Deliverable 1.4

Second Technology Report

Lead Partner	UNITBV
Authors	
Contributors:	UNITBV - VSRO - UNIPR
Date:	30/12/2019
Revision	1.0
Dissemination Level	Public

Project Acronym:	NOAH
Project full title:	NOAH Not Alone At Home
AAL Project Number:	AAL-2015-2-115
With Support of:	

Summary

Introduction 4
Part 1 - Sensors & Behavioral Analysis Module 5
IOT NOAH Home Sensors Kit 5
Behavioural Analysis 7
Data Analytics services 7
Analytics by example: Noah pilot case study 8
Regression models 8
Sensor Profiles 10
Conclusions of Part 1 14
Part 2 – Cloud infrastructure 16
Factors contributing to a reserved attitude regarding the option for cloud computing 17
Cost reduction using cloud computing 18
1. Infrastructure as a service (IaaS) 19
2. Platform as a service (PaaS) 20
3. Software as a service (SaaS) 23
Proposed conceptual architecture 27
Overall architecture 29
System architecture of data storage in cloud computing environments 30
Storage media for the cloud 30
Storage system architecture in cloud computing from the point of view of NOAH project 32
Prices we considered for proposed solution on some environments 33
Prices for the MS Azure cloud in the case of a heterogeneous environment 33
Conclusion of Part 2 42
Part 3 – NOAH System architecture and technologies 44

Server application – cloud tools version 47
Server application – microservices version 47
System security aspects 47
Conclusion of Part 3 49

Introduction

From a technological point of view, the main outcome of the NOAH project has been the creation and the implementation of a real IoT ecosystem well-devised to provide assistive functions to elderly that live alone. Such an ecosystem is composed of:

- the NOAH home sensors kit
- a WiFi network that provides an internet connection
- a cloud infrastructure to store and manage data
- innovative algorithms to analyze the behaviour of elderly inside their home
- an App to return information and services to caregivers

In this Deliverable, the final version of the NOAH system is described.

The document consists of three parts: the first part is dedicated to the description of the 2nd generation of NOAH home sensors kit and to the developing of innovative algorithms to provide behavioural analysis (these topics was mainly in charge to UNIPR); the second part is dedicated to the developing of cloud infrastructure and caregiver App (mainly in charge to UNITBV and VSRO).

In Part1, after a brief description of the innovation carried out in the design of the 2nd generation of the sensors, the analytics services are introduced. Real-life data from NOAH pilots are analysed to show the insights that the analytics system is able to provide.

Part2 is an exposition of the Cloud Infrastructure: a detailed analysis of the opportunities and constraints related to the use of cloud is proposed.

Part3 is mainly dedicated to the description of the final architecture of the NOAH back-end system (Cloud, Data Base, Services for Apps, etc...).

Part 1 - Sensors & Behavioral Analysis Module

IOT NOAH Home Sensors Kit

The NOAH sensor kit consists of a set of five Wi-Fi connected, batteries supplied, devices that have to install in the user's home. The set includes the following elements:

- **Passive Infra-Red (PIR)** sensors for motion detection, suitable for tracing room occupancy. Such sensors are deployed in user's home by fixing them to a wall in the environment where motion needs to be captured
- **Magnetic contact** sensors, useful for monitoring open/close states of different objects. For example, interactions with doors, drawers and medical cabinets can be easily detected with such sensors.
- **Bed occupancy** sensor, useful in tracing sleeping patterns; detection of presence is achieved by a pressure-sensitive resistive pad, usually placed under the mattress. Such signal is read by the sensor module, attached to the bed frame.
- **Chair occupancy** sensor, to gather information on how much time and when a user sits on a chair/armchair/sofa; physical sensing technology is the same as for the bed occupancy sensor.
- **Toilet presence** sensor, specifically developed to keep track of daily toilet use. The sensing element is an active IR sensor with an IR illuminator and a photo-detector: this setup guarantees ranging capability and can be much more selective for close interactions detection, with respect to a PIR. Indeed, the sensor is fixed in close proximity to the toilet.

At the end of 2017, a very first generation of the NOAH system was tested. It was based on TI-CC3200 SoC (a System on Chip equipped with an ARM Cortex-M4 microprocessor and a IEEE 802.11b/g/n network processor).

The sensors exploit MQTT communication protocol to implement a direct connect to the cloud. An "easy to install" procedure was developed, using WPS (Wi-Fi Protected Setup) configuration protocol, to simplify the installation phase at user's home.

Test activities of first generation of IoT sensors, carried out by UNIPR and MOBILAB, had highlighted a series of technical issues, like as:

- anomalous data transmission
- sensor disconnection from the Wi-Fi network
- abnormal batteries consumption
- non-repeatable behaviour

To solve these problems, a strong effort and several mounts of works was dedicated to produce a second generation of NOAH home sensors kit.

The second generation of IoT sensors is based on a new Wi-Fi SoC Module: TI-CC3220 SoC (an evolution of the CC3200 used for the first generation). Furthermore, a pre-certified module available (with PCB antenna) was adopted. New features for security are embedded, as: support for SSL/TLS and Cyphered File System and Secure OTA (Over-the-Air) updates are enabled.

The sensors have been re-designed to improve the user experience by:

- A thinner plastic case with external access for batteries and polarity reversal protection
- A dedicated PCB designed to improve the reliability
- A multicolour led for information signalling
- More effective buttons available for specific functions
- The use of an external power supply is supported

Power consumption optimization strategy have been implemented to improve the batteries consumption. For example, data transmission policy changed: information are sent at intervals of about an hour exploiting a dedicated Non-Volatile Memory, available to store data before sending. Sensors try to retrieve the current date and time using an external internet accessible NTP server using the SNTP protocol. An automatic reset (“factory state”) procedure is available to restore the functionality of the devices in case of malfunction. Besides, a Super-Capacitors was added to the electronic circuit to ensure a better power stability even in case of peak consumption.

Behavioural Analysis

The Behavioural Analysis (BA) module takes care of extracting meaningful insights from raw data gathered by the five home sensors (installed at user's home and above descripted) and stored in the relative DataBase (DB).

Data stemming from field sensors are ingested into the cloud by a MQTT broker instance. Meanwhile, a Node Red module takes care of parsing such data, inserting them into a MySQL DB for further processing.

The Behavioural Analysis (BA) module is, instead, exposed as a REST web service, serving requests over secure HTTPS connections.

Overall, the BA module performs the following tasks:

- Request parsing. Each request is parsed to extract the type of analysis to perform, along with the associated parameters.
- Data retrieval from the MySQL database
- Data cleaning and pre-processing. Raw data are inspected to remove potential errors and transformed to a richer representation (sensor event matching and filtering), better suited for further processing.
- Processing. The transformed data are exploited to infer relevant information and, possibly, make prediction on future states
- Response serving through HTTPS protocol, using JSON encoding.
- In the following, the analytics services are presented, together with some real case studies from installed NOAH pilots.

Data Analytics services

The primary purpose of all analytics services developed for the NOAH project is to model the usual user's behaviours and to detect new or deviating observations. The following services were implemented:

- *Regression analyses with outliers detection*. The purpose of this kind of model is to explain and interpret data by means of given factors, rather than posing the basis for future predictions. In this sense, the analysis is explanatory and retrospective: it can be used to detect if some effects are truly influencing the data (e.g. longitudinal or abrupt trends) and if some observations cannot be explained properly by the accounted factors (outliers). The factors considered in the NOAH project are:
    - *baseline*: the expected, average quantity, if no other effects are present
    - *abrupt trend*, i.e. presence of a statistically significant deviation in the last 5 days.
    - intermediate period, before the abrupt trend, that allows to account for a past abrupt trend, without raising the baseline too much. This factor is not used for user notifications.
    - *linear trend*, to model long term trends over the whole observation window (30 days).
- *Sensor profiles*. Such curves model the expected user-sensor interaction throughout the day. In other words, they provide a daily insight on the user's habits by modeling how likely it is to see the sensor triggered at each time of the day. Such curves may be used to detect changes in patterns of use, by comparing different periods and testing for statistically significant deviations. Another application is as features to assess similarity between users' patterns.

All models can be automatically applied to each sensor type: the analytics pipeline handles all steps of data retrieval and cleaning. This is particularly important, since spurious or missing activations are likely to occur. Exceptions and errors during model computation are handled carefully to prevent service blocking.

Analytics by example: Noah pilot case study

Regression models

Regression models are usually computed once per day, and the returned information is encoded in a JSON response. In Figure 1, we represent daily outputs from the BA module in a graphical way. In particular, the daily hours spent in bed are studied, considering the previously listed factors.

Each data point in the plot is colour coded, depending on which factor is triggered (magenta for abrupt trends, orange for long term ones) or whether the data point is considered an anomaly (crimson) or a normal observation (light blue). Please note that each point classification is supported by the previous 30 days: therefore, the first point in the graph is the output of the analysis of the preceding days, not shown. It can be noticed that the model spots 3 interesting points: two of them represent recent, abrupt changes in the behavior (last 5 days, with respect to baseline, accounting for the other factors), whereas the third one is labeled as anomaly, since it cannot be modeled well by the selected factors.

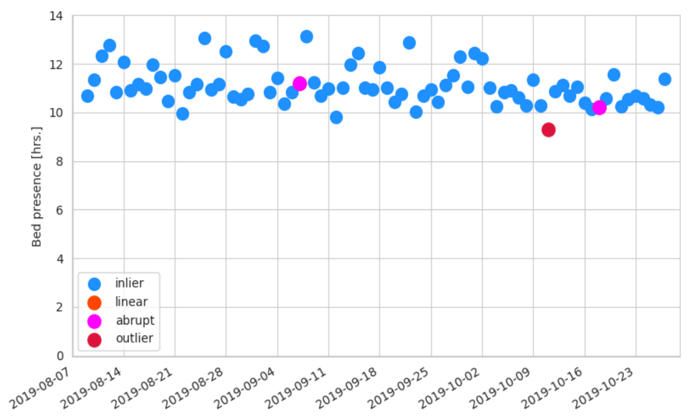


Figure 1 Daily regression model outputs for bed presence

Errore. L'origine riferimento non è stata trovata., on the other hand, shows the output of daily regression modeling of toilet visits: the same consideration apply to the previous case, for interpretation. However, in this case, the quantity being analyzed is no longer a real-valued observed value, but rather a discrete one, namely the number of daily visits. An increasing trend is detected at the figure’s right end.

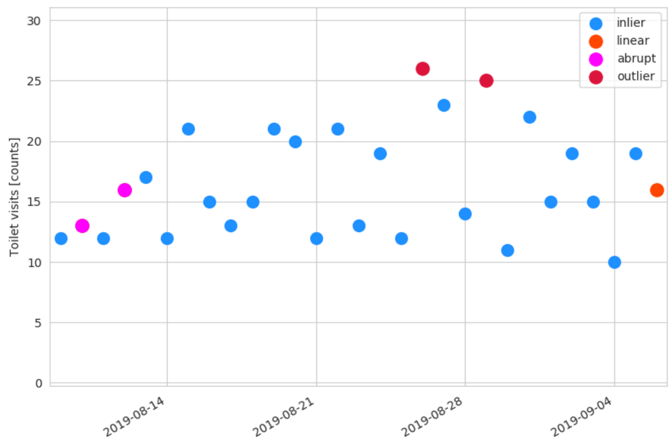


Figure 2 Daily regression model outputs for bed presence

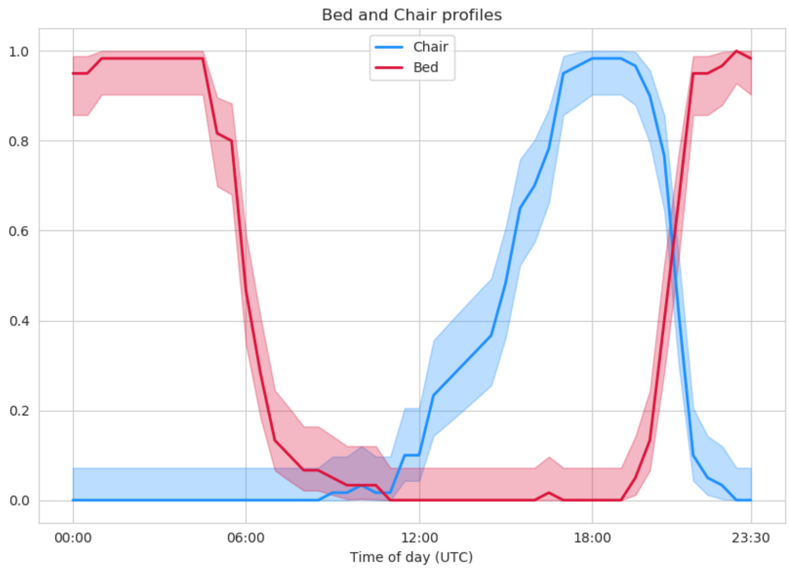


Figure 3 Daily profiles for bed and chair sensors

Sensor Profiles

As mentioned, sensor profiles model the probability of finding the sensor triggered throughout the day. In , an example is given, where a chair and a bed sensor are modeled. In such plot, the solid lines represent the expected probability of being triggered, whereas the shaded area represent the 95% confidence intervals of such estimates.

It can be noticed that the chair and bed sensors are mainly active during the evening and night, respectively. Furthermore, tight confidence interval boundaries reveal a stable, repetitive habit of the user.

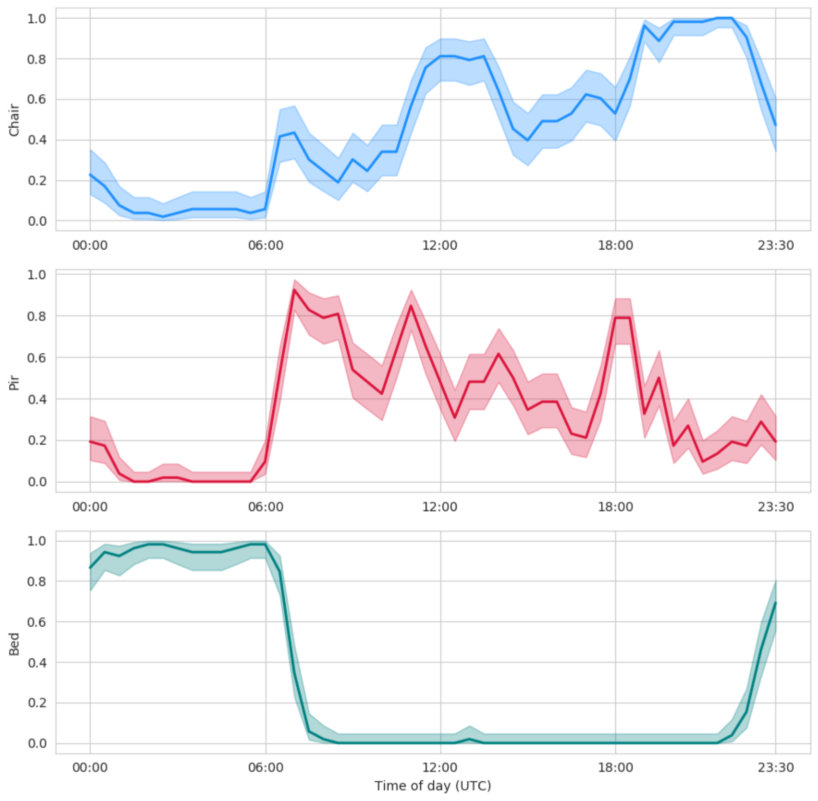


Figure 4 Different sensor profiles for a single resident

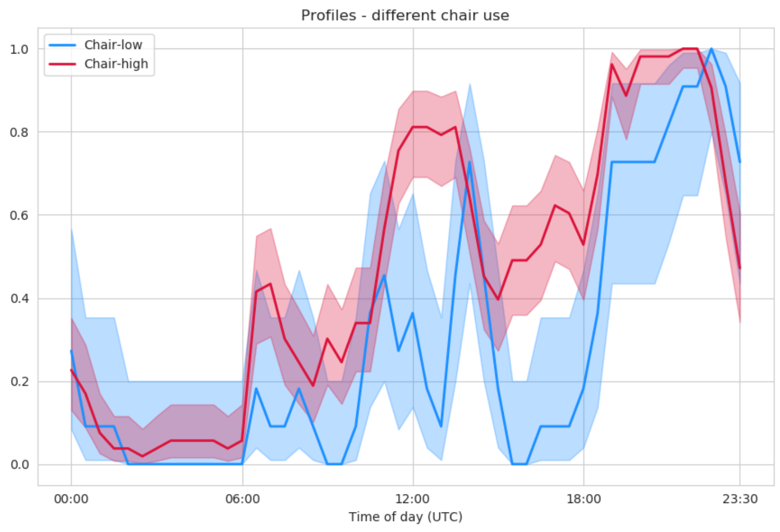


Figure 5 Different chair sensor profiles depending on intensity of use

On the other hand, Figure 4 shows a more complex set of patterns, involving three different sensors, namely the chair, Pir and Bed; it can be noticed that relatively high peaks indicate a more stable behaviour. For example, the Chair sensor is likely to be activated after lunch and during the evening (user reports to be used to fall asleep on the chair before going to bed). At the same time, Pir behaviour is quite expressive; in this pilot the motion sensor is located inside the kitchen: activations for breakfast, lunch and meal preparation are quite apparent. Bed sensor, on the other hand is very stable, indicating a regular sleep routine (user goes to bed between 23:30 UTC and 00:00 UTC).

It is also possible to investigate how different profiles are generated, depending on the intensity of interaction with the sensor. For example, Figure 5 shows different chair use patterns, for moderate and high chair presence throughout the day. In particular, higher chair use is mainly concentrated in the afternoon: such difference, according to the user, is due to recurrent and habitual visits received by her friends. Another possible use of sensor profile is to see whether the sensor activation patterns differ between different time periods.

In this sense, Figure 6 represents an interesting scenario: the user under study exhibited different behaviours throughout the NOAH project: between July and mid-August, he occasionally spent time in bed during the morning, whereas in the next two months he mainly rested just a few hours before lunch. Such difference is significant, as highlighted by non-overlapping confidence intervals. On the other hand, from October on, the user did not spend any time in bed during the day, this highlighting a further behavioural change.

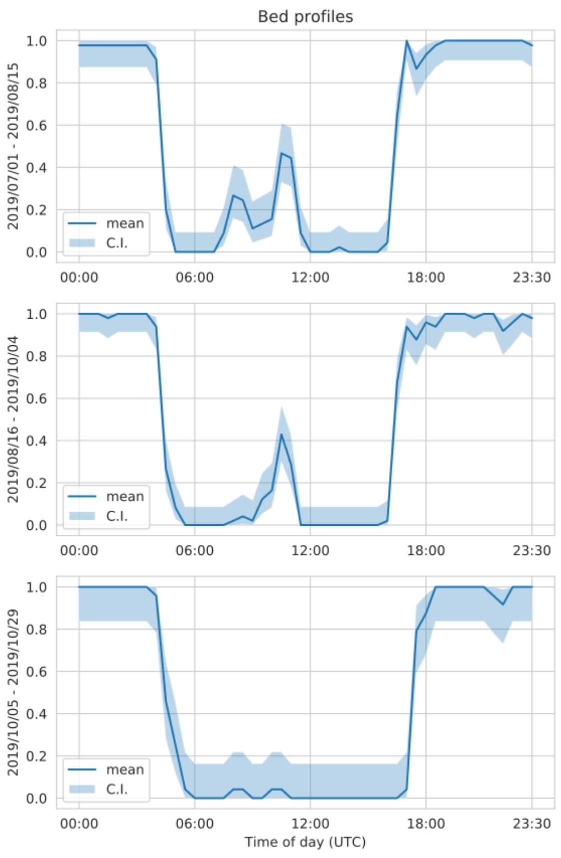


Figure 6 Different sensor profile modes for the bed sensor

Conclusions of Part 1

This document presents the evolution of IoT sensors and the development of the BAM data-analytics services of the NOAH platform. The system gathers data from many different sources, including:

- Bed presence sensor
- Chair presence sensor
- Toilet sensor
- PIR motion sensor
- Magnetic contact (e.g. for medicine cabinet)

A second generation of NOAH home sensors kit was designed, produced and deployed in the pilot installation made in Belgium, Italy and Romania.

Sensors has been redesigned to improve the user experience. Among the various improvements we note:

- the new case now has external access for batteries with polarity reversal protection
- there is a multicolour led for information signalling
- there are some user buttons available for specific functions (like as “hard reset”)
- the power consumption has been optimized (also changing data transmission policy)
- the inclusion of sensors in the network is made using WPS button, as before, and a Python application, to be installed on a PC already connected to the same wi-fi network, is available to receive the notification of inclusion in wi-fi network, with the information about the just connected sensor, in particular, its ID to be used for sensor association to the house/user
- the sensor ID must be associated with the user or house on the server side, to ensure that the information is correctly associated, for data analysis
- every sensor signals its current status and recorded events once every hour (more or less) to save energy. The signalling time is not the same for all the sensors, to try to avoid server congestion
- the communication via MQTT is now protected with SSL, so the CA certificate must be installed in all the sensors during first programming. The username and password login are used for identification and authorization, this information is the same for all sensors

The second generation of NOAH home sensors kit has demonstrated reliability and a sustainable power consumption.

Sensors installed in users' home have provided raw data with continuity. Such data are then used to provide quantitative information about:

- Trend analysis & anomaly detection (*regression analysis*)
- Routine/habits modelling and deviation assessment (*activity curves*)

The services are implemented using cloud-computing frameworks, from data ingestion up to device management and advanced data mining. Services are resilient and can handle common errors which happens in real-world data, such as missing data, incorrect readings and so on. All analytics services are exposed as web-services over secure HTTPS links and require a username and password to be served.

Finally, real-pilot data were analysed to showcase the analytics platform capabilities. Such services are able to produce summarized information (e.g. trend detection, incipient behaviour change in sensor traces), which can be further refined before being presented to the end user.

Furthermore, depending on the type of end-user, such information can be customized: for example, a formal care giver may find useful being notified by the system about incipient trends or on change in specific sensor patterns (e.g. bed/toilet), whereas the elderly person may find more useful to have just a simple indicator that everything is going all right or whether there are some problems with a specific behaviour. Such information can be easily provided through the customized end-user app.

Part 2 – Cloud infrastructure

From a conceptual point of view, at the technical level we can identify the following areas of interest to be considered in order to develop the project:

- cloud computing, with techniques, technologies, methodologies and specificities (compared to a real, physical environment):

  a) available virtualization systems, with their advantages and disadvantages

  b) commonly used hypervisors, with their advantages and disadvantages

  c) system implementation costs of cloud computing

  d) maintenance

  e) aspects to be considered when using a cloud computing environment

  f) aspects to be considered in terms of security

  g) environmental concerns

  h) elasticity and scalability

- the specific of developing applications for the cloud

- remote control of mobile devices (used to transfer data and to run the applications) and their security

Cloud computing gives several options to access the desired services to the users. For this purpose, the user has a graphical user interface that can provide:

1. Self-service on request. Through this feature there are provided on demand IT services, with no intervention from any provider and without the user being required to be familiar with the technologies behind its orders.

2. Permanent and extended access to the network. This feature is based on the provision of IT services anytime and anywhere, by the devices chosen by the user.

3. The facility to use of a group of virtualized resources. The providers supply their customers with scalable services (storing, processing or bandwidth), based on contracts (SLA – service level agreements) according to base system loading and usage

characteristics. Users can change their SLAs without limitations on physical and virtual resources.

4. Rapid elasticity of resources. Elasticity features involve dynamic provisioning of services and users self-serving on demand, with the possibility of amending the contract for services delivery without having to contact the service provider. This is known as scalable provisioning of services, or the ability to offer scalable services. Self-provisioning allows the customers to request a range of services (computing power, storage media, applications, processes, etc) without having to undergo a too cumbersome procedure, thus eliminating many of the expected delays. The elasticity allows users to automatically get extra work space in the cloud or other services.

5. Metering services. IT services can be measured in terms of their use and quantified to establish the costs charged to the users. Payment for services is based on pricing models that consider the load requirements of the basic system and their usage mode, being reflected in the SLA which provides (among others) the QoS - quality of services that are offered. Being flexible, the price models (tariffs) allow, for example, the simple rental of hardware and software resources needed, so the users are not forced to purchase them.

Factors contributing to a reserved attitude regarding the option for cloud computing

1. Maturity – it is not sure whether the technology is ready for implementation at the production level.

2. Standards - are still in the development phase.

3. Security concerns – as many users share the same resources.

4. Interoperability - there are several manufacturers using different techniques and functions to implement cloud computing. The question arises whether the code written by a user can be used regardless of the cloud computing provider, or if a smooth transition can be done from one provider to another.

5. Privacy Control - is different from the traditional IT approach. In most cases, organizations want to retain control of their data. This becomes even more sensitive when using a public cloud.

Because of these issues, we can say that the success factors for adopting a ready to-use cloud computing infrastructure are:

- infrastructure based on open standards
- advanced virtualization and automation
- common processes and components
- advanced security and resilience

Cost reduction using cloud computing

Virtualization + Energetic efficiency + Standardization + Automation = Cost reduction

Cloud computing offers several environmental benefits through:

1. reducing the number of hardware components required to run the applications (compared with the case of internal data centre of the organizations) by reducing the energy required for operation and cooling of the components.

2. consolidation of these systems ((fewer physical components) in remote centres can lead to their better management in groups

3. cloud computing promotes working "from home", with printing remotely, transferring files remotely - helping to reduce the workspace required for offices and their furniture, to reduce the cost of cleaning, and those related to commuting to work, thus reducing carbon emissions.

Costs

The list of possible costs to be taken into account when estimations are made, before all the partners should agree on the final solution:

- Costs caused by the storage environment – they can be very high depending on the type of application (e.g. messaging or data analytics).

- Costs of the networking technology. Transfer into the cloud of a web applications (e.g. messaging or collaborative) should relief strain of own network but could lead to a significant increase in bandwidth requirements for the data transfer.

- Costs related to the backup or archiving of data - savings in terms of backing up data depend on the strategy of their migration into the cloud. The same is true for data archiving. First, it must be determined who is responsible for backups and archives (the client or the service provider). Secondly, it is to consider whether certain data should be backed-up locally.

- Costs due to recovery of data in case of a disaster. Theoretically, the service offered in the cloud has its own possibilities for data recovery, but if this is possible and how much should it cost has to be discussed with the service provider. For example, it must be seen if the cloud service provider has a redundant system, so that in the case of a problematic power supply, service providing could continue from another location. A solution that can be considered could be the ensuring of backups or archives by another provider, specialized in services of this kind.

- Costs due to maintenance of the software that works in the data centre. This is quite hard to evaluate, especially if product licenses are part of a group of licenses covering more applications, or if the app works with other applications. In such situations it is very difficult to assess what the customer is truly consuming.

- Costs due to the platform. Some applications run only in certain specific operating environments, which should be considered in determining the total costs.

- Staff costs – they are charged based on daily costs of operation and application management. It has to be decided whether these costs can be fully transferred to the service provider, or if own dedicated staff is still needed to manage and monitor the services.

One way to estimate costs needs firstly the examination in detail of the expected loads, then the use of a helpful tool for calculating the real costs of these loads in the cloud.

When opting for a cloud computing solution, it has to be seen which of the applications to be transferred into this environment provides a smooth transition and the best return on investment. Assessing the potential for such an application deployment is the key to carrying out such a choice. Some loads are dynamic and unpredictable, which makes sense to spend more to get the required functionality only for a short period of time. For this reason it is important to identify the apps that have this behaviour, because such applications are those that are best suited for such migration. There are on the market a series of tools that can be used to measure the load of an application or a service that is provided via a cloud computing. Most of these products use spreadsheets in which data are introduced and that can provide as output scores on which it can be concluded about the effort to be made by each partner in the NOAH project regarding the investments needed for migrating applications to the cloud computing environment, and the benefits from this approach.

Types of architectures for service provisioning in cloud computing environments

1. Infrastructure as a service (IaaS)

The infrastructure provider grants access to computing infrastructure available as a service. The infrastructure provider manages a group of computing resources and uses virtualization to dynamically resize and transmits the required resources to the userServices to be used based on IaaS:

Virtualized infrastructure

- servers, storage media, networks
- hosted cloud services can be dynamically provisioned

The users (NOAH project partners) would rent processing resources, memory, storage devices and network resources that are provisioned in the cloud.

IaaS can be provisioned either in a public cloud or in a private cloud. In a public cloud, the IaaS user only needs to connect using an authentication mechanism in order to access resources. When the user no longer needs them, their release (”de-provisioning”) is achieved.

This architecture is well suited to use own tools and software to create applications.

Using IaaS does not grant control to a basic infrastructure but control is given over the operating system, storage media, applications and, at some level, over the network components.

Conclusion.

IaaS cloud computing system type will be chosen for the beginning stage when they have to be organized activities for development of applications, to integrate the use of sensors and connections to mobile devices.

2. Platform as a service (PaaS)

Such a platform allows the partners of the NOAH project to develop, test, and deploy web applications on any infrastructure made available by a supplier. In other words, PaaS allows the use of seemingly infinite computing resources within a cloud computing infrastructure. The illusion of infinite resources use is given by the fact that the cloud computing system can be expanded to provide even more resources than would be required.

In this model of service delivery, mediating computing platforms (middleware) and stack solutions are available as a service.

Services to be used in offering applications via PaaS:

- data bases

- messaging
- applications server
- management of the work-processes
- Java execution environment
- Web 2.0 execution environment

Size, configuration and number of hardware components necessary to run the application are not known by those who use it. PaaS providers aim is to create a repeatable and abstract process in order to develop and implement high-quality applications.

Using a PaaS in a public environment will be very different from a traditional platform development and implementation. So:

- Resources are provided in the form of software into the PaaS platform. PaaS environment is hosted so that another actor is granted responsibility for the performance and updating of that software.

- Development and delivery of the services are available in the cloud and not on a particular system.

- Mediating platforms and services are not installed or configured as an integral part of the PaaS platform

- Because PaaS is related to IaaS services, it provides a consistent way (DevOps) of managing and optimizing applications, from development to implementation

When performing a PaaS service delivery, it is recommended to take into account the following:

- Providing developers with a programming environment. For example, there can be provided either open source programming environments, such as Eclipse, or commercial and programming environments such as Visual Studio or Rational platforms.

- Ease of use. PaaS should offer a range of helpful tools to assist programmers such as: various reusable parts created to support the developers, instruments to implement easy-to-use graphic user interfaces, graphical tools for programming, and support for programming environments. It is recommended to implement the iterative development model.

- Providing tools for modelling work processes by which it can also be accomplished the practical application.

- Ensuring availability. The platform chosen must be accessible and available anytime, anywhere.

- Ensuring scalability. The offered platform must include several elements of automation that will produce an elastic medium able to handle the correct loading of the applications it hosts.

- Providing security. To tackle security threats effectively, the platform should consider a number of issues, such as: SQL Injection, Denial of Service, data traffic encryption during the applications implementation. Support has to be available in order to enable access to multiple applications on the same platform, or on other platforms in the cloud, using a single user account, once – “single sign-on”.

- Incorporation, integration and collaboration between existing applications on the platform.

- Independence of the platform from the infrastructure that it is deployed on, allowing later move of applications from one infrastructure to another.

- Providing tools for portability. When moving an application from one platform to another, tools must exist to enable seamless data migration (including tools for data import / export).

- Providing tools needed to create and configure applications so that they can communicate (with the platforms) in order to fulfill their specific requirements.

For PaaS architectures, reusable templates are available to ease the creating of platforms and their configuration. A design template can be defined as a solution that can be reused in other similar situations in a given context and which has proven its correct operation – such a template is identified by a name. The middleware components based on templates are optimized to be able to automatically assemble software.

They are grouped in products and execution environments that allow their automatic assembly in order to dynamically provide different services. For example, IBM offers a series of products, branded as WebSphere that can be used as templates and implemented into platforms.

UTBV owns the licenses for a range of templates and programming environments, under Rational or WebSphere.

Also UTBV can provide Microsoft templates – available under own licenses.

Conclusion

Due to high costs, and increased programming efforts, system type PaaS cloud computing will be chosen for the phase of development, testing and maintenance of applications.

3. Software as a service (SaaS)

SaaS architecture:

Services to be used in offering applications via PaaS:

- Collaboration
- Work processes
- Applications to be developed
- Discovery of data with similarities and their transmission

In the SaaS model, the software provider is responsible for creating, updating and maintaining it, and all the licensing issues. Customers (end users) can rent the software according to usage patterns, or buy an access ticket containing one separate license for each individual who uses the product.

When using this model, the users access just the service itself and not the platform or infrastructure used to support the service. The service is usually accessed as a web application or a web service built into an app – these embedded services are invoked with the help of web API-s (application programming interfaces).

SaaS applications can be divided into two categories: critical services that are absolutely necessary for the good operation of applications and consumer oriented services. Both services are sold on a registration base and payment is only for the consumed resources. However, these two categories seem to have limited usability and more are likely to be added. For example, there may be services providing shared resources or services that partners in the NOAH project do not want in their portfolio and would prefer to be performed by someone from the outside, in order to reduce costs. Another example is the service using external infrastructure for carrying out sporadic activities within the project (such as, distance learning for trained personnel). Apart from consumer oriented services, all other services are profitable only if they are addressed to a large number of users, since there must be covered the costs of infrastructure and the related costs or performance or taxation. All these cost are to be covered by the project in an optimal way.

In order to configure the system to operate normally it is recommended to achieve the best possible concordance between multi-tenancy and virtualization. By multi-tenancy, one instance of the application running on a SaaS platform is used by several partners who lease it. In a multi-tenant data architecture, data and configurations are virtually partitioned to allow each partner to work with an instance. By consolidating IT resources in a single operation, multi-tenancy offers the great advantage of reducing costs; multi-tenancy may not be effective if there are required large resources for storage and processing in applications used by a small number of customers. Another disadvantage of multi-tenancy is that applications running this way require a special programming, in addition to those running in usual environments. Virtualization of servers in SaaS architectures is much more than a simple virtual partitioning of data and configuration, as is the case of usual multi-tenancy.

One of the biggest advantages offered by virtualization is to increase capacity without additional costs of a system, due to dynamic adjustment of the number of logic resources, including storage and databases. But a major problem that can arise is that of virtual servers that cannot be transferred between different virtual environments due to their incompatibility with new virtual environments.

The problem is due to virtualization environments producers, offering virtualization solutions that do not allow interoperability between them, caused by the different approaches of virtualization technology. Therefore, the situation must be avoided by developing an application in a cloud computing environment of and transferring it to another cloud or using applications that must collaborate from different clouds. In such a situation it may be followed the scenario described below: if two SaaS applications should collaborate during operation, one of them using a standard API originated from a manufacturer, and the other one using a different API originated from another manufacturer, they will not work without some adaptations like the introduction of means of communication between the cloud computing media, data reformatting, or even changing the application logic. For the moment, there are no standards for APIs related to data import or export, but there are many other possibilities in this domain.

Synopsis

For the SaaS platforms there are recommended applications that have a service-oriented architecture, in order to enable applications to communicate with each other. This is necessary because the use of services is less expensive than the use of virtual machines. Each software service can act either as a provider or as a recipient of services. SaaS service provider offers its functionality to other applications using public intermediation software components. Recipients use data and functionalities offered by other services. Both service types offer savings in terms of scalability, regarding the implementation and management of SaaS. Web services are usually completely independent functional units, no matter how many resources are needed. In order to avoid that resources used for both service providers and service users become oversized (in case of increased or decreased capacity) there are created web services related to the SaaS applications. This connection affects services independence, and if the web service receives an alert of exceeding the allowed values, connections can be lost to particular services that would be needed in the following.

Choosing a SaaS solution offers the opportunity to reduce costs for the NOAH project applications, as these applications should be used only on request - being no longer necessary to purchase licenses for each device that uses these applications. Cost savings are greater because, according to statistics, most computer systems stay idle – unused about 70% of the time. Using a SaaS environment, applications are effectively used 100% of the time, as the renting of licenses is done only during the time they are used.

Conclusion.

SaaS cloud computing system will be chosen for the final application offering to the users who will access it from their mobile devices.

CONCLUSIONS REGARDING THE ARCHITECTURES FOR SERVICE PROVISIONING IN CLOUD COMPUTING ENVIRONMENTS

IaaS is usually platform independent. The IaaS model has some advantages, such as: reducing costs with hardware resources and human resources, reducing risks in terms of RoI (return of investment), providing computing resources needed without requiring investment in physical infrastructure (no purchase of servers or other equipment is needed for applications running), rapid scalability achieved automatically. Among the disadvantages they can be mentioned: efficiency and productivity depend largely on the capabilities provided by the manufacturer, possible long-term higher costs, reduced security caused by sharing of the same resources. This model is not recommended when CapEx (capital expenditure) is greater than OpEx (operational expenditure).

The PaaS model is particularly addressing the issues of licenses acquisition, ensuring the consumers with the hardware and software infrastructure they need. The model is very suitable for project managers who use Agile methods. With this model of service delivery in the cloud a stack of solutions is offered, with the advantage of a correct versioning. The main problem that can occur when using such a model is that of security caused by centralization.

The SaaS model is offering an application that can be used by end users via a GUI (graphical user interface) that can be accessed also from computers with low resources (”thin clients”). Communication is done via application programming interfaces (APIs) and services have a very low degree of independence. Apps have a client-server architecture, working in a collaborative environment. Nevertheless, choosing such a model avoids significant expenditure with applications development and software resources, reducing investment risks and eliminating costs related to updating apps. Like the other models, centralization requires additional security measures or even the adoption of a new paradigm for the services.

IaaS, PaaS and SaaS architectures could be used together - in such a case, the customer has access to all the resources offered by this combination of models. For example, the SaaS model (offering only the final application to the customer) can be used in combination with PaaS or IaaS, with user access also to the platform and respectively to the infrastructure. From this point of view, SaaS is the most restrictive because nothing more can be used but only the respective application. If PaaS is added, one can develop, test, and deploy the application in the real operation environment, obtaining a more precise control over the application operation, and if IaaS is also added, one can add or remove physical system resources, such as servers, storage media or firewalls.

As discussed above, it is recommended, both in terms of cost and in terms of programming efforts the adoption of a separate solution for each stage of development in NOAH:

1. IaaS will be chosen for the beginning stage when they have to be organized activities for the development of applications, the integration of sensors and the connections to mobile devices.

2. PaaS will be chosen for the phase of development, testing and maintenance of applications.

3. SaaS will be chosen to offer the complete final application to the users who will access it from their mobile devices.

Proposed conceptual architecture

Besides generic service models for cloud computing introduced above, there are other service models on the market, focusing on dedicated users a segment – also such services could be considered in the NOAH project if they look to be more helpful or would have a lower cost:

1. Data services. Own representations can be accomplished, based on data from the available platform. Using such a service assumes to have some data previously stored in the cloud.

2. Testing services. Allowing developers to test their applications in real operating environments. IBM offers comprehensive services combining programming and testing, based on the Rational platform, and known as Cloud Application Management Solutions and Capabilities; Microsoft offers a platform based on Visual Studio, known as Cloud-based Load Testing with Team Foundation Service.

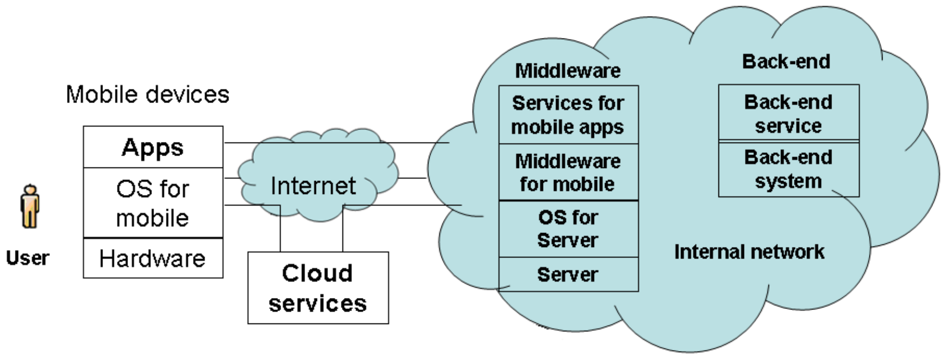
3. Integration Services. This model provides integration of data with different sources and of the applications using these data. Based on cloud platforms and services, an organization can be connected extensively, enable it to pool IT internal systems and applications with one or more external remote IT environments.

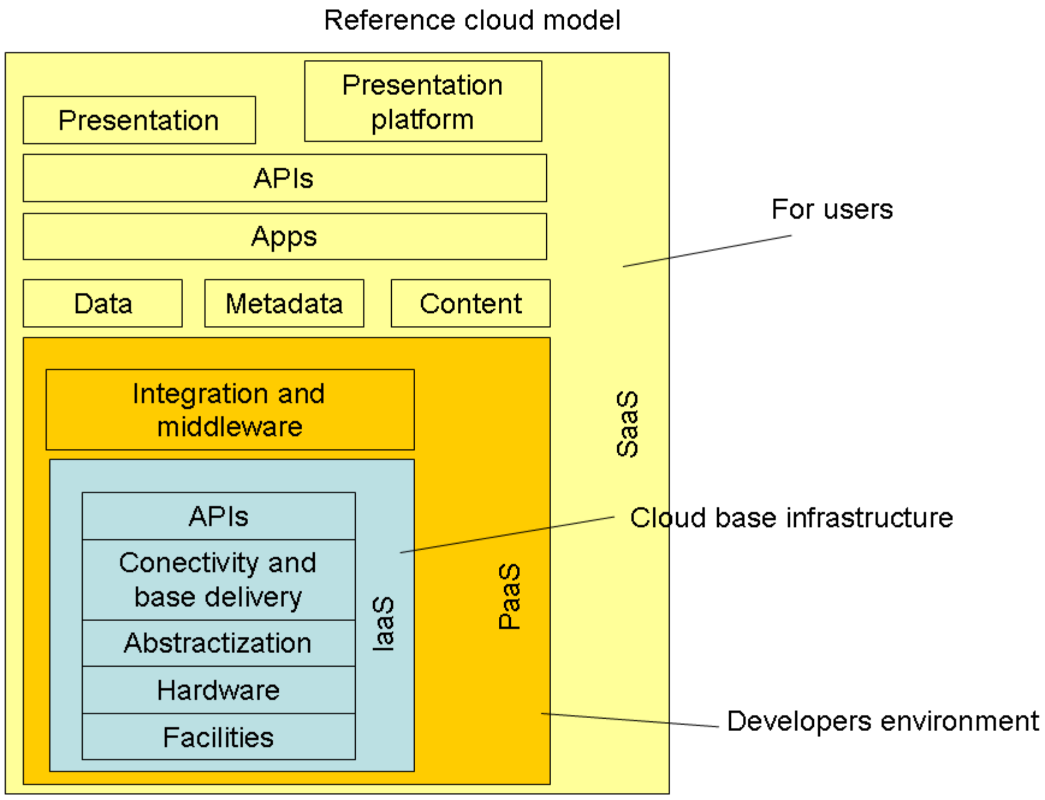
The platform will be implemented as a solution for mobile computing in an cloud computing environment (figure in the next slide). It will allow developers to import and distribute directly applications made in the company making them available to the users through distributing applications by following steps:

1. Developing applications in a dedicated environment for building applications that can be used on mobile devices

2. Transfer applications within the distribution application

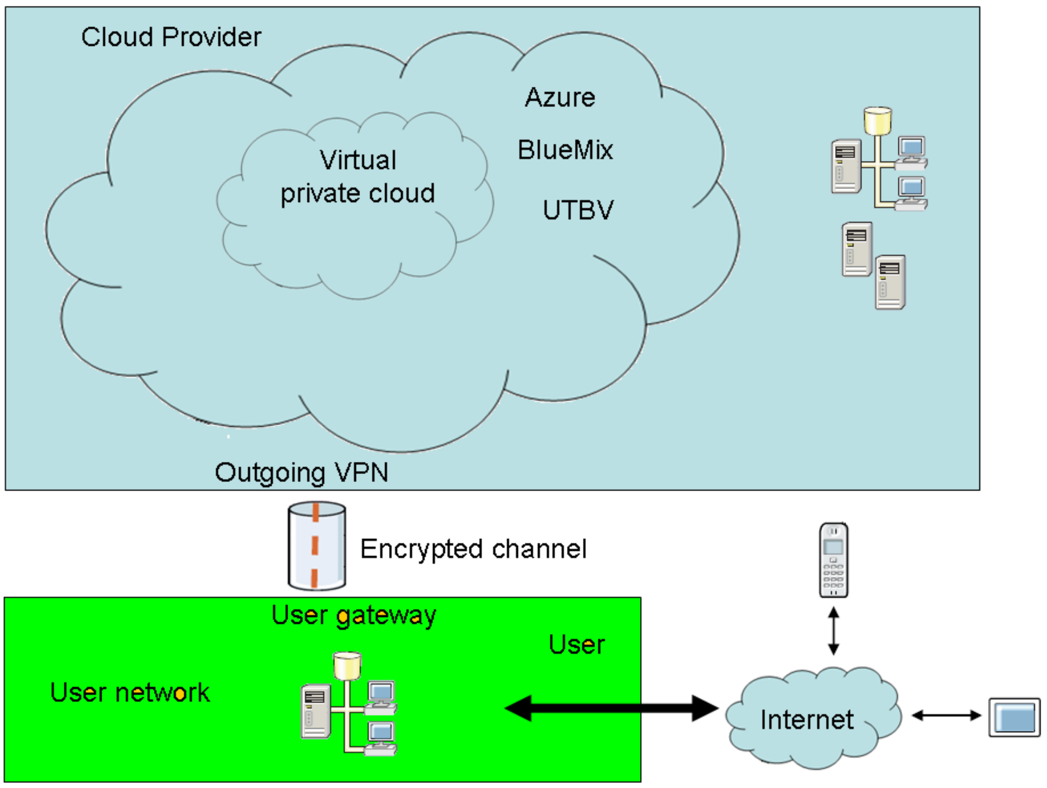
3. Application distribution to the users


Mobile devices
User
Apps
OS for mobile
Hardware
Internet
Cloud services
Middleware
Services for mobile apps
Middleware for mobile
OS for Server
Server
Back-end
Back-end service
Back-end system
Internal network


Reference cloud model
Presentation
Presentation platform
APIs
Apps
Data
Metadata
Content
Integration and middleware
APIs
Conectivity and base delivery
Abstractization
Hardware
Facilities
IaaS
PaaS
SaaS
For users
Cloud base infrastructure
Developers environment

Overall architecture

In conclusion, we propose an overall architecture presented below:



System architecture of data storage in cloud computing environments

The data acquired from the sensors must be kept in secure storage media which can be scaled depending on the load. Load is given by the number of users, the number of sensors that generate data to be acquired and the frequency of data acquisition.

Because data are some of the most important assets of the entire system, they must be kept under maximum security. Therefore, it must be chosen the best solution for storage. Given that running is carried out under a cloud, it could be decided to store data also in a cloud computing environment.

Disks that store data in a cloud computing environment can be connected directly to servers and managed individually through a global distributed file system, or can be part of a network of devices, called Network Attached Storage (NAS) connected directly to a resource group called Switching fabric.

The hierarchy of storage in a cloud computing environment, in terms of a programmer, is the following: a server is made up of a number of processors with multiple cores, a local memory shared coherently and a number of disks attached directly to the server.

Storing in a cloud computing n environment of should be regarded as a service provided to users, regardless of delivery model (public, private or hybrid).

For storage one can use a number of choices, from private storage networks (Storage Area Networks - SAN) to NAS type networks that can be hosted either locally or at other cloud service providers. It is also possible to intend keeping data in other locations, on systems of cloud computing, where the storage is seen as a service that can be paid based on the amount of storage space used. Portable electronic devices can access stored data via the Internet without having to know specific details about the type or location of the storage media used.

The services offered have the ability to allow the storage or retrieval of data based on the behavior of computational processes that are separated from the respective storage service. A storage cloud can be used in combination with a computing cloud, with a private cloud, or as a storage medium for a computing device.

Storage media for the cloud

This is the name commonly used for the storage media implemented in cloud computing, for the provisioning of services offered to users. For example, in order to create a virtual machine it is allocated a certain amount of storage. This storage space is provisioned in the process of creating the virtual machine to support the operating system and execution environment of that instance and it is not provided by a dedicated storage cloud (although it could be also provisioned based on the same infrastructure used for the storage cloud).

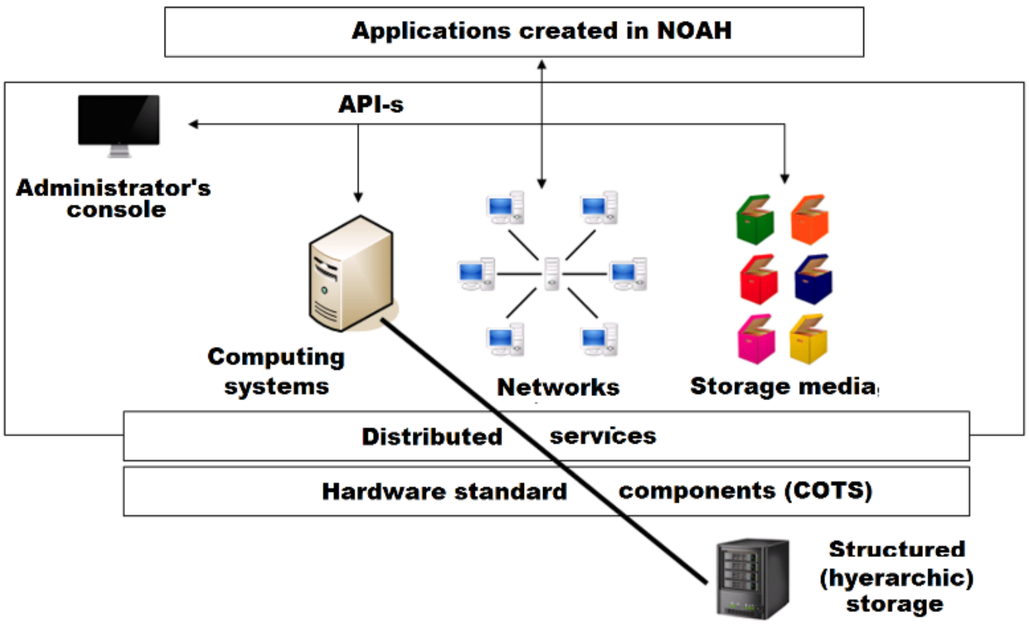
The advantages of storage in a cloud computing environment

Functionality	**Traditional environment**	**Cloud computing environment**
Provisioning of storage	In weeks	In minutes
Continuous access to data	Centralized	Localized – from anywhere, at anytime
Storage capacity	Fixed	Dynamic (elastic)
Reduced administration costs		Down to 50%
Reduced electricity costs		Down to 36%
Increased usage of storage	Up to 50%	Up to 90%

Storage system architecture in cloud computing from the point of view of NOAH project

In the next figure we present the storage system architecture in cloud computing from the point of view of NOAH project:

Cloud storage system architecture proposed for the NOAH project


Applications created in NOAH
API-s
Administrator's console
Computing systems
Networks
Storage media
Distributed services
Hardware standard components (COTS)
Structured (hyerarchic) storage

Prices we considered for proposed solution on some environments

Prices for the MS Azure cloud in the case of a heterogeneous environment

1. Prices for resources

Processor, / RAM capacity / number of disks / hard disk capacity:

A0 Standard: 1 Core, 0,75 GB, 1 disk, 1x500 IOPs, load balancing 12,55 euro/month

A1 Standard: 1 Core, 1,75 GB, 2 disks, 2x500 IOPs, load balancing 37,64 euro/month

A2 Standard: 2 Cores, 3,5 GB, 4 disks, 4x500 IOPs, load balancing 75,29 euro/month

A3 Standard: 4 Cores, 7 GB, 8 disks, 8x500 IOPs, load balancing 150,58 euro/month

A4 Standard: 8 Cores, 14 GB, 16 disks, 16x500 IOPs, load balancing 301,16 euro/month

A5 Standard: 2 Cores, 14 GB, 4 disks, 4x500 IOPs, load balancing 169,40 euro/month

DS1_V2 Standard: 1 Core, 3,5 GB, 2 disks, 3200 IOPs, 7 GB local SSD, load balancing, Premium disk support 42,66 euro/month

DS2_V2 Standard: 2 Cores, 7GB, 4 disks, 6400 IOPs, 14 GB local SSD, load balancing, Premium disk support 85,33 euro/month

DS3_V2 Standard: 4 Cores, 14GB, 8 disks, 12800 IOPs, 28 GB local SSD, load balancing, Premium disk support 170,66 euro/month

DS4_V2 Standard: 8 Cores, 28GB, 16 disks, 25600 IOPs, 56 GB local SSD, load balancing, Premium disk support 341,31 euro/month

DS5_V2 Standard: 16 Cores, 56GB, 32 disks, 51200 IOPs, 112 GB local SSD, load balancing, Premium disk support 682,00 euro/month

DS11_V2 Standard: 2 Cores, 14GB, 4 disks, 6400 IOPs, 28 GB local SSD, load balancing, Premium disk support 119,21 euro/month

DS12_V2 Standard: 4 Cores, 28GB, 8 disks, 12800 IOPs, 56 GB local SSD, load balancing, Premium disk support 237,79 euro/month

DS13_V2 Standard: 8 Cores, 56GB, 16 disks, 25600 IOPs, 112 GB local SSD, load balancing, Premium disk support 476,21 euro/month

DS14_V2 Standard: 16 Cores, 112GB, 32 disks, 50000 IOPs, 224 GB local SSD, load balancing, Premium disk support 952,42 euro/month

DS15_V2 Standard: 20 Cores, 140GB, 40 disks, 62500 IOPs, 280 GB local SSD, load balancing, Premium disk support 1190,21 euro/month

DS1 Standard: 1 Core, 3,5GB, 2 disks, 3200 IOPs, 7 GB local SSD, load balancing, Premium disk support 52,70 euro/month

DS2 Standard: 2 Cores, 7GB, 4 disks, 6400 IOPs, 14 GB local SSD, load balancing, Premium disk support 105,41 euro/month

DS3 Standard: 4 Cores, 14GB, 8 disks, 12800 IOPs, 28 GB local SSD, load balancing, Premium disk support 210,81 euro/month

DS4 Standard: 8 Cores, 28GB, 16 disks, 25600 IOPs, 56 GB local SSD, load balancing, Premium disk support 421,62 euro/month

DS11 Standard: 2 Cores, 14GB, 4 disks, 6400 IOPs, 28 GB local SSD, load balancing, Premium disk support 140,54 euro/month

DS12 Standard: 4 Cores, 28GB, 8 disks, 12800 IOPs, 56 GB local SSD, load balancing, Premium disk support 281,71 euro/month

DS13 Standard: 8 Cores, 56GB, 16 disks, 25600 IOPs, 112 GB local SSD, load balancing, Premium disk support 562,79 euro/month

DS14 Standard: 16 Cores, 112GB, 32 disks, 50000 IOPs, 224 GB local SSD, load balancing, Premium disk support 1109,90 euro/month

D1_V2 Standard: 1 Core, 3,5GB, 2 disks, 2x500 IOPs, 50 GB local SSD, load balancing, 42,66 euro/month

D2_V2 Standard: 2 Cores, 7GB, 4 disks, 4x500 IOPs, 100 GB local SSD, load balancing, 85,33 euro/month

D3_V2 Standard: 4 Cores, 14GB, 8 disks, 8x500 IOPs, 200 GB local SSD, load balancing, 170,66 euro/month

2. Prices for the data bases

Data bases are used together with the other cloud services, thus the cheaper combinations are:

SQL data base

Location: West Europe

Type: Single

Price level: Basic

Performance level: B, 5 DTUs, 2 GB of storage for each data base

Price: 0,0057 euro/hour

4,20 euro/month for 744 hours of operation

Cloud services

Location: West Europe

Type: A0, 1 core 0.75 GB RAM 20 GB HDD 0.017 euro/hour

Price: 12,55 euro/month

Total price: SQL+ Cloud services 16,75 euro/month

The next combination from the price point of view:

Location: West Europe

Type: A1, 1 core 1.75 GB RAM 40 GB HDD 0.067 euro/hour

Price: 50,19 euro/month

Total price: SQL+ Cloud services 54,40 euro/month

3. Prices for Azure IoT Hub

Gratuity for: 500 devices 8000 messages/day 0 euro/month

4. Prices for Stream Analytics

Processed data: 1 GB 0,01 euro/month

Streaming units: 1 x 744 hours / month x 0,026 unitsi / hour = 19,45 euro / month

5. Prices for bandwidth

The first 5 GB / month of data transfer are for free.

For 6 GB / month de data transfer 0,07 euro / month

For 7 GB / month de data transfer 0,15 euro / month

For 8 GB / month de data transfer 0,22 euro / month

6. Prices for IP addresses

Instance-level IP Addresses: 0 euro / month

Load Balanced IP Addresses: 0 euro / month

Reserved IP Addresses: 0 euro / month

IP Address Remaps: 0 euro / month

7. Prices for Gateway

Free if no instancing needed

Gateway: 1 instance x 744 hours / month = 17,57 euro / month

1 GB of processed data: 0,01 euro / month

- first 5 GB are for free

8. Prices for VPN

Inbound data transfer Inter-VNET is for free.

VPN Gateway: 0,03 euro for 1 hour / month

Outbound data transfer Inter-VNET: first 5 GB are for free, then each extra GB cost 0,07 euro / month

Total price VPN for 1 hour gateway and 6 GB data: 0,10 euro / month

9. Prices for DNS

Hosted DNS zones: 0 euro / month

For each hosted DNS: 0,42 euro / month

DNS Queries (millions): 0 euro / month

For each DNS Querry: 0,34 euro / month

10. Prices for storage

Location: West Europe

Storage type: Blob

Price level: Standard – Blob Storage Account

Redundancy: LRS

Access level: COOL

Capacity: 1 GB

Price: 0,01 euro / month

For each 10000 de operations: 0,084 euro / month

Data extraction: for each GB 0,01 euro / month

Data introduction: 0,01 euro / month

Total price: 0,02 euro / month

The price is identical for General Purpose Storage Account

The price este identicale for the HOT access level

Location: West Europe

Storage type: Page Blob and Disk

Price level: Basic

Redundancy: LRS

Capacity: each GB costa 0,04 euro / month

Each 100000 transactions cost 0,01 euro / month

Total price: 0,05 euro / month

11. Prices for Backup

Redundancy: LRS

An instance of 1 GB costs 4,24 euro / month

For each GB there are added 0,02 euro / month

12. Prices for Web cu Mobile – Serviciul App Service

Creation of Web & Mobile applications for any platform and any device.

Location: West Europe

Level: Basic

Instance dimensions: B1, 1 core 1.75 GB RAM 10 GB HDD 0.063 euro / hour

Price: 1 instance x 744 hours = 47,06 euro / month

SSL conexions: gratuity for 1 connexion / month

Each extra connexion costs:

2 connexions: 7,59 euro / month

3 connexions: 15,18 euro / month

Total price for 3 connexions: 109,29 euro / month

13. Prices for Web & Mobile – API Management

Publishing API-s for programamers, partners and employees – in a secure and scalable way

Location: West Europe

Price level: Developer

1 unit x 1 day = 1,33 euro / month

Location: West Europe

Price level: Standard

1 unit x 1 day = 19,02 euro / month

Location: West Europe

Price level: Premium

1 unit x 1 day = 77,50 euro / month

14. Prices for programming instruments – Visual Studio Team Services

Services for teams sharing code, progress monitoring and deployment

Version control, continuous integration etc

Gratuity for first 5 users

For 6 users: 5,06 euro / month

Gratuity for stakeholders

Gratuity for the subscribers of MSDN platforms

14.a. Extensions for the users

For each Test Manager: 43,85 euro / month

For each Package Management: gratuity for u[to 5 users.

Each extra user costs 3,37 euro / month

14.b. Prices for pipelining

For build and release pipelines:

Hosted pipelines: 33,73 euro / month each

Private pipelines: 1 fore free. Any extra one costs 12,65 euro / month

Gratuity for Cloud-based load testing

15. Prices for programming instruments – HockeyApp

Deploys mobile applications, collects the feedback and reports error messages, monitors the usage.

Gratuity for 2 applications with one owner.

For 15 applications with 3 owners: 25,30 euro / month

For 45 applications with 9 owners: 50,60 euro / month

16. Prices for monitoring and management – Application Insights

Detects, sorts and diagnoses problems of the web applications and services.

Gratuity for 1 GB / month.

1,94 euro / month for each extrra GB.

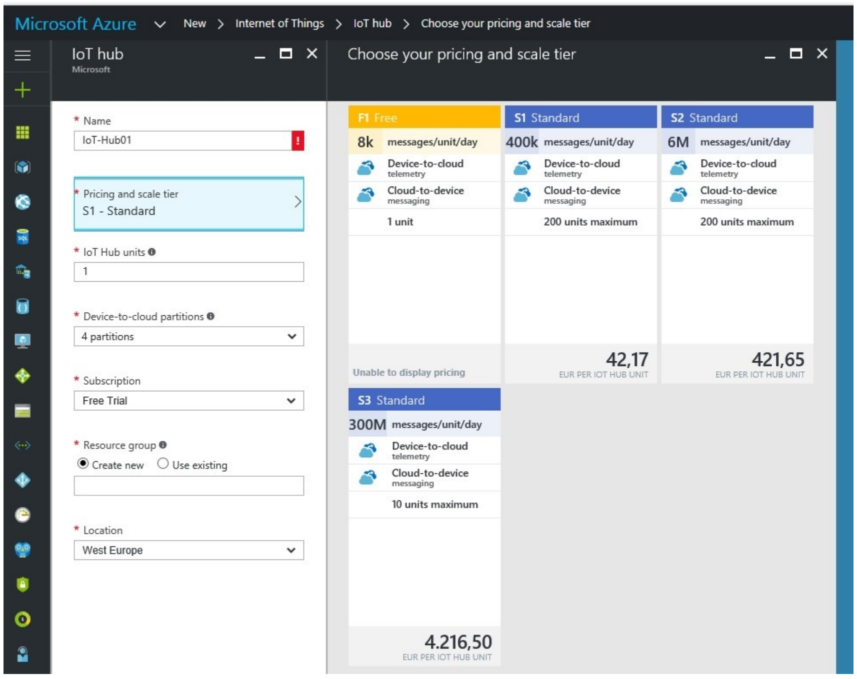
17. Multi-step Web Test

Each test costs 8,43 euro / month

Microsoft Azure IoT cost

Free for 8000 de messages / per device / pe day.

For more messages see the next screenshot.


Microsoft Azure
New > Internet of Things > IoT hub > Choose your pricing and scale tier

IoT hub
Microsoft

* Name: IoT-Hub01

* Pricing and scale tier: S1 - Standard

* IoT Hub units: 1

* Device-to-cloud partitions: 4 partitions

* Subscription: Free Trial

* Resource group: [✓ Create new] Use existing

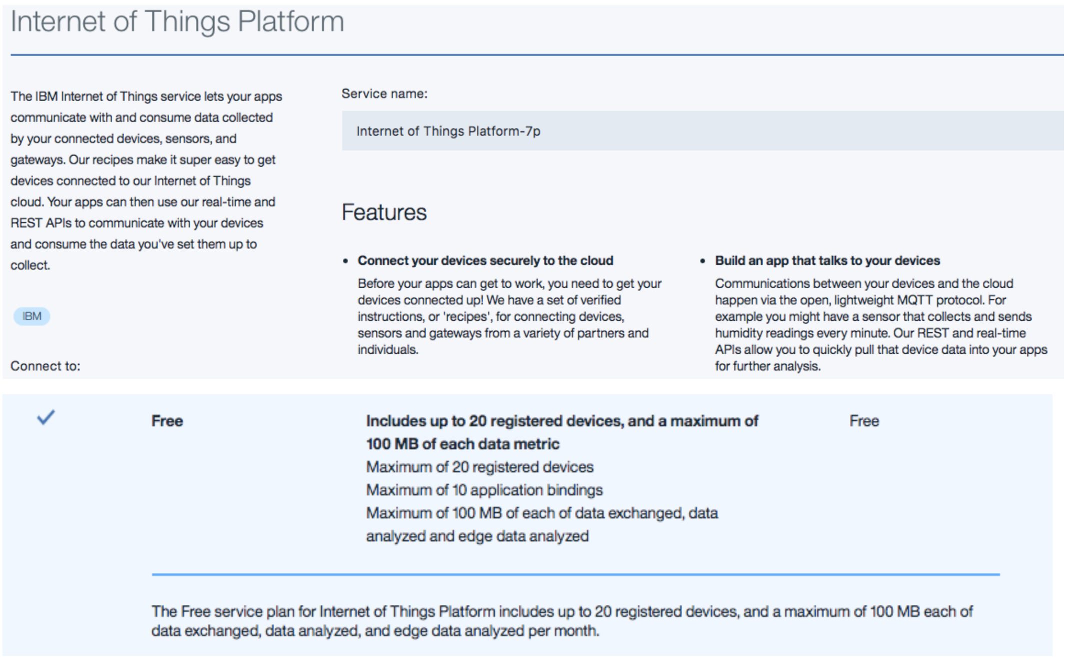
* Location: West Europe

Choose your pricing and scale tier

	F1 Free	S1 Standard	S2 Standard	S3 Standard
messages/unit/day	8k	400k	6M	300M
	Device-to-cloud telemetry	Device-to-cloud telemetry	Device-to-cloud telemetry	Device-to-cloud telemetry
	Cloud-to-device messaging	Cloud-to-device messaging	Cloud-to-device messaging	Cloud-to-device messaging
	1 unit	200 units maximum	200 units maximum	10 units maximum
	Unable to display pricing	42,17 EUR PER IOT HUB UNIT	421,65 EUR PER IOT HUB UNIT	4.216,50 EUR PER IOT HUB UNIT

IBM Bluemix IoT cost

Using Blumix IoT platform is free for the conditions presented in the next screenshot:



Internet of Things Platform

The IBM Internet of Things service lets your apps communicate with and consume data collected by your connected devices, sensors, and gateways. Our recipes make it super easy to get devices connected to our Internet of Things cloud. Your apps can then use our real-time and REST APIs to communicate with your devices and consume the data you've set them up to collect.

IBM

Connect to:

Service name:

Internet of Things Platform-7p

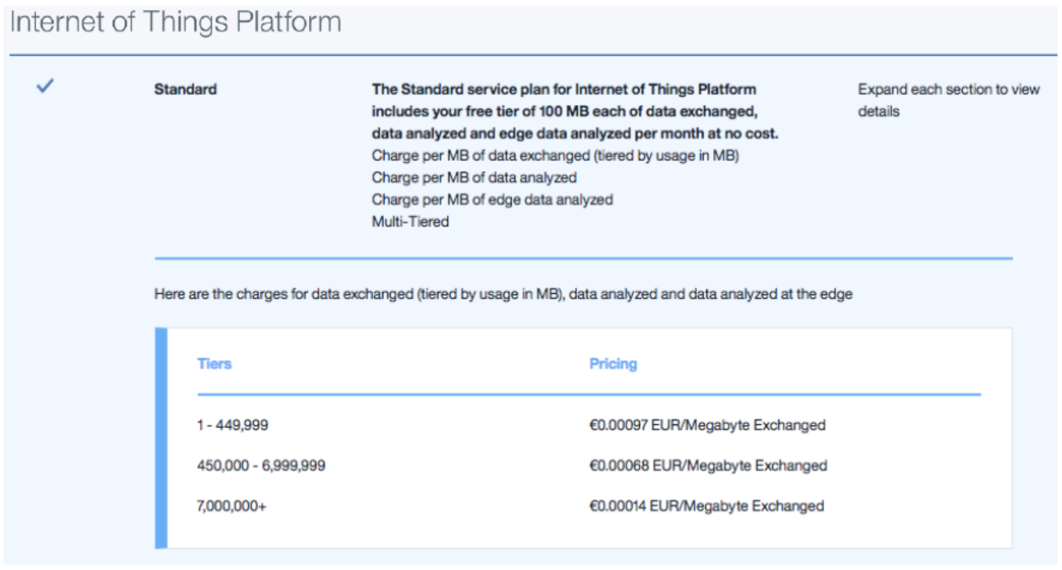
Features

- **Connect your devices securely to the cloud**
  Before your apps can get to work, you need to get your devices connected up! We have a set of verified instructions, or 'recipes', for connecting devices, sensors and gateways from a variety of partners and individuals.
- **Build an app that talks to your devices**
  Communications between your devices and the cloud happen via the open, lightweight MQTT protocol. For example you might have a sensor that collects and sends humidity readings every minute. Our REST and real-time APIs allow you to quickly pull that device data into your apps for further analysis.

✓	**Free**	**Includes up to 20 registered devices, and a maximum of 100 MB of each data metric** Maximum of 20 registered devices Maximum of 10 application bindings Maximum of 100 MB of each of data exchanged, data analyzed and edge data analyzed	Free

The Free service plan for Internet of Things Platform includes up to 20 registered devices, and a maximum of 100 MB each of data exchanged, data analyzed, and edge data analyzed per month.

For exchanged data, cost is like in the next screenshot:



Internet of Things Platform

✓	Standard	**The Standard service plan for Internet of Things Platform includes your free tier of 100 MB each of data exchanged, data analyzed and edge data analyzed per month at no cost.** Charge per MB of data exchanged (tiered by usage in MB) Charge per MB of data analyzed Charge per MB of edge data analyzed Multi-Tiered	Expand each section to view details

Here are the charges for data exchanged (tiered by usage in MB), data analyzed and data analyzed at the edge

Tiers	Pricing
1 - 449,999	€0.00097 EUR/Megabyte Exchanged
450,000 - 6,999,999	€0.00068 EUR/Megabyte Exchanged
7,000,000+	€0.00014 EUR/Megabyte Exchanged

NEW: IoT Analytics	€0.015 EUR/MegabyteAnalyzed €0.0037 EUR/Megabyte Analyzed at Edge

Conclusion of Part 2

Both Microsoft Azure and IBM Bluemix offer a free space for IoT but the registration to the Azure cloud requires providing a credit card number and a sum of $ 170 as a deposit, which is why we have not accessed the Azure platform for evaluation purpose – thus, the information presented hereby is taken only from Microsoft documentation.

IBM Bluemix can be used free of charge for a period of 12 months, as we benefit from this type of account via the Academic Innitiative. Eventually, this facility can be extended for another year on request.

IBM Bluemix provides each user with a unique communication channel (as if each user should have his/her own private cloud).

For this reason, it is possible to organize our NOAH team with different access rights for each partner. We can use an academic account where user groups are created by roles (developers, testers, administrators, etc.) that would allow all partners to work in the same space, each one have its own ”view” / access profile, depending on interest / role.

On page 44 of the document IBM Watson IOT Brasov.pdf (we attach this document to the deliverable) you can see that the application could be used for free by the users as long as the limit values (reasonable for our Pilots) are not exceeded. In the case of exceeding those values, the costs are those on page 45. It can be seen that the additional payment is made only for the number of messages users send.

After completing the application itself (which can be done either in Java or C #, as Bluemix IBM provides both a Java service and a ASP.NET service), data can be processed using the Watson IoT platform that provides also a series of AI (Artificial Intelligence) algorithms (predictive, perspective, cognitive etc) that allow analytical processing (thorough Data Analytics). This can be very useful for the development of the BAM (Behavioral Analysis Model) module which would be harder to develop in MS Azure.

For example, IBM Predictive service engage Machine Learning models to detect abnormal behavior (of things or humans) - anomaly detection is the most sought after algorithm that can forecast trends and detects data that violates patterns. All of these and others are available in the IBM Bluemix platform that includes capabilities for Realtime, Predictive and Cognitive analytics.

We should avoid developing applications in a programming environment and implement them in another environment, for the following reasons:

- Costs of connecting sensors and acquired data processing on Bluemix and the development of graphical user interface are zero for a period of 12 months (that could be extended), regardless of the platform loading (this loading can be assessed on different approaches)

- Development on a cloud and deployment on another cloud has the disadvantage of losing the services provided by the first one and the disadvantage of requiring virtual machines that would increase the operating costs

- IBM Bluemix offers services that make programming very easy, without the need ”to start from scratch” in order to connect to the cloud and to develop stand-alone applications to process data acquired by the sensors.

Part 3 – NOAH System architecture and technologies

The architecture of the NOAH system is based on the client-server concept. An overview of system is presented in the figure 7.

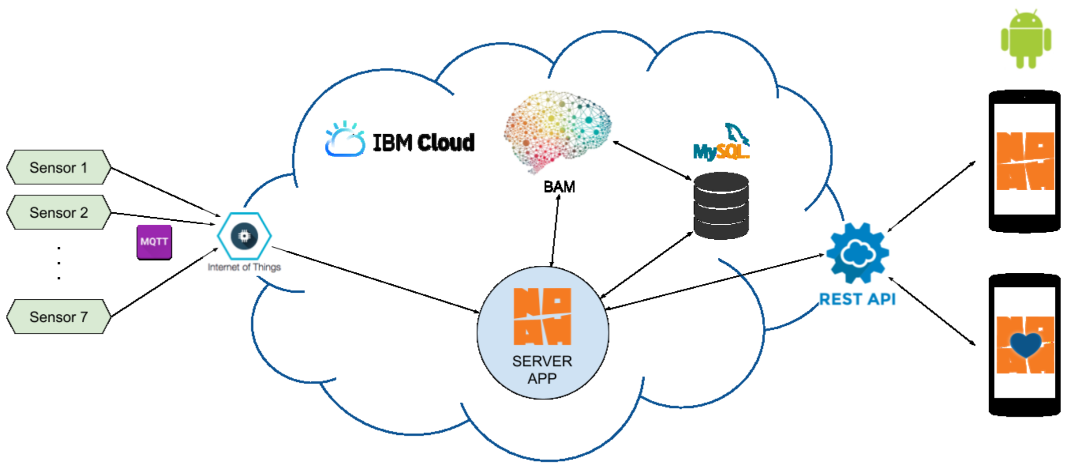


Figure 7 - Overview of NOAH system

In a first development iteration, the server side is developed and hosted by the IMB Cloud (Bluemix) platform. This requires two continuously running services: *Internet of Things Platform* (IoT) and *Compose for MySQL*.

The *IoT platform* communicates with the sensors and its role is to collect the data from the sensors and forward them to the server application. The *Compose for MySQL* service offers support for storing the necessary data for running the system.

The BAM module (*Behavioural Analysis Module*) processes data provided by the sensors to identify behavioural patterns and estimates the well-being of the monitored users; the server application reads the BAM module output and converts them into notifications or alerts that are useful for the caregiver users.

Inter-component communication, respectively between client and server, is done by using REST APIs, through HTTP protocol.

After the optimization process, the NOAH system has been split into several microservices, thus raising the modularity and scalability of the application.

This also bestows to NOAH systems the advantages of microservices-oriented architecture:

- the delivery of newer versions and new services is much faster because the developing, testing and delivery time are shorter.
- availability – in general, the delivery of a newer version for a monolith application requires restarting the entire system. The microservices architecture requires a lot less downtime; with more instances of services running at once, the redundancy of services is assured, so the delivery of new services is seamless.
- efficient management - since microservices behave as small portions of a whole, the software developers that are part of the team will have the opportunity to work independently and with a higher degree of productivity.
- flexibility – higher degree of flexibility in using frameworks, data sources and libraries allows adjusting the application while it is being developed.
- reliability - in case of failure, only one module is affected. Conversely, failures in a monolithic system can cause the entire system to stop.
- scalability – microservices allow for independent scaling of every component.

The second variant of the NOAH system uses a microservices-based architecture, which contains 3 such components.

The microservices architecture consists of developing an application as a suite of small services, each having its role, but communicating through messages, using simple mechanisms, such as the HTTP protocol.

In the NOAH system the MQTT protocol is used to receive data from sensors and HTTP for the interaction between the services that make up the server component.

The Node-RED version of the application developed using the IMB Cloud services represents a prototype system. The microservices version represent the modularization of the application and at the same time its optimization, allowing it to run in a diversity of cloud environments.

The architecture implemented in this regard (figure 8) involves a server application built by interconnecting three microservices that fulfil different functionalities. Thus, there is a service that facilitates communication with the connected sensors including an MQTT broker and which, through a connector, sends the information to a microservice that collects this information and stores it. The third microservice represents the interface between the server and the client, exposing the paths for HTTP requests that serve information to Android applications.

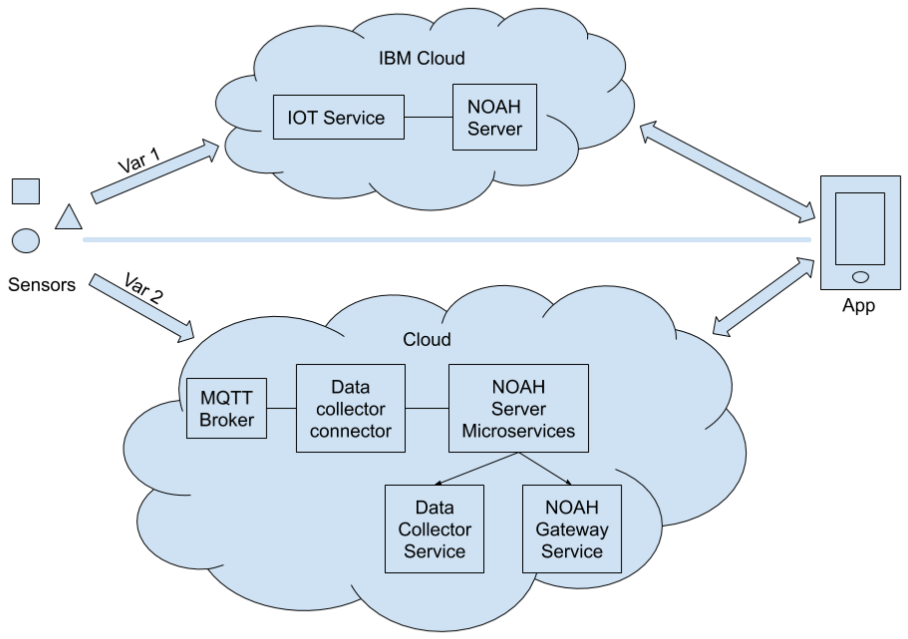


Figure 8 – Versions of the server architecture

Being a modular application, the server component of the NOAH system presents the advantage of diversity, so microservices are developed using different technologies, reaching the same result. In this way the communication part with the connected devices is developed in the Python language, and the collection, processing of the data and their service to the clients have been developed using the Spring Boot framework.

In order to store the data received from the sensors and other details needed to run the application, the management system of relational databases MySQL was chosen.

NOAH System have two main components, a server application and a mobile application. The server application has been developed in two ways. One was developed in Node-Red, an environment provided by the IBM Cloud as a prototype version, and the other one in Java based on the microservices architecture.

Both approaches came to the same results, microservices base implementation brings forward some advantages: increased scalability, modular built, development and release process can be realized for a specific module without interfering with the others.

Server application – cloud tools version

The server application for the NOAH system is a REST API and is built using the IBM Cloud (Bluemix) platform starting from the IoT module. The chosen development environment is Node-RED which runs on a Node.js server and provides developers with a visual programming environment.

The server application serving NOAH is scalable. For starters, it runs in a single instance that uses 512mb of RAM. The application can be scaled according to needs or financial plan.

Server application – microservices version

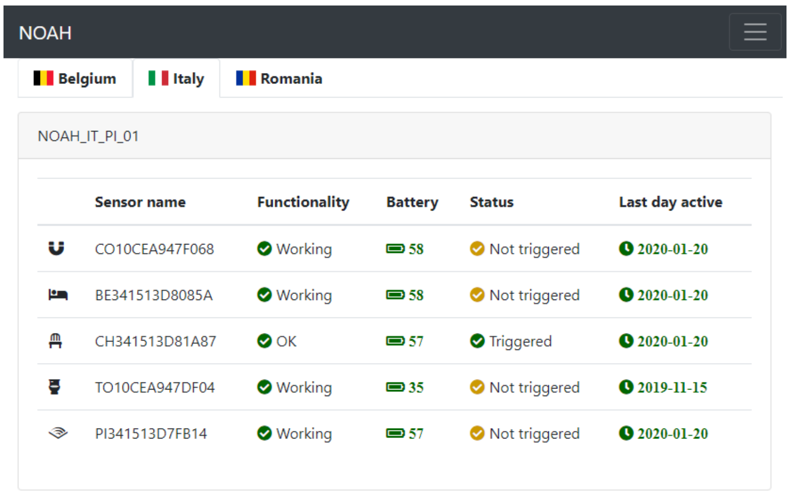
The microservice for the server application of the NOAH system is the central component that serves to collect the data from the sensors. This application receives, via an HTTP request, in JSON format, the data sent by the module that performs the communication with the sensors. These are processed and stored into the MySQL database.

The Gateway application has the role to provide all the information that is needed in the Android application, taking over the second part of the initial Node-Red. It takes the data from the database and process it providing endpoints that are used by the mobile user interface.

The *AdminCenter* web application (figure 9) is part of the system; an administrator user has the possibility to register sensors, as IoT connected devices, and group them into kits. Also, there is an overview of kits and the last received states of the sensors.

System security aspects

The security of the system is ensured by multiple methods depending on the characteristics of each working flow.



NOAH_IT_PI_01

	Sensor name	Functionality	Battery	Status	Last day active
	CO10CEA947F068	✓ Working	58	✓ Not triggered	2020-01-20
	BE341513D8085A	✓ Working	58	✓ Not triggered	2020-01-20
	CH341513D81A87	✓ OK	57	✓ Triggered	2020-01-20
	TO10CEA947DF04	✓ Working	35	✓ Not triggered	2019-11-15
	PI341513D7FB14	✓ Working	57	✓ Not triggered	2020-01-20

Figure 9 - AdminCenter Overview of NOAH kits

Users' access to the application is controlled based on the username-password pair authentication mechanism, improved with the rights assigned to a specific user, defining roles which restrict the features of the application that can be used.

Authentication on the *AdminCenter* can be performed only by the users that have administrator role, each country has its own user/s that can manage the NOAH kits.

Mobile applications can be accessed by two types of users: caregivers and end-users, roles that provide access to specific features. These applications provide an auto-login feature secured by a token which is generated when user logs in and is deleted when user logs out.

The application requires a link between the two types of end users. For a caregiver user to associate with an end-user, he/she must enter the secret 4-digit PIN as an initial pairing procedure, a PIN that is owned/known only by the end-user.

The sensors used by the system are from the IoT sphere, directly connected to the Internet, without the need for an intermediate computing system. Communication with the server is secured by using the SSL / TLS protocol (PKI – *Public Key Infrastructure*).

Measures were taken for guarding users' identities. Data is kept in anonymized database tables, not directly linked to the users. All communication interfaces with the database are secured using the internal mechanism of the database management systems.

Conclusion of Part 3

The NOAH system uses cloud technologies and IoT principles to provide quality services, in a flexible and scalable product.

The two equivalent deployments as facilities (the first, using services provided by IBM Cloud, the second representing a more flexible solution from the point of view of the cloud service provider), give a clear advantage to the system, being widely scalable, depending on by the number of users and budget.

The technologies chosen to develop the system's modules are Java-based, the microservices being developed using the SpringBoot framework, compatible with most of the containerization environments offered by cloud providers.