Developing a cloud based system for bird nests environment and behavior data monitoring and analysis, following the Internet of Things paradigm

Desarrollo de un sistema basado en la nube para el monitoreo de un nido de aves, su ambiente y comportamiento, siguiendo el paradigma del Internet de las Cosas

Alexis Gamboa-Soto[1]

Gamboa-Soto, A. Developing a cloud based system for bird nests environment and behavior data monitoring and analysis, following the Internet of Things paradigm. *Tecnología en Marcha*. Vol. 33, especial Movilidad estudiantil. Pág 32-44.

https://doi.org/10.18845/tm.v33i7.5472

1 Computer engineer. Email: agamboa@tum.de München, Germany Master of Data Analysis and Engineering at the Technical University of Munich. Germany.

Keywords

NodeJS; IoT; Cloud Computing; API; Bird Nests Monitoring; animal behavior; environment monitoring; smart environments; wren bird; Artificial Intelligence.

Abstract

This work presents the design and development of a cloud based system prototype for monitoring birds in the wild, following the Internet of Things paradigm principles and tendencies. The behavior and population of birds represent an important metric to measure the health of an ecosystem, becoming a very important tool to study the climate change influence on the environment. The system is built using NodeJs for the backend and a Cloud Computing Platform as a Service technology is utilized for the deployment to enhance the accessibility and scalability, while reducing operation and deployment costs. For this project, it is assumed the challenge of deploying the prototype without any monetary investment. The research main purpose is to set a real time bird nest monitoring web system, for environment and animal behavior data collection and analysis, designed to define a communication standard for the integration from custom made devices utilized to collect the data, allowing for a better integration of different contraptions built to obtain wildlife data with minimum interference and giving enough dynamism with stateless communication protocols to handle new data types from a wide variety of sensors. The standardization and centralization aims to solve the current issue of remotely collecting data from monitoring devices and the ability to share raw data among wildlife researchers, offering worldwide availability through a web portal for the users to access the collected data.

Palabras clave

NodeJS; Internet de las cosas; Computación en la nube; interfaz de programación de aplicaciones (API); monitoreo de vida salvaje; comportamiento animal; entornos inteligentes; reyezuelo; inteligencia artificial.

Resumen

Este trabajo presenta el diseño y desarrollo de un prototipo basado en la nube para monitorear aves en en su entorno natural, siguiendo los principios y tendencias del paradigma de Internet de las Cosas. El comportamiento y las poblaciones de aves representan una útil métrica para medir la salud de un ecosistema, volviéndose así en una importante herramienta para estudiar las repercusiones del cambio climático en el medio ambiente. El sistema se construye utilizando NodeJs para el backend y se despliega utilizando una plataforma de computación en la nube como servicio a fin de mejorar la accesibilidad y la escalabilidad, mientras se reducen los costos de implementación y operación. Para este proyecto, se asume el reto de cero inversión en la implementación del prototipo. El objetivo principal de la investigación es establecer un sistema web de monitoreo de nidos de aves en tiempo real, para recopilar y analizar datos sobre el comportamiento animal y ambiental, diseñado para definir un estándar de comunicación para la integración de dispositivos personalizados utilizados para recopilar los datos, permitiendo una mejor integración de diferentes artilugios construidos para obtener datos de vida salvaje con la mínima interferencia y dotar de suficiente dinamismo a los protocolos de comunicación para manejar nuevos tipos de datos obtenidos de una amplia variedad de sensores. La estandarización y centralización tiene como objetivo resolver el problema actual de recopilación remota de datos de dispositivos de monitoreo y la capacidad de compartir datos entre investigadores de vida salvaje, ofreciendo disponibilidad mundial a través de un portal web para que los usuarios accedan a los las lecturas recolectadas en distintos entornos.

Introduction

Climate change becomes more relevant with the increasing global warming effect accelerating during the last decades [1], and the research of its repercussions on the ecosystems worldwide is vital to get a better understanding of the consequences and find solutions to help mitigate the current and projected harm on the environment. Birds in specific, are vital for the ecosystems and for their migratory nature, serve as a very good metric to measure the health of different ecosystems, like the impact from the climate change in the populations [2] and the relationship between the birds behavior and their vital role within the ecosystems [3]. Monitoring wildlife and its interaction with a given environment, grants the researchers valuable information to find correlations between environment, the animal behavior and their ecosystem.

Given the importance of data acquisition of natural habitats and animal populations, the mechatronic department of the cooperative University of Baden Wurttemberg Karlsruhe (DHBW Karlsruhe) developed different monitoring devices, but the external communication and the management of data limited the value that all the projects could have given to the researchers biologists -specifically birds research for this project-, and the lack of a standardize communication between devices disable any integration with old systems.

The IoT (Internet of Things) paradigm revolves around connectivity and how new smaller and simpler devices, can connect to a network and work simultaneously for a common purpose, generating each time more data about our environment and creating a new way to relate collected digital data with the surroundings, discovering useful information and patterns and allowing for a better digital interpretation of the physical world. On the challenges met by the researchers working on the monitoring devices, the IoT paradigm presents a structure for a platform where devices can communicate on one common language and data can be collected, analyzed and published under one standard, while keeping the flexibility for new devices and data structures to be integrated into the whole platform ecosystem. A prototype for this cloud base system for the integration of sensing devices, data storage and data presentation for the real time monitoring of bird nests is developed on this research, and its name is Cloud Nest.

Related Work

Implementations to utilize electronic sensors to monitor animals, collect and analyze data have been implemented on different scenarios, from poultry farm monitoring using Internet of Things and Wireless Sensor Networks [4], to wildlife image processing to automatically identify a specific species from a monitoring network [5]. On this research, the main scope is over data acquisition and sharing potential, by offering an open platform for researchers to upload, store and share data collected from their different studies. The Cloud Nest platform is specifically designed for birds monitoring, but might as well be expanded to include any other use case for animal monitoring. The idea of creating platform is to create a common ground for researchers to implement different analysis tools over the data, giving them the opportunity to easily reutilize historic records and incorporate other researchers data. A secondary attractive aspect of the project presented on this document, consist on develop a working prototype being able to be deployed and tested with no monetary investment.

DHBW Mechatronic department projects background

The Mechatronic department of the DHBW Karlsruhe have developed different monitoring devices for birds nests, among other data collecting devices for animal research, e.g. a squirrel feeder which takes samples and collect data about the animal and its surroundings. The collected data from the animal behavior and its environment, which involves animal pictures and video, environment temperature, light levels, animal weight, pressure and humidity levels, are very important for biologist research, and automatizing the sample data collection turns this monitoring devices, alongside other advantages like minimizing the human intervention into the animal environment, which can directly interfere with the metrics and the animal behavior, and the capacity to register anomalies in real time on a continue monitoring system, turns these devices into a powerful tool for the biologists and their research, since this could cover the Data Collection, Data Storage and the platform for the Data Analysis required for an animal monitoring plan [6].

One of the most recent project developed by the mechatronics department of the DHBW for this purpose is named “Control and registration of different parameters in a nesting box for birds” (“*Überwachung und Registrierung verschiedener Parameter in einem Nistkasten für Singvögel*”). It was developed by the students Florian Schopp and Steven Watterott on 2013 and was supervised by the professor Dr. Thomas Haalboom. The student project consists on a device, able to measure the birds weight, nest temperature, pressure levels, light levels, besides taking pictures and a temperature array of the interior of the nest, and save this data on a SD-Card or send them via bluetooth to be accessible for the end user. This project inspired the origin of the research and helped to shape different aspects, like communication constraints of the data collecting devices, the different hardware components available for device developers, the type of data and connectivity of the devices, the automatization of the systems to required the less human interaction and finally, the integration challenges and problems to use the appliance as part of a network. This problems can be solved on the IoT paradigm, term first coined by Kevin Ashton in 1999 in the context of supply chain management [7], which main goal is making a computer sense information without the aid of human intervention [8].

The project was intended to be used by the ornithologist Dr. Stefan Bosch, who equipped bird nests with IP cameras to make observations of birds and its behavior and required a more complete system to monitor the nests, nevertheless, it was tested and used just as a prototype, the data accessibility and system difficult operation, limited its use, and Dr. Stefan Bosch was not able to incorporate it on his monitored nests. This is not the first project which have the same ending, different past devices developed to monitor the nests were never incorporated into Dr. Stefan research. This scenario shows the importance of the automatization of the data collection process and the accessibility of the data, where a cloud centric application following the IoT paradigm structure can give collection process the automatization and the data the worldwide exposition [8] required to solve both issues.

General System Description

System design and description

The system, named Cloud Nest, is an online platform that uses a standard for the communication and for the data structures, which can be as complete to be used by different systems and devices, but as flexible to let different type of data to be sent and received, and can handle and display the data from any client with internet connection. The needed standard can be found on the representational state transfer (REST) software architectural style, under the HTTP protocol for the communication and JSON as an open format model for the data. This system designed to standardize the transferred data, solves the problem of handling different data sets representing different measures, and the integration of different devices. This measure leads to an increasing potential with each device that joins the system and offers the researchers a tool to exploit all the generated data with no need to use new different applications for each new device, while gives them a remote access to analyze and manage the data collected from devices worldwide. The implemented system prototype consist on two main modules, one oriented to end-users, researchers and system managers (Web-application), and the other one to the device and hardware developers (Web-Service).

The backend, programed in NodeJs, backups the web-application and the external services, both through a REST-ful API. The authentication of the application is implemented by user sessions, stored in the database, and it is required to make any transaction with the API. This backend will handle all the incoming traffic from the devices, which can scale horizontally or vertically to support a heavier load of data, task that can be done within minutes thanks to its cloud service structure. All the model interactions form the devices and sensors are made under the API service of the application. The communication with the devices works under the REST architecture, selected by its potential on stateless (no sessions required) communication [9], and the authentication will work under a device key, which is a value stored in the Database to authenticate the device and allow the access just for the device and their sensors data.

The front-end is a minimalist web-application created with Angular framework, following Google Material Design pattern and a hierarchy Drill-Down user interface for the user interaction and navigation through the assets of the system.

Features

The main capabilities of the system are summarize on five features:

- Register and storage of Device reading data: Automated registry of data on the system from the sensor to the data base.
- Access and management of the collected data: Access to all the collected data registered from the devices sent data.
- Users, devices and sensors management: Capability to add, delete an edit users, devices and sensors in the system.
- Devices and sensors definition and configuration management: Capability to set individual attributes to each device and sensor, a long as being able to set the different configurations own attributes.
- Live view of devices readings: Display of live data from the selected devices as they are registered in the system.

More details and technical documentation can be found on the project repository: https://github.com/agamboa23/CloudNest

System Structure and Architecture Design

Architecture of the Model

The prototype consist on a client-server application. The server side of the system runs entirely on cloud technologies. For the application server, Heroku will be the cloud service running a NodeJs server and hosting the web application. The storage are also cloud provided. The devices will communicate directly with the NodeJs server and the end user will use a web application which will be powered by the same NodeJs server. Both, system administrator and researchers roles, are handle by the back-office running on the same web application.

Backend structure

The backend of the system (see figure 1) is configured over three different cloud services, one for the application and two for the storage. Heroku is a cloud PaaS (Platform as Services) where the NodeJs server of the system prototype runs. The two services of storage are MongoLab -now called mLab- and Google Drive. Google Drive will store all the media and Blobs (Binary large objects) of the system, while mLab (Mongo Lab) will be the main data-base of the system. All the business logic data, the readings meta-data and the primitive type readings data will be store on mLab. The device interacts through the API service of cloudNest for data posting and configuration setting. When it has to upload a large file, like a media file, it has to post the meta-data of the file over the API service. The api will answer with one URI, already authenticated with he Google Drive service and with the metadata set, ready to upload. The upload then is made directly with the Google Drive Service as a Resumable Upload. The device then handles by its own the way it is going to upload the file, by chunks or the whole file. The URI is unique to the meta-data related to the file. This way, the prototype does not have to handle all the blobs uploads and downloads directly, which would consume most of the system resources in the case of handle them. This is a workaround to keep the costs of deployment at $0, without the common restrictions from services like Amazon Web Services, where the free tier expires within 12 months and a valid credit card has to be registered. Nevertheless, on a production environment, it is strongly recommended to use a dedicated service for the storage, since Google Drive have limit on reads and post on their services which restrict scalability.

The whole system uses a framework named SailsJs. This framework emulates the Model View Controller pattern, like Ruby on Rails, but also provide different features to better suit Cloud Nest applications requirements, like data-driven APIs with a scalable, service-oriented architecture. The Front End of the system is powered by the backend API built from the SailsJs framework.

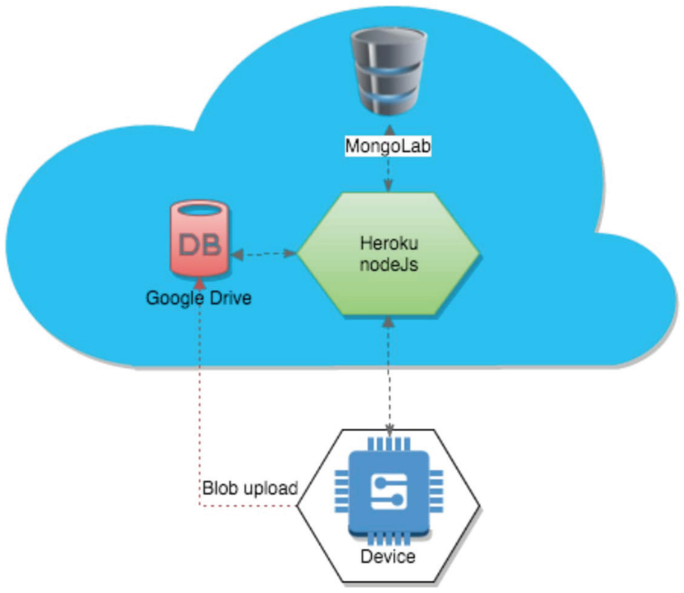


Figure 1. A model for the basic structure of the system.

Data Model

Mongo is used as the main database of the system, and Google Drive is limited for the storage of media and big files. Mongo is a documental, NoSql, scheme-less data base. The system is really flexible about the data models, but some of the critic attributes have to be set correctly to be successfully stored on the system, as the required attributes or the types of some specific attributes. The Models with the most flexibility are the configuration and the ModuleDataStructure collections, which allows any new attribute. The data model has a strong hierarchy pattern, but it allows the system to support different instances of the system independently from each other, using the organization as the root, which makes possible to offer the system like an Software as a Service (SaaS) solution. The navigation for the user interface is required to have a flexible and user-friendly drill-down pattern, so it can become intuitive and easy to use and understand for the end user.

Each entity is represented by a collection on the mongo data base. Here is a list with a description for each of the entities of the data model (see figure 2):

Organization: Symbolize an organization or institution, it is the highest hierarchy concept over the model is constructed. This approach allows a perspective of the system as a Saas (Software as a Service), allowing different organizations to use the same platform.

User: For every organization, a group of users are associated with it. Every organization has an unique set of users.

Region: Represents a geographic region. Each organization is related with one or more regions. Since the system is oriented to biology research, the regions are not constraint within political borders, since political and ecological borders hardly coincide. The region is highly abstract.

Location: Represents an specific location within a region. Each Region is related with one or more locations.

Spot: Represents an specific Spot within a location, usually related with a nest. Each Location is related with one or more spots. The Spot is more accurate and represents a geographical point, not an area like the last two.

Device: Represents an specific Device. A spot can be related with one or more devices.

Module: Represents a sensor or a group of electronic components within a Device designated to get one specific data measurement or metric. A Device can be related with one or more modules. The module must be based on the most atomic data reading unit the device get from the measurer module, regardless of the electronic components needed to collect it.

ModuleData: The measurements data or generated data is the representation of the modules sensors readings data sent from the device to the system.

Configuration:

Device Configuration: Each device has an individual configuration profile. The attributes can change from device to device.

Module Configuration: Each module within a device has an individual configuration profile. The attributes can change from module to module.

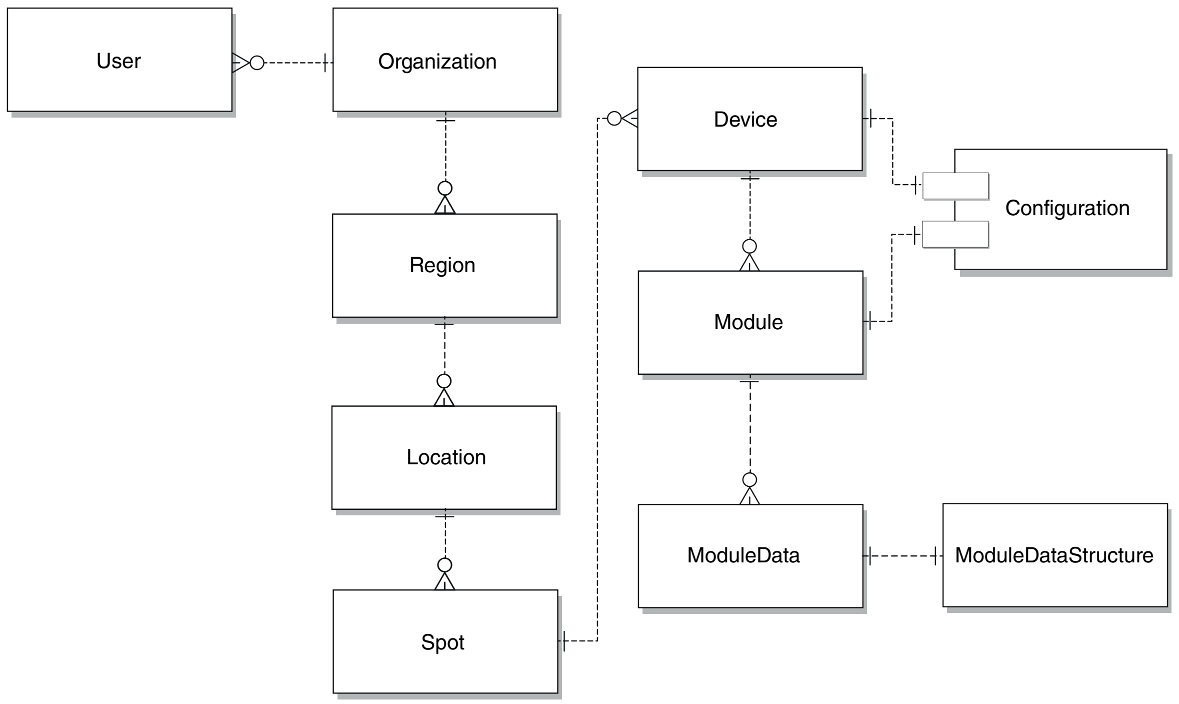


Figure 2. Domain Model.

IoT paradigm influence and technical decisions justifications

Cloud Nest Architecture

The technologies elected for the architecture and design of the system are based on the IoT paradigm and the proposed technologies are analyze and selected to better suit the requirements. One of the most difficult decision is how to base the communication and the architecture of the system. For the architecture there are two big choices, a first generation standard represented by Web Service Definition Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery and Integration (UDDI) or a newer Representational State Transfer (REST) architecture. The software community tend to find REST architecture not only easier to learn and develop, but also more suitable to connect smart things, which is an elemental principles of the IoT paradigm, as research results have shown [10]. There are already different successful platforms developed for IoT applications using a REST architecture, as Sensor Network (https://sensor.network.com/REST/login.jsp) which is a global exchange for sensor information, or Pachtube (https://www.pachtube.com), now named “xively”, which REST architecture implementation evidences its suitability for an IoT platform [11].

Since there is to much flexibility on the REST architecture, the general structure of the service of Cloud Nest was developed using this protocol, keeping the communication stateless and the messages lightweight, two of REST’s strongest points. For similar scenarios where standardized communication among web components is required, big software companies support REST as a great alternative[12].

Cloud Nest DataBase

An IoT platform is planed to collect, store and transform a huge amount of data, and the scalability of the system is vital for its potential. Traditional databases, like SQL relational Data Bases, use a scheme for the data model and making multiple writes at the same set of data is not allowed, which causes a bottleneck when millions of registers need to be stored in the database. The SQL databases tend to have consistency at all time, which requires to block a table when data is written into it, and it becomes a problem when there are thousands of devices sending readings each second to be stored in the data base, limiting the concurrency. Some NoSQL databases, like the documental database MongoDB, have the flexibility to allow multiple writing operations simultaneously and work with a more granular locking system, on the premise that the consistency is not met at all time, but it temporary arrives. It may be a problem for critical systems like an internet banking application, but for an IoT platform with millions of register, to false-read one of the registers, have a minimum impact over the system and the results. The all-time consistency is a price to pay for the advantages of the NoSQL database, like its high speed access to mass data. One example is MongoDB, which can outperform MySQL by a factor of 10 accessing mass data sets that exceeds 50 GB [13].

For Cloud Nest, MongoDB was elected. It is scheme-less, and the attributes of the collections can change. It is one of the biggest advantages of this database, since the sensors and devices will use different data types and different attributes.

Cloud Nest Runtime and server engine

Traditional web server applications made in Java or .NET Core are well built for applications which requires heavy processing for the engines behind Java or C# languages, which can easily outperform javascript ones. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine, this is why NodeJS is not ideal for heavy data processing but Node.js uses an event-driven, non-blocking Input/Output (I/O) lightweight and efficient model, which makes it a great choice for request handling and transferring data. The asynchronous event driven I/O feature, helps NodeJS with concurrent request handling, that is the reason why NodeJs outperform traditional Java or .Net web services at requests per second [14] and suits it better to handle multiple requests from a high amount of devices concurrently. NodeJs have other different advantages, like simplicity, big community of developers, which represent big amount of support and documentation, becoming a viable technology to be applied on Cloud Nest.

Cloud Nest Infrastructure

One of the challenges of the prototype was to deploy it without having any own computing infrastructure to install and run the system, for which using cloud services was not only a design decision from the IoT paradigm recommendations perspective, but was also an alternative to deploy an online prototype without the hardware infrastructure. Other of the advantages of using a cloud architecture is the deployment time, since all the environment is already set and the deployment consist on the environment configuration and upload of the system source code.

The cloud Infrastructure as a Service perspective is built around the configuration over the development of the infrastructure and the environment, simplifying the deployment and management of a whole platform, as shown on the figures from Heroku (figure 3), where all the tasks required to set up an infrastructure is abstracted into a service which only needs to be configure and managed. Heroku was selected as the cloud service to host Cloud Nest for its simplicity and low requirements. For storage, mLab, a cloud storage service for MongoDB, and Google drive, were elected. MLab offers a 500 MB storage free package, for all the business and collected data for the system, while Google Drive will handle all the media and large files, with a free tier limit of 10 Gigabytes.

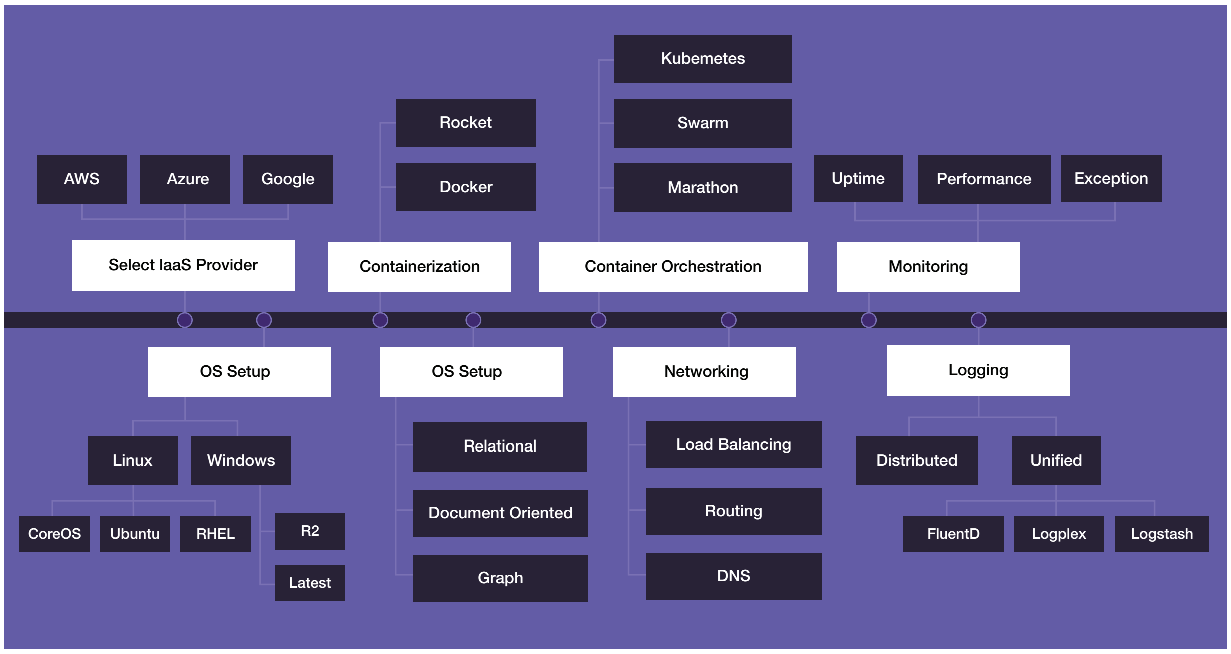


Figure 3. Structure of the Heroku service. Source: extracted from [15]

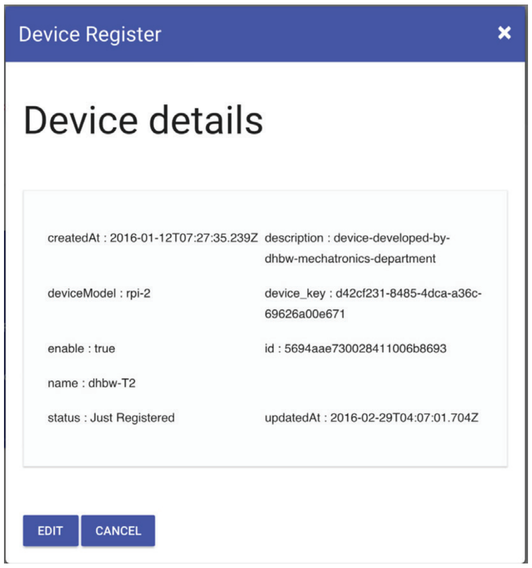
It is important to highlight that the increase of resources needed on an eventual increase of the active users of the system, have to be backed up by a payment proportional with the new resources required. Cloud services costs are based on the resources demanded.

Prototype and results

Deployment of the prototype

The system was deployed on a Heroku instance (a "dynamo") and it is being used in the cooperative university of Baden-Wurttemberg as a prototype for further monitoring systems development. The repository of the system, alongside the API reference of CloudNest, can be both found in the following link: https://github.com/agamboa23/CloudNest. Some examples to post data are provided on the repository. The system is currently live on: https://cloudnest.herokuapp.com/. The deployment was successful. Data is successfully posted and retrieved form the device API web-services using the given token as identification, and the users can log in the web application and find all the assets related to their organization. A main administrator user can access the backend to handle new users and new organizations, which can work completely independently from each other. Data was posted from a raspberry pi running the data posting sample written on the programming language C available in the repository.

The web application allows the user to register organizations, locations, spots, devices and custom configuration for any device or module (see figure 4). An example for a registered spot with its corresponding geographic location and a registered device can be found on figure 5. For each device, a custom configuration can be set, allowing for custom data types to be stored and displayed on the system. Time series data is shown on a chart (figure 5), if the data type is not recognized, the raw data will be displayed.



Device Register

Device details

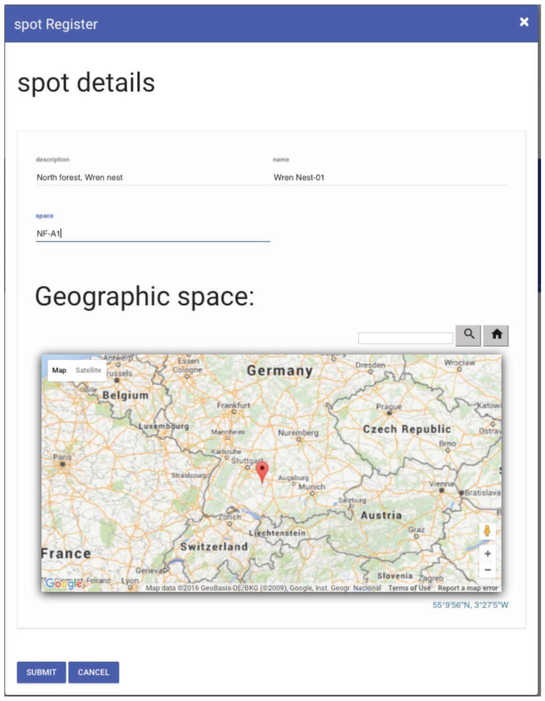
createdAt : 2016-01-12T07:27:35.239Z description : device-developed-by-dhbw-mechatronics-department

deviceModel : rpi-2 device_key : d42cf231-8485-4dca-a36c-69626a00e671

enable : true id : 5694aae730028411006b8693

name : dhbw-T2

status : Just Registered updatedAt : 2016-02-29T04:07:01.704Z



spot Register

spot details

description: North forest, Wren nest

name: Wren Nest-01

space: NF-A1

Geographic space:

55°9'56"N, 3°27'5"W

Figure 4. Examples of a registered spot and device on the system.

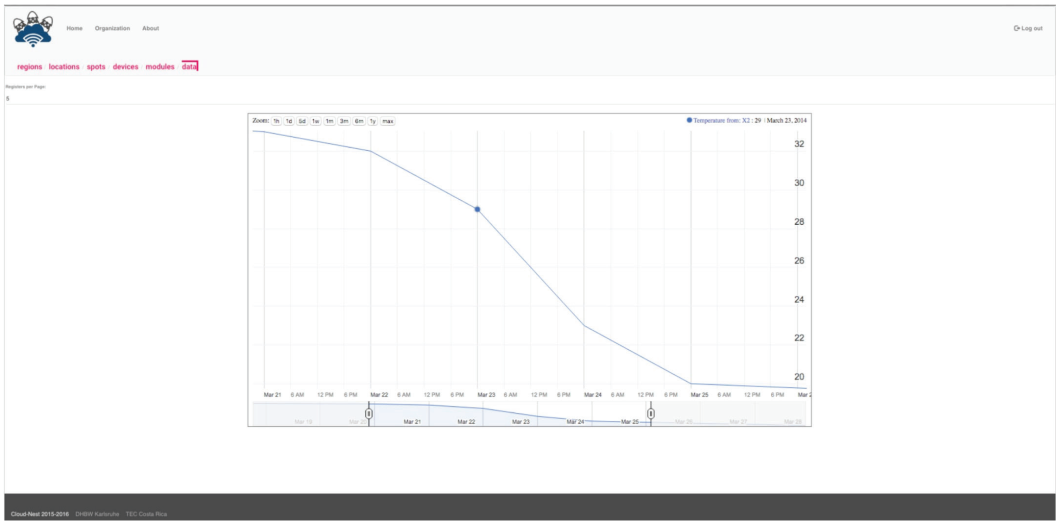


Figure 5. Example of a chart displaying readings from a module.

There were some aspects that could not be completed in the system. One, is the live-streaming of video, alongside the web-socket connection of the system to sustain the live-data view from the system. User is required to manually interact with the system to get the last stored data. Another aspect which was not completed were the users policies and users permissions, since for the moment, any validated user of the system, has permissions to run any query and access any web method of the API. Other than this issues, all the other features were successfully developed and deployed. Any device capable of making a HTTP request over the internet is able to post and read data form the system, once it is registered, and any user with a browser and internet connection can access the web application to manage and access the system remotely.

Conclusions and further work

Further work

Cloud Nest is just a functional prototype, it does not comply with the software quality requirements needed for a production system. It may present important bugs and need to be tested and maintained for its good operation. The live-stream modules need to be incorporated to the server side and the web socket need to be enabled for the web application to comply with the live-stream feature. The cloud services elected may not be the best to host the application depending on the needed scale, since Amazon web services could offer a better price package for higher resource requirements, it all depends on the budget and estimated resource usage. For a production system, a new process should be made to decide the best services to host the platform, considering the new variables of budget and planned users. The Module Data Readings should be redesign as time-series to have a better management of the readings and make it more suitable for analysis. An ETL (Extract, transform and load) module have to be integrated to the design of the system, since it is necessary for more complicated queries and data analysis. For the communication protocol, HTTP was selected, nevertheless, there are different benchmarks that show that different protocols to communicate the devices may be more efficient for an IoT platform, like the CoAP, an Internet Engineering Task Force (IETF) protocol providing a Low-power and Lossy Network (LLNs) with a RESTful architecture, which represent advantages for power consumption and performance over HTTP [16].

For the User-interface, a Tree map UI element could be more attractive and more easy for the user to interact with, instead of drilling down through a master-detail model, giving a better general perspective of all the assets of the system.

Conclusion

Cloud Nest is a functional prototype, free of deployment and testing costs, which proves that when building a cloud based platform to monitor and collect data of animals environments, like birds nest, an IoT platform running on cloud services and built using modern available technologies, is a suitable solution. Cloud Nest complies the requirements for an animal monitoring data collection system, allowing for real time and historical data access, measurement devices integration and remote monitoring of the studied environment. Cloud Nest solves the data integration problem setting a standard for the communication, based on the IoT paradigm, establishing HTTP as protocol and JSON as data structures to send and receive data using a REST architecture, enabling further monitoring devices to become part of the system and not leaving old devices obsoletes, centralizing all the collected data and giving the end user, like the biology researchers, an user friendly platform to analyze and access the monitoring data remotely from any web browser, without any installation required.

Acknowledgment

I would like to thank professors Thomas Haalboom from the DHBW Karlsruhe and Cesar Garita from the ITCR for all the support before, during and after the whole research, making possible to accomplish the development of the Cloud Nest prototype. I would also like to thank the university ITCR and the Baden Wurttemberg *Stiftung* for granting the financial support for the research, and finally I would to thank the DHBW Karlsruhe for allowing me to work with them in the project, giving me access to all the resources required for its development.

References

[1] IPCC, 2014: Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. IPCC, Geneva, Switzerland, 151 pp.

[2] Both, C., Bouwhuis, S., Lessells, C. M., & Visser, M. E. (2006). Climate change and population declines in a long-distance migratory bird. *Nature, 441*(7089), 81.

[3] Pejchar, L., Pringle, R. M., Ranganathan, J., Zook, J. R., Duran, G., Oviedo, F., & Daily, G. C. (2008). Birds as agents of seed dispersal in a human-dominated landscape in southern Costa Rica. *Biological Conservation, 141*(2), 536-544.

[4] Mahale, R. B., & Sonavane, S. S. (2016). Smart Poultry Farm Monitoring Using IOT and Wireless Sensor Networks. *International Journal of Advanced Research in Computer Science, 7*(3).

[5] Elias, A. R., Golubovic, N., Krintz, C., & Wolski, R. (2017, April). Where's the Bear?-Automating Wildlife Image Processing Using IoT and Edge Cloud Systems. In *Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on* (pp. 247-258). IEEE.

[6] Morrison, M. L. (2013). Wildlife restoration: techniques for habitat analysis and animal monitoring (Vol. 1). Island Press.

[7] Ashton, K. (2009). RFID Journal, 1.*That 'Internet of Things' Thing.* Recovered from http://www.rfidjournal.com/articles/view?4986

[8] Gubbi, J., Buyya, R., Marusic, S. & Palaniswami, M. (September 2013). *Internet of Things (IoT): A vision, architectural elements, and future directions Future Generation Computer Systems*, 29, 1645-1660. doi: 10.1016/j.future.2013.01.010

[9] Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation, University of California, Irvine).

[10] Guinard, D., Ion, I., & Mayer, S. (2011, December). *In search of an internet of things service architecture: REST or WS-*? A developers' perspective.* In International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services (pp. 326-337). Springer Berlin Heidelberg.

[11] Zhang, X., Wen, Z., Wu, Y., & Zou, J. (2011, May). *The implementation and application of the internet of things platform based on the rest architecture.* In Business Management and Electronic Information (BMEI), 2011 International Conference on (Vol. 2, pp. 43-45). IEEE.

[12] Rodriguez, A. (2015). *Restful web services: The basics. IBM developerWorks.* Recovered from https://www.ibm.com/developerworks/library/ws-restful/index.html

[13] Tauro, C. J., Aravindh, S., & Shreeharsha, A. B. (2012). *Comparative study of the new generation, agile, scalable, high performance NOSQL databases.* International Journal of Computer Applications, 48(20), 1-4.

[14] Doglio, F. (2015). *Pro REST API Development with Node. js.* Apress. Recovered from https://www.apress.com/gp/book/9781484209172#otherversion=9781484209189

[15] Rockford, M (2017). *8 Reasons This Salesforce Developer Loves Heroku.* Recovered from https://medium.com/@GoRadialspark/8-reasons-this-salesforce-developer-loves-heroku-60159426d167

[16] Colitti, W., Steenhaut, K., & De Caro, N. (2011). *Integrating wireless sensor networks with the web. Extending the Internet to Low power and Lossy Networks* (IP+ SN 2011).