Cadi Ayyad University

Higher School of Technology in Essaouira

Internship Report

In order to obtain the Professional License in Computer Systems Engineering and Software (ISIL) :

Under the Theme :

Cows Management System

Carried out at Agri4.0, Agadir

From 03/04/2023 to 06/06/2023

Framed by :
Bella Abdelouahab

Supervised By :
Mr. Fouad Tabit
Mr. Azdin Guezaz

Academic Year : 2022-2023

Acknowledgment

First and foremost, I extend my deepest gratitude to God, whose unwavering guidance has been instrumental in both achieving this significant milestone and navigating the various aspects of my life's journey.

I would like to take this opportunity to wholeheartedly express my heartfelt appreciation to my beloved parents. Their boundless love, care, and daily connection have had an immeasurable impact on my overall well-being. Their unwavering support has not only nurtured my soul but also contributed to a profound sense of fulfillment.

Furthermore, I am indebted to the entire staff of the esteemed High School of Technology Essaouira. Their dedication to delivering exceptional training and education has undoubtedly played a crucial role in enhancing my skills and knowledge, ultimately leading to a more proficient execution of this internship. Their unwavering dedication to excellence has truly made a lasting impact on my professional growth, leaving an indelible mark that has shaped me profoundly.

Lastly, I want to express my deepest and heartfelt gratitude to the remarkable Fouad Tabit. His exceptional intellect, perceptive insights, and careful approach have been the cornerstone of our collective success. It is through his visionary leadership and tireless efforts that we not only completed this project but also thrived in numerous simultaneous endeavors. Fouad's guidance has been invaluable, and I am truly fortunate to have had the opportunity to work alongside someone of such extraordinary caliber.

In reflecting on this journey, I am overwhelmed with gratitude for the unwavering support, divine guidance, and invaluable contributions of each person involved. Their presence has been an unwavering source of inspiration and encouragement, shaping the trajectory of this achievement in the most meaningful way. I am deeply humbled and grateful for the genuine human connections that have enriched this experience and propelled us towards success.

Summary

During my two-month internship at Agri 4.0 in Agadir, I developed a cow management system as part of my end-of-study project. The system utilizes modern technologies like Spring Boot, microservices, and live monitoring to improve cow productivity and automate farm operations using IoT devices. Its key features include detecting cows ready for breeding, monitoring their health, and optimizing their feeding system. Throughout the internship, This project aims to created a functional prototype of the system, demonstrating its scalability potential.

Table des matières

Table des figures

Abstract

In today's world, effective cow management is crucial for farmers to optimize performance, ensure animal welfare, and scale their operations. This project presents an innovative approach to cow management through the development of a comprehensive system. The aim is to empower farmers with advanced technologies and data-driven insights, enabling them to make informed decisions and drive success in the industry.

Through the utilization of modern tools and frameworks, including Spring Boot, microservices architecture, and Docker, a robust and scalable system is created. This system encompasses various components, including frontend, backend, mobile, and Internet of Things (IoT) integration. It enables farmers to monitor cow health, track performance metrics, and streamline daily operations efficiently.

During the project, we had the opportunity to enhance our skills as a backend developer and gain hands-on experience in data analysis. The implementation of the system involved optimizing performance, ensuring reliability, and designing a user-friendly interface to meet the specific needs of farmers.

The key findings of this project showcase the successful development of a unique cow management system that leverages cutting-edge technologies. It empowers farmers to overcome challenges, improve productivity, and make data-driven decisions to enhance their cows' performance. The system's real-time monitoring capabilities, advanced analytics, and intuitive user interface contribute to its effectiveness and usability.

This end-of-study project, conducted as part of the Professional Bachelor's Degree in Information Systems and Software Engineering program at EST Essaouira, reflects our dedication to innovation in the field of cow management. The project's outcomes contribute to the advancement of the agricultural industry and provide valuable insights for future research and development in this domain.

General Introduction

This report documents my internship experience at Agri 4.0 in Agadir, where I completed a two-month internship as part of my Professional Bachelor's Degree in Information Systems and Software Engineering program at EST Essaouira. The internship focused on developing a cow management system to enhance productivity and automate farm operations for farmers.

The cow management system utilizes modern technologies, including Spring Boot, microservices, and live monitoring, to optimize cow performance and streamline farm processes. Key features of the system include detecting cows ready for breeding, monitoring their health, and optimizing their feeding system. The ultimate goal is to empower farmers with advanced tools and insights to make informed decisions and elevate their business.

During the internship, I successfully created a working version of the system, showcasing its scalability potential. The report highlights the various tasks completed during the internship, including the development of additional projects such as HiveCare and Masser, which focus on beekeeping and greenhouse monitoring, respectively.

This report aims to provide a comprehensive overview of my internship experience, the projects I worked on, and the skills I acquired during this period. It also reflects the application of advanced technologies in the agricultural industry and the impact they can have on improving farm management practices.

Chapter1

Company Presentation

1.1 Introduction

The internship was conducted at AGRI 4.0, a Moroccan company located in the Ryad Essalam neighborhood in Agadir. Founded in 2018, AGRI 4.0 aims to integrate a range of innovative technologies applied to agricultural production, such as the Internet of Things (IoT) and artificial intelligence (AI). These technologies enable farmers to enhance the efficiency of their crop yields by leveraging IoT and AI in their farming practices.

This company is engaged in the design and production of electronic sensors, as well as the development of web and mobile applications dedicated to agriculture. These applications enable intelligent farm management through real-time tracking and analysis tools. By utilizing these innovative technologies, AGRI 4.0 empowers farmers to monitor various aspects of their farms, such as soil moisture levels, temperature, and crop growth, facilitating informed decision-making for enhanced productivity. Additionally, these applications provide farmers with valuable insights and recommendations to optimize resource allocation, mitigate risks, and improve overall efficiency in agricultural operations.

1.2 Organizational structure

Agri4.0 Company's organizational structure is designed to ensure efficient operations and optimal collaboration. At the helm is the Manager, providing strategic guidance. Supporting the Manager are key roles such as the Project Manager, HR Manager, Communication Specialist, Head of Software Development, Research and Development Lead, Marketing Director, Design and Electronics Expert, and Finance and Accounting Manager. With their expertise and cohesive teamwork, Agri4.0 Company excels in the agricultural technology sector, driving innovation and achieving its strategic goals.

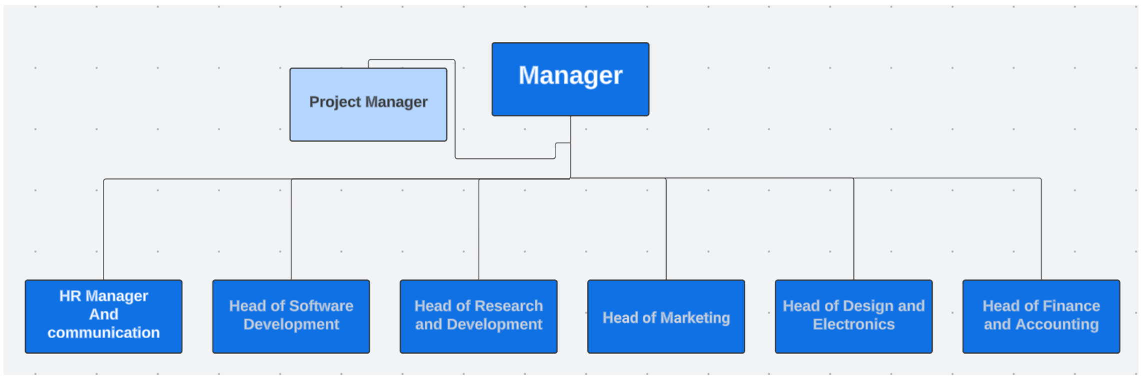

Manager
Project Manager
HR Manager And communication
Head of Software Development
Head of Research and Development
Head of Marketing
Head of Design and Electronics
Head of Finance and Accounting

FIGURE 1.1 – company organisational diagram

1.3 Our role in the company

Being a Computer Systems Engineering and Software student with prior experience in Data Science, I possess a versatile skill set in diverse domains. After careful consideration, I have chosen to specialize in back-end development (server-side) due to the high demand for proficient back-end developers in the industry. This decision stems from my goal to make meaningful contributions to organizations by delivering robust and scalable back-end solutions. This report elucidates the reasoning behind my decision and emphasizes the abundant opportunities available within the realm of back-end development.

1.4 conclusion

In conclusion, this chapter has presented a thorough account of the host company where I completed my last internship, as well as the organizational structure and our role in this company. With this framework in place, we can now turn our attention to digging further into the project.

Chapter2

Tasks Completed

2.1 Introduction

During the internship, significant contributions were made to various projects alongside the primary project. This section highlights the tasks undertaken during the internship period.

2.2 HiveCare

One of the projects involved the development of "HiveCare", a sophisticated hive management system designed to facilitate beekeepers in monitoring, maintaining the health and activities of their bees and honey production. The primary objective of the project was to enable beekeepers to track essential data collected from specialized devices.
The system provides beekeepers with valuable insights into their bee colonies. For example, temperature monitoring allows identification of potential risks, such as excessively high temperatures that could endanger both the bees and the plant species they rely on for sustenance. By analyzing temperature patterns over time, beekeepers can take preventive measures to protect their colonies and preserve the delicate ecosystem.
In addition to temperature, the system incorporates humidity monitoring, which plays a critical role in maximizing bee productivity. By leveraging scientific principles related to humidity levels, beekeepers can optimize their beekeeping practices to ensure optimal conditions.
The system's capabilities extend to analyzing bee sounds, which can provide valuable information about the bees' behavior, health, and potential issues that need attention.
Another noteworthy feature of the system is the ability to measure the weight of bee colonies. This information allows beekeepers to gauge the productivity and overall status of the hive, assisting them in making informed management decisions.

2.3 Masser : Greenhouse Management System

Another project focused on is the development of Masser, a greenhouse monitoring system that enables farmers to effectively monitor and manage environmental conditions within the

greenhouse environment. The system incorporates a range of sensors to collect critical data, including temperature, oxygen and carbon dioxide levels, battery status, and pressure.
By continuously monitoring these parameters, farmers can maintain optimal growing conditions for their crops within the greenhouse. Deviations from desired conditions can be promptly identified, allowing farmers to take appropriate corrective actions.
The integration of Masser into the larger project framework contributes to the overall objective of providing comprehensive solutions for agricultural management.

2.4 Conclusion

In conclusion, the internship provided an opportunity to contribute to multiple projects, such as hive management and greenhouse monitoring, addressing challenges in beekeeping and greenhouse farming. These experiences fostered professional growth and provided exposure to real-world applications of technology in the agricultural domain.

Chapter3

Project presentation and conception

3.1 Project framework

In the field of farming, farmers who manage a large number of cows face significant challenges related to cow productivity. One such challenge is accurately identifying cows that are ready for pregnancy, as they exhibit specific signs indicating their readiness. Typically, cows have a gestation period of approximately 283 days, followed by a post-pregnancy period of 60 to 90 days. Ideally, this would result in a calf being born every 13 months. However, in reality, farmers with over 20 cows often struggle to detect the subtle signs indicating a cow's readiness for pregnancy, resulting in wasted time of an additional 3 to 4 months and substantial financial losses.
Furthermore, during the late stages of pregnancy, it is crucial for the cow owner to cease milking the cow to avoid potential health complications for both the cow and its offspring. Additionally, monitoring the cow's activities such as sleep patterns, food consumption, and water intake is necessary to prevent diseases and maintain their overall well-being. However, as the number of cows increases, manually monitoring each cow becomes an insurmountable task.
To address these challenges, technological advancements and scientific approaches can be employed. Implementing automated systems utilizing sensors and data analytics can help farmers identify cows in heat accurately. These systems can detect behavioral changes, such as increased activity and mounting behavior, indicating the optimal time for insemination. Additionally, wearable devices can track vital parameters, such as rumination, body temperature, and activity levels, enabling early detection of health issues and facilitating timely intervention.
By leveraging scientific innovations and implementing data-driven approaches, farmers can overcome the hurdles associated with cow productivity management. These technologies offer the potential to improve breeding efficiency, reduce wastage of valuable time and resources, and enhance the overall health and productivity of the cow herd. Embracing these advancements can lead to more sustainable and profitable farming practices in the modern agricultural landscape.

3.2 Objectives

The project has set forth specific objectives across different categories to address the needs of cow management effectively. These objectives encompass :

Monitoring :

— Gain a holistic view of the herd's performance, allowing for better decision-making.
— Detect the physical well-being of each cow, ensuring timely attention to their health needs.
— Identify cows that are ready for insemination, maximizing reproductive success.
— Enable real-time monitoring of cows through sensor data, ensuring their well-being.
— Track important factors such as sleep patterns, food intake, and water consumption for proactive care.

Communication :

— Simplify communication with inseminators through a user-friendly mobile application.
— Ensure that important notifications reach the right people promptly via email, SMS, and application notifications.

By focusing on these objectives, the project aims to create a system that empowers farmers with comprehensive monitoring capabilities and streamlined communication channels. This human-centric approach will enhance the overall management of cows, promoting their health, productivity, and reproductive efficiency.

3.3 Challenges

During the implementation of the project, it is important to anticipate and address potential challenges that may arise. Some of the key areas of concern include :

Technical Complexity :

— Managing the intricacies involved in developing a comprehensive monitoring system and integrating diverse sensors.
— Designing the system to handle a large number of cows and scale seamlessly as the farm expands.
— Ensuring that the system can accommodate the growing data volume and processing requirements without compromising performance.
— Developing a cost-effective solution that provides value for money and is affordable for farmers

3.4 Project analysis and conceptualization

We chose the UML modeling language as a software architecture modeling standard because it simplifies the creation of documents required for the development of object-oriented software.

3.4.1 Use case diagram

The use case diagram is a UML diagram that depicts a software system's overall functional behavior. A use case is a discrete unit of interaction between a system and a user (human or machine). In a use case diagram, users are actors, and they interact with the system and manage use cases. The following images depict a broad use case diagram for the project :

3.4.1.1 Farmer and Inseminator

The use case diagram shown in Figure X illustrates the functional behavior of the proposed cow management system. The diagram represents the interactions between the system and the actors involved, namely the User (Farmer) and the Inseminator.
The primary objective of the system is to address the challenges faced by farmers in managing their cow herds effectively. By leveraging technological advancements, the system aims to improve cow productivity, reproductive efficiency, and overall herd health.
The User (Farmer) is a key actor in the system and is responsible for performing various tasks related to farm management. These tasks include managing groups of cows, giving ratings to inseminators, creating and tracking job requests, and accessing the dashboard for an overview of the farm's performance. Additionally, the User can manage their profile, add events, and read notifications to stay informed about important updates.
On the other hand, the Inseminator, another actor in the system, has specific roles such as accepting and rejecting job requests and updating the status of cows. These actions enable effective communication and collaboration between the User (Farmer) and the Inseminator, ensuring seamless coordination in the cow management process.
The use case diagram provides a high-level overview of the system's functionalities and the interactions between the actors and the system. It serves as a visual representation of the core features and workflows that the system supports, helping stakeholders understand the scope and purpose of the cow management project.

3.4.1.2 Streams

The Streams use case in the cow management system is responsible for capturing and processing data from different sensors and devices to gather valuable insights about the cows' behavior and well-being. This use case involves multiple actors and extension points to handle various types of data streams.
The primary actor in this use case is the Farmer, who is interested in monitoring and analyzing the streaming data to make informed decisions regarding the cows' health and behavior. The Streams use case interacts with different devices and sensors, such as the Pedometer, Collar, and RFID devices, to collect specific data points.
The Pedometer device measures the number of steps taken by cows and gives the position of the cow's foot related to the ground, and the system checks the number of steps to determine specific conditions, as defined in the extension points. If the number of steps exceeds a certain

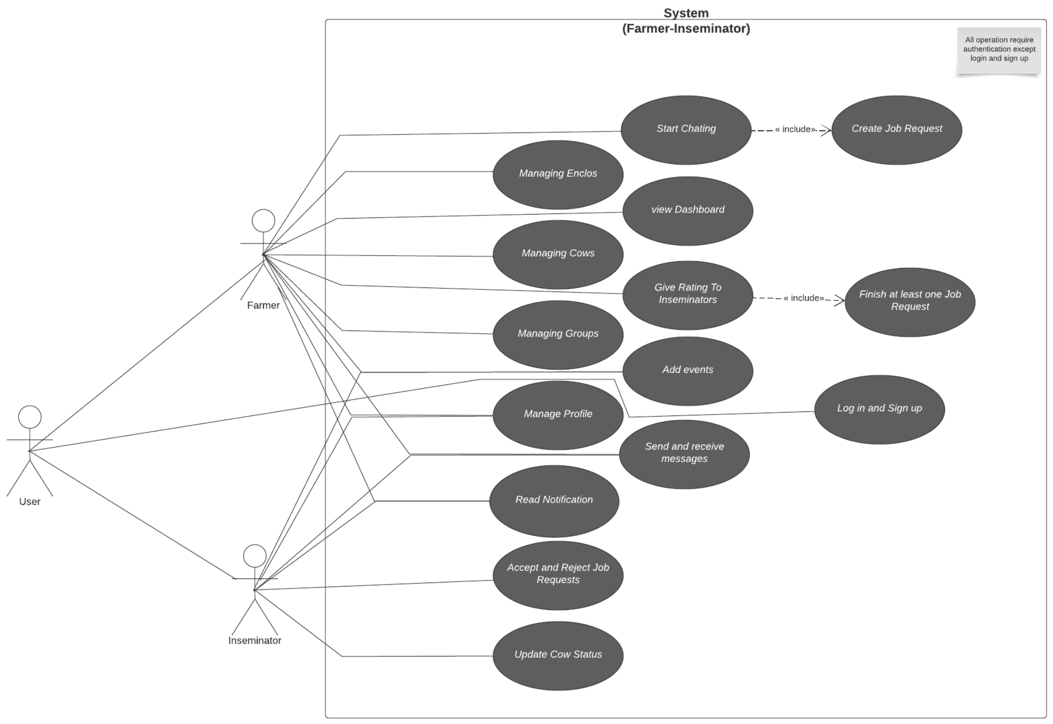


Figure 3.1 – use case diagram for farmer and inseminator

average value, the system triggers actions such as notifying the farmer and if exceeds the value more than 3 times the system sends an SMS.
Similarly, the collar device provides data related to the cows' health, including factors like body temperature and heart rate. Combining those factors, it generates a value that indicates the overall cow status. The system analyzes this data and updates the cow's status accordingly. In case the system detects anomalies or specific conditions, such as the height of the cow's head passing a certain value, extension points are triggered, and the farmer is notified via notifications.
The device also captures the cow's head position in 3D space and sends the data to the system. This data is used to track the cows' movements and behaviors. The system saves the data and can provide insights to the farmer regarding the cows' activities and locations.
Finally, the RFID devices play a crucial role in updating the cows' status, particularly related to eating and drinking behaviors. The system monitors the data from RFID devices and updates the cow's status accordingly. This information helps the farmer track the cows' feeding and drinking patterns, ensuring their proper nutrition and health.

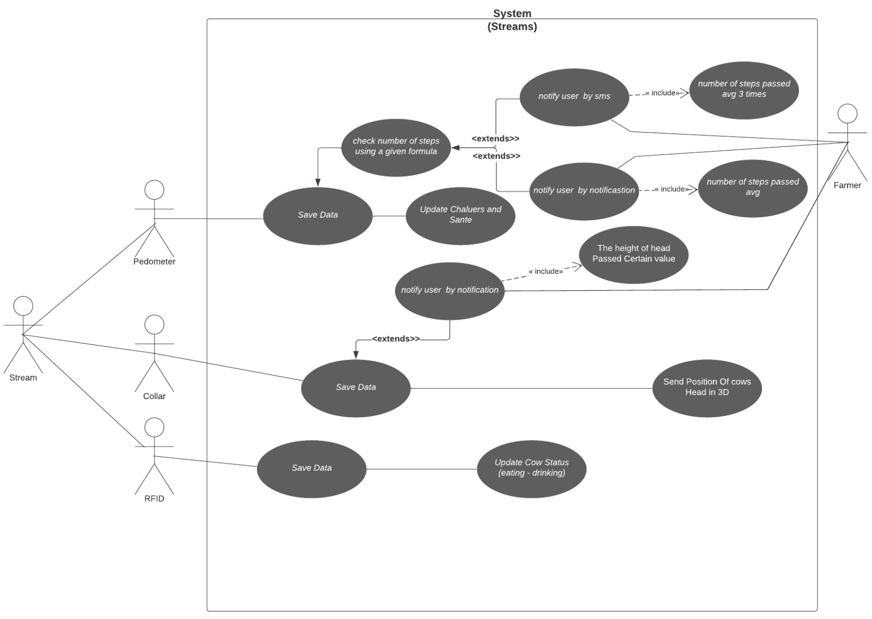

System
(Streams)
notify user by sms
« include»
number of steps passed avg 3 times
check number of steps using a given formula
<extends>>
<extends>>
notify user by notificastion
« include»
number of steps passed avg
Farmer
Pedometer
Save Data
Update Chaluers and Sante
The height of head Passed Certain value
« include»
notify user by notification
<extends>>
Stream
Collar
Save Data
Send Position Of cows Head in 3D
RFID
Save Data
Update Cow Status (eating - drinking)

Figure 3.2 – use case diagram for streams

3.4.2 Class Diagram

— **User** A system user entity is represented by the "User" class. It contains data like the user ID, login credentials, personal information (first name, last name, email), activation status, language key, profile image URL, activation and reset keys, reset date, and authorities. It also contains data like the language key. It offers a variety of ways to manage user accounts, such as creating new accounts, activating existing ones, checking authentication status, retrieving account information, updating user profiles, changing passwords, and handling password resets. It also connects to one-to-many roles with notification messages and job requests, as well as one-to-one roles with profile classes because each user has a profile.

— **Farmer** A farmer entity in the system is represented by the "Farmer" class. It extends the "User" class and has a one-to-many relationship with a group. It includes characteristics and practices that are peculiar to farmers, like managing groups and the enclosures that go with them. In addition to performing operations on the linked enclosures within their groups, farmers can create, update, and obtain information about their groups.

— **Inseminator** An inseminator entity in the system is represented by the "Inseminator" class. It is linked to job requests and extends the "User" class. Specialized duties linked to offering insemination services fall under the purview of inseminators. Users are able to access and handle work requests, including retrieving pertinent data about the customer and supplier associated with each request as well as accepting or rejecting requests, altering task statuses, and more.

— **Notification**

The "Notification" class represents a notification entity in the system. It contains information such as the notification ID, date, seen status, content, cow ID, sender, and receiver. The class provides methods to set a notification as seen and retrieve all unseen notifications for a specific receiver. It has a one-to-many relationship with the "User" class, as multiple notifications can belong to a single user.

— **JobRequest**
  The "JobRequest" class represents a job request entity in the system. It stores details such as the job request ID, consumer, provider, service status, room ID, and cow ID. The class provides methods to retrieve job requests based on the consumer or provider, change the status of a job request, and get confirmed or completed jobs. It has a many-to-one relationship with the "User" class, as multiple job requests can be associated with a single user.

— **Message**
  The "Message" class represents a message entity in the system. It contains attributes such as the message ID, created date and time, message type, content, room, and username. The class provides methods to retrieve messages by room and get the last message in a room. It has a many-to-one relationship with the "User" class, as multiple messages can be associated with a single user.

— **Group**
  The "Group" class represents a group entity in the system. It contains attributes such as the group ID, lactation, GPS address, user ID, and name. The class provides methods to retrieve groups by user ID. It has a one-to-many relationship with the "Enclosure" class, as multiple enclosures can belong to a single group, and also a composition relationship since a group is composed from enclose

— **Enclosure**
  The "Enclosure" class represents an enclosure entity in the system. It stores information such as the enclosure ID, user ID, group ID, and name. The class provides methods to retrieve enclosures by user ID and group ID. It has a one-to-many relationship with the "Group" class, as multiple enclosures can belong to a single group.

— **Cow**
  The "Cow" class represents a cow entity in the system. It stores information such as the cow ID, number, group ID, enclosure ID, responder, name, user ID, waiting for inseminator status, RFID, pedometer, and collar. The class provides methods to partially update cow information, retrieve cows by enclosure, set the waiting for inseminator status, and get cows waiting for inseminator. It has a many-to-one relationship with the "Enclosure" class, as multiple cows can belong to a single enclosure.

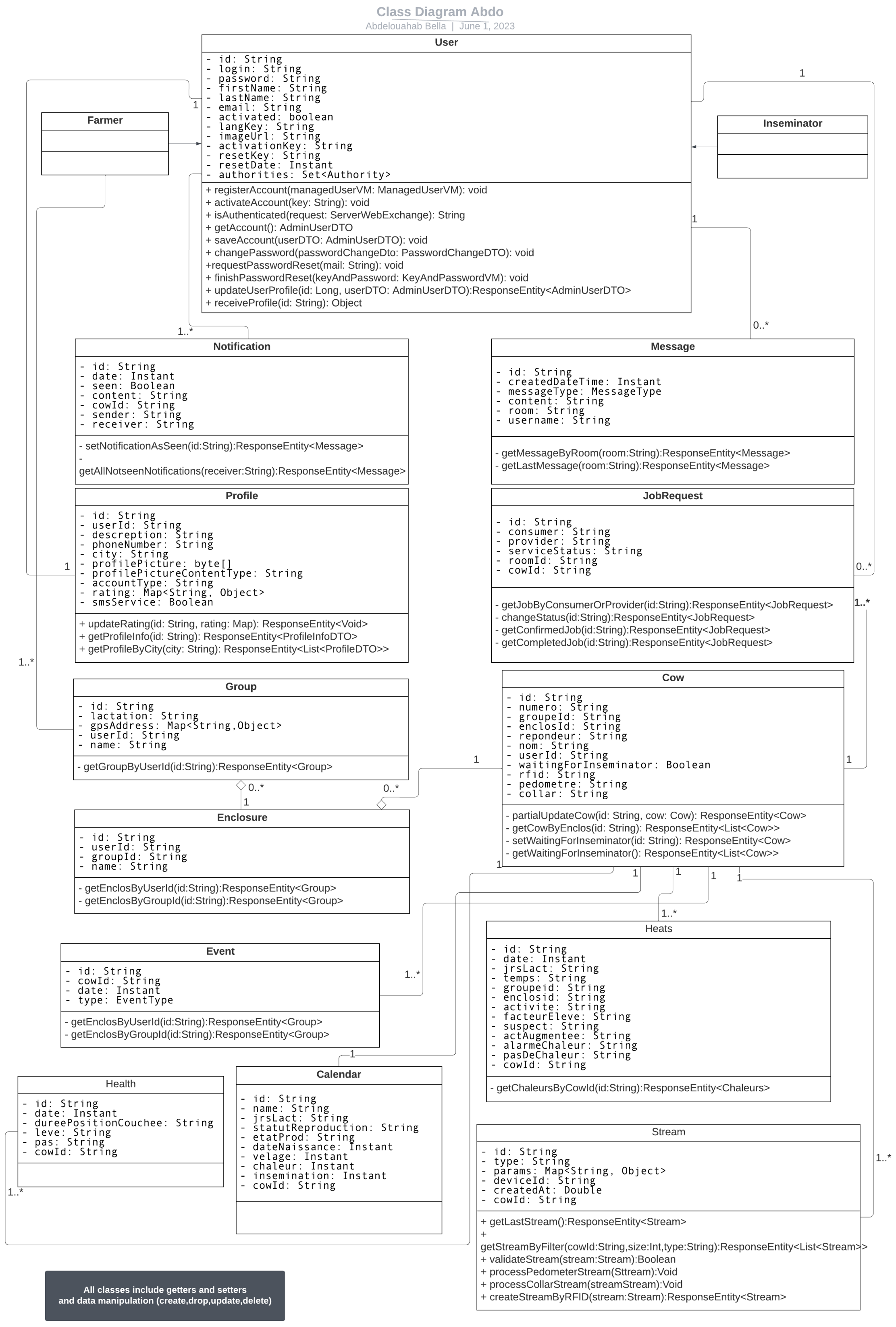
— **Calendar**
  The "Calendar" class represents a calendar entity in the system. It stores information such as the calendar ID, name, lactation, reproduction status, production status, date of birth, calving date, heat date, insemination date, and cow ID. It is associated with a cow to track various events and statuses related to the cow.

— **Event**
The "Event" class represents an event entity in the system. It contains attributes such as the event ID, cow ID, date, and event type. The class provides methods to retrieve events related to an enclosure or group. It has a one-to-many relationship with the "Enclosure" class and the "Group" class, as multiple events can be associated with an enclosure or group.

— **Heats**
The "Heats" class represents a heat entity in the system. It stores information such as the heat ID, date, lactation, duration, group ID, enclosure ID, activity, breeding factor, suspicion status, increased activity status, heat alarm status, no heat status, and cow ID. The class provides methods to retrieve heats by cow ID. It has a one-to-many relationship with the "Cow" class, as multiple heat records can be associated with a single cow.

— **Health**
The "Health" class represents a health entity in the system. It stores information such as the health ID, date, duration of lying position, standing status, walking status, and cow ID. The class provides methods to retrieve health records by cow ID. It has a one-to-many relationship with the "Cow" class, as multiple health records can be associated with a single cow.

— **Stream**
The "Stream" class represents a stream entity in the system. It contains attributes such as the stream ID, type, parameters, device ID, creation timestamp, and cow ID. The class provides methods to retrieve the last stream, filter streams by cow ID, validate streams, process pedometer streams, process collar streams, and create streams by RFID. It is associated with a cow to track and process different types of streams.



Class Diagram Abdo
Abdelouahab Bella | June 1, 2023
User
- id: String
- login: String
- password: String
- firstName: String
- lastName: String
- email: String
- activated: boolean
- langKey: String
- imageUrl: String
- activationKey: String
- resetKey: String
- resetDate: Instant
- authorities: Set<Authority>
+ registerAccount(managedUserVM: ManagedUserVM): void
+ activateAccount(key: String): void
+ isAuthenticated(request: ServerWebExchange): String
+ getAccount(): AdminUserDTO
+ saveAccount(userDTO: AdminUserDTO): void
+ changePassword(passwordChangeDto: PasswordChangeDTO): void
+requestPasswordReset(mail: String): void
+ finishPasswordReset(keyAndPassword: KeyAndPasswordVM): void
+ updateUserProfile(id: Long, userDTO: AdminUserDTO):ResponseEntity<AdminUserDTO>
+ receiveProfile(id: String): Object
Farmer
Inseminator
Notification
- id: String
- date: Instant
- seen: Boolean
- content: String
- cowId: String
- sender: String
- receiver: String
- setNotificationAsSeen(id:String):ResponseEntity<Message>
- getAllNotseenNotifications(receiver:String):ResponseEntity<Message>
Message
- id: String
- createdDateTime: Instant
- messageType: MessageType
- content: String
- room: String
- username: String
- getMessageByRoom(room:String):ResponseEntity<Message>
- getLastMessage(room:String):ResponseEntity<Message>
Profile
- id: String
- userId: String
- descreption: String
- phoneNumber: String
- city: String
- profilePicture: byte[]
- profilePictureContentType: String
- accountType: String
- rating: Map<String, Object>
- smsService: Boolean
+ updateRating(id: String, rating: Map): ResponseEntity<Void>
+ getProfileInfo(id: String): ResponseEntity<ProfileInfoDTO>
+ getProfileByCity(city: String): ResponseEntity<List<ProfileDTO>>
JobRequest
- id: String
- consumer: String
- provider: String
- serviceStatus: String
- roomId: String
- cowId: String
- getJobByConsumerOrProvider(id:String):ResponseEntity<JobRequest>
- changeStatus(id:String):ResponseEntity<JobRequest>
- getConfirmedJob(id:String):ResponseEntity<JobRequest>
- getCompletedJob(id:String):ResponseEntity<JobRequest>
Group
- id: String
- lactation: String
- gpsAddress: Map<String,Object>
- userId: String
- name: String
- getGroupByUserId(id:String):ResponseEntity<Group>
Cow
- id: String
- numero: String
- groupeId: String
- enclosId: String
- repondeur: String
- nom: String
- userId: String
- waitingForInseminator: Boolean
- rfid: String
- pedometre: String
- collar: String
- partialUpdateCow(id: String, cow: Cow): ResponseEntity<Cow>
- getCowByEnclos(id: String): ResponseEntity<List<Cow>>
- setWaitingForInseminator(id: String): ResponseEntity<Cow>
- getWaitingForInseminator(): ResponseEntity<List<Cow>>
Enclosure
- id: String
- userId: String
- groupId: String
- name: String
- getEnclosByUserId(id:String):ResponseEntity<Group>
- getEnclosByGroupId(id:String):ResponseEntity<Group>
Event
- id: String
- cowId: String
- date: Instant
- type: EventType
- getEnclosByUserId(id:String):ResponseEntity<Group>
- getEnclosByGroupId(id:String):ResponseEntity<Group>
Heats
- id: String
- date: Instant
- jrsLact: String
- temps: String
- groupeid: String
- enclosid: String
- activite: String
- facteurEleve: String
- suspect: String
- actAugmentee: String
- alarmeChaleur: String
- pasDeChaleur: String
- cowId: String
- getChaleursByCowId(id:String):ResponseEntity<Chaleurs>
Health
- id: String
- date: Instant
- dureePositionCouchee: String
- leve: String
- pas: String
- cowId: String
Calendar
- id: String
- name: String
- jrsLact: String
- statutReproduction: String
- etatProd: String
- dateNaissance: Instant
- velage: Instant
- chaleur: Instant
- insemination: Instant
- cowId: String
Stream
- id: String
- type: String
- params: Map<String, Object>
- deviceId: String
- createdAt: Double
- cowId: String
+ getLastStream():ResponseEntity<Stream>
+ getStreamByFilter(cowId:String,size:Int,type:String):ResponseEntity<List<Stream>>
+ validateStream(stream:Stream):Boolean
+ processPedometerStream(Sttream):Void
+ processCollarStream(streamStream):Void
+ createStreamByRFID(stream:Stream):ResponseEntity<Stream>
All classes include getters and setters and data manipulation (create,drop,update,delete)
1
1
1..*
1
0..*
1
1..*
0..*
1..*
1
0..*
1
0..*
1
1
1
1
1
1
1..*
1..*
1
1..*
1..*

FIGURE 3.3 – class diagram

3.4.3 Sequence Diagram

3.4.3.1 Pedometer

The below diagram illustrates the interaction flow of the pedometer system components. The pedometer device sends step data to the system, which verifies and saves the data in the database. If the number of steps exceeds the average threshold, a notification is sent to RabbitMQ. Additionally, if the threshold is exceeded three times, an SMS notification is triggered using Telesign. This diagram showcases the essential actions involved in capturing and processing step data, as well as the integration with external services for notifications.

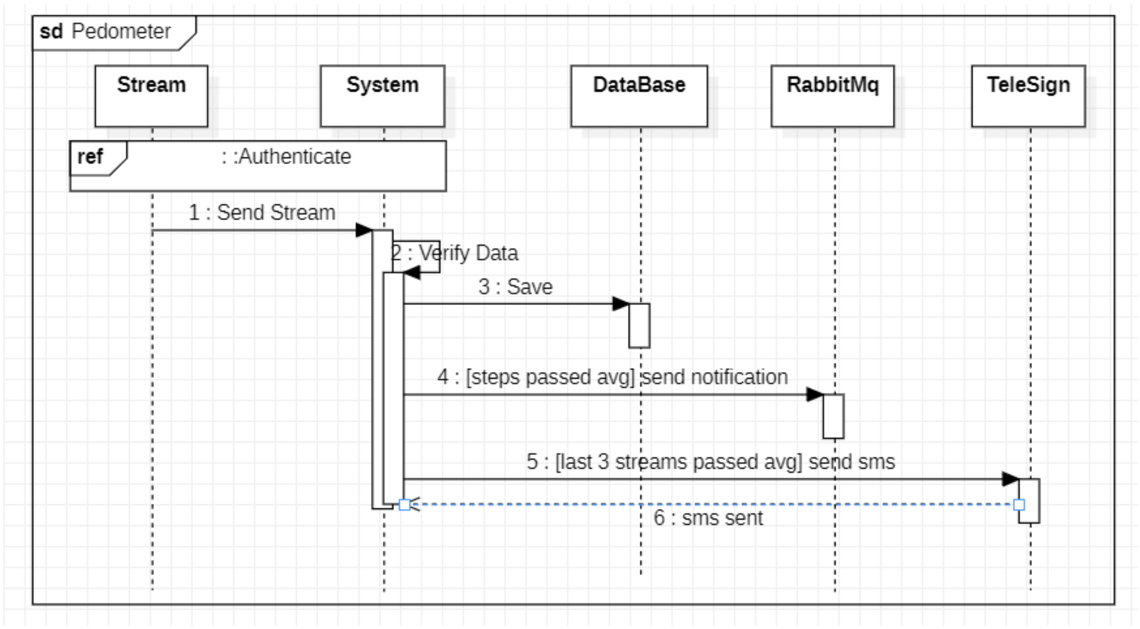


FIGURE 3.4 – diagram sequence pedometer

3.4.3.2 Collar

The following diagram demonstrates the flow of interactions in the collar system. The collar device communicates with the system, transmitting data streams. The system authenticates and verifies the received data, subsequently storing it in the database. In the event that the measured head height exceeds the maximum height threshold, a notification is generated and sent through RabbitMQ. This diagram depicts the key actions involved in data transmission, verification, storage, and triggering notifications based on specific conditions in the collar system.

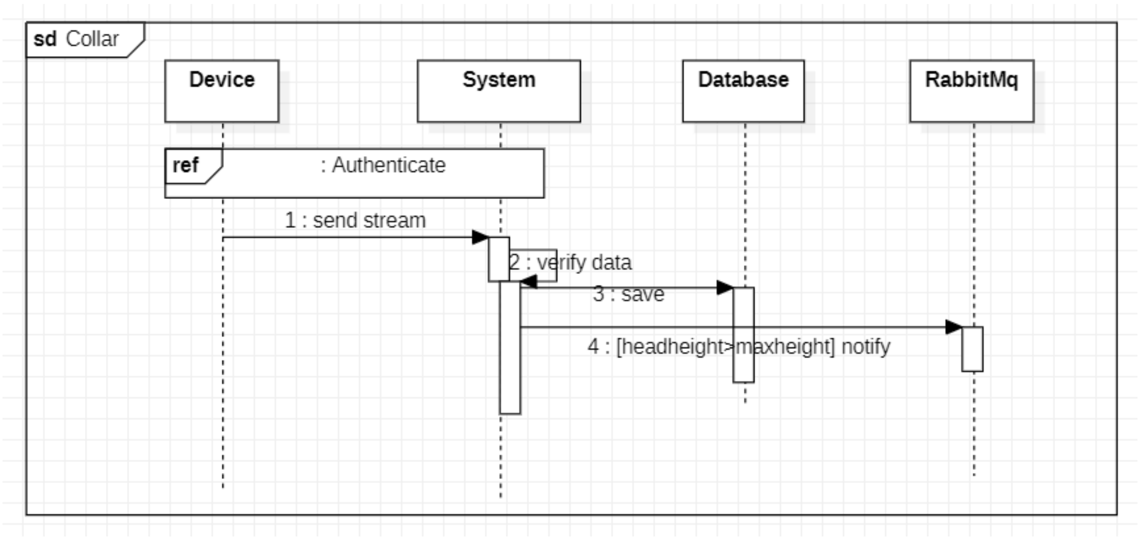


FIGURE 3.5 – diagram sequence collar

Chapter4

Project Overview And Realization

4.1 Intoduction

This final chapter is dedicated to the project's implementation phase, which is a subsequent step in the project's development. We will begin the implementation phase of the project after the above-mentioned steps have been completed.

4.2 Tools and Technologies

In the implementation phase of the project, several tools have been used to facilitate the development process and enhance the functionality of the system. In the following sections, we will provide detailed information about each technology used.

4.2.1 Backend Technologies

Java	Java is a widely used object-oriented programming language known for its portability and versatility. It is designed to be platform-independent, allowing developers to write code once and run it on any Java Virtual Machine (JVM). Java has a vast ecosystem of libraries, frameworks, and tools that support various types of applications, from desktop to web and enterprise systems. It is renowned for its performance, security, and strong community support.
	JHipster is a development platform that generates a fully-featured and customizable application stack for modern web development. It combines popular frameworks and tools such as Spring Boot, Angular or React, and Apache Kafka to create a production-ready application architecture. JHipster provides scaffolding, code generation, and a development workflow that streamlines the creation of robust and scalable web applications.
Cucumber JUnit	JUnit and Cucumber are complementary testing frameworks used in software development. JUnit is a unit testing framework for Java, while Cucumber is a behavior-driven development (BDD) tool that facilitates collaboration between developers and non-technical stakeholders. Together, they enable efficient and comprehensive testing of Java applications with clear and readable test scenarios.
	WebSocket is a communication protocol that provides full-duplex communication channels over a single TCP connection. It enables real-time, bidirectional communication between clients and servers, allowing for interactive and dynamic web applications. WebSocket provides a persistent connection that avoids the overhead of establishing new connections for every request. It is widely used for applications requiring real-time updates, such as chat applications, collaborative tools, and live data streaming.
{…}	Your trusted companion for API development. Simplify API creation, documentation, and consumption with this open-source framework. Generate interactive docs and client SDKs effortlessly. Focus on building great applications while Swagger handles the technical details. Efficiency, collaboration, and reliability for modern software teams.

4.2.2 Persistence Technology :

mongoDB	MongoDB is a popular open-source NoSQL database that stores data in flexible, JSON-like documents instead of traditional tables. It provides high scalability, availability, and flexibility, making it suitable for handling diverse and evolving data types. MongoDB's document model allows for agile development, seamless horizontal scaling, and fast query performance. It is commonly used in modern web applications, content management systems, and real-time analytics.

4.2.3 Industrialization Tools :

	Git is a distributed version control system (a system that records changes to a file or set of files over time so that you can recall specific versions later.) for tracking source code changes during software development. It is designed to coordinate the work between programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed and non-linear workflows.
docker	Docker is an open-source containerization platform that allows developers to package applications and their dependencies into lightweight, portable containers. Containers provide an isolated and consistent environment for running applications, making it easier to deploy and scale them across different environments. Docker simplifies the deployment and management of applications, promotes consistency, and enhances scalability in modern software development.
	GitHub is a web-based platform built on top of Git that provides a hosting service for Git repositories. It offers additional features such as issue tracking, pull requests, and collaboration tools, making it popular among developers for sharing and collaborating on code projects. GitHub simplifies the process of contributing to open-source projects and enables seamless team collaboration.

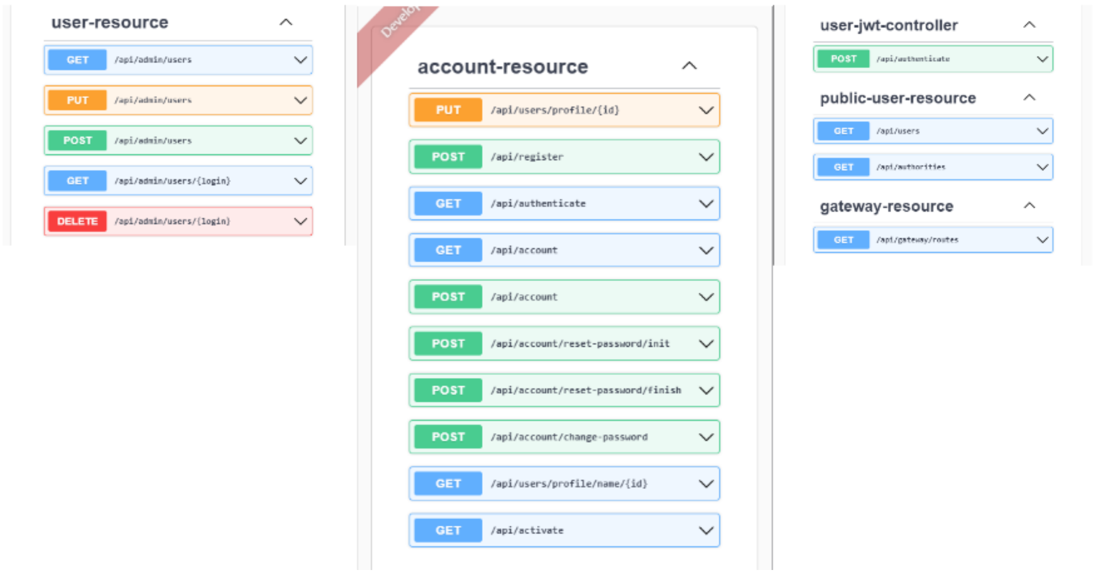
4.3 Project graphical interface

4.3.1 Introduction

In modern software development, user interfaces play a crucial role in creating intuitive and interactive experiences for end-users. However, as a backend developer focused on the server-side logic and data management, our direct involvement in front-end development is limited. To address this limitation, we have leveraged API documentation interfaces from Swagger to illustrate the functionality and usage of the underlying APIs.

4.3.2 Gateway microservice

In a microservices architecture, the service gateway serves as the main access point for client requests. User accounts, including registration, login, password management, and profile updates, are managed by the account resource. The user resource manages user-related operations like locating user data, revising user information, and controlling user responsibilities and permissions. For user authentication and authorization, the user JWT controller manages the creation and validation of JSON Web Tokens (JWT). The configuration files, filters, and interceptors included in the gateway resources also add to the service gateway's overall functionality and security.



user-resource

```
GET     /api/admin/users
PUT     /api/admin/users
POST    /api/admin/users
GET     /api/admin/users/{login}
DELETE  /api/admin/users/{login}
```

account-resource

```
PUT     /api/users/profile/{id}
POST    /api/register
GET     /api/authenticate
GET     /api/account
POST    /api/account
POST    /api/account/reset-password/init
POST    /api/account/reset-password/finish
POST    /api/account/change-password
GET     /api/users/profile/name/{id}
GET     /api/activate
```

user-jwt-controller

```
POST    /api/authenticate
```

public-user-resource

```
GET     /api/users
GET     /api/authorities
```

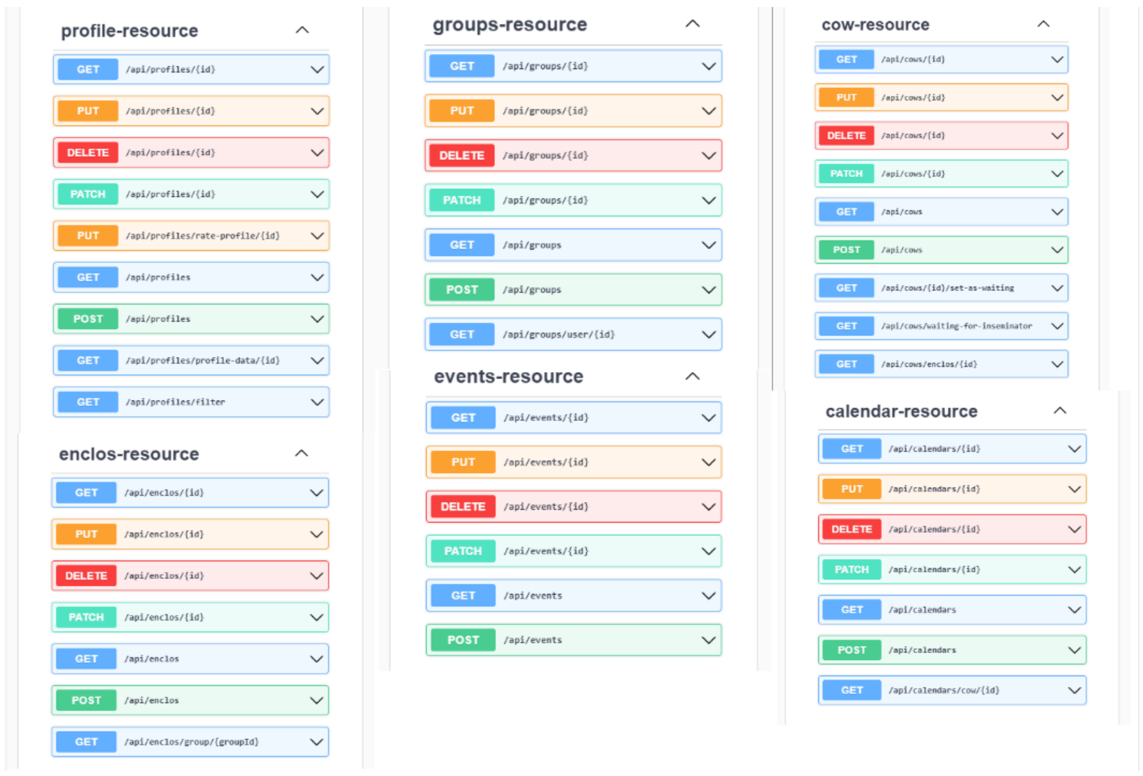
gateway-resource

```
GET     /api/gateway/routes
```

FIGURE 4.1 – gateway microservice

4.3.3 Managment microservice

The management microservice's api endpoints, which include operations for profiles, groups, events, enclosures, cows, and calendars, are shown in the image below. These endpoints enable managing cows in various states, filtering profiles, executing CRUD operations on various resources, rating profiles, and accessing user-specific group and calendar data.


profile-resource

GET /api/profiles/{id}
PUT /api/profiles/{id}
DELETE /api/profiles/{id}
PATCH /api/profiles/{id}
PUT /api/profiles/rate-profile/{id}
GET /api/profiles
POST /api/profiles
GET /api/profiles/profile-data/{id}
GET /api/profiles/filter

enclos-resource

GET /api/enclos/{id}
PUT /api/enclos/{id}
DELETE /api/enclos/{id}
PATCH /api/enclos/{id}
GET /api/enclos
POST /api/enclos
GET /api/enclos/group/{groupId}

groups-resource

GET /api/groups/{id}
PUT /api/groups/{id}
DELETE /api/groups/{id}
PATCH /api/groups/{id}
GET /api/groups
POST /api/groups
GET /api/groups/user/{id}

events-resource

GET /api/events/{id}
PUT /api/events/{id}
DELETE /api/events/{id}
PATCH /api/events/{id}
GET /api/events
POST /api/events

cow-resource

GET /api/cows/{id}
PUT /api/cows/{id}
DELETE /api/cows/{id}
PATCH /api/cows/{id}
GET /api/cows
POST /api/cows
GET /api/cows/{id}/set-as-waiting
GET /api/cows/waiting-for-inseminator
GET /api/cows/enclos/{id}

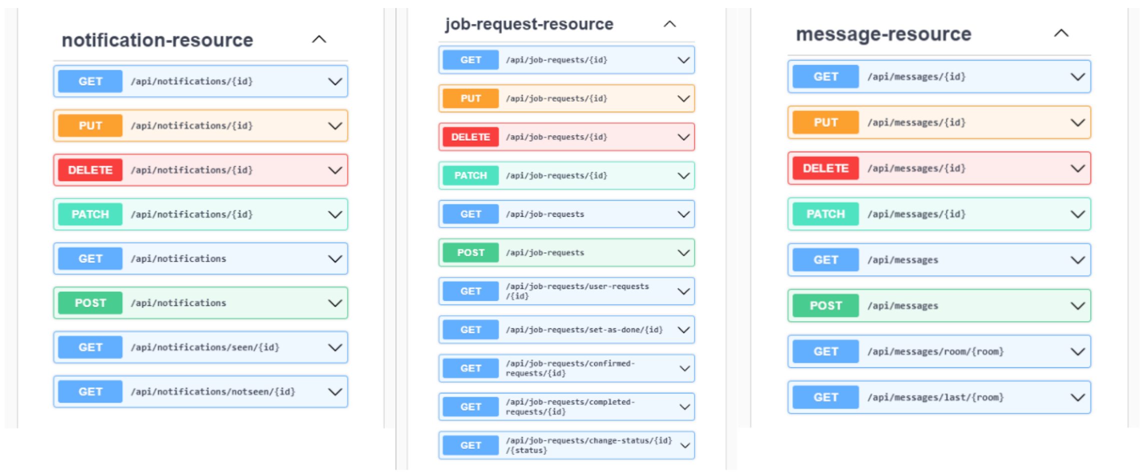
calendar-resource

GET /api/calendars/{id}
PUT /api/calendars/{id}
DELETE /api/calendars/{id}
PATCH /api/calendars/{id}
GET /api/calendars
POST /api/calendars
GET /api/calendars/cow/{id}

FIGURE 4.2 – managment microservice

4.3.4 Notification microservice

An API endpoint for the notification microservice, which offers capabilities for handling notifications, messages, and job requests, is shown in the picture below. These endpoints allow controlling task requests' status and related data, retrieving specific resources, marking notifications as seen or unseen, retrieving messages for a given room, and performing CRUD operations on notifications, messages, and job requests.


notification-resource

GET /api/notifications/{id}
PUT /api/notifications/{id}
DELETE /api/notifications/{id}
PATCH /api/notifications/{id}
GET /api/notifications
POST /api/notifications
GET /api/notifications/seen/{id}
GET /api/notifications/notseen/{id}

job-request-resource

GET /api/job-requests/{id}
PUT /api/job-requests/{id}
DELETE /api/job-requests/{id}
PATCH /api/job-requests/{id}
GET /api/job-requests
POST /api/job-requests
GET /api/job-requests/user-requests/{id}
GET /api/job-requests/set-as-done/{id}
GET /api/job-requests/confirmed-requests/{id}
GET /api/job-requests/completed-requests/{id}
GET /api/job-requests/change-status/{id}/{status}

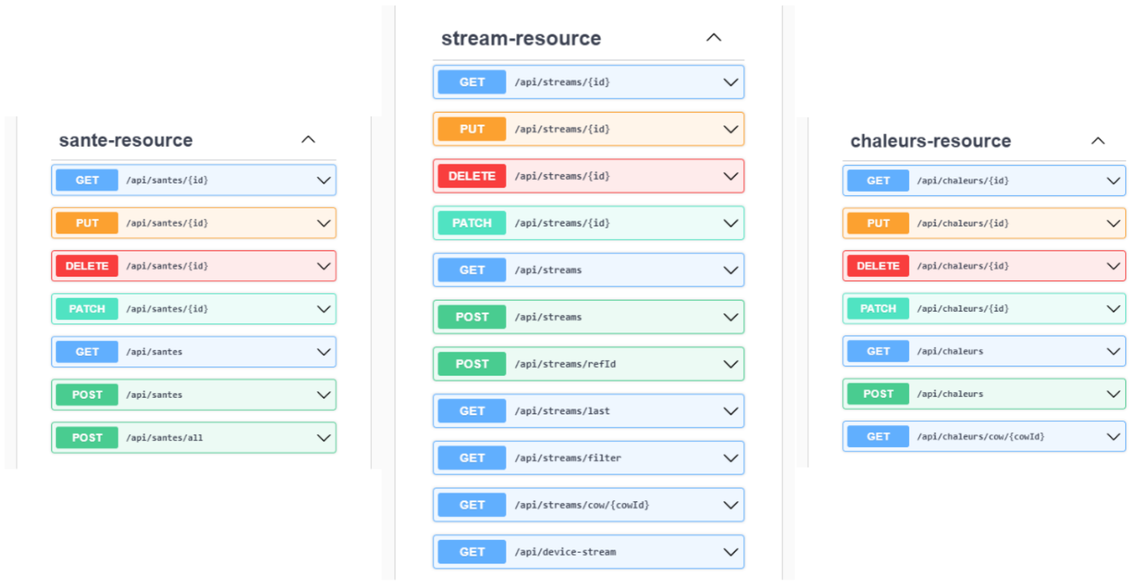
message-resource

GET /api/messages/{id}
PUT /api/messages/{id}
DELETE /api/messages/{id}
PATCH /api/messages/{id}
GET /api/messages
POST /api/messages
GET /api/messages/room/{room}
GET /api/messages/last/{room}

FIGURE 4.3 – notification microservice

4.3.5 Sensoring microservice

An API endpoint for the sensoring microservice, which facilitates activities involving streams, health data (health), and heat detection, is shown below (heats). These endpoints provide for the retrieval, updating, and deletion of streams, the retrieval of the most recent stream data, the filtering of streams, the retrieval of streams related to a particular cow, the management of health data, and the detection of cow heat events.


sante-resource
GET /api/santes/{id}
PUT /api/santes/{id}
DELETE /api/santes/{id}
PATCH /api/santes/{id}
GET /api/santes
POST /api/santes
POST /api/santes/all

stream-resource
GET /api/streams/{id}
PUT /api/streams/{id}
DELETE /api/streams/{id}
PATCH /api/streams/{id}
GET /api/streams
POST /api/streams
POST /api/streams/refId
GET /api/streams/last
GET /api/streams/filter
GET /api/streams/cow/{cowId}
GET /api/device-stream

chaleurs-resource
GET /api/chaleurs/{id}
PUT /api/chaleurs/{id}
DELETE /api/chaleurs/{id}
PATCH /api/chaleurs/{id}
GET /api/chaleurs
POST /api/chaleurs
GET /api/chaleurs/cow/{cowId}

FIGURE 4.4 – sensoring microservice

General Conclusion

During my internship, I had the chance to embark on an incredible journey of personal and professional growth. It has been an amazing opportunity for me to gain hands-on experience in the field of agriculture and cutting-edge technologies like Industry 4.0. I was fortunate to work alongside experts in microservices technologies, which opened my eyes to new possibilities and expanded my knowledge.
Throughout my internship, I explored various technologies such as Apache and Eureka Server, as well as Spring Cloud and Spring MVC. These tools have given me a broader skill set to tackle complex challenges. One project that stood out to me was the development of a cow management system. It's exciting to see how this cost-effective solution can potentially replace more expensive alternatives in the industry. However, I recognize that there is still work to be done to make it more robust and competitive on an international scale.
During my time in Agri 4.0, I had the opportunity to contribute to different projects, some of which I mentioned earlier. I also worked on Arduino programming and 3D graphic design, although I didn't highlight them in detail. I am grateful for the support, guidance, and opportunities provided by the company throughout my internship. These experiences have shaped my skills and knowledge, providing a strong foundation for my future career.
Overall, this internship has been an invaluable experience that has allowed me to grow both personally and professionally. I am excited to take what I have learned and apply it to future endeavors.