

1. Patient

The **main page** is a dashboard that includes all the users main information: devices status, notification/alerts/events (basic and aggregated). By basic we mean per device and by aggregated we mean information generated by aggregating data from multiple devices (eg. is the patient lazier than normal - info from watch, show and maybe indoor cameras).

The **devices page** will have as main content a table for all the user's devices. Each row will include the device id, UUID, name, description, alerts and maybe last sent data. Each row can be clicked and the device edited or removed. The table has sort by name and UUID and search filtering for name, UUID and description.

Also, there will be an add button to add new devices. This will present the user with a pop-up form where he can define the device name, description, UUID. The form has validation and the user will be presented with error/success messages.

The **account page** will include info about the user: first name, last name, address, phone. This can be editable. The form has validation and the user will be presented with error/success messages.

It will also include a user UUID that the user can share with anybody that has a *Family* account. Using this UUID, a *Family* user can add the user to his account to monitor it.

2. Family

This is can monitor multiple *Patient* users.

The **main page** will be a dashboard that presents info about all the associated users. This can be a grid of user 'cards', where each card includes: user name, user status (status of different evaluations - can be represented with icons or smileys), user alerts/notifications/events.

The **user page** will be a user dashboard with more details. This can be the same dashboard as the *Patient* user sees.

The **account page** will include info about the user: first name, last name, address, phone. This can be editable. The form has validation and the user will be presented with error/success messages.

The family does not manage devices.

3. Organization

This can be an institution/asylum that can monitor its patients.

The **main page** is a dashboard that presents the latest alerts/notification/events.

The **users page** will have as main content a list of the organization's users. Each row will include the user id, UUID, name, description, number of devices (upon hover or click, a list of the user's devices will be shown). Each row can be clicked and the user edited or removed. The table has sort by name and UUID and search filtering for name, UUID and description. There will also be a link/button on each row to open the user's dashboard and the list of devices.

Also, there will be an add button to add new users. This will present the user with a pop-up form where he can define the user name, description. The form has validation and the user will be presented with error/success messages. The form will have a '*generate user account' checkbox for the 2 cases*:

- Patient that is not interested to have an account and no family that wants to monitor him - in this case the email address is optional
- Patient wants to have an account and/or the family wants to monitor him - in this case the email address is required and the user is activated only after he confirms with a link sent on his email

The **devices page** will have as main content a table for all the users' devices. Each row will include the device id, UUID, name, user name, description, alerts and maybe last sent data. Each row can be clicked and the device edited or removed. The table has sort by name, UUID, user name and search filtering for name, UUID, user name and description.

Also, there will be an add button to add new devices. This will present the user with a pop-up form where he can define the device name, description, UUID and choose the associated user. The form has validation and the user will be presented with error/success messages.

The **user dashboard** is the same a the dashboard of a *Patient*.

The **account page** will include info about the organization: name, address, phone. This can be editable. The form has validation and the user will be presented with error/success messages.

4. System administrator

This user only has management capabilities. He shouldn't have and doesn't need to have access to dashboards with device data, aggregated data, etc.

The **organizations page** will have as main content a table for all the organizations. Each row will include the organization id, name, description, number of users and number of devices. Each row can be clicked and the device edited or removed. The table has sort by name and search filtering for name. The table will have links/buttons to show the list of users and devices of the selected organization.

Also, there will be an add button to add new organizations. This will present the user with a pop-up form where he can define the organization name, description and email. The form has validation and the user will be presented with error/success messages.

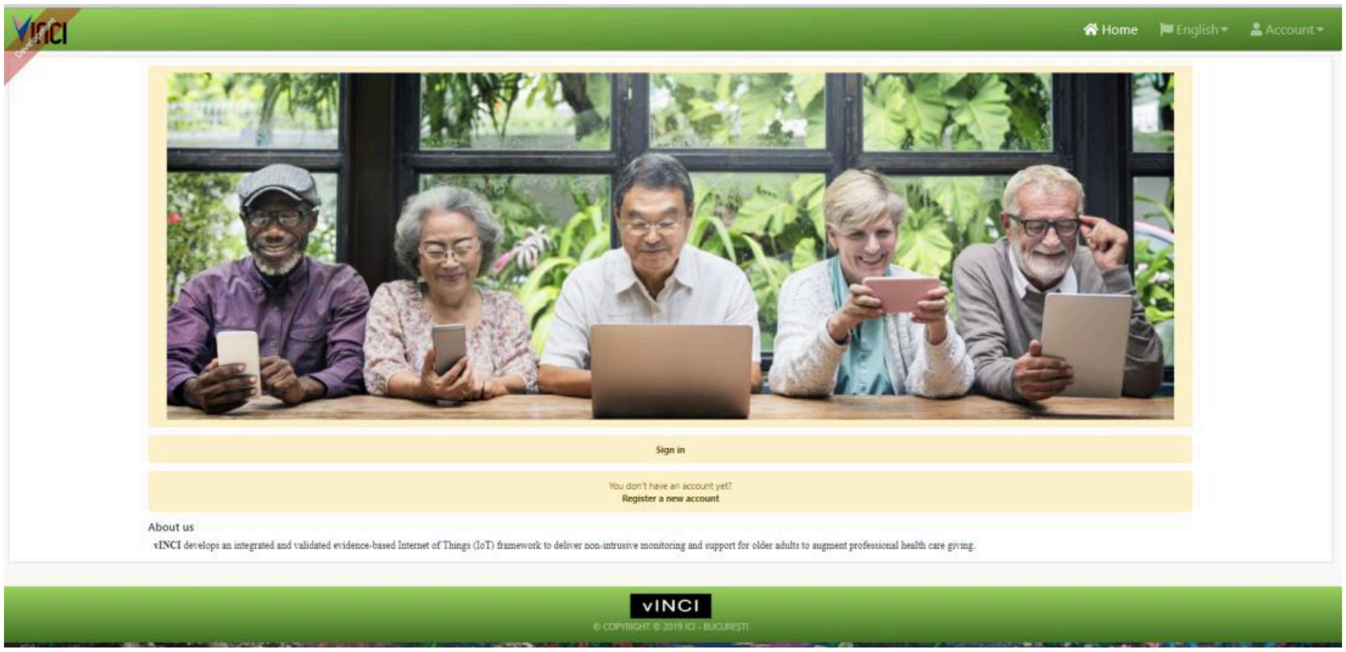
Upon adding a new organization, an email will be sent to that organization to confirm its account.

The **users page** will have as main content a list of all the users. Each row will include the user id, UUID, name, description, number of devices (upon hover or click, a list of the user's devices will be shown), organization name. Each row can be clicked and the user edited or removed. The table has sort by name, UUID and organization name and search filtering for name, UUID, organization name and description. There will also be a link/button on each row to open the list of devices.

Also, there will be an add button to add new users. This will present the admin with a pop-up form where he can define the user name, description, email (optional) and choose the associated organization. The form has validation and the admin will be presented with error/success messages.

The **devices page** will have as main content a table for all the users' devices. Each row will include the device id, UUID, name, user name, organization name, description, alerts and maybe last sent data. Each row can be clicked and the device edited or removed. The table has sort by name, UUID, user name, organization name and search filtering for name, UUID, user name, organization name and description.

Also, there will be an add button to add new devices. This will present the admin with a pop-up form where he can define the device name, description, UUID and choose the associated user. The form has validation and the admin will be presented with error/success messages.


Home
English
Account
Sign in
You don't have an account yet?
Register a new account
vINCI
© COPYRIGHT © 2019 ICI - BUCUREȘTI

About us

vINCI develops an integrated and validated evidence-based Internet of Things (IoT) framework to deliver non-intrusive monitoring and support for older adults to augment professional health care giving.

Fig. 12. Login page.

Following these specs, the interface is now implemented using React. It is part of the Vinci gateway, the sole entrypoint of the Vinci platform.

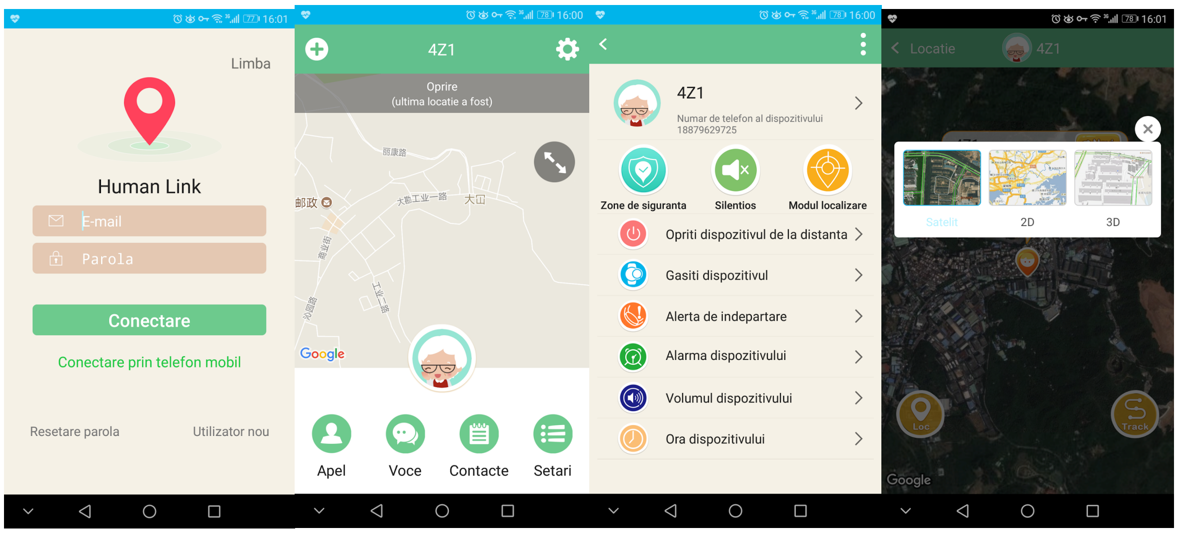


Fig. 13. Examples of UI views for the smartwatch component.

The gateway contains both the interface and the Java Spring microservice that manages the access to the Vinci backend. No matter the interface requests, they will all pass through this gateway and it will redirect them to the corresponding microservice.

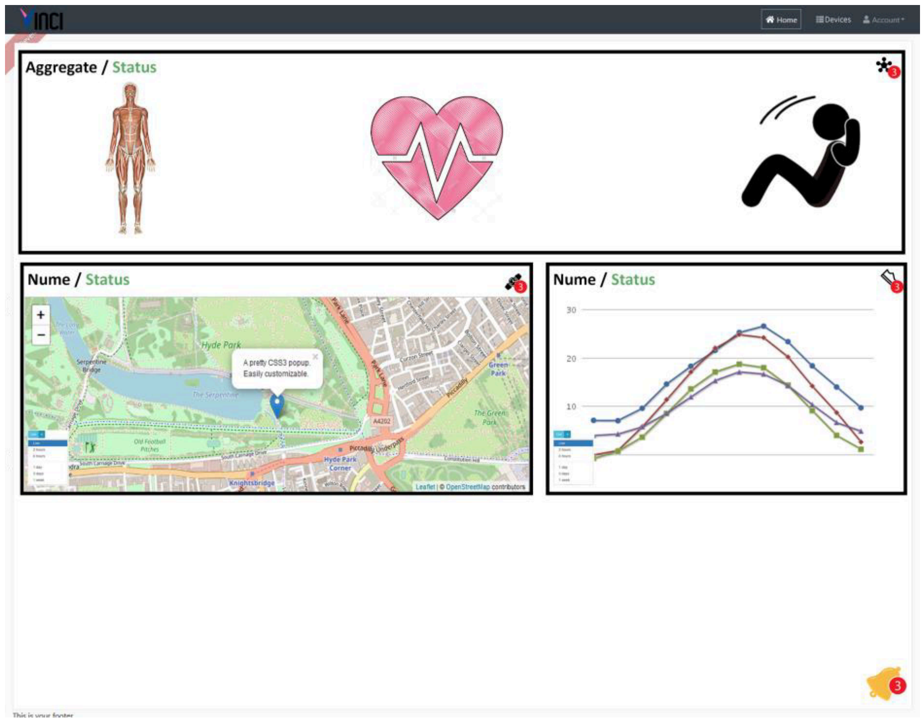


Fig. 14. Partial interface as seen by the Subject in her account.

Everything else is also part of JHipster: interface, gateway, backend.

The interface uses the CI/CD features of GitLab and automatically built itself upon push on the master branch. It also uses the docker registry available from GitLab to automatically bush its image and deploy itself on the corresponding server (staging, production).

In Figure above, we can see several zones:

- *Zone 1*: widget containing aggregated information – the subject is ok, or we instruct her to do something, etc. The content is to be defined currently in the project, being retrieved either from a notification / alerts / user event table or from microservices, remains to be seen. But an initial mock can be made with some hardcoded info. A widget creation for each device type (watch, shoe, camera, etc.). A widget should contain at least:
    - a device name
    - a device type
    - a current status
    - a graphic / map / any other visual elements required. These are done by each person for his microservice. Initially we put a mock, even pictures.
    - a history button - opens a pop-up or a separate page with the history, depending on how much information should be displayed
    - a notifications / alerts - an icon that displays a number of unread alerts ONLY from that device (https://material-ui.com/demos/badges/). Device alerts are stuff like connection problems, operating errors. They will be defined more clearly further.

- *Zone 2*: integrating them into the user's home page (a grid / flex that displays widgets depending on which devices the patient has in mind)

- *Zone 3*: fixed widget / icon in a corner of the alerts / general notifications screen (also a badge, as above). It would include things like 'you have to do more movement', 'abnormal pulse', etc.

- optimized to be responsive

Bibliography

1. TODO