Universitat de Lleida

TREBALL FINAL DE MÀSTER

Estudiant: **Joan Pau Castells Gasia**

Titulació: Màster en Enginyeria Informàtica

Títol de Treball Final de Màster: Design and implementation of an architecture to assist decision making in livestock 4.0

Director/a: Jordi Mateo Fornés, Adela Pagès Bernaus

Presentació

Mes: Juliol

Any: 2021

Design and implementation of an architecture to assist decision making in livestock 4.0.

Joan Pau Castells Gasia

February-July 2021

Contents

List of Figures

Abstract

A wide variety of low-cost sensors that describe di□erent environmental variables are currently available for all sectors. However, more value can be derived from these data by combining it with models and other sources of information. To support these sectors in adapting this technology, this work presents a general cloud architecture that is able to integrate any type of sensor, detect and correct possible sensor errors, obtain information from other sources, incorporate models that generate value from the collected data, provide user-friendly graphical interfaces to interact with complex system processes, authentication and user roles mechanisms, and technologies that make easier to adapt the architecture to di□erent environments. Moreover this architecture is applied in a livestock scenario in order to assist in farmer decisions.
Keywords— Big data; Internet of Things; Microservices Architecture; Cloud Computing; Precision Livestock Farming; Industry 4.0

1 Introduction

Developments in the digital era are expected to boost industrial and business processes making them more e□cient, automated and competitive. Nowadays, the line that separates the virtual and the physical world is getting closer, and this approximation brings new paradigms and challenges to becoming a 4.0 society. From the Internet of Things (IoT) perspective, the core idea is that each physical object in the real world is equipped with sensors that connect it to the virtual world. Predictive, prescriptive models or Artificial Intelligence (AI) algorithms can be fed with these data and other context information to generate new insights to assist the decision making process. Several innovations, like sensor technology [20, 3], positioning systems[30], digital image processing [21], cloud [29] and fog computing [9] among others make this transformation possible.

Nowadays the automatic acquisition of data, also known as the digitization process, is undertaken by many sectors given that sensors are becoming cheaper and energy-e□cient [13]. The value of such data is linked to the use given. A monitoring use of the sensor data leads to descriptive analytics and using the information only in this way, it does not exploit its full potential [2]. Thus, there is a need of gathering, crossing and processing data from di□erent formats and sources to make a smarter use of the data.

Several challenges arise on the data collection process: a) raw data coming from the sensor may be inaccurate or even erroneous due to systematic and random errors [12], b) loss of the information due to network congestion , continuous environmental interference [10] or far distances and remote places [31], c) fraudulent manipulation of data by attackers a□ecting its veracity [1], d) sensors from several manufacturers that can vary the structure of their data and their communication interface [11] , e) huge volumes of real-time data that traditional systems cannot handle [26] , among others. These result in an inadequate representation of the data [5].

Another challenge to be solved is how the multiple types of sources, devices, and other information are connected in order to obtain a global vision. Data comes from di□erent installations, places, databases, enterprise information systems and more, each with its own format. There is a need to develop platforms and services where this data can be exchanged among internal and external tools.

With the aim to increase the value of the data collected, data scientists in the fields of Operational Research, Artificial Intelligence or Statistics are developing advanced modelling techniques. These techniques may rely at some points in solving intensive computational processes regarding CPU (Central Processing Unit) and Memory such as solving optimization models or training neural networks. The management of these resources requires platforms where services can scale as resource needs.

Moreover, the human interaction with these complex techniques and smart data generated needs to be presented with a user friendly perspective, smoothing the learning curve and pro-

viding targeted information. Decision-makers needs to use these tools with a natural flow to access different information levels depending on their roles and decisions.

To overcome the challenges exposed until now, in this work we propose a generic architecture and to provide guidelines on how to adapt the architecture to different scenarios. Therefore the main contributions of this work is the design of an architecture with the following properties:

- Support for the integration of many type of sensors, regardless of how it has been provided, in order to collect real-time and store historical data independently of its format.
- Assistance to the adoption of methodologies and mechanisms to detect and correct the possible errors in order to provide the most accurate information.
- The capability to obtain data from other data sources such as databases, web services, etc... to complement the sensor data.
- The interoperability between internal and external systems.
- The ability to incorporate models in order to use the collected data to provide an extra value.
- The provision of the needed resources to ensure the correct use of the different tools and methodologies of the digital platform.
- The incorporation of tools that facilitates the interaction with the different complex processes and presents insights to the users.
- Security mechanisms to provide the needed information and access to the different processes for the different user roles.
- Easy to use, implement, and deploy the architecture into different scenarios.

In order to validate this architecture design this work presents a case study for the pig sector. A platform using this architecture has been implemented. Data collected from feed silos equipped with a level sensor is used to predict the growth of a batch of fattening pigs. Procedures for data cleaning are integrated in order to provide data of quality to proceed with the analysis. Furthermore some models are integrated in order to estimate new variables from these sensor data such as the growth of these pigs in order to provide more knowledge to the farmer such as if these pigs are growing as expected. Moreover it is added an alert system to warn the farmer if some anomaly is detected.

The rest of this paper is structured as follows. Section 2 compares similar works. Section 3 presents the design of the cloud architecture, the technologies that have been chosen to implement it and how it has been distributed and deployed. Section 4 introduces the functionalities that have been implemented by the use of the selected technologies. Section 5 applies this cloud platform in a livestock scenario. Finally, section 6 highlights the main conclusions of this work and future features that could be integrated.

2 Bibliography review

In this section we review the state of the art of the proposals in the literature addressing the digitization focusing on the Agriculture sector, which is related to our case study.

The main objective of this research is to analyse if these studies present IoT platforms that accomplish with a set of characteristics that we think they are essential for IoT scenarios. These characteristics are the following:

1. Decision making: To check which tools of an IoT scenario such as visualisation, predictive or prescriptive analysis, generation of alarms, among others are described in the studies.
2. Scalability: In order to analyse if the platform is able to integrate new IoT devices and services to meet the growing demand.
3. Interoperability: The architecture lets the communication between di□erent systems independently of their technology.
4. Portability: Check if the architecture provides some mechanisms that facilitates the system to be run in di□erent environments.

The items listed below provide a general summary of the purpose of IoT platforms architecture described by several authors in this context.

1. Authors of [8] presents a platform that is able to acquire, process, store and monitor data from grow cropping systems with the purpose to automate the maintenance of crop lands and control the conditions that determine the proper development of a crop such as soil moisture, water and soil, and luminance.
2. In [28] present the design of a generic architecture that is able to manage IoT devices, acquire, analyse, generate events/alerts from these data. They validate the solution in vineyards in order to monitor the environment such as the temperature, air/soil humidity, among others and provide models that helps the farmer to predict mildew diseases.
3. The study presented in [4] proposes a smart IoT communication system used as a low controller irrigation system. It collects real time data such as temperature, humidity air, rain in order to monitor it, control the actuators (components that perform mechanical actions) and detect dangers in the field.
4. In [27], the authors present an IoT platform for animal behavior analysis and health monitoring. It focus in di□erent parameters of diary farms in order to monitor the health of the cows and detect possible anomalies.
5. In [6], a low cost, modular IoT platform is presented to improve the management of generic farms. Authors validate the platform in a real farm in order to collect environment variables related to the growth of grapes and greenhouse vegetables.
6. A cloud-based framework called WALLeSMART is presented in [19] which proposes an architecture to address the challenges of acquisition, processing, and visualization of massive amount of data. They focus in collect parameters related to the diary farms and the weather.
7. The authors of MICROCEA [24] present an architecture in order to automate the growth of plants in urban indoor residential. The purpose is to monitor sensor data such as light, temperature, humidity, among others and let users program events to automate the process.

Although most of the solutions provide a tool to visualize the data (historical sensor data, real-time sensor data, external services data) and perform some management tasks, only three works present the required tools for IoT scenarios in order to take more profit for the data collected. In [28] they provide models to detect diseases in the crop and generate alarms since the farmers can deal with the crops the earliest possible. The system developed by [4] generate

alarms and create actions to deal with the adequate growth of the crops and the solution implemented by [27] provides analysis to predict heat detection or anomalies of the cows and generate alarms to warn the farmer. Such solutions are mostly self-created platforms. However in [8] uses the Ubidots IoT platform in order to provide with generic and basic visualisation and analysis tools.

From a scalability point of view, all these solutions use the cloud computing technology in order to obtain the needed resources to integrate new services and IoT devices. However the type of database that is used and how these applications are distributed must be also in consideration in order for the platform to be able to cope with the new needs of the environment. There are some solutions that use relational databases [4, 6]. These kind of databases are not change tolerant since they require that the structure of the data is defined before store it and a small update in the schema can cause a lot of modifications in the system that must be carefully controlled. Use NoSQL [27, 19] or Time serie databases [28] are a better choice for IoT scenarios due to the heterogeneous data of the sensors. For example NoSQL databases provide flexible schemas that let store unstructured data without having to define a structure.

Another important feature is how the different applications of the architecture are distributed to ensure that it is able to support the growing demands. Authors in [6] propose a platform that is able to integrate new IoT devices and application modules according to the demands of farmers. Moreover in [27, 8] take some responsibilities from the cloud such as computing and analytics in order to be performed at the edge of the network and provide better time-responses. In addition in [27, 28] present a microservice based approach in order to distribute the IoT platform in a set of services that are responsible to perform specific functionalities and are independent from each other.

Most of the studies take in consideration to provide HTTP (Hyper Text Transfer Protocol) APIs (Application Programming Interface) to the services in order to be able to interact with the consumer applications. A HTTP API is an interface that allow interact two applications by using the HTTP protocol. In addition the most used message protocol in these works that lets connect the IoT devices and the cloud platform is the MQTT (Message Queueing Telemetry Transport). However in order to integrate heterogeneous IoT devices, they might require other kind of network protocols. In [28] they use RabbitMQ in order to also be able to operate with AMQP (Advanced Message Queuing Protocol) and STOMP (Streaming Text Oriented Messaging Protocol) clients.

Finally there is also the need to provide mechanisms that facilitates the use of the architecture into different environments and scenarios. In [28, 27, 19] there are the only studies that provide these facilities. In fact they use Docker in order to package the architecture and its dependencies in order to be deployed in any environment with the advantage to select which component might need in order to adapt it to a specific IoT scenario.

In table 1 is summarized the comparison of related works and ours with the sensors and actuators used, what they do with the data, the characteristics of the architecture and in which sector is applied. We can conclude that there are few related studies in the livestock sector. So there is the need of studies that provide scalable, portable, interoperable architectures in this sector that let integrate models that take profit of the data collected by the sensors. In our work the purpose is to provide the design of an architecture that can be reused in different scenarios since the use of the Microservice approach, Docker and the integration of models that allow to extract more value from the data obtained from sensors such as in our case study the growth of the pigs from feed silos.

Study Reference	Sensors/Actuators	Summary	Decision Making	Sector
[8]	temperature, humidity, electrovalves, lights, electric pumps	Use these data to observe the growth of the plant and act if some condition is accomplished	Cloud Computing; Edge Computing; Descriptive Analysis	Agriculture
[28]	temperature, air/soil humidity, wind speed, wind direction ,rainfall	Monitor sensor data and train models to predict diseases in vineyards and send alerts to the users	Cloud Computing; Docker; Microservices; Time series DB; Descriptive and Predictive Analysis	Agriculture
[4]	temperature, humidity air, rain, UAV, Irrigation controllers	Monitor sensor data, human interaction with irrigation controllers, danger detection, generate alarms, controllers act from the smart system decisions	Cloud Computing; Relational Database; Descriptive, Predictive and Prescriptive Analysis	Agriculture
[27]	Pedometer sensor	Monitor the health of the cows, predictive detection of anomalies	Cloud Computing; Docker; Fog Computing; Microservices; No SQL; Descriptive and Predictive Analysis	Livestock
[6]	humidity, temperature	Monitor the sensor data from a greenhouse and a vineyard	Cloud Computing; relational database; Descriptive analysis	Agriculture
[19]	17 parameters diary farms, temperature, precipitations, humidity, sunshine, wind speed	Processing and monitoring massive amount of data	Cloud Computing; Docker; No SQL and relational database; Descriptive analysis	Livestock
[24]	Light, temperature and humidity air, CO2, PH, water temperature, electro conductivity, heater, humidifier, fan circulation, fan vent	Real time monitoring and events programming	Cloud Computing; NoSQL database; Descriptive Analysis	Agriculture
Ours	Silometric sensor	Real time monitoring and prediction pigs growth, generation of alarms	Cloud Computing; Docker; Microservices; NoSQL database; Descriptive and Predictive Analysis	Livestock

Table 1: Comparison between related studies

3 Architecture

Basically in an IoT scenario there are sensors that are able to share its data through the Internet and interoperate with clould platforms. See figure 1.

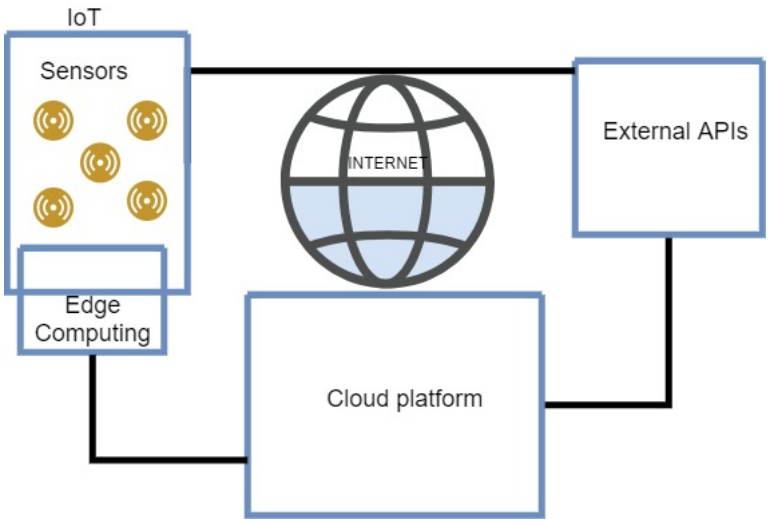


Figure 1: General Internet of Things (IoT) scenario

These sensors can give access to its data through APIs or Edge computing nodes. Usually in most of the IoT scenarios sensors provide APIs that allow obtain its last reads (they are updated periodically) in semi-real time by using the HTTP protocol. However there are other scenarios where is required that the response time is the lowest possible. In order to provide quicker responses there is the need that the processing and storage capabilities of the cloud are moved near to these IoT devices but also to use more lightweight protocols than HTTP such as MQTT for data transmission to provide real-time interaction. Edge Computing nodes o□er these capabilities. Nevertheless the resources of these computing nodes are limited and in order to store historical data or perform complex tasks they must also transmit the preprocessed sensor data to the cloud. The cloud platform is the key to manage the di□erent requirements of IoT scenarios such as huge volumes of heterogeneous data, security, interoperability and scalability by providing the on-demand computing resources such as networks, databases, servers, storage and others through Internet thanks to its Cloud computing infrastructure.

This generic cloud architecture (Figure 2) has been designed by taking into account the requirements presented in Introduction but with the condition that the cloud platform must be connected with third party APIs or Edge nodes to give response to these input data.

This cloud architecture is composed of three di□erent layers: The presentation layer, the logic layer and the data layer. The main purpose of the presentation layer is to provide a user interface to collect the user data and display the relevant information to the user. The logic layer represents the processes to obtain the sensor data, the treatment of this heterogeneous data such as data cleaning methodologies, the generation of alarms, the execution of di□erent algorithms to obtain valuable data for the users and the interoperability mechanisms to interact with external systems. Finally the data layer is in charge to store the data that come from the di□erent sources.

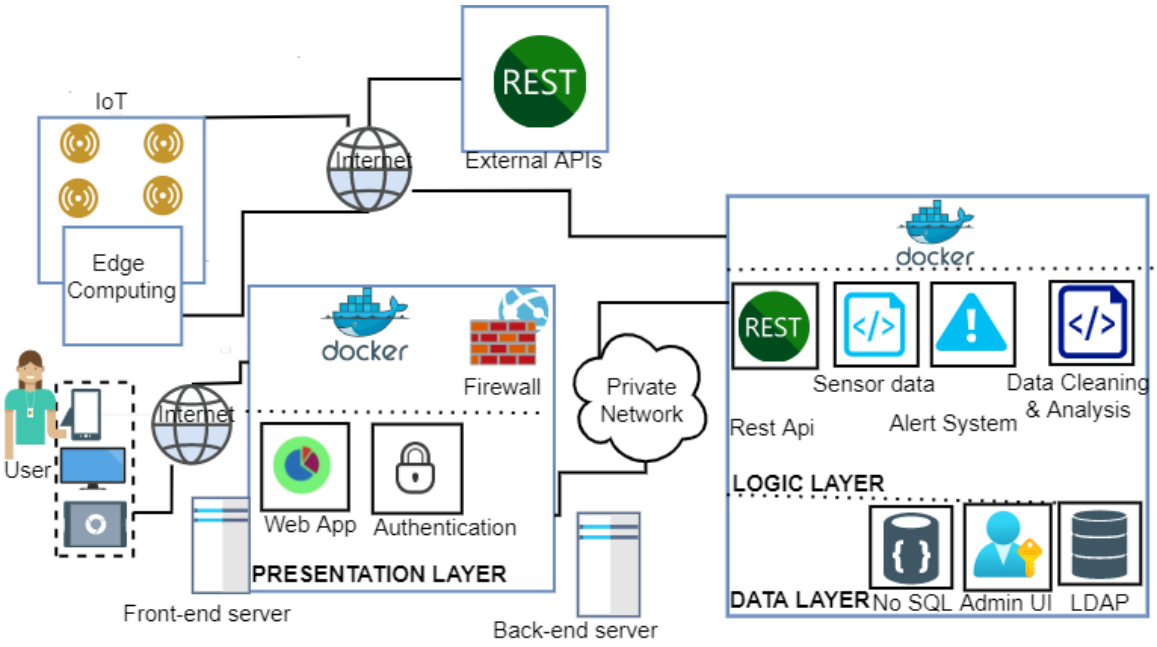


Figure 2: Cloud Architecture Design

3.1 Presentation layer

The presentation layer is the visible part of the cloud architecture. It represents the tool that users use to interact with the complex functionalities and models that they might require. Basically it is a user-friendly web application with authentication and authorization mechanisms that can be accessed from any user device connected to Internet. It hides the complexity of the functionalities and operations that are performed in order to obtain the desired information. Two technologies have been chosen in order to implement this layer.

- R Shiny [22]. This technology has been chosen in order to implement the Web Application. It is a package for the R language that facilitates the built of interactive web apps by providing the required components to integrate the di□erents analysis developed in R in the web without having the knowledge of Javascript, HTML (HyperText Markup Language) and CSS (Cascading Style Sheets). It also provide the mechanisms to interoperate with external systems such as APIs in order to obtain and transmit the required information.
- ShinyProxy [23]. It is an open source software that provides the authentication and authorization mechanisms in order to deploy Shiny apps in a production context. It o□ers a login page that users introduce its credentials and apply the authentication and authorization mechanisms. Moreover it provides a totally isolated workspace for every user session by launching a Docker container with the web application accessed by the user.

3.2 Logic layer

The purpose of this layer is to process the data that comes from the presentation layer, the sensors and external services through defined business operations and to query the data layer. It acts as the bridge that allows the communication between the presentation layer, the IoT devices, the exernal services and the data layer. In order to provide communication between

the di□erent resources of the architecture there are used the APIs and scripts in order to obtain the sensor data by using the mechanisms that the source provider o□ers (APIs, Machine to Machine protocols). APIs allow interoperate between the di□erent components of the architecture independently of its technology stack that has been used but also it lets the integration of new components to provide more data such as more sensors, more external services and others. Most of the time these APIs use the REST (Representational State Transfer) principles to operate respecting the HTTP protocol and transfer the required resources by using data formats such as JSON (Java Script Object Notation) and XML (Extensible Markup Language). Moreover di□erent scripts have been developed in order to perform data collection, data analysis to extract value of this data and generate alarms. The technologies that have been applied in order to implement this layer are the following.

- Node js [17]. Node js is an application runtime environment that allows to write Java Script for web applications. It comes with many libraries that are adequate for backend development such as file system management, HTTP streams, database management, etc. Its dependencies are managed using Node package manager(npm) which are defined in a file called package.json. It has been chosen in order to develop the web service but also to implement an alert notification system since it facilitates to deal with multiple client requests and provides mechanisms that help to scale the applications. The web service is in charge of retrieving the data from the database but also receiving the data from the client to process the data and store it. This data is transmitted and received using the JSON format. The communication between the client and the web service is made via the REST API with the HTTP protocol that defines the di□erent CRUD (Create, Read, Update, Delete) operations which are defined by di□erent controllers. These controllers can operate with the NoSQL database by using a library called mongoose that permit to map the models defined by Schemas into collections of the database. The alert notification system uses the nodemailer library in order to send emails to the user when the system detects an alert.
- Python. In order to perform the data collection process python has been chosen. It provides the mechanisms to perform HTTP calls, MQTT protocol and to query NoSQL databases. Basically it has been implemented a python script in order to collect the sensor data provided by an External API and using a package called pymongo to store the data into the NoSQL database. In order to integrate new sensors, it would be necessary to develop the corresponding script using the protocol that would be required for that sensor.
- R. In order to perform the data cleaning and analysis R scripts have been developed. These scripts interact with the Web service in order to store the cleaned data and store the outputs of the analysis methods.

3.3 Data layer

The data layer is in charge to store data that come from the sensors, the user profiles, the user roles, the data coming from external services, among others. No SQL databases are a good choice to store the heterogeneous data such as sensor data and LDAP (Ligthweight Directory Access Protocol) directories for store user information and roles. In order to implement this layer two technologies have been selected.

- MongoDB [16]. It is an open-source NoSQL database oriented to documents. It allows store unstructured data as a document in a representation called BSON (Binary JSON) in a collection of documents. It provides a dynamic schema that lets build flexible models by updating the schema without a□ecting the other documents. This features allow to adapt quickly to changes that can be vital when the system starts to store huge volumes of data. This database is also designed to support horizontal scalability that is achieved by adding new machines and is managed by the cloud computing technology and provide

high-performance in order to perform simple operations. Finally it also save costs and complexity in comparison to relational databases. These features make this database ideal for IoT scenarios.

- OpenLDAP[18]. OpenLDAP is an open source implementation of LDAP that allows to access to directory services. These directories are specialized databases optimized for reading, browsing, searching and simple update operations. These features make these kind of databases ideal for providing centralised user administration to store sensitive information and other account details of the users. Furthermore in order to make easier the management of this LDAP directory an administration user interface is provided. This is the LAM (LDAP Account Manager) [14] service which basically provides a web frontend in order to manage the entries of OpenLDAP. The main reason for choosing LAM is providing a user friendly interface to manage a LDAP directory without the need that the administrator has any knowledge in managing LDAP entries.

3.4 Deployment

In this section is explained how the architecture has been deployed by leveraging the advantages that o□ers the application of Virtual Machines and Docker.

Virtual Machines are digital representations of physical computers with its own Operative System and assigned resources (Memory, CPU, storage and networks) that are created from a software component called hypervisor. This virtualization allows the cloud infrastructure to be split up by independent Virtual Machines that can act as di□erent servers by sharing the resources managed by the hypervisor. Moreover, in these Virtual Machines can be installed Docker [7]. Docker is an open platform that allows split up the applications from the infrastructure into Docker containers and deal with process isolation. Docker containers allow separate a service and its dependencies from the underlying operating system and other containers so it avoids dependency conflicts . All these containerized applications share a unique operating system so it permits saving resources and also to be started and stopped more quickly in comparison to Virtual Machines that need each one an Operative System.

To deploy this architecture it has been used the Stormy platform [25]. It is a private cloud based on OpenNebula owned by the University of Lleida that provides us the infrastructure, the virtualization, among others. However this architecture is not limited to this cloud provider since it has been given the needed mechanisms to be reused in di□erent environments such as Amazon Web Services(AWS) or Google Cloud. Two Virtual Machines have been instantiated using a Centos7 OS image in order to host the di□erent layers. The first machine that contains the presentation layer acts as a gateway. It is the open door that allow this machine to be accessed through the Internet but also to communicate with the services that are deployed in the second machine. In this second machine is hosted the logic layer and the data layer and only is accessible via a Virtual Private Network. Moreover these two machines have a firewall configured in order to deny all the connections except the ones that use http, https and ssh protocols. The advantage of this configuration is that makes the data protected inside the cloud.

All the services that contain these layers are deployed and managed using the Docker and any host with the Docker runtime installed can run these services. This containerisation enables that this architecture can be scaled and reused for the reason that it can replace the services used in this architecture for the specific services and its suitable technologies to implement the specific IoT scenario and also the environment where these services are deployed but also duplicate the services that require more availability. So it facilitates that this cloud architecture can be reused in any environment. Moreover the client docker compose can also be installed. It is used to manage applications with a considerable amount of services by declaring them in a YAML file that makes easy their deployment and management.

4 Implementation

In this section is mentioned the di□erent functionalities that are obtained by the use of the technologies stated above.

- Roles and authentication. This functionality is provided by the combination of the Shinyproxy and LDAP services. The Shinyproxy service o□ers a login page that sends the credentials to the LDAP directory service in order to check if that user exists and the roles that the user has assigned in order to provide the web application that it might require.
- Dashboards and management of data. RShiny applications present in a user-friendly manner the data collected by the sensors and the analysis provided by the cloud. The data is provided by the REST API o□ered by the Node js web service.
- Data collection: This task is performed by the python scripts, they are responsible to collect data periodically and store it into the MongoDB database. There is the need to assign them with the corresponding external service or Edge Node by using the required communication protocol. Moreover they are controlled by a cron job. A cron job is a utility provided in Unix Operative Systems to schedule di□erent tasks such as these scripts.
- Data processing and Analysis: This task is performed by the R scripts. These scripts are responsible to execute di□erent methodologies such as data cleaning methods and other procedures in order to provide data of value. They are also controlled by a cron job.
- User administration: These service is provided by the combination of the LDAP and LAM. The administrator can access to the LDAP directory through a user-friendly interface provided by the LAM service and manage the user data.
- Alert Notification: This service operates with the MongoDB database in order to obtain the most recent data and check if some conditions are accomplished in order to advise the users via email to be warned about possible problems in their environment.

  Note that these services follow a Microservices architecture approach. The full functionality of the cloud platform is provided by these collection of services. As it is observed above each service use its own technology stack and interoperate between them by using lightweight protocols and APIs. Observe that if a service needs to be updated then only this service becomes not operational without a□ecting the others. Another highlighting feature is that this decoupling not limits to build the overall platform by using a unique technology, instead it can be used the most adequate technology in order to built a specific functionality.

  The code of this implementation can be found in this Github repository: `https://github.com/sensors-agritech/sensors-udl-cat`

5 Case Study

In the Livestock sector, there are available a huge amount of sensors that provide data for di□erent variables of their environment. However there is the lack of tools that supports the integration of its current management information systems with the sensors, external services and other sources of data, but also integrate predictive and prescriptive models that generate insights from this data due to its computational requirements. So they need cloud platforms that provide user-friendly interfaces that allow interact with these complex models but also it supports the on-demand requirements of the environment. In this section we present an application built under this cloud architecture. It can be accessed through this link `https://gcd007.udl.cat/login` and a user (demo) to test it with the password (demo@alba25.udl.cat). The main purpose of this application is to estimate the weight evolution of the pigs in a fattening farm based on their feed consumption. For this reason there is the need to know the available feed in a silo on the assumption that this growth of the animals is due to what they consume. Once the amount of feed that has been unloaded is known, how this amount is distributed among the animals in the batch and finally estimate the weight from this consumption.

In order to obtain the available amount of feed in a silo we have used the Silometric LM sensor [15] for this livestock scenario.

Figure 3: The MC Systems Silometric sensor

It is a wireless sensor that has been built by the enterprise MC Systems. This sensor is powered by a battery due to its low consumption and must not depend on electrical installations that in farms the most of the times are not necessary. It is placed on the top on the silo and it is used to measure the level of solids inside them using laser technology. See figure 4.

In order to compute the level taking into account the height of the silo it uses a time-to-flight computation to a specific point. So it measures the time that the laser pulse goes to this specific point and it backs. In addition, these devices are in standby, so every two hours a reading of the sensor is done and it is sent through a radio network of wide coverage and low electricity consumption to the platform called Digitplan that belongs to MC Systems where the users can visualise and manage this data. This platform extrapolates the volume of the silo by using the level measures that have been sent by the sensor. Finally this platform exposes a REST API in order obtain the silo data via HTTP protocol.

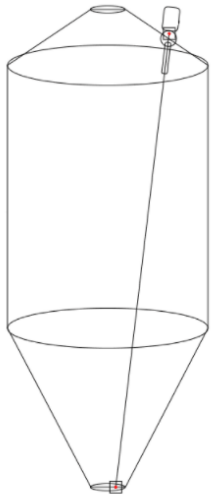


Figure 4: Level Measurement using laser technology

In order to perform an automatic data collection of the sensor data a Python script has been implemented in order to obtain the data from the di□erent silos that the University of Lleida has been given access. A cron job has been defined so every two hours this script is executed. See algorithm 1.

Algorithm 1 Get sensor data pseudocode

```
1: Authenticate to the Digitplan platform;
2: sensors = Get sensors;
3: for s in sensors do
4:     s*= Select attributes of s;
5:     Store s* into the database;
6: end for
```

Moreover in the R web application there are four di□erent views where the user must perform a manual entry of data that correspond to a specific batch since there are parameters that this sensor not provides. The first view shows general parameters for the batch (Figure 5).

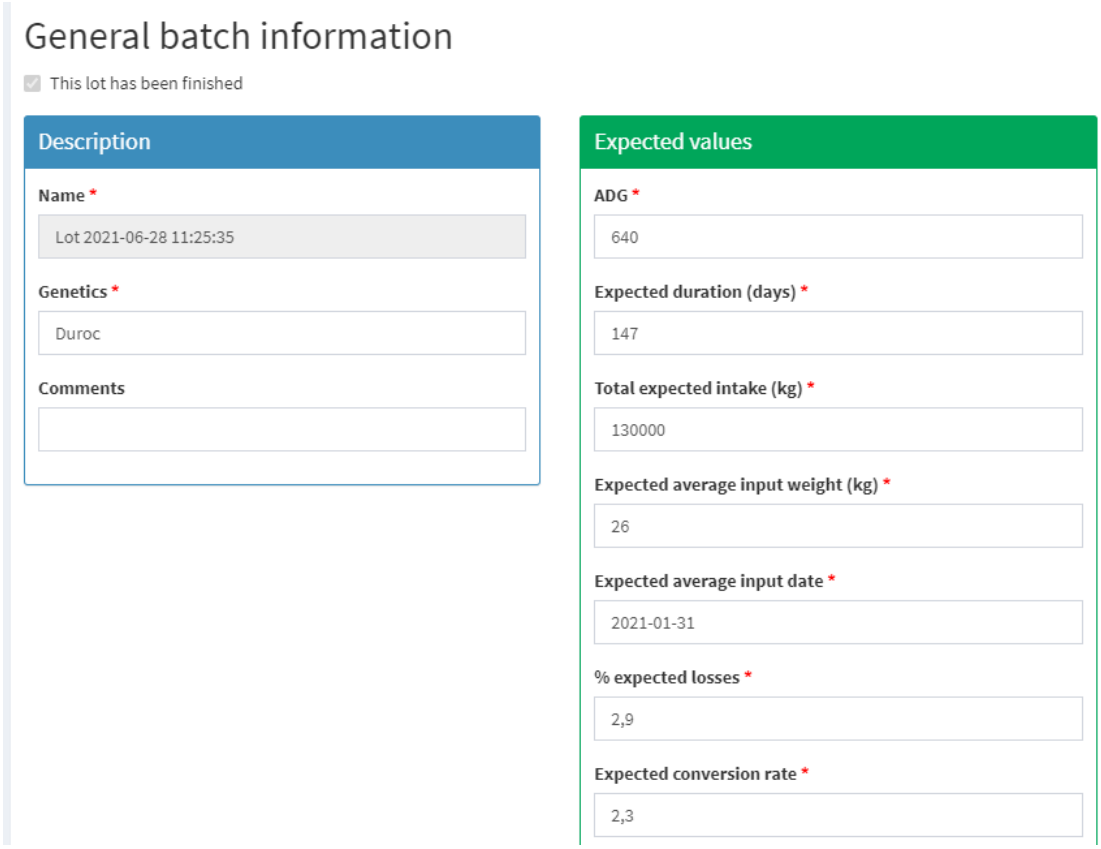


Figure 5: General parameters for a batch

The second views display the inputs (Figure 6). It represents the amount of pigs that arrived in a specific date for this batch. These animals inputs can vary in age. Whereas the third view represent the outputs of animals that are sent to the slaughterhouse in a specific date (Figure 7). These entries are currently entered manually, however they could be automated in the future by implementing a script that obtains these data from the management information system of the company.

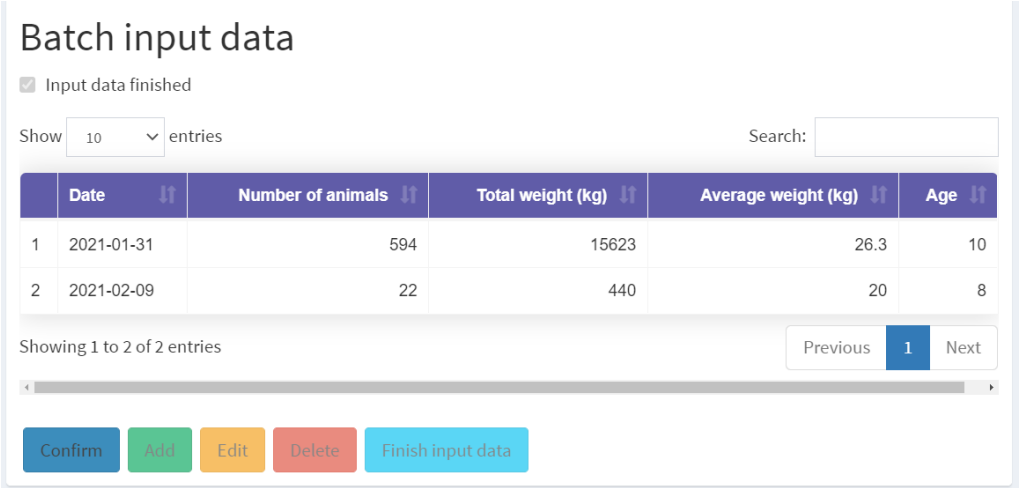


Figure 6: Input data for a batch

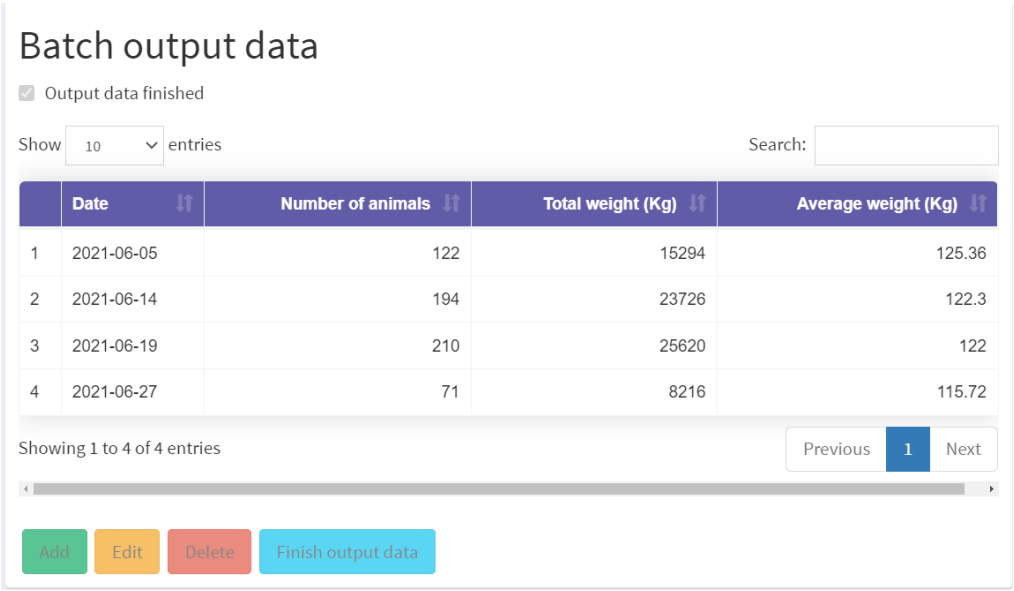
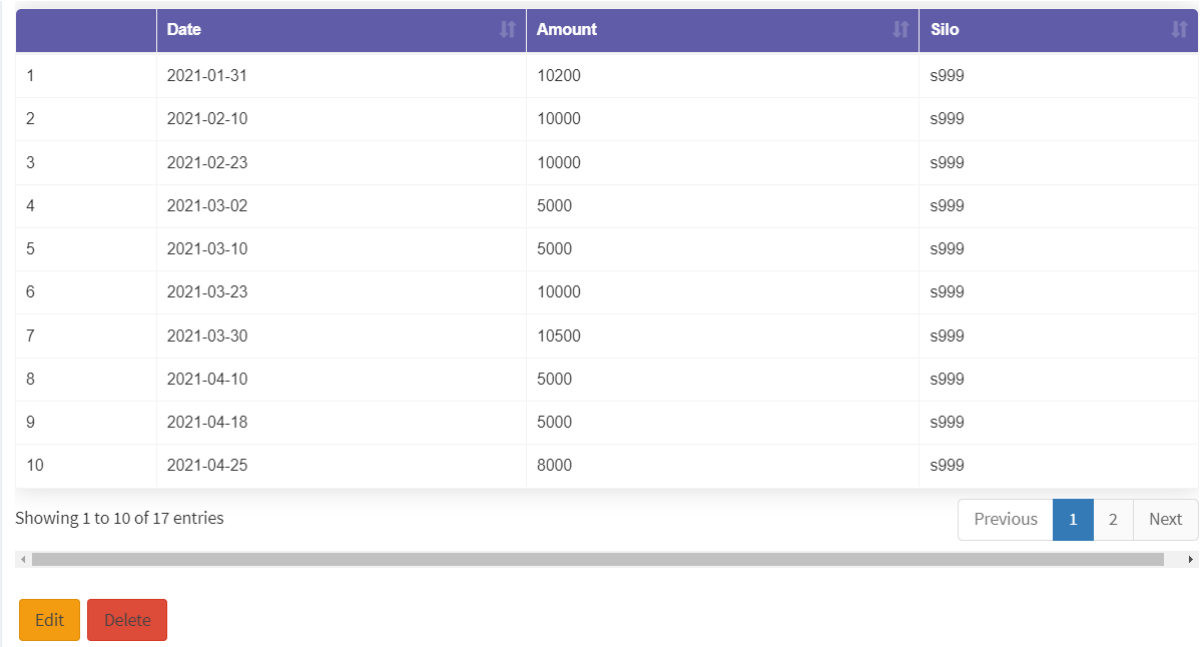


Figure 7: Output data for a batch

The fourth view shows the estimated amounts of feed loads where the user can update them specifying the real amount (Figure 8). These inputs are required in order to perform di□erent analysis for the batch.



	Date	Amount	Silo
1	2021-01-31	10200	s999
2	2021-02-10	10000	s999
3	2021-02-23	10000	s999
4	2021-03-02	5000	s999
5	2021-03-10	5000	s999
6	2021-03-23	10000	s999
7	2021-03-30	10500	s999
8	2021-04-10	5000	s999
9	2021-04-18	5000	s999
10	2021-04-25	8000	s999

Figure 8: Trucks for a batch

These automatic or manual input data is processed by some R scripts that have been created. One of them is in charge of cleaning the sensor data by correcting specific points that are detected as outliers. Another R script is responsible to process this sensor data and determine possible feed loads by using the default density of the sensor. The relation between the volume and the weight of the feed loads change according to the feed density. This value is estimated from the information of the weight unloaded by the trucks to the silo. So the user can update these feed loads by introducing the real amount to this load (Figure 8). So there is a script that check for these amount updates to compute a new density and modifying the sensor data between a range of dates by taking into account the new density. Finally the animals intake is computed by week so there is a R script that is responsible to compute a weekly intake for the pigs in a farm.

A view to control the available feed in the silo has been integrated to the R web application (Figure 9). It shows a processed range of historical data by correcting the points that are considered to be out of the normal that might be produced by di□erent scenarios in the sensor reading. The usefulness of this feature is that it deletes the need of the farmer to climb the silo and observe the amount of available feed. To know if there is the enough amount of feed and plan in time the demand of feed avoiding holidays.

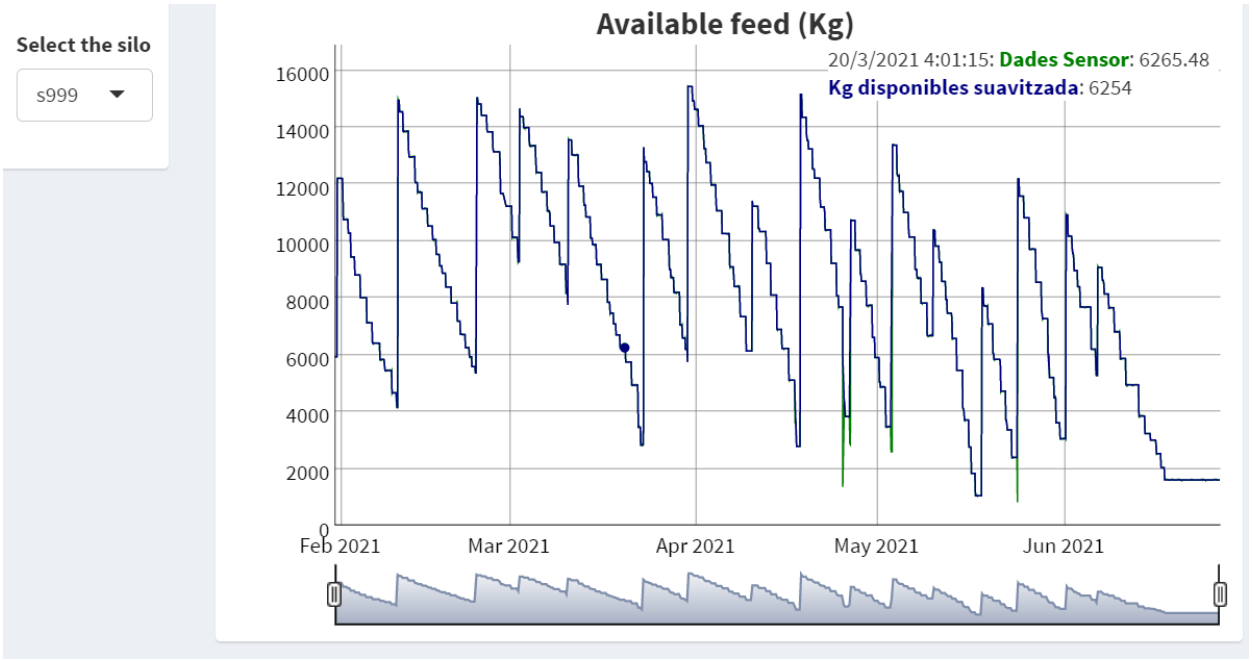


Figure 9: Available feed

This sensor data describe to the farmer what is inside a silo. However by combining this sensor data with other information and models it is possible to get more value of these data such as determining the growth of these animals in a batch. A predictive model has been built in order to estimate the expected intake for the pigs in a batch and compare it with the intake estimated by the sensor (Figure 10).

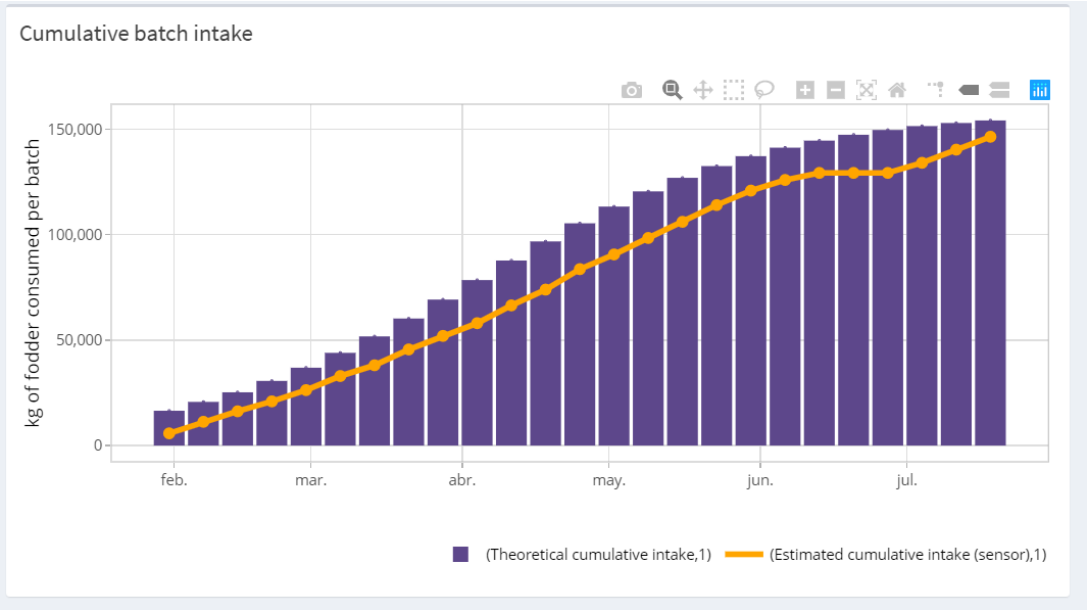


Figure 10: Cumulative batch intake

With this feature the farmer can observe if their animals are consuming as expected or not, that could be caused due to a possible problem that should be detected.

Moreover once it is known how much feed has been assigned to every input of the batch it is possible to determine its growth(Figure 11). It provides the average weight for every input of animals in a specific week.

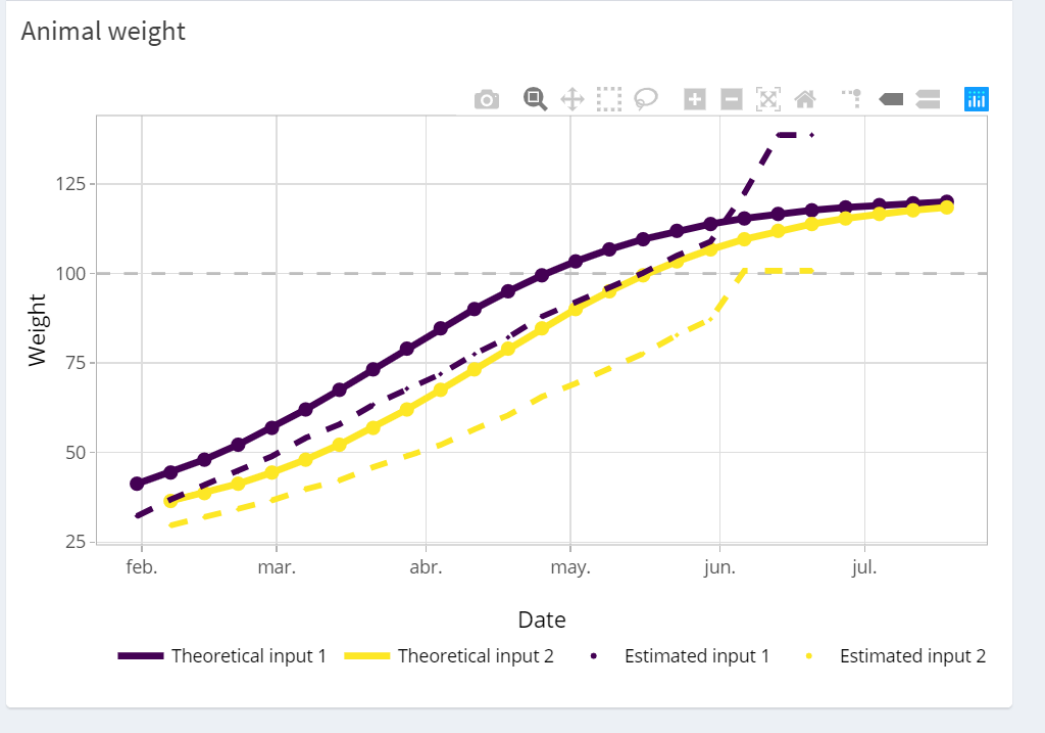


Figure 11: Animal average weight

However it is also interesting to determine the groups of animals that belongs to a range of weights in a specific week to provide more concrete information to the farmer (See Figure 12).

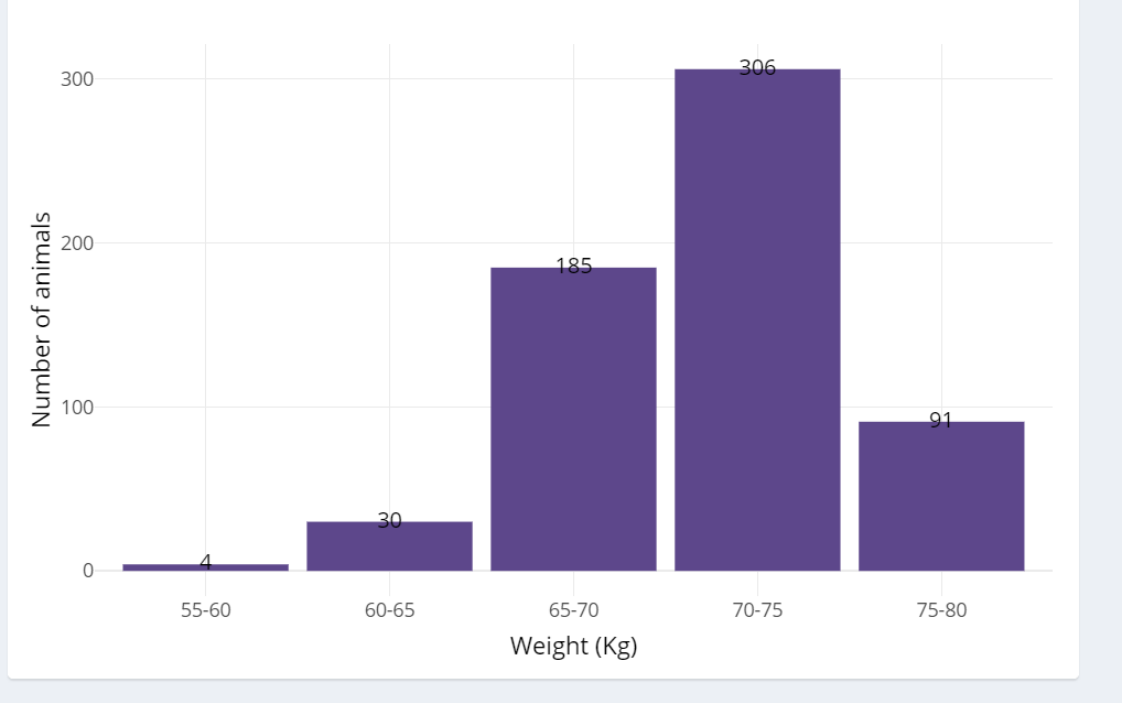


Figure 12: Animal weight distribution in a specific week

Another feature that is integrated in the application is the generation of alarms. A basic alarm has been implemented. It checks for the two last measures obtained by the sensors and it computes a proportion that if is lower than a boundary it sends an email to notify the user of the anomaly.

This case study shows how a sensor that describes the level of solids inside a silo can be used to estimate the evolution of the weight of the animals based on the feed they have consumed and to generate alarms warning of possible anomalies. This generated information allows farmers to detect possible problems in the animals on the farm, such as a growth that is di□erent to the expected, which could indicate that the animals are sick. Therefore, this tool would help in improving farmers' decision making, since the farmer would be able to detect unwanted behaviour in the animals and take the adequate actions in time to alleviate the problem.

6 Conclusions and Future Work

In order to facilitate the process of the digital transformation there is the need of general architectures that supports the evolving requirements that emerge from a wide variety of IoT scenarios. A general architecture provides the guidelines of how can be implemented an IoT solution. One of the most important components of an IoT architecture is the cloud platform that is in charge for collecting , processing, analysing the data and provide the needed tools to the users to interact with the system. For this reason in this work we have presented a cloud architecture that must accomplish with the requirements of IoT scenarios. One of these requirements is to deal with the heterogeneous data of sensors and external services that can be managed by providing the corresponding scripts that are able to interoperate with these devices by using the corresponding protocol such as HTTP and MQTT and store the data into NoSQL databases such as MongoDB. Another important characteristic that must provide the cloud platform is the scalability. It is achieved by providing a modular design such as a microservice approach,distributing these services in di□erent layers such as the presentation layer in order to interact with the user, the logic layer in order to collect the data,proccess it and analyse it and the data layer in order to store the data, but also leveraging container technologies such as Docker in order to be able to integrate more sensors, analysis methods,services and provide horizontal scaling. Docker technology also facilitates the portability of the platform to other IoT scenarios and Cloud providers by only installing this technology into the cloud. Moreover there is also the need to provide security mechanisms such as roles and authentication in order to provide to access to the needed tools and prevent undesired access to the platform. If an IoT solution accompish with all these characteristics then it is able to give support to the use case for a long time. Finally regarding the future work, we are interested in integrating new kind of sensors to analyse critical tasks for the case study that can be assigned to Edge Nodes and provide new analysis to provide valuable output. A functionality that would be interesting to integrate would be the ability to register IoT devices to the platform by using for example QR(Quick Response) codes and the configuration/management of these devices in order to control proprietary devices. Another interesting characteristic for IoT scenarios is product traceability that can be implemented by leveraging Blockchain technologies. Also in order to improve the analysis performance it could be integrated a processing distributed framework such as Apache Spark that provides support for R and Python programming to provide complex analysis with huge amounts of data. Moreover this general cloud architecture could be applied to other scenarios such as healthcare,transportation, supply chains, and others by integrating,removing or replacing the needed components. An analysis should also be carried out to study the performance and robustness of the platform to see how it behaves by performing stress tests to see how many IoT devices it is able to support, as well as the number of users using the application or how it behaves when a user enters valid or unwanted values in the application.

References

[1] Abderrahmen Belfkih, Claude Duvallet, and Bruno Sadeg. “A survey on wireless sensor network databases”. In: *Wireless Networks* 25.8 (2019), pp. 4921–4946. ISSN: 15728196. DOI: 10.1007/s11276-019-02070-y.

[2] D. Berckmans. “General introduction to precision livestock farming”. In: *Animal Frontiers* 7.1 (2017), pp. 6–11. ISSN: 21606064. DOI: 10.2527/af.2017.0102.

[3] Mardiana Binti Mohamad Noor and Wan Haslina Hassan. “Current research on Internet of Things (IoT) security: A survey”. In: *Computer Networks* 148 (Jan. 2019), pp. 283–294. ISSN: 13891286. DOI: 10.1016/j.comnet.2018.11.025.

[4] Carlos Cambra, Sandra Sendra, and Lloret Jaime. “An IoT Service-Oriented System for Agriculture Monitoring”. In: (2017).

[5] Xu Chu et al. “Data cleaning: Overview and emerging challenges”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data* 26-June-20 (2016), pp. 2201–2206. ISSN: 07308078. DOI: 10.1145/2882903.2912574.

[6] Gaia Codeluppi et al. “LoraFarM: A LoRaWAN-based smart farming modular IoT architecture”. In: *Sensors (Switzerland)* 20.7 (2020). ISSN: 14248220. DOI: 10.3390/s20072028.

[7] *Docker v19.03.8*. https://www.docker.com/. Accessed: 2021-04-21. URL: https://www.docker.com/ (visited on 04/21/2021).

[8] Francisco Javier Ferrández-Pastor et al. “Developing ubiquitous sensor network platform using internet of things: Application in precision agriculture”. In: *Sensors (Switzerland)* 16.7 (2016). ISSN: 14248220. DOI: 10.3390/s16071141.

[9] Pengfei Hu et al. “Survey on fog computing: architecture, key technologies, applications and open issues”. In: *Journal of Network and Computer Applications* 98 (2017), pp. 27–42. ISSN: 10958592. DOI: 10.1016/j.jnca.2017.09.002. URL: http://dx.doi.org/10.1016/j.jnca.2017.09.002.

[10] Gonçalo Jesus, António Casimiro, and Anabela Oliveira. “A survey on data quality for dependable monitoring in wireless sensor networks”. In: *Sensors (Switzerland)* 17.9 (2017), pp. 1–23. ISSN: 14248220. DOI: 10.3390/s17092010.

[11] Vaclav Jirkovsky, Marek Obitko, and Vladimir Marik. “Understanding data heterogeneity in the context of cyber-physical systems integration”. In: *IEEE Transactions on Industrial Informatics* 13.2 (2017), pp. 660–667. ISSN: 15513203. DOI: 10.1109/TII.2016.2596101.

[12] Gerda Kamberova. “Understanding the Systematic and Random Errors in Video Sensor Data Department of Computer and Information Science University of Pennsylvania”. In: August (2013).

[13] Vassili Karanassios. “Sensors trends: Smaller, cheaper, smarter, faster and under wireless control”. In: *FLEPS 2019 - IEEE International Conference on Flexible and Printable Sensors and Systems, Proceedings* (2019), pp. 2019–2022. DOI: 10.1109/FLEPS.2019.8792272.

[14] *LDAP Account Manager v7.1*. https://www.ldap-account-manager.org/lamcms/. Accessed: 2021-04-21. URL: https://www.ldap-account-manager.org/lamcms/ (visited on 04/21/2021).

[15] *MCSystems*. https://mcsystems.es/ca/. Accessed: 2021-04-21. URL: https://mcsystems.es/ca/ (visited on 04/21/2021).

[16] *MongoDB v4.0.14*. https://www.mongodb.com/. Accessed: 2021-04-21. URL: https://www.mongodb.com/ (visited on 04/21/2021).

[17] *Node.js v8.16.1*. https://nodejs.org/es/. Accessed: 2021-04-21. URL: https://nodejs.org/es/ (visited on 04/21/2021).

[18] *OpenLDAP v1.2.5*. https://www.openldap.org/. Accessed: 2021-04-21. URL: https://www.openldap.org/ (visited on 04/21/2021).

[19] Amine Roukh et al. "WALLeSMART: Cloud Platform for Smart Farming". In: *ACM International Conference Proceeding Series* (2020), pp. 1–4. DOI: 10.1145/3400903.3401690.

[20] Kinza Shafique et al. "Internet of Things (IoT) for Next-Generation Smart Systems : A Review of Current Challenges , Future Trends and Prospects for Emerging 5G-IoT Scenarios". In: 8 (2021).

[21] Amit Sharma, Pradeep Kumar Singh, and Yugal Kumar. "An integrated fire detection system using IoT and image processing technique for smart cities". In: *Sustainable Cities and Society* 61.December 2019 (2020), p. 102332. ISSN: 22106707. DOI: 10.1016/j.scs.2020.102332. URL: https://doi.org/10.1016/j.scs.2020.102332.

[22] *Shiny v1.3.2*. https://shiny.rstudio.com/. Accessed: 2021-04-21. URL: https://shiny.rstudio.com/ (visited on 04/21/2021).

[23] *ShinyProxy v2.3.0*. https://shinyproxy.io/. Accessed: 2021-04-21. URL: https://shinyproxy.io/ (visited on 04/21/2021).

[24] Joseph D. Stevens and Talal Shaikh. "MicroCEA: Developing a Personal Urban Smart Farming Device". In: *2nd International Conference on Smart Grid and Smart Cities, ICSGSC 2018* (2018), pp. 49–56. DOI: 10.1109/ICSGSC.2018.8541311.

[25] *Stormy*. https://stormy.udl.cat/. Accessed: 2021-04-21. URL: https://stormy.udl.cat/ (visited on 04/21/2021).

[26] Ikbal Taleb, Mohamed Adel Serhani, and Rachida Dssouli. "Big Data Quality: A Survey". In: *Proceedings - 2018 IEEE International Congress on Big Data, BigData Congress 2018 - Part of the 2018 IEEE World Congress on Services* (2018), pp. 166–173. DOI: 10.1109/BigDataCongress.2018.00029.

[27] Mohit Taneja et al. "Fog assisted application support for animal behaviour analysis and health monitoring in dairy farming". In: *IEEE World Forum on Internet of Things, WF-IoT 2018 - Proceedings* 2018-Janua (2018), pp. 819–824. DOI: 10.1109/WF-IoT.2018.8355141.

[28] Sergio Trilles, Alberto González-Pérez, and Joaquín Huerta. "An IoT platform based on microservices and serverless paradigms for smart farming purposes". In: *Sensors (Switzerland)* 20.8 (2020). ISSN: 14248220. DOI: 10.3390/s20082418.

[29] Blesson Varghese and Rajkumar Buyya. "Next generation cloud computing: New trends and research directions". In: *Future Generation Computer Systems* 79 (2018), pp. 849–861. ISSN: 0167739X. DOI: 10.1016/j.future.2017.09.020. arXiv: 1707.07452. URL: http://dx.doi.org/10.1016/j.future.2017.09.020.

[30] Andreas Waadt, Guido Bruck, and Peter Jung. “Positioning Systems and Technologies”. In: (2017).

[31] Cholatip Yawut and Sathapath Kilaso. “A Wireless Sensor Network for Weather and Disaster Alarm Systems”. In: *International Conference on Information and Electronics Engineering* 6.February (2011), pp. 155–159.