“Software Architectures”

Course

a.a. 2017-2018

Version 4.0

Lecturer: Prof. Henry Muccini (henry.muccini@univaq.it)

Masaccio - Monitoring for urbAn SAfety with the IoT

Date	30/01/2018
Deliverable	4.0
Team (Name)	Primavera

Name & Surname	Matriculation number	Email address
Luca Grillo	254377	lucag.8595@gmail.com
Paul Noorland	255351	paulnoorland93@gmail.com
Aysel Yusubzade	-	ayusubzade@std.qu.edu.az

Table of Contents

System introduction

The Masaccio system is software and hardware complex (Wi-Fi video cameras, NFC readers, monitors, recorders, RFID readers and other equipment) intended for organizing video monitoring both on local and on territorially-distributed objects such as museums, malls, universities, office, etc. The system can be seen as part of the smart city movement. The particular focus of the system in this movement is citizen safety.

The functions of the system is monitoring the crowd area in order to inform people about the crowd density inside specific locations. It may monitor visitors and staff in the office, in a warehouse or in a store, monitoring activities and movements people in any room to get the exact number of people.

Protecting the people in enclosed spaces is one of the most important tasks of our system which is called the Masaccio. To ensure the safety of people our system is able to monitor and count people in the close area overall, and in each room separately by using the services that are described below. This information provided by the system helps first responders and emergency agencies perform their job better and faster.

Application domain

The Masaccio system is an indoor location technology that consists of software and hardware parts. It permits to monitor the areas such as museums, malls, offices, universities, hospitals.

Museums, malls, offices, hospitals using the Masaccio system can get many advantages such as know exactly number of people on each day and then they can make analytics of the month or a year.

In Universities and schools it would be very good to implement in order to know the density of the laboratories, libraries,canteen.

Portfolio solutions of our project are system of access control and monitoring of people and their movements, management of automatic security system and guarantee the people safety inside the area.

The main goal of the system is to avoid the density of people in exactly places in order to guarantee the safety of people by sending to people the notification or information through the user recommendation service that some areas are crowd of people and it is best to go to other ones. To make the services work the system use: RFID readers which will save time and money, as well as reduce costs for staff, NFC cards that helps to avoid unauthorized people access, Wi-Fi video cameras and other hardware.

Also, Masaccio allows to protect the people in the case of emergency by sending them the available (the most suitable for him at the moment) exit, system has emergency service that automatically sens the information to the police, fireman, ambulance, etc.

Another advantages of the system are possibility of see the information beforehand and make a reservation (to the event that will take place at university, museum, mall), and moreover, it even permits to check the availability of the car spots if the visitor (student, professor, staff) drive to the area.

Access Control service: Different users (with different rights) are enabled to access to offices, laboratories, parking spaces etc. The basic service consists in checking the users' rights, track entry/exit time, and deny or allow access (e.g., preventing access to high-risk areas close to an hazard). The access can be also conditional to some conditions, such as load, temperature, available seats. Different stakeholders with different roles may receive different treatments.

Evacuation service: The city can be equipped with fire and other security sensors. When one (or more) of those sensors are activated, information from cameras, and sensors are collected to locate people in the building and to guide them towards the best exit. Once people are located, they are reported on a map owned by the security people. In this way, they know areas that are more populated, with respect to the diagnosed security problem. They can then be directed to different emergency exits, so to avoid the overload of only a subset of those exits. Smart emergency exit devices (e.g., people counters or cameras linked to emergency exit intelligent signs) inform the users on the load of the specific exit.

First responders communication service: the system shall include a platform that enables “first responders” (like operators, fireman service, emergency, police) to interact and exchange information even in case of emergency.

Smart Information service: Our system collects a big amount of data, that while computed, provides situational aware information about the monitored area. Some of this information can be made available to the population, so to increase their awareness on the surrounding.

User event information: See information about an event/building and enroll to the event.

User recommendation: Send a user a recommendation on whether to go to an event or not based on how crowded it is.

“I am safe” service: In case of an evacuation a user can register her/himself as being safe via the system. Buildings can have people registered via the NFC entry system. These people will all be stored in a DB and be connected to this particular building. If people use the system to say that they are safe, this will show on the list. This helps emergency teams to see if people are missing or not and might be still (trapped) inside. There might also be a ‘Safe me’ option in the system that sends the user’s location when he/she is trapped and needs to be saved.

Employee monitoring service: Employees can check in with their cards. This allows the management team to see when people are in and out of the office. Also, when employees want to have lunch or take some office products from a vending machine in the office this card and system can be used.

Rental service: NFC cards can be used to rent a room, laboratory, parking space or other location. The entry time is saved and the time the person leaves as well. This combined forms the total time a person has used a commodity and the costs of the rental can be charged to the person's account. Also, an overview of the availability can be made available, so people can see which rooms or parking spaces are still available.

Find the nearest parking spot: Use image recognition on camera images on the streets to look for free parking spaces and guide the user, based on the current location, to the nearest parking spot.

In the chapter concerning the user stories there is a further elaboration of the requirements and basic services that make up these services. Also, the most important stakeholders of the system are described there.

Challenges/Risk Analysis

Very simply, a risk is a *potential* problem. It's an activity or event that may compromise the success of a software development project.
Identify and plan how the risks could been managed is very important for the success of the project. The following table shows the most important risks that have been identified and how they have been managed.

Risk	Date the risk is identified	Date the risk is resolved	Explanation on how the risk has been managed
Communication from sensor to the system/No Internet connection	31/10/17	28/01/18	We will use sensors that, in absence of Internet connection, are able to store the data in a local storage and then send them to the system once the connection will work again.
Sensor input is too big for the system to manage(If one or more sensors are broken outputs will be wrong)	31/10/17	23/12/17	Modifying some Kafka properties, it is possible to send a large messages without problems.
The system cannot handle 40000 messages in a hour	31/10/17	04/12/17	Thanks to Apache Kafka that it is designed as a distributed system which is very easy to scale up, we can write large volumes of data into it without problems.
DB may crash	31/10/17	04/12/17	We have chosen Cassandra database for store the real-time data. It provides scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. So data is automatically replicated to multiple nodes for fault-tolerance. All these reasons may solve a DB crash.
Server may crash	31/10/17	12/01/18	We will use a database and server cloud and this means there is no physical server, so it can't get damaged or overheat.
Critical message is lost	31/10/17	04/12/17	For each topic, Kafka cluster maintains a partitioned log that represent an ordered and immutable sequence of messages. The partitions of the log are distributed over the servers in the Kafka cluster. So Kafka achieves fault tolerance by replicating each partition over a number of servers.
Critical message is not delivered within 5 seconds	31/10/17	28/01/18	Some tests have been performed about this risk through a tool called Siege. We have discovered that the critical messages are delivered within 5 seconds, as shown in Table 5.
Authorization process - unauthorized user has access	31/10/17	15/01/17	For this purpose we have used Spring Security. You can tell it that all URLs with a specific pattern e.g. **/static** are accessible to all users.

			You can tell it that all URLs with a specific pattern e.g. **/admin** are only available to users with a specific role. e.g. ROLE_ADMIN. We can also add annotations on the java methods to make them secure.
Authorization process-authorized user has no access	31/10/17	15/01/17	For this purpose we have used Spring Security. You can tell it that all URLs with a specific pattern e.g. **/static** are accessible to all users. You can tell it that all URLs with a specific pattern e.g. **/admin** are only available to users with a specific role. e.g. ROLE_ADMIN. We can also add annotations on the java methods to make them secure.
Data loss between sensor and the system	31/10/17	04/12/17	Kafka also replicates its logs over multiple servers for fault-tolerance.
Actuator gives false alarms	31/10/17	28/01/18	If an actuator gives false alarm, a message to the administrator dashboard will be sent. Therefore, administrator can send someone to solve the problem.

Table 1. Risk management table

List of Assumptions

At the beginning of building a system there are always uncertainties. These uncertainties can be of different natures, such as environmental, institutional, social or economical. Therefore, in the beginning it is essential to make some assumptions about the system environment. These assumptions need later to be tested in order to validate them. If an assumption proves to be false the system might need to adapt to this or else it might fail to be deployed and or penetrate the market.

The following assumptions are made by the team concerning the system.

Assumption	Description
Usage existing hardware	The assumptions is that the existing hardware in buildings and on the streets can be used. Examples of such hardware are camera's and NFC readers.
Hardware distribution	The assumption is that the sensors are sufficiently distributed to deliver valuable and useful data to our system. This means that for instance there need to be camera's at correct places to monitor the number of people and NFC readers at entrances to authorize and monitor entrance. The same goes for fire, humidity, temperature, rain and other sensors that need to be distributed through the city.
Legal and ownership of camera imagery	The assumption is that monitoring and analysing the camera imagery is (legally) allowed and that we do not need to pay third parties for this data.
Ownership of NFC cards	The users of the system or occupants/visitors of buildings that use our system are in possession of a NFC card or can be easily given one.
Participation	Organizations of event, owners of buildings and (local) governments are open to participation in our project.
Floor plans and blueprints	The floor plans and building blueprints can be used by the system. That means that they are or can be digitised and can be stored in our database. This is necessary for the evacuation service.
Algorithms	Algorithms are available or can be written to determine the number of people in a location based on camera images and to determine the best route for an evacuation. Also, these algorithms need to perform within sufficient accuracy.
First-responder adoption	A sufficient number of 'first-responders' will use the platform for it to be useful during emergencies.

Table 2. Assumption overview table

State of the art

Introduction on why we need the state of the art

Analysing the state of the art has a lot of benefits such as; assessment of the current state of the research on a similar project, identification of the experts, identification of key questions about the system that need further research, determination a methodologies and implementations used in past studies of the same or similar system.The search of the following information sources helped us determine what is already known about the system in our mind and also how extensively that kind of system has already been researched. It is often useful to review the types of studies that previous researchers have launched as a means of determining what approaches might be of most benefit in further developing a topic.

Results of what we found

Accuware Video Tracker (1)

Build a solution to analyze visitors' behavior by following the movements of people inside specific venues, identifying and tracking them by their visual signature.Implement people counting solutions for venues such as museums and shopping malls. Counts per sector can pretty accurately reflect visitors' volumes at different times per day, week and month. Create a system that enables security personnel to attach a name to any specific person seen by a camera and follow their movements on a floor plan. Build solutions for hospitality venues, such as hotels or casinos, to track VIP customers' whereabouts, enabling personalized assistance, such as delivering an order at the right table as the client moves around.

Features of the Accuware Video Tracker

- Locates people moving through monitored venues;
- No mobile device or app required;
- Provides daily log of people tracked;
- People counting in real-time and overtime;
- Dashboard for site management;
- Enables tagging individuals to follow automatically .

Components:

- Video camera at venue;
- Server: cloud based or standalone;
- API for integration;
- Administrative dashboard.

Setup (2):

- Ensure cameras at the site provide desired coverage;
- Set up video feed from all cameras to reach the server;
- Sign in to the dashboard;
    - Define levels at the site;
    - Upload floor plans for each level;
    - Mark location of all cameras on floor plans;
- Train the system to geo-locate coverage by each camera;
    - For each camera's coverage, mark reference points on corresponding floor plan;
    - Verify location accuracy.

“RFID-system for access control and monitoring of movement of people” developed by «АйТиПроект» company (3)

RFID technology is based on radio waves that react to finding an object (a person with a special label on clothing - a badge or bracelet) within a radius of 1 to 5 meters in the range of readers, and instantly send information to a computer or PDA where information is synchronized with the database.
What is the use of the RFID system?

- Automatic tracking of the movement of people. The system cannot be deceived: it recognizes the mark, even if the person is 1 to 5 meters from the reader - you do not even need to attach a card. So, looking in the database, you will immediately see who exactly, how much and where to go;
- Safety and quick response. Unlike the cameras that a person should monitor, the RFID system acts automatically on the specified parameter: for example, it sends a signal to the system if someone has entered the closed zone without access;
- Convenient tags. Depending on the purpose, RFID tags can be placed on badges, bracelets, plastic cards. In this case, they do not necessarily have to be in sight - the system considers them even out of pocket;
- Durability and reliability. The system is extremely simple - and therefore reliable.

Features of the system:

- In-store reading (statistics collection) Fixing regular customers and their movement;
- Readout: access control;
- Office reading: working time control;
- Office reading: control of movement of people;
- Search for a person.

Figure 1. Example of RFID system in the shop center.[3]

Using the RFID system, you can:
Register events such as the time of entry, exit, finding a person in the zone, for example, when a person entered the room, how many stayed when he left.
The movement of people: you no longer need to attach a card or sign with the guard, and lost tags can easily be replaced.
Monitor working hours: you can create schedules for employees and track all delays, as well as the presence or absence of an employee in the workplace. It is possible to calculate the efficiency of the whole staff.
Find and track any person: search the database you can find an employee or a visitor, wherever he is.
Collect useful statistics: it will now be easy to determine which parts of the business center or store are the most visited, which departments in the office last longer, and thus optimize many business processes.
Configure access: you can enter information about the access of certain people to a specific zone, and if temporary access is required - to limit the time of their stay there.

Implemented RFID projects:
"Tinkoff Bank" is a Russian commercial bank focused entirely on remote maintenance, without retail outlets and ATMs. The head office of the bank is located in Moscow.
Fitness center "Mendeleef Fitness" is a modern fitness center of international level from the managing company Fitness Holding. The fitness center is equipped with the most modern sports and engineering equipment. The total area of the club is more than 3000 sq.m.

TRAF-SYS people counting system (4)

Traf-Sys' people counting software provides near real-time access to dependable and accurate people traffic data via an automatic online platform. It has easy-to-use interface, clients can access traffic data on any computer with having a network connection. This system can get data from the traffic counters and do some reports to use in online generation.
Advanced software functionality includes detailed business reporting options that enable users to generate custom reports. In order to monitor essential business metrics the data from the Traf-Sys has given in many formats.
Key Software Features of the people counting system:

- It has an ability to connect the retail traffic counters along the LAN or WAN, PC serial port, modem.;
- Menus permit for quick system setup and report generation;
- Ascendable database structure allows to easily customize the business to traffic counting systems analysis;
- Extensive reports can be presented in chart, tabular or drill-down formats;
- Unique snapshot reports, such as the Weather and Special Events Report, enable to examine traffic with what else is happening at concrete site;
- Four different formats of report styles, and numerous trending and period analyses;
- Flexible data import and export;;
- Interfaces are available to link sales and staffing data to analyze each in the context of traffic;
- Can create reports and generate them from the latest data.
- No need any software or data storage for the system.

FootfallCam People counting System (5)

FootfallCam is a British company, started by a team of experienced engineers with the vision of creating the most advanced people counting system in the market. Through many years of research and development, FootfallCam has developed into a technologically sophisticated system with many world's first innovations.

FootfallCam Software Suite is designed for customers who are managing large number of counters. It collects data from all counters and store them in a single place where user can carry out deep analytics.
Stereo Vision (3D) People Counting-has the ability to see an object in 3D form in details such as height, width and depth by viewing a scene from two slightly different angles.
Benefits
Wi-Fi counting- i uses ses Wi-Fi receiver to pick up unique Wi-Fi beacon signals emitted from the smartphones with a range of up to 100 metres.Combining both technologies together allow Wi-Fi data to be normalized with the large sample size statically neutral video counting data to have an accurate picture of the overall customer shopping behaviour
Most accurate.Using 3D counting technology. Most features rich.Wi-Fi analytics, video counting, etc.
Most powerful hardware.With quad-core 1GHz processor and built-in graphics card. Most comprehensive software suite.

Possible uses of things we found in the system

What can we use in our services?

Access control service:
RFID technology is the badge or bracelet. Safety and quick response of the RFID system, it acts automatically on the specified parameter: for example, it sends a signal to the system if someone has entered the closed zone without access. If someone who get inside the room without access you can find and track him, search the database you can find, wherever he is.
Monitoring service:
RFID system is very convenient and good to track and monitor people, it is possible to calculate them and even know the exactly time when a person entered the room, how many stayed when he left.
Traf-Sys can also count people by using LAN or WAN, a PC serial port, or a modem.
Stereo Vision (3D) People Counting- we can see the object in our case the people in a format of 3D by viewing a pictures from two different angles.
Wi-Fi and video counting, Wi-Fi beacon signals emitted from the smartphones with a range of up to 100 metres can help us to know exactly the number of people inside the room.

Informal Description of our system and its Software/System Architecture

Introduction

In this part of the document the system and its (software) architecture is described. The chapter starts with a description of the system we have in mind. After that follows a possible scenario in which our system would be used, which helps to create a more concrete and realistic view of the system. Then the system architecture is described by a component diagram. This diagram displays the components, subcomponents and the connections/interfaces of the designed system. After that a suitable architectural pattern is identified that will be used to build our system. The chapter is concluded by a distinction between system components that will be produced in house and those that will be produced or owned by other parties.

System description

The system we have in mind can be especially deployed in close spaces (some services concern also open spaces).

The system architecture consists of two main macro-components:

1. A monitoring sw/hw sub-system, taking care of collecting, storing, organizing, and analysing big amount of sensor data coming from the environment;
2. A dashboard for first responder and administrator and an user app for event/building information, parking, renting and "I am safe" service, user recommendation, evacuation guidance.

The monitoring sub-system may contain buildings, rooms and parkings where sensors are placed in order to collect row data. The basic kind of sensors we can think of are: cameras, earthquake sensors, RFID/NFC sensors and readers, and so on (e.g., mobile phones, social media).

The dashboard sub-system for the administrator, instead, has the main objective of showing the current state of the buildings/rooms/parkings on a configurable dashboard. The dashboard shall highlight the critical situations so to inform the civil protection personnel.

The following list represents all the services that our system could implement:

- **Access Control service:** Different users (with different rights) are enabled to access to offices, laboratories, parking spaces etc. The basic service consists in checking the users' rights, track entry/exit time, and deny or allow access (e.g., preventing access to high-risk areas close to an hazard). Different stakeholders with different roles may receive different treatments. Employees can check in with their cards. This allows the management team to see when people are in and out of the office. Also, when employees want to have lunch or take some office products from a vending machine in the office this card and system can be used;
- **Monitoring service**: check through cameras how many people there are in the room;
- **Rental service**: NFC cards can be used to rent a room, laboratory, parking space or other location. The entry time is saved and the time the person leaves as well. This combined forms the total time a person has used a commodity and the costs of the rental can be charged to the person's account. Also, an overview of the availability can be made available, so people can see which rooms or parking spaces are still available;
- **Find the nearest parking spot**: Use image recognition on camera images on the streets to look for free parking spaces and guide the user, based on the current location, to the nearest parking spot;
- **User event information:** See information about an event/building and enroll to the event;

- **User recommendation**: Send a user a recommendation on whether to go to an event or not based on how crowded it is;
- **"I am safe" service**: In case of an evacuation a user can register her/himself as being safe via the system. Buildings can have people registered via the NFC entry system. These people will all be stored in a DB and be connected to this particular building. If people use the system to say that they are safe, this will show on the list. This helps emergency teams to see if people are missing or not and might be still (trapped) inside. There might also be a 'Safe me' option in the system that sends the user's location when he/she is trapped and needs to be saved;
- **First responders communication service**: the system shall include a platform that enables "first responders" (like operators, fireman service, emergency, police) to interact and exchange information even in case of emergency;
- **Evacuation guidance:** the building can be equipped with fire and other security sensors. When one (or more) of those sensors are activated, information from cameras are collected to locate people in the building and to guide them towards the best exit. Once people are located, they are reported on a map owned by the security people. In this way, they know areas that are more populated, with respect to the diagnosed security problem. They can then be directed to different emergency exits, so to avoid the overload of only a subset of those exits;
- **Smart Information service**: Our system collects a big amount of data, that while computed, provides situational aware information about the monitored area. Some of this information can be made available to the population, so to increase their awareness on the surrounding.

A day at the Uffizi: a practical example

An example of the building where our system could be deployed is a museum, such as the Uffizi. In the morning the employees and guests of the Uffizi could arrive by car at the museum. They can use the 'Find nearest parking spot' service of the system to find a parking spot in the crowded streets of Florence. When the user has parked, the parking spot can be rented using the NFC card of the user and the 'Rental' service of the system.

When the visitors of the museum enter the museum they can use our app to see information on how crowded the museum is and how crowded the different rooms are. Based on this the system can provide the user with advice to go to a certain place or to wait until the place has become less crowded. This uses the 'User event information' service and the 'User recommendation' service. These services are supported by the 'Smart information' and 'Monitoring' service.

Employees have to check in/out with their cards so the system can track the enter and exit time. This allows the administrator team to see when people are in and out of the office. If an employee wants to have lunch or take some office products from a vending machine in the office he can use this same card to do this.
Some paintings and the other works are displayed in glass show cases. An employee can only use his card to open the show case (only the employees having the authorization). This is part of the access control service.

In each room of the museum there are some cameras to count the number of people that are inside. So, thanks to this information the administrators can see the current state of the museum, rooms and parking spots on a configurable dashboard.

In case of an emergency, for example when a fire breaks out, all the people inside the museum are guided towards the best exit by using our application. Operators/fireman/police are called through the service “First responders” where they can interact and exchange information.
When the visitors are outside the museum, they are guided to use the “I am safe” service. Thanks to this service and the system information concerning the number of visitors that were in the museum, the operators and administrator know if there are people missing. These people might be trapped somewhere. These people should use the service “Safe me” so the system sends the user’s location to the first responders and he or she may be saved.

System architecture

In figure 1 a model of the architecture of the Masaccio can be seen. At the center of the system is the ‘System manager’. The System Manager is a software component that is located on a server and serves as the communication channel and logical part of the system. This ‘System manager’ component consists of four subcomponents.

The first sub component is Spark Streaming. The Spark Streaming component handles the data streaming from Kafka. Because the Kafka component sends data to the Spark Streaming component, the Spark Streaming component provides an interface to the Kafka component. The Spark Streaming component consists of two subcomponents. The first subcomponent is the Spark Consumer. This component has a streaming channel that receives the data stream from Kafka and converts this to an JavaInputDStream. This can then be used by the second subcomponent of the Spark Streaming component, the Spark Processor component. This component processes the data. This means that it can store the data in the Cassandra real-time database and or call other functions depending on the input. It is connected to the ACL because the Spark Processor component needs to know if a user or event is allowed to call a certain function. Because it requests a service from the ACL the ACL provides the interface. Also, the Spark Processor component can call the Actuator Software component. This is for instance for opening doors when a user uses the Access Control service. Because the Spark Processor component calls the Actuator Software component the interface is provided by the later.

The second subcomponent of the System Manager component is the Access Control Layer (ACL) component. This component is used to check whether a user or a process has a right to perform a certain action or gain access to a particular resource. The Spark Processor component uses this to control whether a uses may access a particular room, as part of the Access Control service. The DashBoard REST and the app REST components use the ACL to see whether a user may send messages or request certain information or resources. Because the other components all use the ACL, the ACL provides an interface for these components. The ACL component uses the App Data component to determine whether a user is allowed to perform a particular action. Since the ACL uses the App Data component, the App Data Component provides an interface to the ACL.

The Kafka component receives IoT data events from the sensor SW component and publishes this data with a particular topic. This data is then processed by the Spark Streaming component and then put in the database. Also, the (critical) messages that can be send using the Administrator Dashboard component are published by Kafka. These messages first go through the Dashboard REST component. Because the Dashboard REST component and the Sensor SW component send information to Kafka, it is the Kafka component that provides the interface. Since Kafka publishes this data, which is again consumed by the Spark Streaming component, it consumes the interface of the latter.

The fourth and final component is the Spring component. This component provides the direct back-end communication for the front end application. The first subcomponent is the Dashboard REST component. This provides all services for the Administrator Dashboard. This includes insight in information and the first responder communication. It provides an interface to the administrator dashboard and uses the interfaces of Kafka, the ACL and the real-time database. The second subcomponent is the App REST component. This component provides all the services for the User App front end component. This is the app that normal users can use to view information on events, receive recommendations, find a parking spot and use the other services. The App REST component provides an interface to the User App component of the Front End Application component. It uses the ACL, Real-Time, App Data, Kafka and SpotFi component. The SpotFi[9] component is a piece of software that is used to precisely locate a device using wifi access points and a technique called triangulation. The App REST component uses the SpotFi component in the evacuation service, to locate the user and use this location to find the best evacuation route for this user. Therefore, the SpotFi component provides an interface to the App REST component.

The database component provides an interface to the ‘System manager’. This is because the ‘System manager’ will always instantiate the calls for CRUD operations on the database. Therefore, the database is the provider and the ‘System manager’ is the consumer. The database component consists of three sub-components. The first sub-component is ‘App data’. This part of the database contains the information concerning users, building and rentals. Rentals include objects or spaces that can be rented, but also if these spaces are rented at the moment and how much people eventually need to pay for a rental. This database is a MySQL database. The second sub-component in the database is the ‘sensor data’ component. This component contains all information regarding the data that the sensors produce and the analyses that have been performed on this data. The ‘sensor data’ component is again split into two sub-components. The first sub-component is the ‘real-time’ component. This component stores the data that comes in (for only one hour) and allows fast access by the ‘System manager’ so analytics on the data can be performed in real time and used to create data insights in the front-end application. The focus of this component is that it needs to be fast and allow concurrent access to the data. The second component is the ‘long-term’ component. After an hour the data from the ‘Real-time’ component will be transferred to the ‘long-term’ component. Here the data will be stored for a longer time and it is less important to have super fast access. This operation is initiated by the ‘long-term’ component. Therefore, the ‘real-time’ component provides an interface to the ‘long-term’ component. Both these databases are Cassandra databases. Cassandra is uses because it is extremely scalable, distributed, fast and fault tolerant. Therefore, it is very suitable for the large amount of data that the system gathers and it can easily when more organizations use the system.

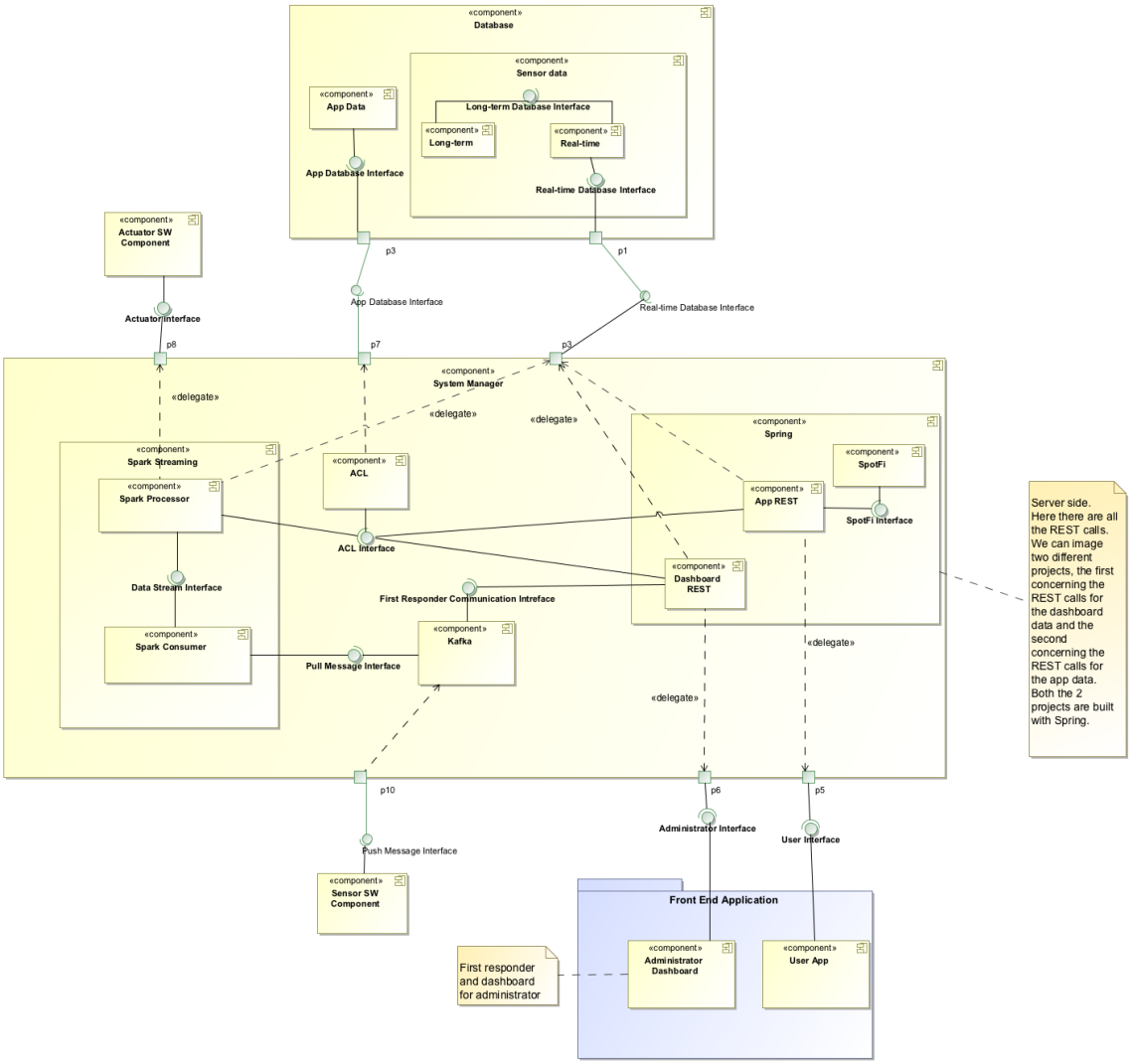


Figure 2. Component diagram of the Masaccio system

To the right of the Database component is the ‘Actuator SW’ component. This component contains the hardware that can ‘do’ something for the system. An example would be opening a door or providing feedback in the form of sound or a green/red light. The ‘Actuator SW’ component provides an interface to the ‘System manager’ because the ‘System manager’ calls the hardware of the ‘Actuator SW’ component to perform an action. Since this means that the ‘System manager’ uses the ‘Actuator SW’ component the interface is provided by the ‘Actuator SW’ component.

Below the System Manager the ‘Sensor SW’ component can be seen. This component includes all the hardware and software that are used for measuring the sensor data and sending this data to the ‘System manager’. The interface is provided by the ‘System manager’ because the ‘Sensor SW component sends the data to the ‘System manager’ and the ‘System manager’ does not request this data from the ‘Sensor SW. The data generated by the ‘Sensor SW’ component is sent to the database by the ‘System manager’ and stored there.

The last component is the ‘Front end application’ component. This component is divided in two sub-components. The first sub-component is the ‘Administrator app’. This app has a dashboard that provides an overview of the information on the locations that are monitored. Alarms can be sound here and messages received. It is also provides a platform for the first responders to communicate. Furthermore, rights can be administered to roles and roles can be administered to users by using the administrator app. The second sub-component of the ‘Front end application’ component is the ‘User

app' component. This component is for 'normal' users of the system and provides them with information and recommendations concerning events and/or locations. Furthermore, it allows the user to use the parking, rental and 'I am safe' or 'I need saving' service. Also, in case of an emergency evacuation, the application helps the users exit the area through the least crowded exit points.

Suitable architectural patterns

A suitable architectural pattern for the Masaccio system would be Model-View-Controller. Len Bass *et al*[8], mention that Model-View-Controller, from now on MVC, is good when users interact with the system and want to see the data from different perspectives. Therefore, the architecture is good for providing different overviews and dashboards for monitoring the locations. Furthermore, it is suitable for developing the rental, user information and user recommendation services. Also, it makes it easier to modify the front end of the system in order to add perspectives to the dashboards or extra services. This is because there is a strict distinction between the user interface (the view), the logic (the controller) and the data (the model). In the case of our architecture, this MVC architecture will also be a client-server architecture, where the view will be located at the client and the server will host the controller and the model.

In the Masaccio system this architecture model is used by the Front End, Spring and Database component. The front End Application component is the view, Spring the controller and the database is the model. The view contains the dashboard that shows the information and graphical user interfaces that show information or allow the user to request a service. These requests are then send to the Spring component, the controller. This controller handles the request by performing the logic, retrieving or storing the information in the database and returning the requested information or service. Also, the controller uses the ACL to control whether a uses has the right authorization to request a service or certain data. For all these functionalities MVC is very suitable because it provides a clear distinction between different parts of the code. These parts can then be internally changed as long as they provide the same interface. This is a useful abstraction and thus an advantage provided by the MVC architecture.

The second architecture that is used in our system is Kafka. Kafka is an architecture that uses a publish-subscribe pattern. Publish–subscribe is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be. In our system, the publishers are the sensors like cameras or NFC Readers and the subscriber is the Spark Streaming component. Also, the first responder communication is published via Kafka. So, Masaccio is a topic-based system in which messages are published to "topics" or named logical channels. Subscribers in a topic-based system will receive all messages published to the topics to which they subscribe, and all subscribers to a topic will receive the same messages. The publisher is responsible for defining the classes of messages to which subscribers can subscribe.

For this reason the system uses the Kafka architecture. All Kafka messages are organized into topics. If you wish to send a message you send it to a specific topic and if you wish to read a message you read it from a specific topic. A consumer pulls messages off of a Kafka topic while producers push messages into a Kafka topic. Lastly, Kafka, as a distributed system, runs in a cluster. Each node in the cluster is called a Kafka broker. This again is very scalable and therefore useful when the number of organizations that use the system increases.

Inhouse versus external components

In order to realise the designed system, some components will be developed in house and be owned by the organization building the system and other components will be developed and owned by third parties.

Inhouse

The ‘Front end application’ component will be developed in house. This is because this component is very tailored to our system and therefore it is not likely that there is a third party that can provide a premade solution for the ‘Front end application’ component. In the component there will be third party libraries used, for instance for data visualization or Javascript usage, but the overall design and functionalities will be developed in house.

The same reasoning can be used for the development of the ‘System manager’ This back-end component will probably be realised using a framework (such as Eclipse Spring). However, the functionalities that will provide the specified services still need to be created. Again, because this is very tailored to the system needs and specifications it is not likely that there is a premade solution by a third party and thus better to develop in house.

The analytics component transforms the crude data into insights that are useful to the administrator of the system. This analytics component can use libraries and/or software from third parties but will run on the servers of the system. Therefore, it is not owned by a third party and thus an in house component.

Hybrid

The database component will be realised by using Infrastructure as a Service (IaaS). This means that the servers on which the database will run will not be owned by us. The reason for this is that we are not hardware experts and therefore we are not good at maintaining servers. In order to remain focused on our core business we use IaaS. Then the physical servers are owned and maintained by another party, but we have all the control over the database and how to run this.

External

As mentioned in the assumptions chapter of this report, it is assumed that the hardware is in place or can be placed at the monitored locations. The system can use the data that these sensors produce but the sensors are not owned by the system. For example the camera’s, NFC readers and the infrastructure to send this data to other (parts of) systems is owned by the owner of the locations. For an open location such as a street or a square this might be the local government and in a museum or office this is the owner of the building. Because the system only uses the data of the sensors and is not the owner of the hardware, the ‘Sensor hardware’ component is considered an externally realised component.

The same reasoning can be used for the ‘System actuator’ component. These actuators, such as door locks or feedback hardware (sound or visual), are located in the buildings or on the streets. Therefore, this hardware is not owned by the system, it is only used by the system. Therefore, the ‘System actuator’ is an external component of the system.

User Stories

List of actors

The following actors have been identified that interact with the system.

- User;
- Civil protection operator;
- Security manager;
- Sensor network Administrator;
- System Administrator;
- Organisator of an event;
- Owner of a building;
- The local government/city.

Users are the guests or clients who use the system to get an access to specific area such as libraries, museums, universities, malls, etc. Each of users have different authentication so their access can be denied, if they don't have proper rights to go inside. For instance in museum, the staff and guests authority is very different, ones has access to each rooms, other only to these which they can visit.

In order to get information about free rooms, libraries, closest parking spot and to rent the place from these areas each user can utilize his own NFC card, by checking smart information, find the parking spot and user event information services.To get information about event and enrollment they have user recommendation service in which the event organisator publish every details about the event or the whole building.

Moreover, to inform the organizational staff of the event about their safety users have an access to "I am safe" service. It ensures to know, that this person is safe when he get out from the building.

Civil Protection Operator is plays the role of the first responders in communication service, so in the case of emergency he interacts with the system administrator in order to exchange information and inform the latest situation inside. He uses the smart information service to inform and exchange information.

Security Manager is also one of the first responders who can deal with smart information to be ensure about the security and safety.

Sensor Network Administrator checks the access control to be sure than no authorized people can get access or conversely the one who has access cannot get inside.

System Administrator team see people movements (going in or out from the room/event) , checks how many people inside the room to collect the data about density of crowd get the smart analyses from the monitoring service. It can be some devices that gives the accurate numbers or a pictures which later analysed due to get concrete number of people.

Organizer of an event is responsible to arrange the event, check and publish the information about the enrollment and its conditions to do it easier for users. He someone who has access to both user event information and user recommendation services.

Owner of a building is connected to smart information to provide the event or building with the all needed conditions and things to ensure that nothing can affect the process of an event.

The local government/city has the control over the images on the street taken by cameras.

Functional requirements

List of basic services/requirements

Described below are the basic services which make up the ten main services that we have described before.

1. Read sensor data from a NFC card, from RFIDs, turnstile, cameras and more;
2. User rights check: the system checks if the user has the right to access a certain space- sees only the authority can you go inside or not;
3. Run-time Access control: depending on runtime data a user's rights may be changed to disallow the user to enter a place. For example when that place is too crowded;
4. Check-in, Check-out: when the user accesses a space, its presence is registered in a DB; (entering a building or renting something);
5. Statistics: to compute statistics out of the system usage;
6. Data Analytics services: the service in which the data from the system is analysed in order to get concrete numbers and trends;
7. Perform actuator action: actuators are responsible for moving or controlling a mechanism of access in our system;
8. Store and retrieve data in/from DB;
9. Show data on dashboard for administrators;
10. Show data in user application: event/building information, parking, renting, "I am safe" service, user recommendation and evacuation guidance;
11. First responders communication: the platform where operators can interact and exchange information;
12. Administer rights to users: give roles to users. Different for organization staff and guests of an event;
13. Locate person in case of emergency: the Civil Protection Operator is responsible for this service, in cases of emergency he informs people about this;
14. Guide users to best exit in case of emergency using the app;
15. Report users/visitors on map owned by the security operators;
16. Let user enroll to an event: user event information service in which users can get information about event/room and can enroll;
17. Recommend user to go to an event or not (based on how crowded it is): user recommendation service in which the system recommends users to go to an event based on their preferences or not to go because it is too crowded;
18. Allow user to identify him/herself as being safe or in need of saving: after exiting an area/event users can register themselves as being safe by using the "I am safe" service;
19. Buy lunch or products from a vending machine with NFC card: the rental service which can be used to rent a place in a library, room or parking spot can also be used to register the purchase of a lunch or office products using the NFC card;
20. Show availability of rooms and parking spaces that can be rented: user event information service also provides some information about free and close parking spots;
21. Analyse street images from camera to identify free parking spaces: the monitoring service uses cameras to recognise free parking spots and send information to the free parking spot service;
22. Guide a user to a parking space based on its current location: using the monitoring and parking spot services in order to find the nearest parking spot.

Linking the functional requirements to the services

In figure 3 an use case diagram is displayed, showing which stakeholders make use of which particular use case.

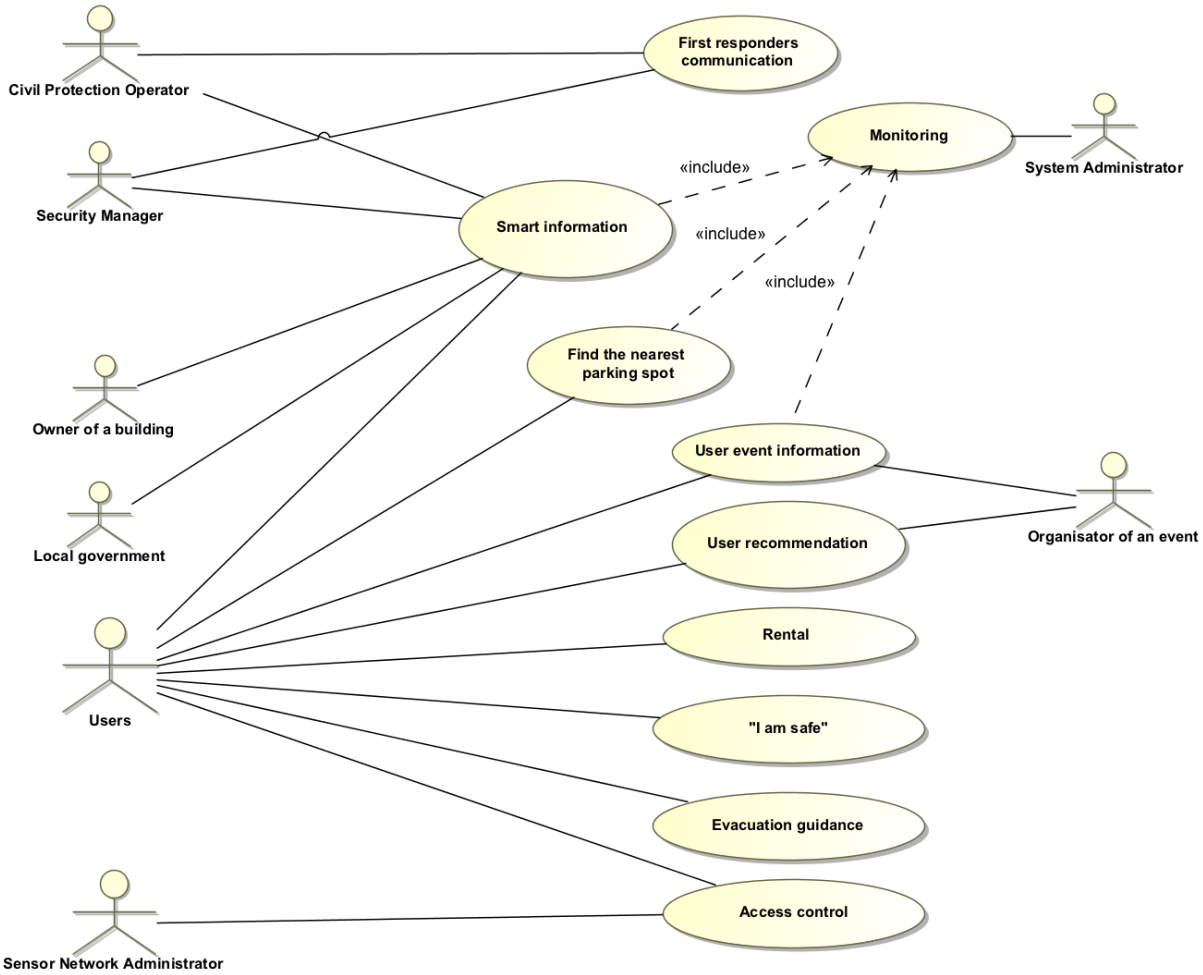


Figure 3. Use case diagram of the Masaccio services

Below are the descriptions of the different user stories, which are written using the template **<**stakeholder**> - <**main service**>, <**basic services**>**. The main service is described by its name and the basic services that constitute the main service are written by their number, which has been assigned to them in the list above. The services are ordered based on their architectural importance. The five most important “main service” - ”basic services” pairs are described in detail. For the others pairs only the linking will be given.

User - ”I am safe” - 10, 18:
The service “ I am safe” is consumed by the stakeholder user and consists of the basic services 10 and 18. With the help of app/website the user can inform the organisators, security of the event that he/she needs a help by sending a message or just “I am safe” notification, which is allowed to send both after going out the building and inside the building in case if you have a problem. The sending message text may look like this: ”Save me”.

User - access control - 1, 2, 3, 4, 7, 12:
The service Access Control consists of the 1st, 2nd, 3rd, 4th, 7th and 12th basic services. At the first to be accepted the service need to read data from NFC card, RFID readers and so on, which is

exactly the first basic service. It is clear that without giving an authorization or rights there will be no differences among the event staff, members of organisation, guests and visitors. So using the second basic service our system recognize who has access to event/room. For some reasons such as density of people in room/event some guests cannot enter the room even if they have access control with the aim to their safety. To be sure that how many people are at the moment in each room the data about the access inside or outside is sending to the DB.

Civil Protection Operator - first responders communication - 9, 11, 13, 14, 15:
The first responders communication service is consumed by the stakeholder Civil Protection Operator and consists of the basic services 9,11,13,14,15. This is a service by which Civil Protection Operator can inform people about emergency situations, send them the map and show the emergency exits of the building/event by using the dashboard for administrators.

Security Manager - first responders communication - 9 , 11, 13, 14, 15:
It is responsible of the safety of people either in the process event or in cases of emergency. Uses almost the same basic services as a Civil Protection Operator.

System administrator - monitoring - 1, 2, 5, 6, 8, 9:
Monitoring is one of the main services in our system, that is consumed by stakeholder System Administrator and consists of the basic services: 12, 5, 6, 8, 9. In order to get right statistics first system need to read a data from each DB real-time and long term.

The given below list is the user stories which described in the same template that are less architecturally important than the ones which described above

- User - smart information - 1, 5, 6, 8,10;
- User - find nearest parking spot - 8, 10, 20, 21, 22;
- User - user event information - 8, 16;
- User - rental - 8, 10, 19;
- User - user recommendation - 8, 16, 17, 19;
- Civil Protection Operator - smart information - 5, 6, 9;
- Security Manager - smart information - 1, 5, 6, 9, 10;
- Owner of a building - smart information - 1, 5, 6, 8, 10;
- Local government - smart information - 1, 5, 6, 8, 10;
- Sensor network administrator - access control - 1, 2, 3, 4, 12;
- Organisator of an event - user recommendation - 14, 16, 17, 19, 20, 21, 22.

Basic Hardware components

The following list shows the hardware components that we are going to use in our system and how they will be used.

- Situational Awareness Sensors: cameras. They are used for check how many people are in each room and camera images on the streets to look for free parking spaces;
- Readers: RFID, NFC, card readers. Through NFC/RFID readers we can check the users' rights, track entry/exit time, and deny or allow access. NFC cards can be used to rent a room, laboratory, parking space or other location;
- Displays: kiosk, tablets, interactive displays, etc. They are used by employees, guests or administrators;
- Actuators: alarms, automatic door actuators, etc.
- Servers for DB, REST API and hosting the user application (events, dashboard, renting and parking spot location).

Extra-Functional requirements

- **Reliability**: Reliability, or dependability, describes the ability of a system or component to function under stated conditions for a specified period of time;
- **Security**: the protection of computer systems from the theft and damage to their hardware, software or information, as well as from disruption or misdirection of the services they provide;
- **Scalability**: the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth;
- **Fault tolerance:** the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components;

- **Portability**: The usability of the same software/system in different environments;
- **Interoperability**: a characteristic of a product or system, whose interfaces are completely understood, to work with other products or systems, at present or future, in either implementation or access, without any restrictions;
- **Performance:** The amount of work accomplished by the system. This may be measured using for instance the response time or the throughput;
- **Maintainability:** The ease at which the system can be kept operable. This concerns the difficulty of finding and fixing bugs, introducing new features and cope with change of the environment;
- **Usability:** The ease at which the system can be used by humans. In other words the user-friendliness. This also concerns how much new users of the system need to learn before they can use the system.

Why do these non-functional requirements have an impact on our system and why are they important?

- **Reliability**: each our system's component (software, or hardware, or a network, for example) that consistently performs according to its specifications;
- **Security:** if the system is not secure (hacked by someone) we can lose our data (the data about the number of people, density of crowd, information about employees or event), which has a huge impact on the safety of people that are aim of the system;
- **Scalability:** our system should be scalable in order to be ready to add some hardware components in each time they are needed;
- **Fault tolerance**: in case of disaster, the system should guarantee that no critical messages are lost and are delivered in at most five seconds;
- **Portability:** the system/software should be portable in order to be used by other close areas or events (It can be museum, university, school, mall or other);
- **Interoperability**: our system should be able to run application program from different vendors, and to interact with other computers across local or wide-area networks regardless of their physical architecture and operating system;
- **Performance**: how Chris Boss (the owner of Computer Workshop, which is a software development business) says: "The performance of software can have an effect on up to 80 percent of the total cost to run the datacenter - better software uses less power and requires less hardware";
- Software that isn't **maintainable** is prone to being replaced, because the company needs to be able to extend it, modify it, upgrade it, scale it, sell it, or else it's the same as junk;
- **Usability:** users should feel at ease using our products (app, dashboard, etc.). The latter should be "user-friendly".

Views and Viewpoints

Concerns are issues that an organization (different stakeholders) must pay attention to. They may affect all system stakeholders and the requirements from these stakeholders.
The following list represents the most important concerns for developing IoT system as Masaccio.

Concerns

1. Security;
2. Dependability;
3. Interoperability;
4. Connectivity;
5. Integration with Hardware;
6. Cost;
7. Performance;
8. Privacy;
9. Complexity;
10. Maintenance;
11. Data Analytics.

The following list of stakeholders represents the actors that interact with the system.
They are the same we have been identified in the “User Stories” chapter.

Stakeholders

1. User (Both administrator and final user);
2. Civil Protection Operator;
3. Security Manager;
4. Sensor Network Administrator;
5. System Administrator;
6. Organizer of an event;
7. Owner of a building.

At the highest level of abstraction, we can discern patterns that give each system a particular architectural style. For all systems of sufficient complexity, we can only understand and reason about a system’s architecture from multiple points of view, each view representing the concerns of a particular set of stakeholders.
The following table shows these multiple points of view.

Concern-Stakeholder Traceability

	User	System Administrator	Security Manager	Sensor Administrator	Civil Protection Operator	Organizer of an event	Owner of a building
Security		X	X		X	X	X
Dependability	X	X			X	X	X
Interoperability				X			
Connectivity	X	X		X	X	X	X
Integration with Hardware				X			
Cost		X	X	X			
Performance	X	X			X	X	X
Privacy	X		X			X	
Complexity			X	X			
Maintenance			X	X			
Data Analytics		X					

Table 3. Table showing the Concern-Stakeholder Traceability

User:

1. User wants “trust” the app itself and thus to be able to use it without particular worries;
2. User needs connectivity for some services (Event information, “I am safe” service, Find the nearest parking spot);
3. It would be nice for user having services as fast as possible;
4. User would like to have his or her data protected.

System administrator:

1. The most important job for a security manager is make and maintain the building secure;
2. System administrator wants “trust” the system itself and thus to be able to use it without particular worries;
3. System administrator needs connectivity for some services (Event information, “I am safe” service, Find the nearest parking spot);
4. System administrator can work based on his or her job’s cost;
5. It would be nice for system administrator having services as fast as possible;
6. System administrator occupies to manage statistic and data analytics.

Security Manager:

1. The job of a security manager is make and maintain the system secure;
2. Security manager can work based on his or her job’s cost;
3. Make the system secure may increase user’s privacy;
4. Make the system as secure as possible may increase system’s complexity;
5. Security manager should maintain the system always secure.

Sensor Administrator:

1. Sensor Administrator should be able to make different sensors communicate with each other and with the system;
2. Sensor Administrator must be concerned to not drop the connection otherwise the sensors cannot communicate with the system;
3. Sensor Administrator works with the hardware part of the system;
4. Sensor Administrator can work based on his or her job’s cost;
5. Sensor Administrator should be able to make the system as scalable as possible;
6. Sensor Administrator should maintain the hardware components updated.

Civil Protection Operator:

1. Civil Protection Operator wants “trust” the system itself and thus to be able to use it without particular worries;
2. Civil Protection Operator needs connectivity for the “I am safe” service;
3. It would be nice for civil protection operator having services as fast as possible.

Organize of an event:

1. Organizer of an event have to check if nothing can affect the process of an event;
2. Organizer of an event wants “trust” the system itself and thus to be able to use it without particular worries;
3. Organizer of an event needs connectivity for his or her job;
4. It would be nice for organize of an event having services as fast as possible;
5. Organize of an event would like to have some of his or her information protected.

Owner of a building:

1. Owner of a building have to check if nothing can affect the process of an event;
2. Owner of a building wants “trust” the system itself and thus to be able to use it without particular worries;
3. Organizer of a building needs connectivity for his or her job;
4. It would be nice for organize of a building having services as fast as possible.

UML static and dynamic architecture view

Component Diagram

The component diagram has been explained in the chapter concerning the system architecture.

Sequence Diagram

UML sequence diagrams of each service are described below.

Access control service

The figure 4 which is given below describes the access control service in Masaccio system. Users use NFC card to get access. Sensor (Sensor SW Component) sends an IoT event/topic {userId: “”, hardwareId: “”, topic: “”} and saves the information then sends it to the Kafka with a certain topic. Kafka in its turn pull the event/topic to the Spark Consumer which then processed in Spark Processor. Then system-ACL checks the users right role which gives access to the hardware item. ACL component after checking and getting the right sends the permission or deny it. Whereupon hardware perform the access or does not perform it. The information is stored in the database, including if access was granted. An event is send to the topic to which only the specific hardware item is subscribed. If the hardware item receives an event from this topic, it performs it actuator function.

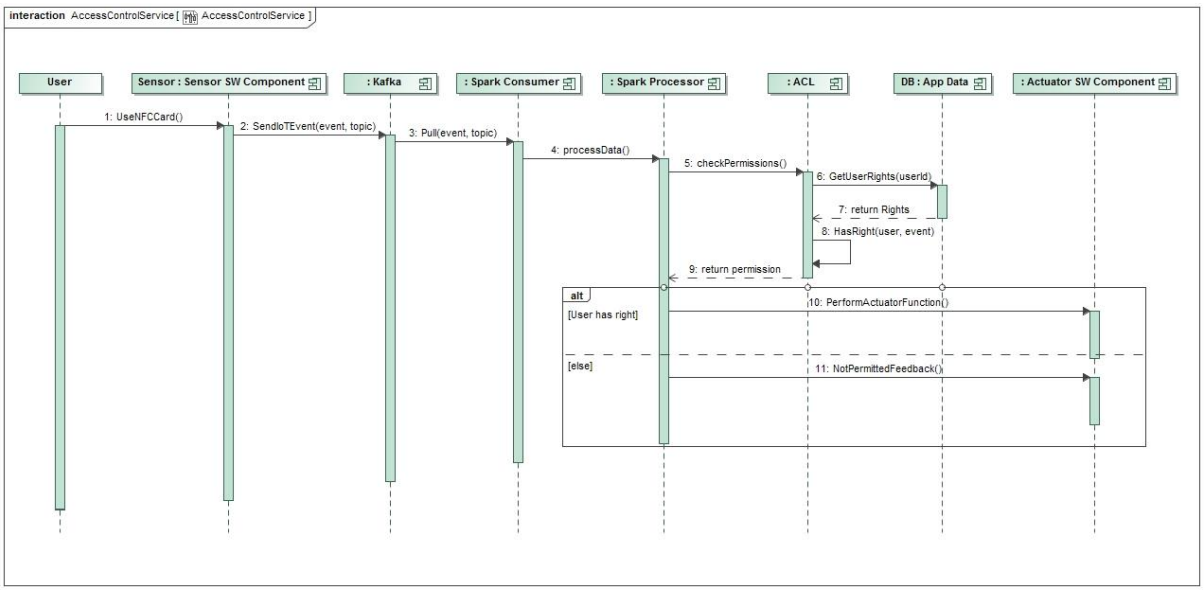


Figure 4. Sequence diagram of the Access Control Service in Masaccio system

Evacuation service

The given figure 5 illustrates an evacuation service of the system. Camera's monitor the number of people at an exit. This is done in the same way as the number of people in a room are monitored, since the exit will be from a particular room. In the database the number of people that can go through an exit at the same time is known. The Sensor gets the data from the cameras, then sends it to Kafka. Whereupon Kafka pull the event/topic to the Spark Consumer which then processed in Spark Processor and subsequently store data in Real-time database. When the user sends request to the App Rest through User App, it will try to use SpotFi (software using information from the wifi points send to the server) to locate the user based on the wifi beacons to which the person is connected. If this person is not connected to the beacons or no connection can be established the user is asked to enter the room number he or she is currently in. The system then determines the nearest exit that is not too crowded. It saves the information that a user will take this exit, so not all users will be guided to an exit that is only best at that time. Taking this into regard the system leads the user here. (9)

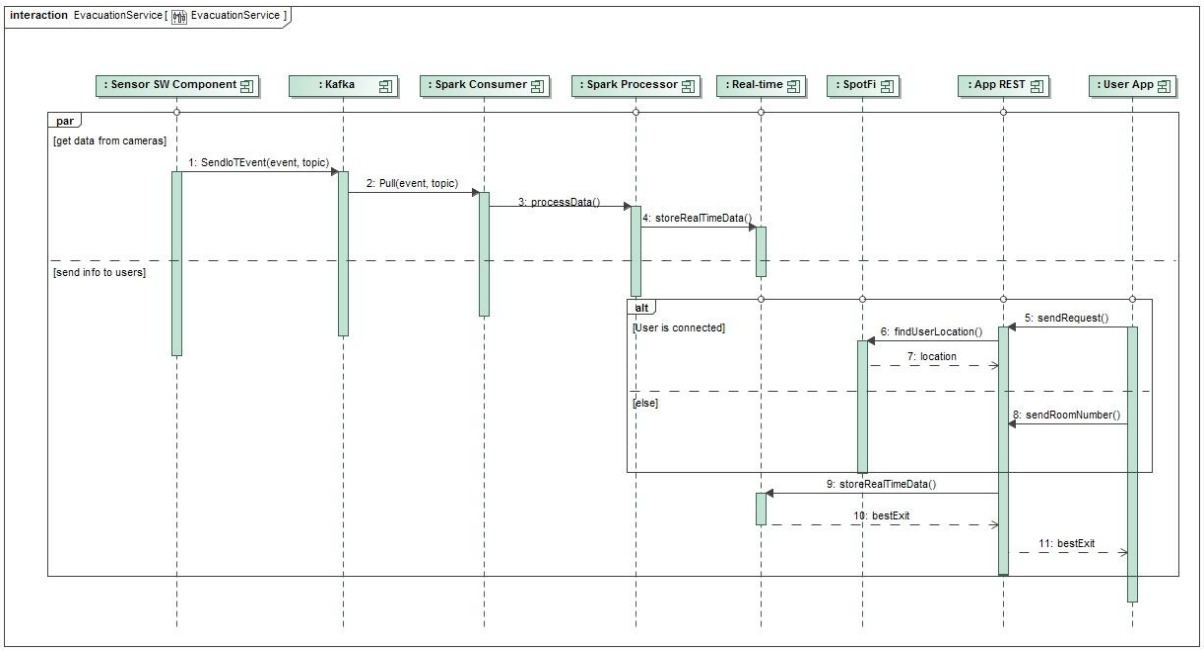


Figure 5. Sequence diagram of the Evacuation Service in Masaccio system

First responders communication service

The presented figure 6 gives information about first responders communication service. A first responder is both a potential producer and a consumer. The chat/message board is a certain topic in Kafka. All the first responders are subscribed to this topic, so they receive every message that is sent. If a first responder wants to send a message, this will be done as a kafka event. This message is sent from the dashboard to the Dashboard REST component. Then this checks with the ACL if the user can send this. Then the message is converted to a Kafka event, which is processed, stored in the database and eventually pushed by the Dashboard REST component to all the subscribers of the topic.

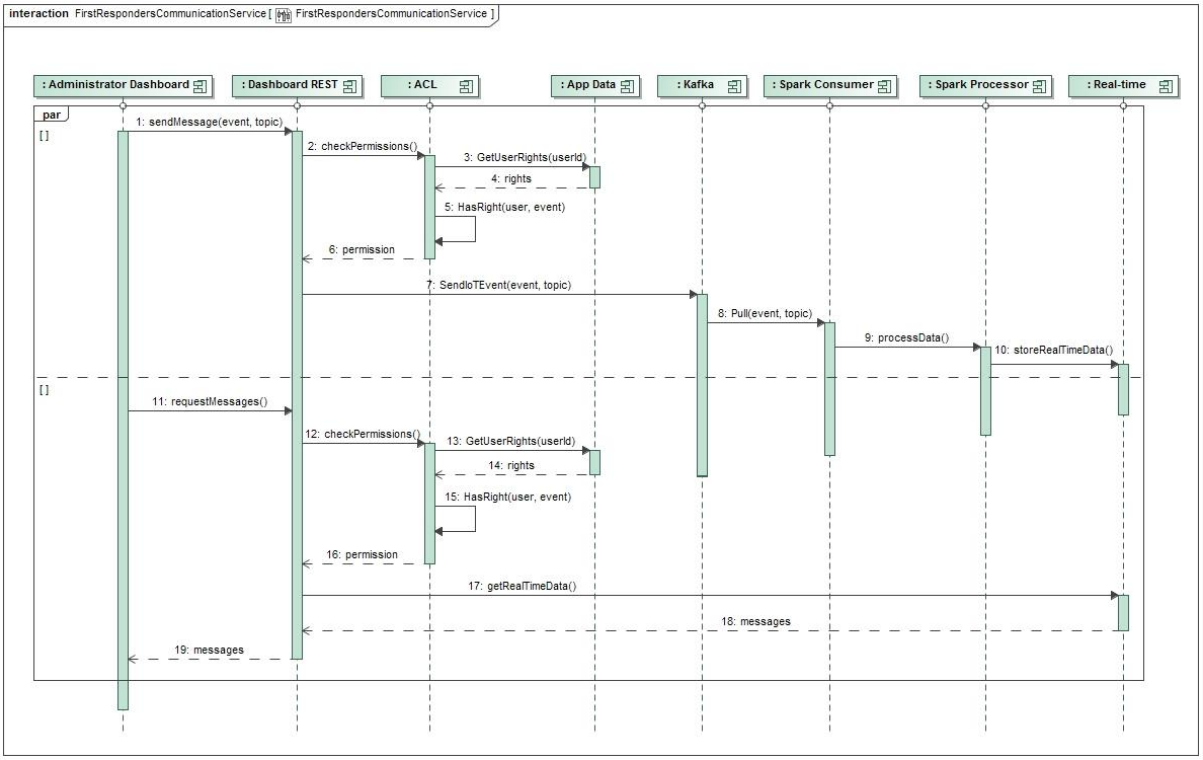


Figure 6. Sequence diagram of the First Responders Communication Service in Masaccio system

Smart Information service

The given below figure 7 describes the Smart Information service of the Masaccio system. IoT data production and Dashboard Data retrieval are produced in parallel. Namely, the sensor SW component creates IoTDataEvents with the format {locationRoomId: “”, numberPeopleId: “”}. This is sent through Kafka and consumed and processed by Spark Streaming. Then this information is stored in the Real time database-Cassandra database.administrator app requests data, then Dashboard REST component takes the information at regular intervals from the database and pushes it to all the subscribers, which are the people looking at the database.

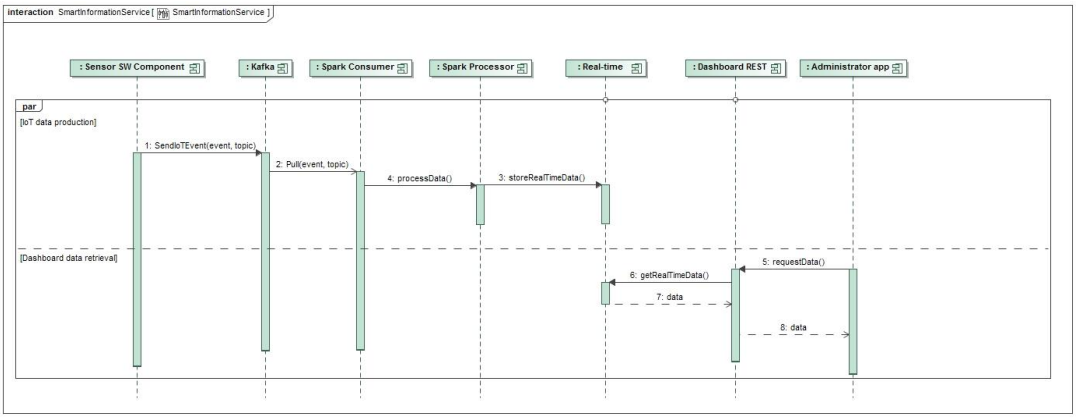


Figure 7. Sequence diagram of the Smart Information Service in Masaccio system

User event information

Simple MVC style request illustrated in the following figure 8 which gives description of User event information service. User uses the app(User App component) get the information about the event, by sending request to the App REST. Then check with the ACL component if the user is allowed to see this event and returns the rights. If the user has the right role he/se get event information. Then request the information from the App Data Database. Which is then returned and shown to the user on the User App.

For enrolling it is basically the same. However, the database stores the userId of the user that wants to enroll to an event.

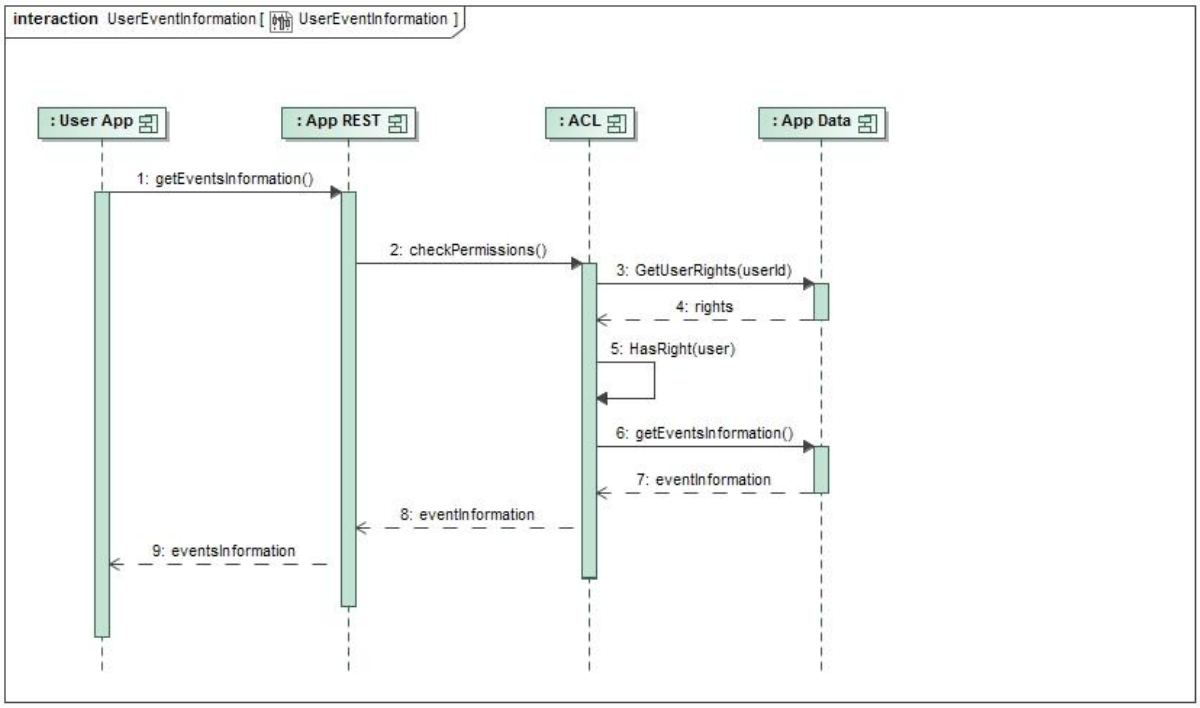


Figure 8. Sequence diagram of the User Event Information Service in Masaccio system

User recommendation

User recommendation service is illustrated on the figure 9 below. Administrator Dashboard requests the recommendation of an event to the Dashboard REST then send it to the Kafka which then pull and processed in Spark Streaming component. Checking the ACL rights it stores the data in App Data. If a person enrols to an event, this user also subscribes to the Kafka topic. The event is this topic. When recommendations regarding the event are published by the system administrator, all subscribers will receive this information.

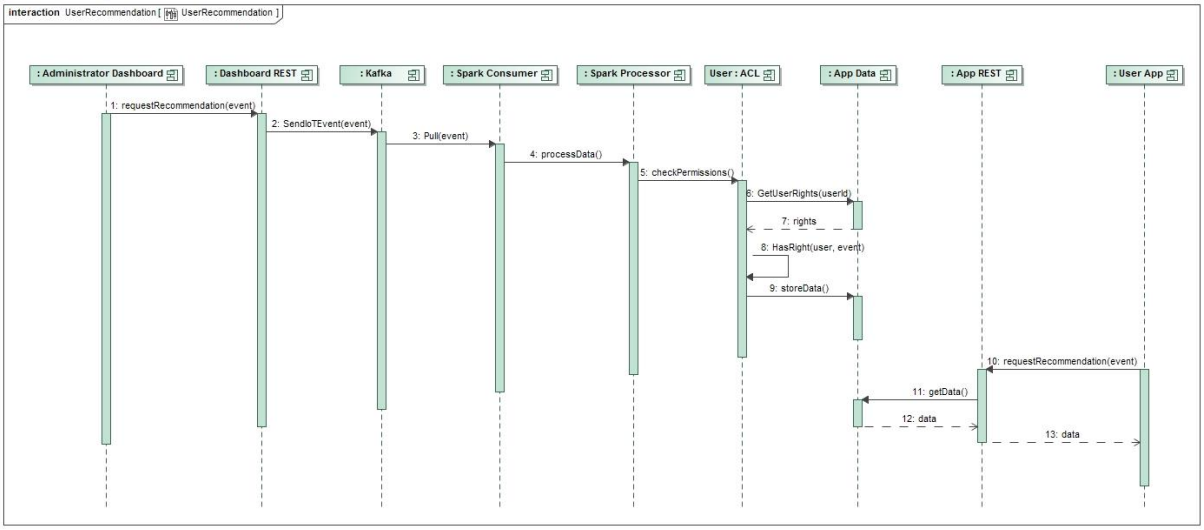


Figure 9. Sequence diagram of the User Recommendation service in Masaccio system

“I am safe” service

The supplied below figure 10 shows the sequence diagram of the service “I am safe” There are two block streams one of them is an option another one is parallel. Visitor by using the app sends sends a “I am safe” or “I need saving” message to the app REST component, which forwards this to Kafka. This is then processed by Spark Streaming. The “I am safe” messages will be shown on the dashboard, so will go through the Dashboard REST component. The “I need saving” messages will be shown on the dashboard and on the first responder communication message board. This will also go through the Dashboard REST component.

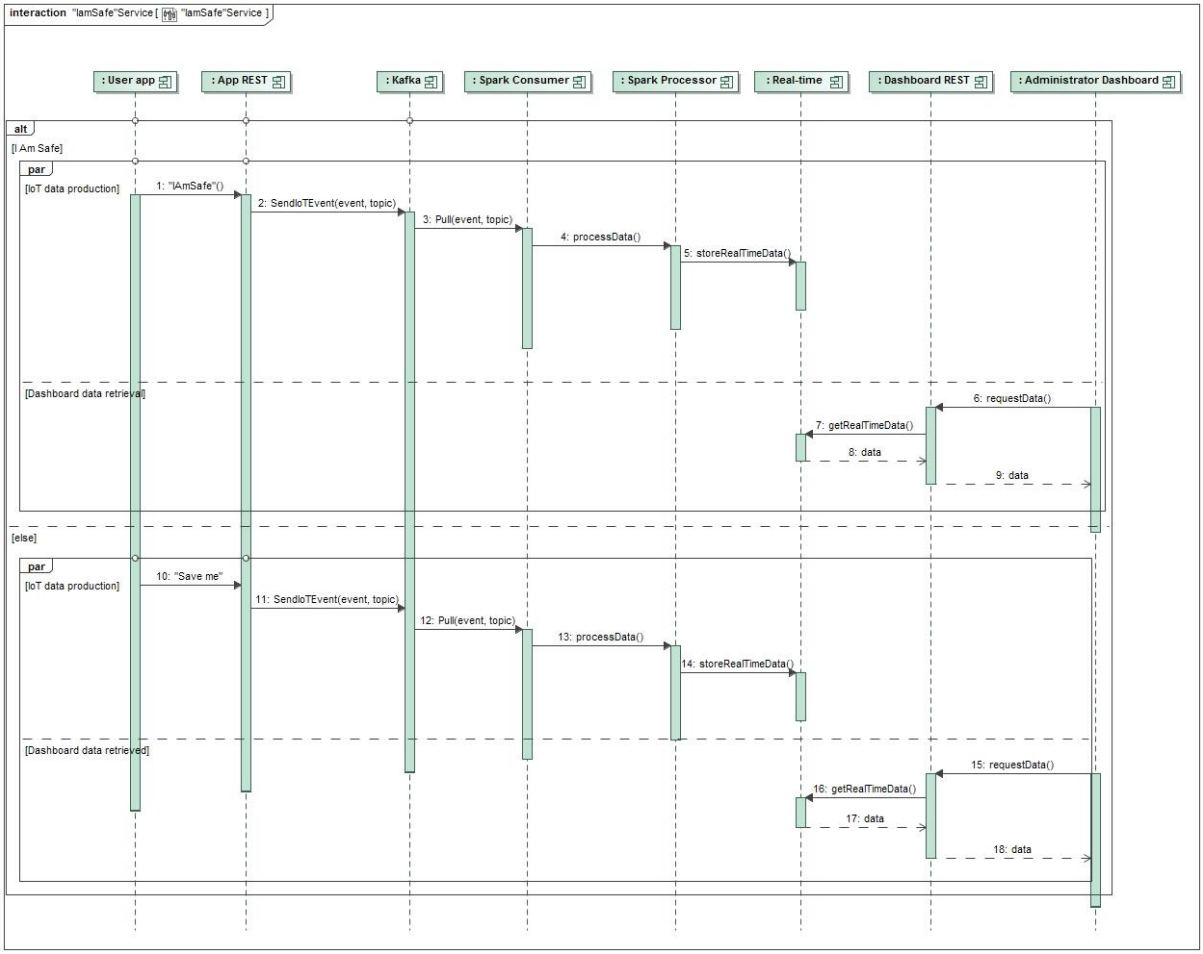


Figure 10. Sequence diagram of the "I am Safe" servin Masaccio system

Employee monitoring service

The supplied figure 11 gives information on Employee monitoring service. A check in will create an IoTDataEvent which will be sent to the database via Kafka. When a user wants to buy something this is sent as a Kafka event. It will be consumed, processed and put in the database. Then the Spark Processor will send an event to the topic to which only the specific payment hardware is subscribed. If the hardware item receives an event from this topic, it performs it actuator function.

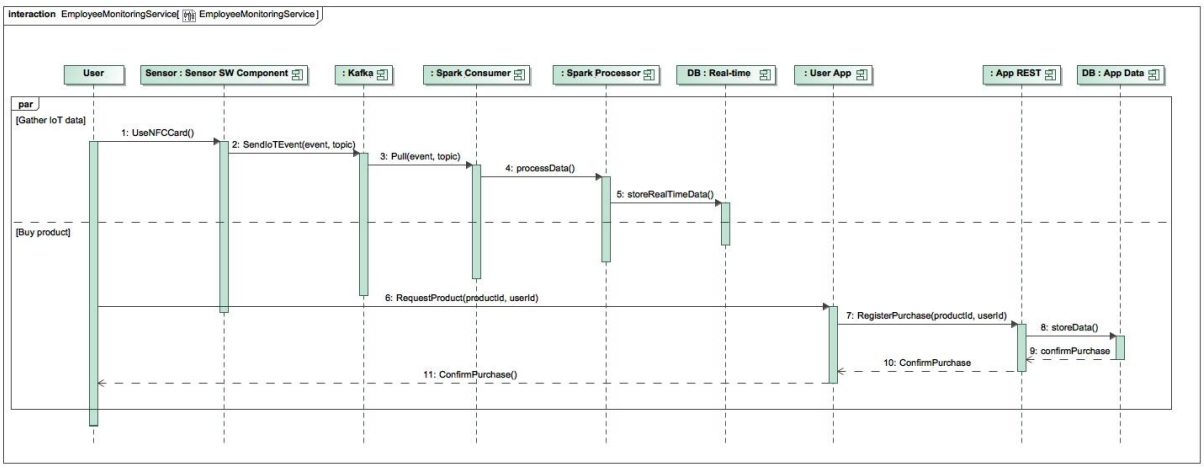


Figure 11. Sequence diagram of the Employee Monitoring Service in Masaccio system

Rental service

The provided figure 11 shows data about rental service by using sequence diagram. Then a user uses the NFC card an IoTDataEvent is sent to Kafka. Register the start time, userId and the itemId and send back an event of confirmation to the topic to which the specific NFC hardware item is connected. The system also registers when the user uses the NFC card to stop renting the item. Then the start and the end time will be used to create a transaction that will be added to the user's transaction list, which will be used to charge the user eventually.

Furthermore, when a rental starts, the item in the database is marked as "in use". This can then be used to show to the user which items are in use. The user can request from the user app which items are available. The App REST component will request this information from the App data database. This information can then be send back via the same route to the user.

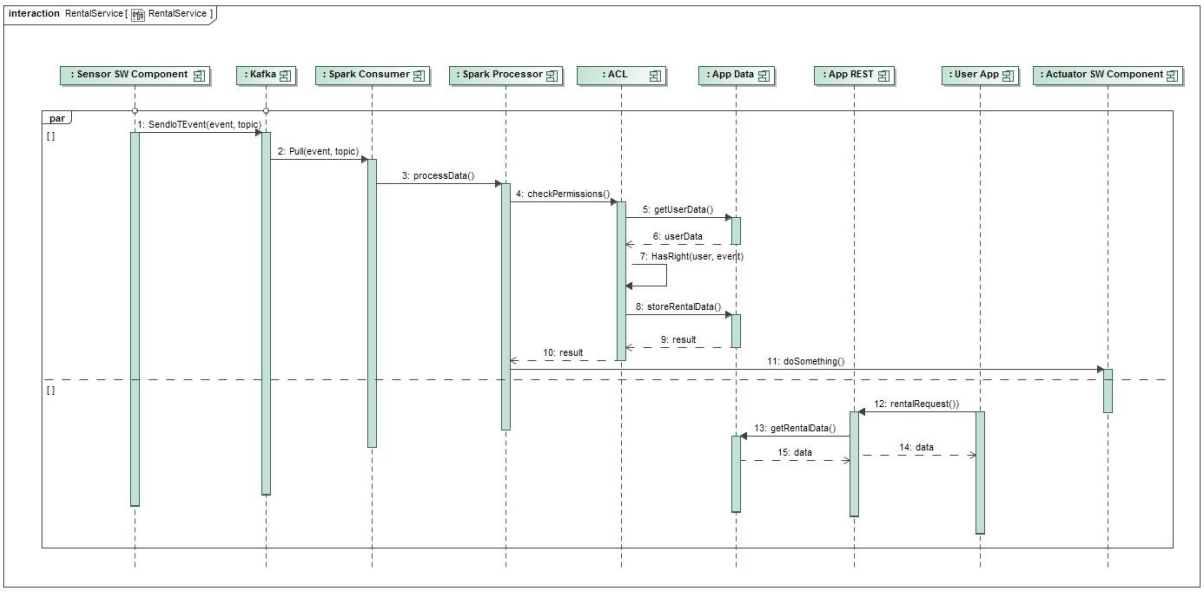


Figure 11. Sequence diagram of the Rental Service in Masaccio system

Finding the nearest parking spot

Finding the nearest parking spot service explained by using sequence diagram which is given in the figure 12. The street camera's send IoTEventData to Kafka. This has the format {streetId: "", freeParkingSpaceIdList: ""}. This information is used to update the App data database on which parking spaces are free. The user can then request the service in the User app, which takes the user's current location and sends this to the App REST component. This component checks which free parking space is nearest to the user and sends the coordinates to the user. The user can use these coordinates in Google Maps or Apple Maps and then coordinate to the free parking space.

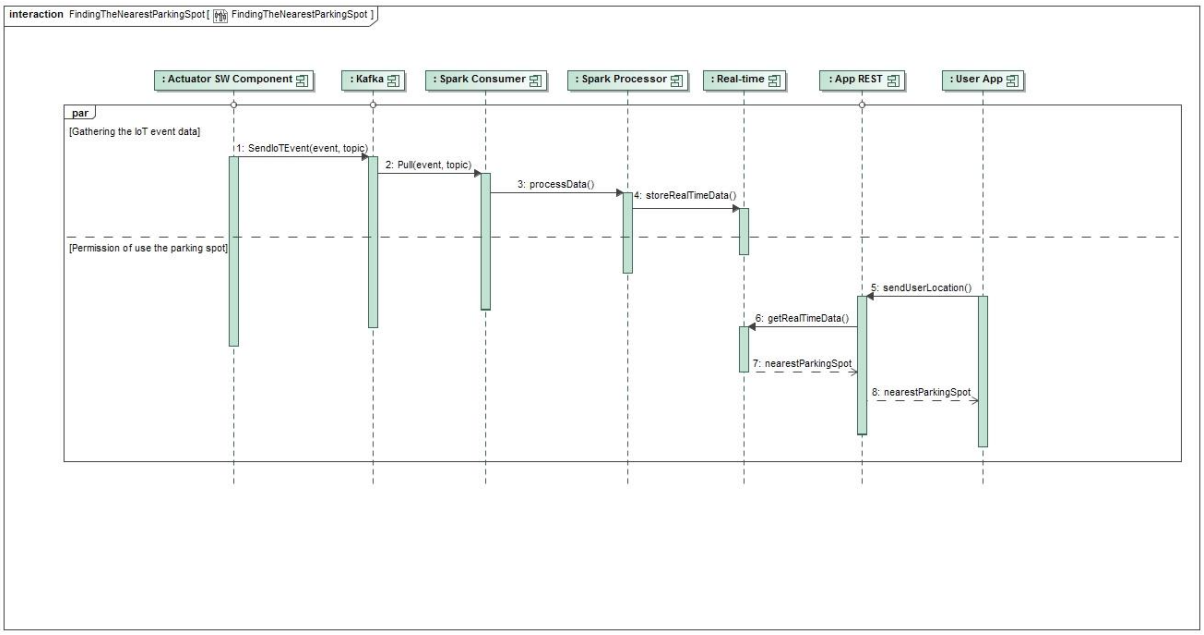


Figure 12. Sequence diagram of the Finding the nearest parking spot Service in Masaccio system

Design Decisions

Introduction

The design decisions addressed in this part of the document concern the decisions that are important for the system architecture and have a big impact on the architecture if these decisions need to be revised. The five most important design decisions will be discussed using the QOC template. This template uses for each design decision a question, options and the concerns. The design decision answers this question, by choosing one or more of the options, based on their satisfaction of the concerns. The is used for each design decision both in a text and graphical representation. Table 4 describes the colors and notations that are used in the QOC graphs.

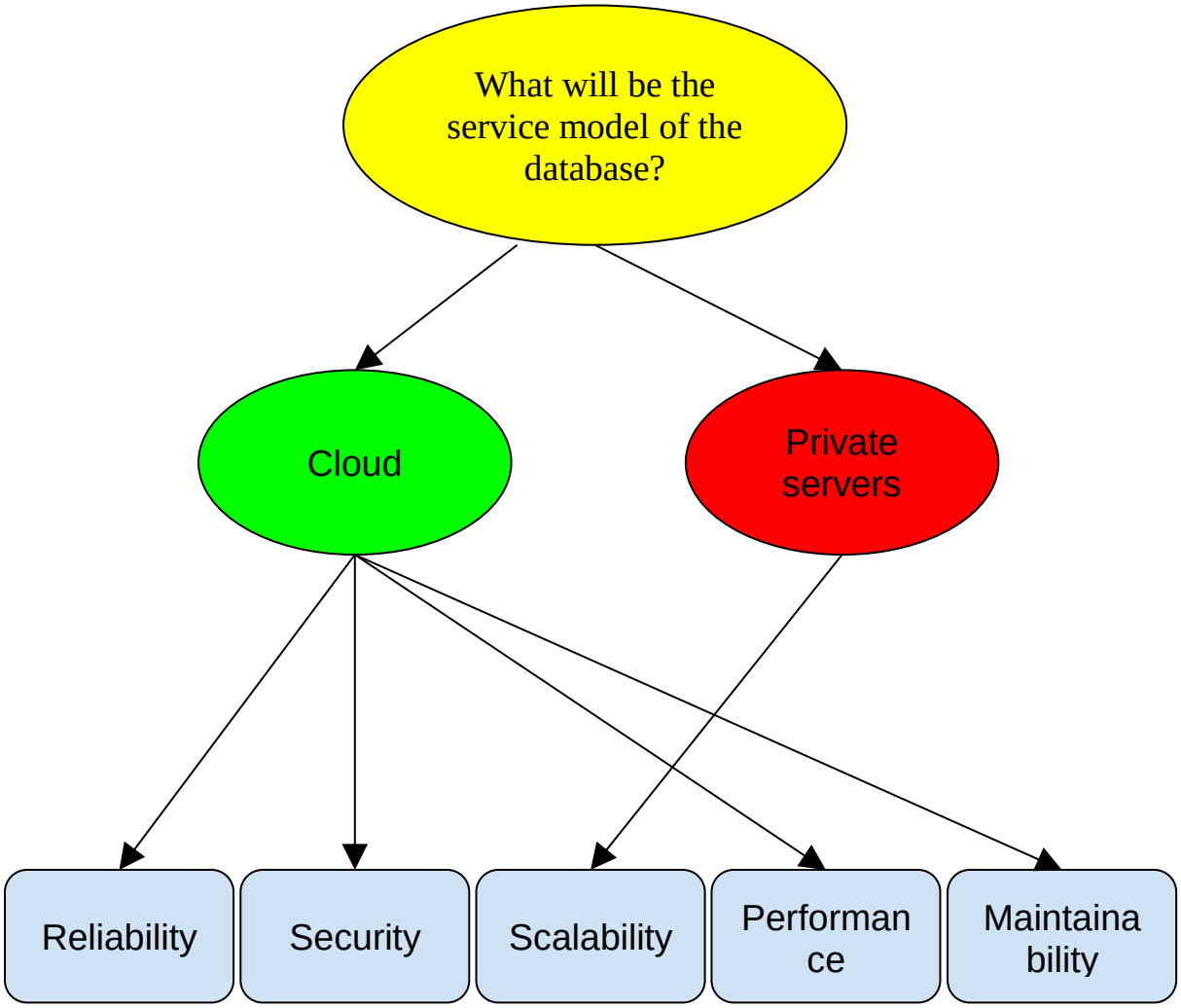
Color/Notation	**Description**
Red	Rejected option
Green	Decided option
Blue	Quality requirement
Yellow	Question
Straight arrow	Option satisfies the quality requirement
No arrow	Option does not satisfy the quality requirement

Table 4. Description of the colors and notations used in the QOC graphs

Design decision 1

Concern (identifier: description)		Con#1: What will be the service model of the database? The service model concerns the servers on which the data of the system is stored and who owns this servers. Potential models can be cloud and private servers.
Ranking criteria (Identifier: Name)		The NFRs affiliated with this concern are: Cr#1: Reliability Cr#2: Security Cr#3: Scalability Cr#4: Performance Cr#5: Maintainability
	Identifier: Name	Con#1-Opt#1: Cloud
	Description	Masaccio system will use a cloud database. This means that the hardware is not owned by the system, but that the system is provided a service in the form of hardware usage for servers and the database.
	Status	Accepted
	Relationship(s)	Forbids Con#1-Opt#2

Options	**Evaluation**	Cr#1: the option does satisfy the reliability criteria because cloud services are very fault tolerant and SLAs are created that promise a certain level of reliability. Cr#2: the option satisfies the security criteria because companies specialising in cloud services also specialise in how to secure the servers and the system. Cr#3: the option does not satisfy the scalability requirement. When more bandwidth or memory is needed the cloud provider can provide this. However, this is against a steep price. This makes it not very scalable in the long run. Cr#4: The performance criterion is also satisfied because the cloud service providers specialize in making the servers run as fast as possible. These companies have many more experiences employees for this than our system. Cr#5: The maintainability factor is also met because it takes our team little work to maintain the servers, since the cloud company will do this.
	Identifier: Name	Con#1-Opt#2: private server database
	Description	This option means that the development team needs to invest in the servers, install and maintain them.
	Status	Rejected
	Relationship(s)	Forbids Con#1-Opt#1
	Evaluation	Cr#1: the option does not satisfy the reliability criteria. The servers need to be managed by our team and none of us have the expertise to do this. Therefore, if we do this it might not be reliable and will surely not be as reliable as when the cloud provider maintains the servers. Cr#2: we would have to manage all the hardware and software security ourselves. Since this is a complex and specific task for servers, we would not be experts in it and thus the security criteria is not satisfied. Cr#3: the option does satisfy the scalability criteria. In the beginning an investment is needed, but after that servers can be added to scale the database. Cr#4: the performance criteria is not satisfied because we would not be able to deliver a performance as optimised as an IaaS or cloud provider can. Cr#5: the criteria maintainability is not satisfied. It would take us a lot of time to manage the data center.
	Rationale of decision	Only Cr#3 is satisfied, so therefore Con#1-Opt#2 is rejected.

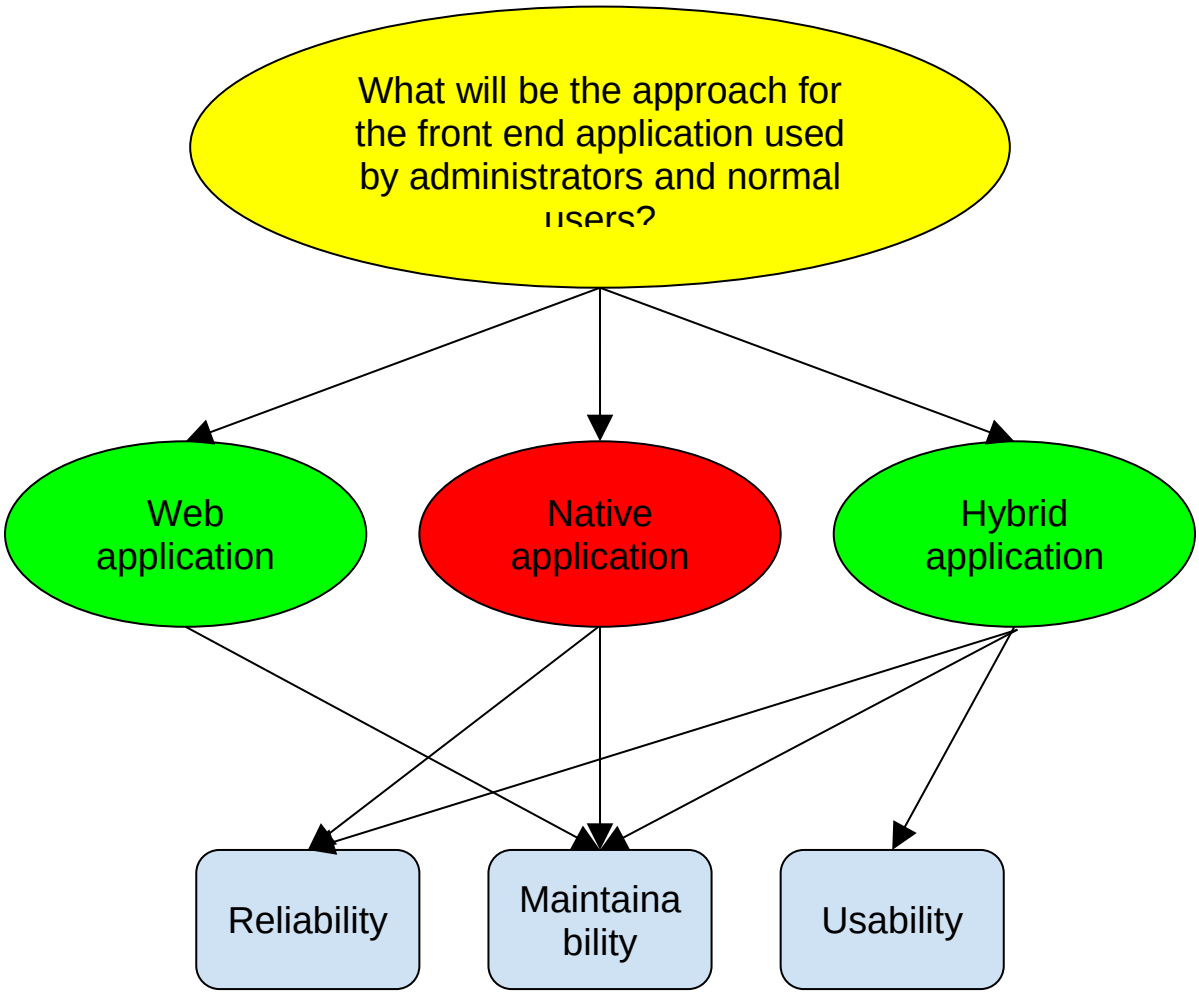


Design decision 2

Concern (identifier: description)		Con#2: What will be the approach for the front end application used by administrators and normal users? The administrators and normal users need to consume the services provided by the system. In order to enable this a front end of the system is needed where the administrators and users can interact with the system.
Ranking criteria (Identifier: Name)		Cr#1: Reliability Cr#2: Maintainability Cr#3: Usability
	Identifier: Name	Con#2-Opt#1: Web application
	Description	A web application as front end would mean that the users will use a browser (mobile or desktop) to access the

Options		services.
	Status	Accepted
	Relationship(s)	Is related to: Con#2-Opt#2 and Con#2-Opt#3. In principle all three solutions can coexist. So they are related to each other, but not the same. The preference goes to building only one of the solutions due to time.
	Evaluation	Cr#1: the option does not satisfy the reliability requirement because people won't have any data when there is no internet. Cr#2: the option does satisfy the maintainability requirement because only one platform needs to be maintained. Cr#3: the option does not satisfy the usability criteria because the user needs to go to the website every time instead of having an app on the phone.
	Rationale of decision	Satisfies only Cr#2, but is necessary to display the administrator dashboard
	Identifier: Name	Con#2-Opt#2: Native app
	Description	A native app is a downloadable app for iOS and or Android
	Status	Rejected.
	Relationship(s)	Is related to: Con#2-Opt#1 and Con#2-Opt#3.
	Evaluation	Cr#1: the option does satisfy the reliability criteria because even when there is no internet (temporarily) an user can still use some of the services. Cr#2: the option does not satisfy the maintainability criteria because the development team needs to develop both for the iOS, the Android and the desktop platform and make sure all new services work on the platforms. Cr#3: the option does satisfy the usability criteria because an app can be downloaded and the user does not have to go to a browser each time.
	Rationale of decision	Does not satisfy all criteria.
	Identifier: Name	Con#2-Opt#3: Hybrid app
	Description	Creating Hybrid Mobile application that has access to functions such as camera and GPS. It is based on a cross-platform application that provides a new dimension and helps to expand the business.

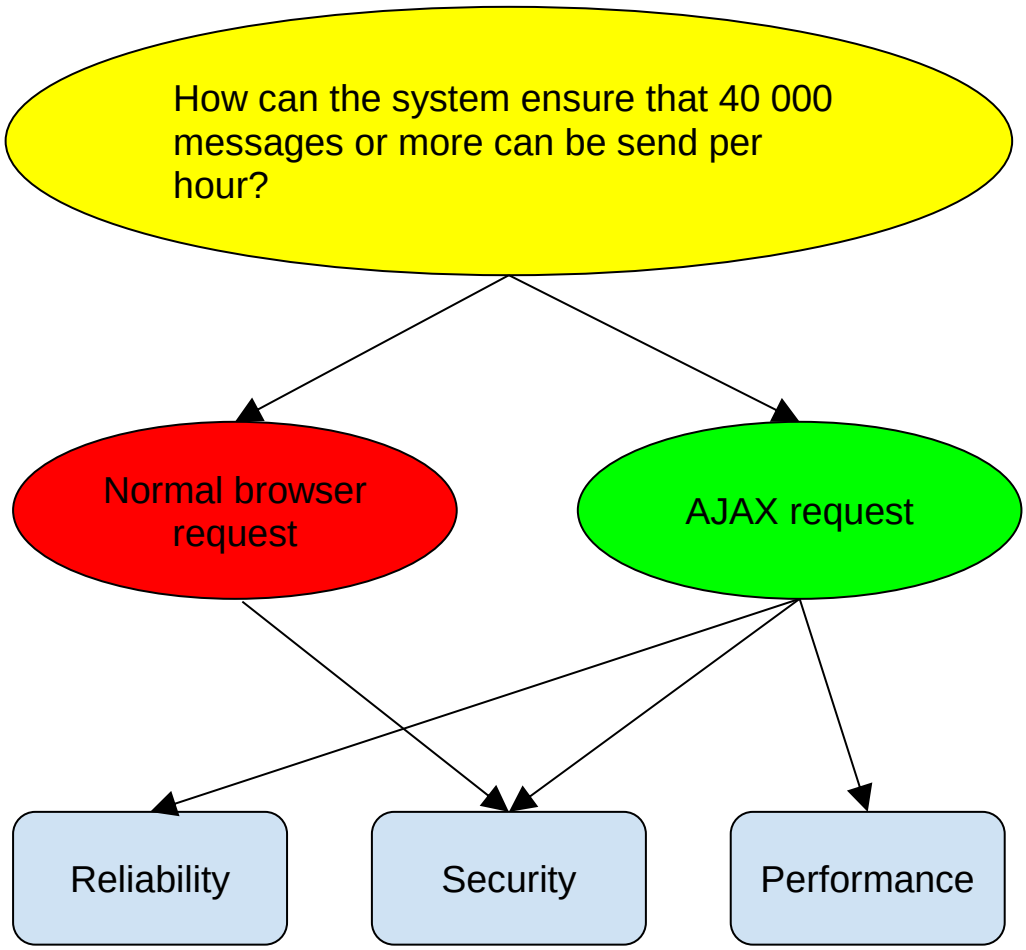
	Status	Accepted
	Relationship(s)	Is related to:
	Evaluation	Cr#1:Hybrid mobile app uses the API of the mobile phone with the purpose of securing and key information.It lets to use app when connectivity is very poor and allows to user to store some information. Cr#2: the option is satisfy the maintainability criteria, because it is possible to build it once and submit it to all of the platforms (iPhone, Android, Windows Phone) Cr#3: the option satisfy the usability criteria because the user does not need to go to the website each time, he can avoid it just having an app on phone.
	Rationale of decision	Satisfy all criteria



Design decision 3

Concern (identifier: description)		Con#3: How can the system ensure that 40 000 messages or more can be send per hour?
Ranking criteria (Identifier: Name)		Cr#1: Reliability Cr#2: Security Cr#3: Performance
Options	**Identifier: Name**	Con#3-Opt#1: Regular browser request (with reload)
	Description	The message is sent by a regular browser request that reloads the page.
	Status	Rejected
	Relationship(s)	Forbids Con#3-Opt#2
	Evaluation	Cr#1: The option does not satisfy the reliability criterium because the message might be lost and the browser would not have knowledge of this, which it can use to resend the message. Cr#2: The option does satisfy the security criterium because before adding the message to the database the application first checks if the user has the right permission to do this. Cr#3: The option does not satisfy the performance requirement because it needs to reload the whole page. This is quite slow and unnecessary since it only needs to send and add one message.
	Rationale of decision	Rejected because it only satisfies the second requirement.
	Identifier: Name	Con#3-Opt#2: AJAX post request
	Description	The message is sent to and received from the database by AJAX requests.
	Status	Accepted
	Relationship(s)	Forbids Con#3-Opt#1
	Evaluation	Cr#1: The option does satisfy the reliability requirement because if a message is lost, then the browser will not receive a 200 response and thus can decide, depending on the error, to send it again or at least warn the user that the message is not stored in the database. Cr#2: The option satisfies the security requirement because before the message is sent to the database the system checks whether the sender has the right permissions to send the message.

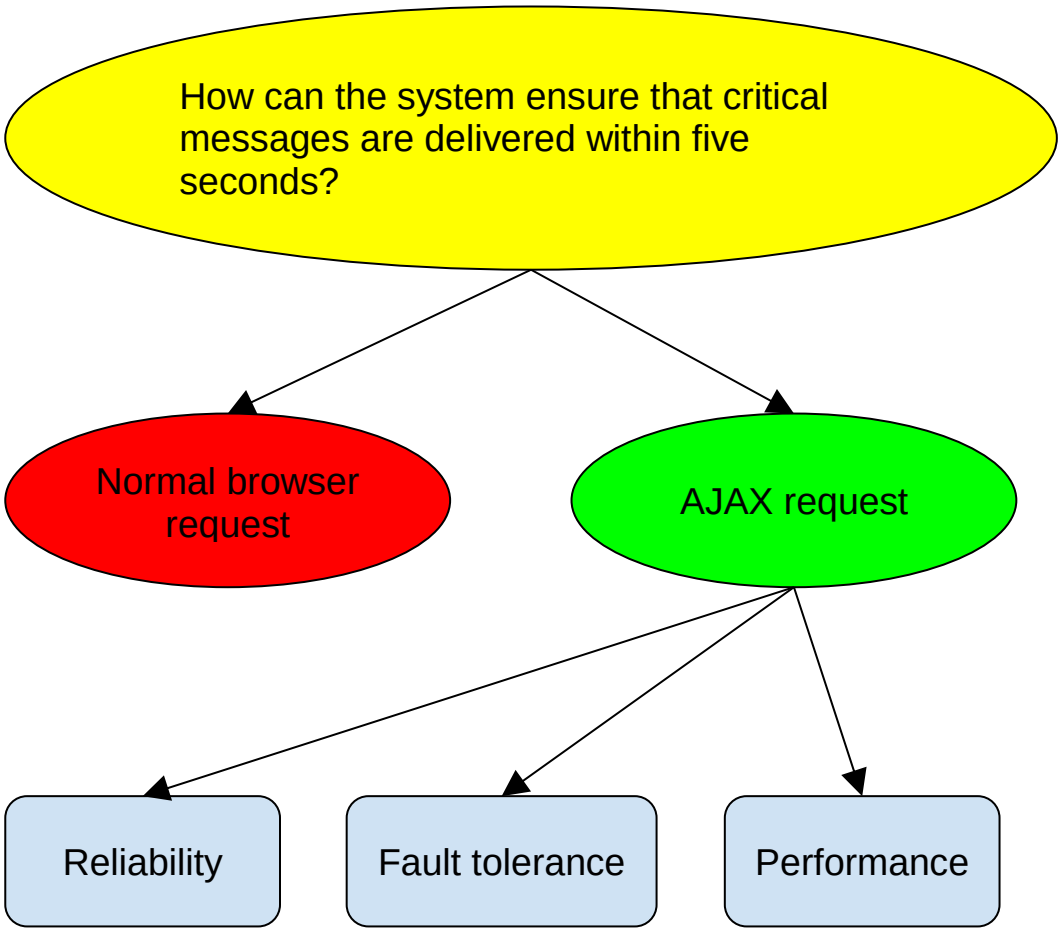
		Cr#3: The option satisfies the performance requirement because only the message is send and received and not the whole page needs to be reloaded. This enhances the performance.
	Rationale of decision	Satisfies all requirements.



Design decision 4

Concern (identifier: description)		Con#4: How can the system ensure that critical messages are delivered within five seconds?
Ranking criteria (Identifier: Name)		Cr#1: Reliability Cr#2: Fault tolerance Cr#3: Performance
	Identifier: Name	Con#4-Opt#1: Regular browser request (with reload)

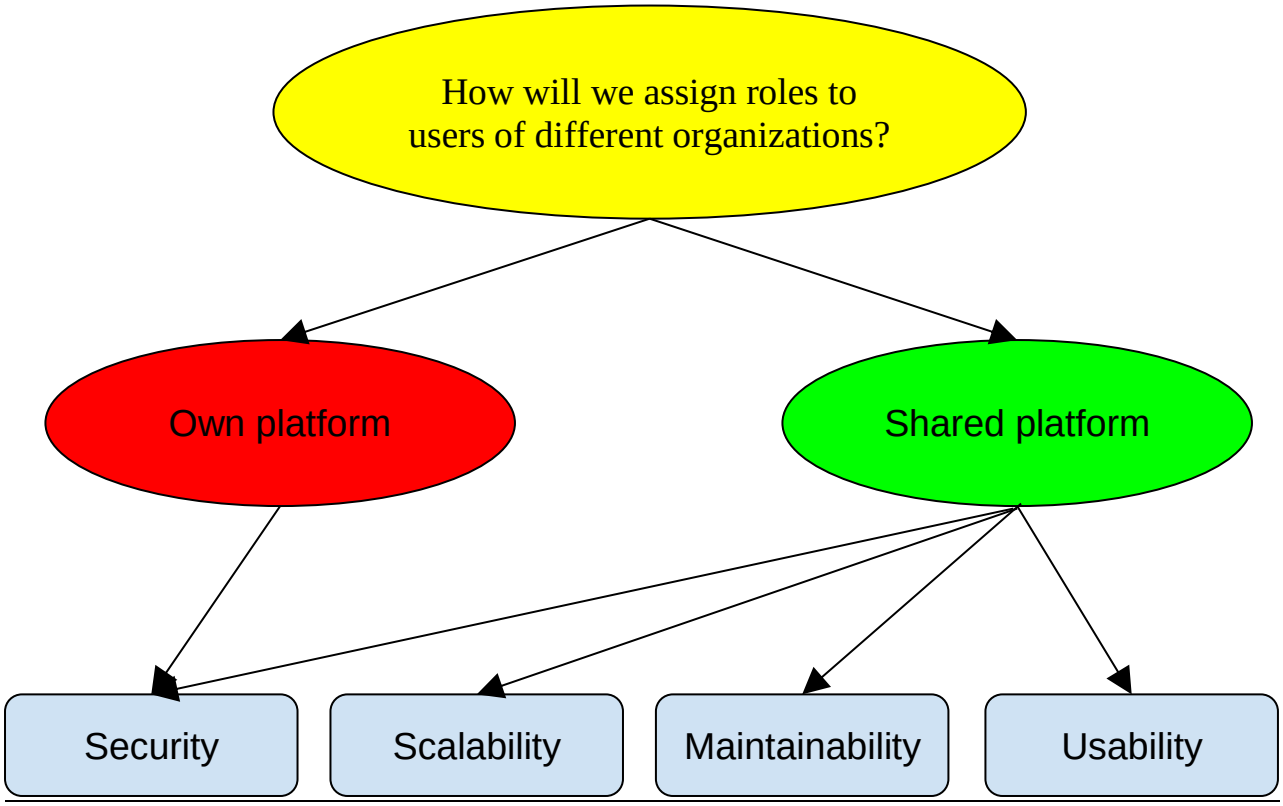
Options	**Description**	The message is sent by a browser request that reloads the page.
	Status	Rejected.
	Relationship(s)	Forbids Con#4-Opt#2.
	Evaluation	Cr#1: The option does not satisfy the reliability criterium because the message can be lost and no feedback will be given to the browser. Cr#2: The option does not satisfy the fault tolerance criterium for the same reason as with Cr#1. Cr#3: The option does not satisfy the performance criterium because it needs to reload the whole page instead of sending only the necessary part. This makes it slow.
	Rationale of decision	Does not satisfy any of the criteria.
	Identifier: Name	Con#4-Opt#2: AJAX request
	Description	The message is sent via an AJAX post request.
	Status	Accepted
	Relationship(s)	Forbids Con#4-Opt#1.
	Evaluation	Cr#1: The option satisfies the reliability criterium because the browser receives feedback when a message has not been delivered and can resend the message. Cr#2: The option satisfies the fault tolerance criterium because when something goes wrong it receives feedback and can resend the message. Cr#3: The option satisfies the performance criterium because AJAX only sends the message. This is fast. This was also tested, this is further elaborated on in the “From architecture to code” part, and the tests confirmed that the messages were sent within 5 seconds.
	Rationale of decision	Satisfies all requirements



Design decision 5

Concern (identifier: description)		Con#5: How will we assign roles to users of different organizations?
Ranking criteria (Identifier: Name)		Cr#1: Security Cr#2: Scalability Cr#3: Maintainability Cr#4: Usability
	Identifier: Name	Con#5-Opt#1: every organization gets its own platform with users
	Description	This means that every organization that uses the Masaccio system will have its own platform with its own users. If two organizations might have the same users, these users need to have different accounts for these organizations because they each have their own platform.
	Status	Rejected
	Relationship(s)	Forbids: Con#5-Opt#2

Options	**Evaluation**	Cr#1: the option satisfies the security criteria. The ACL of the platform has roles with certain rights assigned to users, so the user can only do or see what the user is authorized for. Cr#2: the option does not satisfy the scalability criteria. Because every organization needs its own platform, it will be difficult to manage all the platforms when there are many organizations using the system. Cr#3: the option does not satisfy the maintainability criteria. As the system grows and more organizations have their own platform it will be difficult for the team to maintain it because of the workload. Cr#4: the option does not satisfy the usability criteria because for every new organization the user needs to make an account again.
	Rationale of decision	It only satisfies criteria 2.
	Identifier: Name	Con#5-Opt#2: All organizations will share one platform
	Description	This means that based on roles within an organization a user or administrator is authorized to view and perform certain information or action. This means that users can easily use the system for multiple organizations. Also, a user can be an administrator at certain organization and a normal user at other organizations. This is because all the actions are performed on one platform and all information is stored in one database, but only the information is shown for which a user has the authorization.
	Status	Accepted
	Relationship(s)	Forbids: Con#5-Opt#1
	Evaluation	Cr#1: the option satisfies the security criteria because this ACL allows users to have different roles which determine their authorization per organization. Cr#2: the option satisfies the scalability criteria because there is only one platform that needs to be managed and organizations and users can all be added to that platform. Cr#3: the option satisfies the maintainability criteria because there is only one platform to maintain. This keeps the work load on the team bearable. Cr#4: the option satisfies the usability criteria because a user only needs to register once and has only one app. Also, a user can have different roles at different organizations, all using the same app.
	Rationale of decision	Satisfies all requirements.


How will we assign roles to
users of different organizations?
Own platform
Shared platform
Security
Scalability
Maintainability
Usability

CAPS Architecture View

In this chapter we will represent a scenario of Masaccio system. A simple scenario describes the monitoring of the number of people inside a room of the building, for example a museum, the checking entering and leaving of an employee from the building and checking the rental process for rent a room or a laboratory of the building.

CAPS SAML

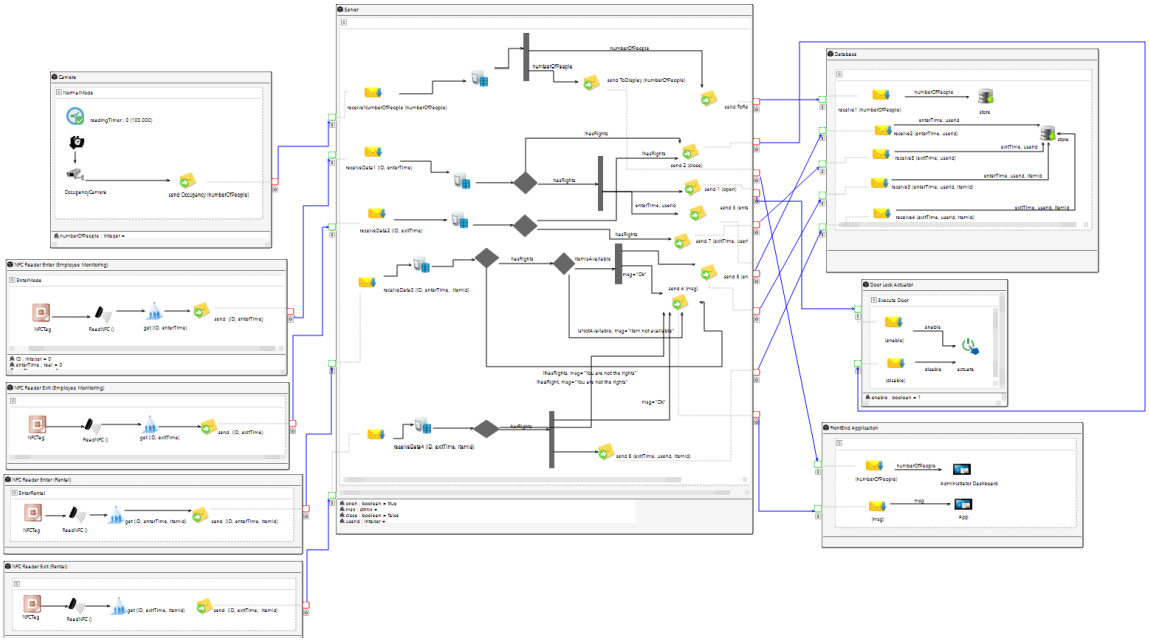


Figure 13: The software architecture of a scenario in Masaccio system

Figure 13 shows the SAML model of the Masaccio Architecture.

From a structural point of view, the SAML model is composed of seven main components: Camera component, NFC Reader component for Employee Monitoring, NFC Reader component for Rental service, Door Lock Actuator component, Server component, Database component and Front-end Application component.

- Camera component: is responsible of monitoring the number of people inside a room. It includes only the Normal mode. In this mode, the camera Sensor reads the number of people inside the room every 100 seconds. A timer is set in this mode to schedule the reading from the Camera sensor. A message carrying the number of people (numberOfPeople) is sent from the output message port of the Camera component to the input port of the Server.
- NFC Reader Enter component for Employee Monitoring: is responsible of tracking each employee who wants to enter in the building where he/she works. It includes only the Enter mode.
  In this mode, the NFC Reader reads the scanned NFC Tag ID and the enter time. The ID is the tag unique identifier. A message carrying the ID and the enter time (ID, enterTime) is sent from the output message port of the NFC Reader component to the input port of the Server.

- NFC Reader Exit component for Employee Monitoring: is responsible of tracking each employee who wants to exit from the building where he/she works. It includes only the Exit mode. In this mode, the NFC Reader reads the scanned NFC Tag ID and the exit time. A message carrying the ID and the exit time (ID, exitTime) is sent from the output message port of the NFC Reader component to the input port of the Server.

- NFC Reader Enter component for Rental Service: is responsible of tracking the rental of a room or a laboratory and the entry time. It includes only the Enter mode. In this mode, the NFC Reader reads the scanned NFC Tag ID, the enter time and the item ID. The ID is the tag unique identifier and the item ID is the item unique identifier that the person wants to rent. A message carrying the ID, the enter time and the item ID (ID, enterTime, itemID) is sent from the output message port of the NFC Reader component to the input port of the Server.

- NFC Reader Exit component for Rental Service: is responsible of tracking the rental of a room or a laboratory and the exit time. It includes only the Exit mode. In this mode, the NFC Reader reads the scanned NFC Tag ID, the exit time and the item ID. A message carrying the ID, the exit time and the item ID (ID, exitTime, itemID) is sent from the output message port of the NFC Reader component to the input port of the Server.

- Server component: is responsible of processing the different data received through its ports from other components. The data received from the Camera component are processed, sent to the real-time database (send ToRealTimeDatabase(numberOfPeople)) for store it and sent to the Front-end Application component (send ToDisplay(nummberOfPeople)) to show it on the administrator dashboard.
  Concerning the data received from both the NFC Reader component, the data are processed based on rules that are already set for entering/coming out the room and rental a room or a laboratory. So, when the data are processed, from the ID of the respective NFC tag, the user ID of the user who has the tag is derived in order to check the permissions. Concerning the data received from the NFC Reader from Employee Monitoring, the employee must have the rules to enter/ exit from the building. If he/she has the rules, the employee is allowed to enter/exit from the building and three messages are sent through the output port of the Server component to the input message port of the Database component and the Front-end Application component.
  Concerning the data received from the NFC Reader from Rental Service, first, the user must have the rules to rent the item, second the item must be available. Thus, if all the rules have been satisfied, the user is allowed to rent the room/laboratory, and messages are sent through the output port of the Server component to the input message port of the Database component and the Front-end Application component.

- Door Lock Actuator: is responsible for opening/closing the door when an employee is entering/coming out from the building. It has one mode called Execute Door. If it receives from its in port a message containing the true value coming from the output port of the Server component, it enables the actuator and thus it opens the door. Otherwise, if it receives from its in port a message containing the false value coming from the output port of the Server component, it enables the actuator and thus it closes the door.
- Display component: it represents our Front-end application. It has only one mode and it responsible for displaying the number of people in the room on the administrator dashboard web application and the messages about the rental process on the user app.

- Database component: is liable of stored our data received through its ports from the output ports of the Server component. In this scenario, the number of people in the room is stored on a Cassandra database and the data received from the NFC Reader on a Relational database.

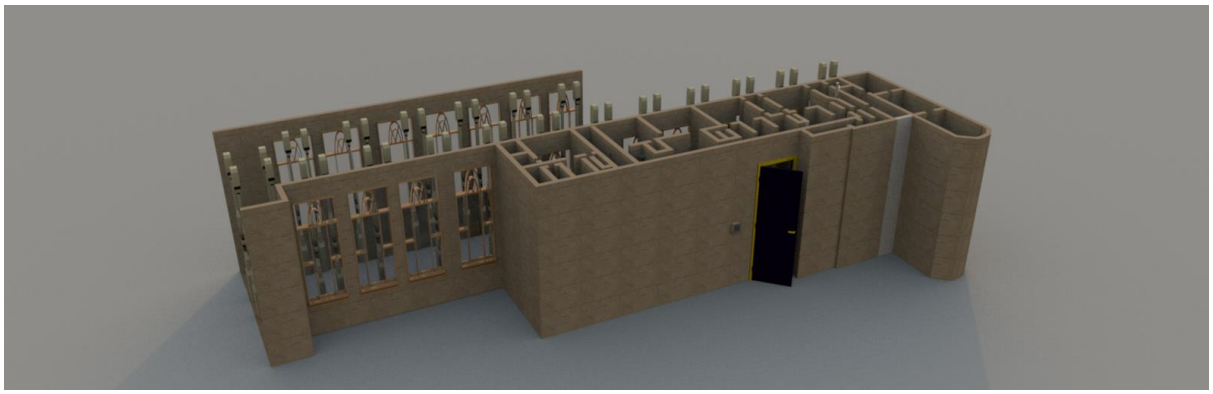


Figure 14: Space modelling of the first floor of our scenario

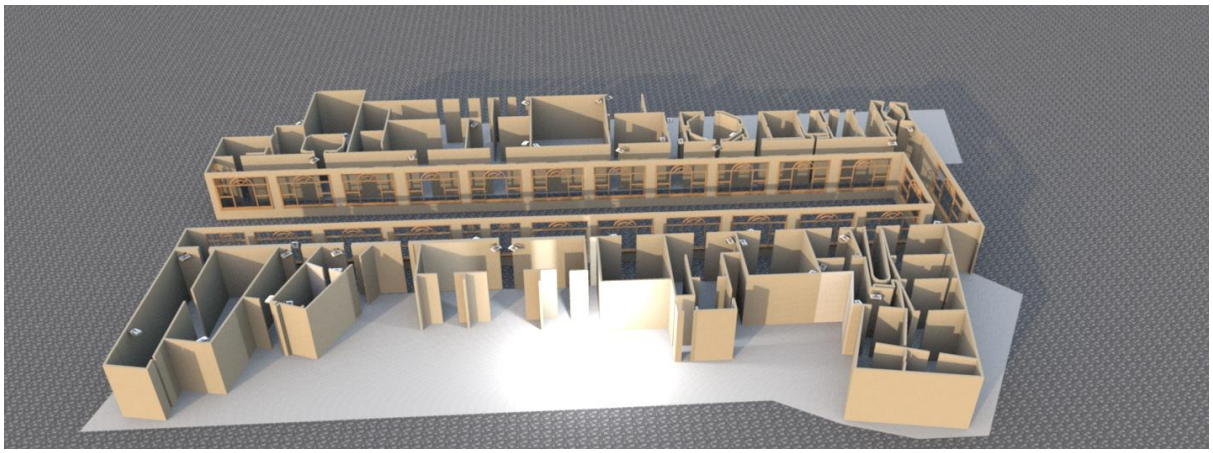


Figure 15: Space modelling of the second floor of our scenario

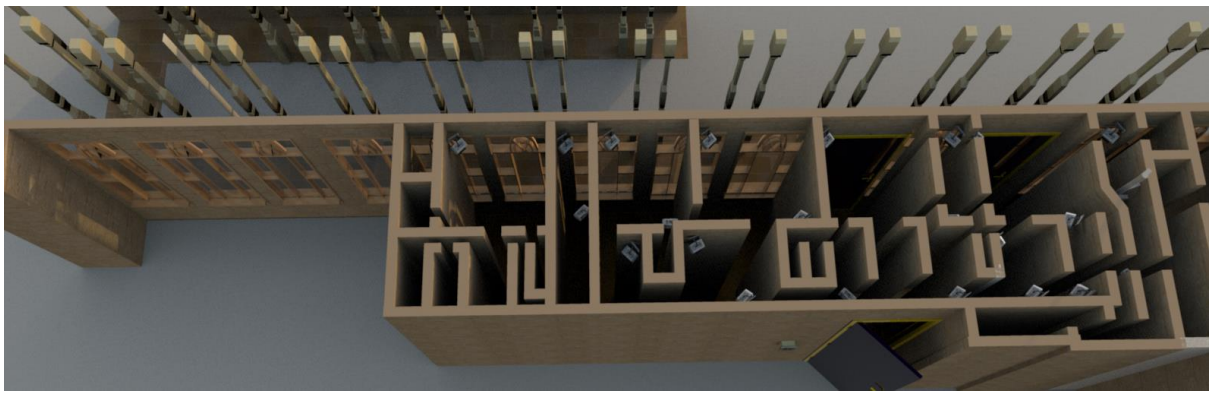


Figure 16: Use of cameras in our scenario

Figure 17: Use of NFC Reader in our scenario

Concerning the CAPS ENVML, we decided to model a museum as we can see in figure 14 and 15 using the Sweet Home 3D which allows to use the plan picture of the city, office, museum and so on. Using the Masaccio system, it can get many advantages such as know exactly number of people in each room through the use of many cameras, as we can see in figure 16, and check entering and leaving of an employee from the building through the use of a NFC Reader, as we can see in figure 17. In this kind of model we can put the cameras in such order and location that we can get the best results from them.

CAPS HWML

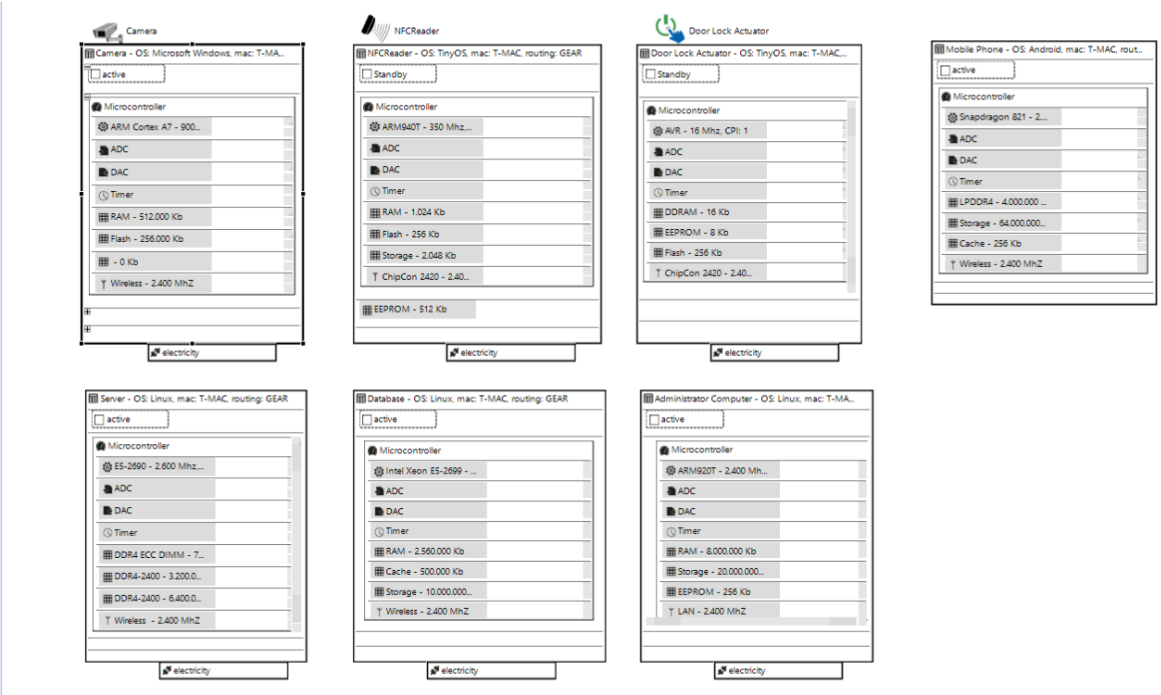


Figure 18: hardware modelling of a scenario in Masaccio system

Figure 18 shows the HWML model developed for our scenario and seven hardware configurations have been defined:

- Camera is responsible for sensing and counting people inside the room. This equipment is powered by a classical electrical plug connected to the electrical system of the building. The micro-controller is equipped with a processor ARM Cortex A7, Ram and Cache memory. The Camera is always active and it uses Wireless network for communication.

- NFC Reader is responsible for sensing the signal coming from a smart card or a phone. This equipment is powered by a classical electrical plug connected to the electrical system of the building. The micro-controller is equipped with a processor ARM940T and different type of memory. The NFC Reader is always on standby power mode which will be changed to activate mode by an interrupt.

- Door Lock Actuator has a single actuator device which will send the active signal to the door. The micro-controller equipped with a processor AVR and different type of memory.

- Server processes different type of data coming from the different devices. The micro-controller is equipped with a processor E5-26990 and different types of memory. The Server is always active and it uses Wireless network for communication.

- Database stores all the data coming from the server. The micro-controller is equipped with an Intel Xeon E5-2699 processor and different types of memory. The Database is always active and it uses Wireless network for communication.
- Administrator Computer is the physical computer where the administrators can see the number of people in the room. The micro-controller is equipped with a processor ARM920T and different types of memory. The Computer is always active and it uses LAN for communication.

- Mobile Phone is the phone where the user who wants to rent a room received messages. The micro-controller is equipped with a Snapdragon 821 and different type of memory. The Mobile Phone is always active and it uses Wireless network or LET for communication.

From architecture to code

Which services have been implemented

The main function of the Masaccio system is to monitor crowds in an indoor environment. Based on this information decisions can be made regarding safety and evacuation. Therefore, this is the central pilar around which the rest of the system should be build. Also, the system should be able to process vast amounts of incoming data in the shape of IoT data events. These IoT data events are produced by the camera images and NFC readers.

In the architectural part of this project it was decided that a Kafka based architecture would be suitable to build this part of the system. The service that provides the whole cycle from data produced by the sensors, sending the data to the database, storing it and then retrieving it to show it on the front end dashboard, is the Smart Information service. Therefore, implementing this service would provide us with the backbone of the system, on which the rest of the system can be constructed. It provides the IoT data production, the handling of this data with Kafka and Spark Streaming, a database with the first database table and a small front end application to display this information on a dashboard. Therefore, implementing this service in the second sprint would increase the business value of the project most.

The implementation also helped with visualizing risks. It provides the team to see if the general architecture works and if Kafka and MVC can be combined properly. Also, the team can now test how many events can be processed by the system and how the system performs when the number of events increase. Furthermore, it provides the basic code on which the team can build another critical service, being the first responder communication.

This service has been implemented second. This is because this service is essential to test design decision three and four, and therefore, also essential to the system. By implementing the first responder communication service the system can be tested on the handling of critical messages and if it can handle 40 000 messages per hour. Furthermore, because not all users are allowed to send and read messages from the first responder communication board, an access control layer is implemented. This service is also the basis for the third service that will be implemented, being the Access Control Service. This service allows people access to rooms or to open things.

The third service that has been implemented is the Access Control Service. This service uses the ACL that has been implemented. The request access events are sent by Kafka and stored in the database. These events are then retrieved by the Spring application that controls if the user that has created the request had the right permission to have access to the item that the user has sent the request for. Implementing this service required an important extension of the database and usage of the ACL. Therefore, it increased the business value of the product significantly and was thus chosen to implement as the third and final service of the prototype.

How the services has been implemented

First service

The source code of the implemented services is available at the following Github repository: https://github.com/Lucgril/Masaccio .
Please check the README.md file to see all the technologies that we have used and our development organization.

A demonstration of the first service that we realized is shown in the First Service video that is available on the Dropbox repository. Firstly we have run our server-side application built with Spring. We have developed only the 'Dashboard Rest' component where there all the REST calls to support Dashboard administrator services, as we can see at the end of the video.
From the second project we have run we can see the generation of the messages every ten seconds in a semi-random way. Then the messages are subscribed with Spark Streaming and stored in a Cassandra database (third project that is run). Then all the data are shown in the administrator dashboard, a single HTML page. The page receives data with AJAX requests every ten seconds to the server-side application.

Second service

Also for the second service, the source code of the implemented one is available at the following Github repository: https://github.com/Lucgril/Masaccio .
Please check the README.md file to see all the technologies that we have used and our development organization.

A demonstration of the second service that we realized is shown in the Second Service video that is available on the Dropbox repository.
For the realization of the second service we have also used Spring Security. It is used for authentication and authorization purposes in our application. We added role based access to our pages, also known as Authorization, so that a user, after logging in, can only browse the pages that he or she is allowed to browse. We have secured our Rest back-end with it. We have used a relational database (MySQL) to store all the information about the registered users and their permissions.
The view part has been moved on the server project as well. In additional to the login page there are two jsp pages connected by a menu: IoTRoomData.jsp and firstResponderCommunication.jps. The first menu item is the administrator dashboard that shows the number of people in each room and it is the one we realized for the first service. The second menu item is the platform for the first responder communication: there the people allowed to use the platform, can send a new message and see the list of messages already sent. When a message is written down, it is sent to the server by a Rest Api via an AJAX post request and stored in the Cassandra database. The last menu item allows the user to log out.

In order to test whether the system can handle 40 000 messages per hour and deliver critical messages within 5 seconds, some tests have been performed. The tests have been performed with a tool called Siege. Siege is an http load testing and benchmarking utility. We have run the tests on one machine and the server was located on the local host of this machine. This means that the latency was very low. The tests would need to be run again when deploying the application on a real server, with a more realistic latency, in order to determine the real output of the tests.

During the test we simulated AJAX JSON post messages. The JSON object that was used has the following format: {"text": "Hello. This is a test message that we will use for load testing the application!", "topic": "firstResponderCommunication", "timestamp": "1517149900368"}. The number of concurrent users, delay (of the user) and the test time was changed in different tests. The variables that have been monitored are number of messages, number of messages per hour, longest transaction and the succes rate. In table 5 the results of these tests are displayed.

Number of concurrent users	Delay	Test time	Number of messages	Number of messages per hour	Longest transaction	Succes rate
10	1 sec	60 sec	1163	70k	0.04 sec	100%
25	1 sec	60 sec	2889	173k	0.13 sec	100%
50	1 sec	60 sec	5968	358k	0.13 sec	100%
80	5 sec	60 sec	1940	116k	0.14 sec	100%
50	3 sec	120 sec	3973	119k	0.08 sec	100%

Table 5. Load test results of the message service.

As can bee seen from the table all test results showed that more than 40 000 messages per hour could be send by the system. Furthermore, the longest transaction during those tests did not surpass 5 seconds, the threshold within a critical message should be delivered. This satisfies the previously stated system requirements. However, the tests should be run again when the application is deployed on a server because the results might be different due to different processing power and latency.

Third service

Also for the third service, the source code of the implemented one is available at the following Github repository: https://github.com/Lucgril/Masaccio .
Please check the README.md file to see all the technologies that we have used and our development organization.

A demonstration of the third service that we realized is shown in the Third Service video that is available on the Dropbox repository.
For the realization of the third video (Access Control) we have added another web page: accessRequest.jsp. Administrator are the only one allowed to see this page. This page allows the administrator to check if a user is enabled to access in a specific item at a specific time. The information are taken from the server by a Rest Api via an AJAX GET request.
The data mentioned above are generated by our Kafka application. For this purpose, Multithreading capability has been added to our Kafka application. The main class creates now two threads: the first one generates semi-random data for the first services and the second one generates random accessRequest data in JSON with the below format: {"item": "room1", "carduid": "abcdef12", "timestamp": "2018-01-13 03:26:18"}.
This data are then consumed by our Spark application and then stored on the Cassandra database.

We have updated our MySQL database adding tables for the items and user card.
We have insered three doors, three showcases and three gates. We have added four card as well, one for each user role.
The following list explains which users are allowed to access in a specific item.

- Admin: has access to all doors, showcases and gates;
- Employee: has access to all showcases and doors;
- Responder: has access to all doors and gates;
- User: has access to the first gate.

Summary

Context

The project was coming up from the problem of crowd people areas where people even can get harmed. Moreover, people lose the time instead can enjoy the other things while waiting so much time in the queue to see something (for instance in the museum) and it is the reason of empty visit rooms.

Objective (the main goal of the project)

The project set out to increase the awareness of the importance of protection people from the crowd areas in the closed venues (museums, malls, universities). The project has contributed to improving the monitoring system both inside and outside that contains track of the people movement in order to get an exact number, track the available places (rooms, cabins, parking spots); collection and store the data and use it to show to people (visitors, staff), to help people when they need to do reservation/enrollment or in case of emergency help them find the nearest and the most suitable exit.

Outputs

To achieve the purpose that we mentioned above, we decided to concentrate on the Masaccio system that has 10 main services. All of them have one common objective - the protection of people in density areas, while separately they have many responsibilities. For instance, access control service is responsible for protection the areas by checking the users authority and rights to allow or deny the access inside.

Identifying the risks is very important to the project and its success future. Surely, the Masaccio systems have many important risks like DB crash, server crash, data lost, loos of messages and to decrease the risks we have made some research in order to manage them.

In the list of assumption we mentioned the things such as hardware, plans, blueprints, adopters, etc. which we can use from the building (in our case closed area).

In the section “State of the art” we made some research in the global market to find the projects and systems which has the main goal with the Masaccio or at least the similar idea. We analyzed the systems such as “Accuware Video Tracker”, “RFID-system for access control and monitoring of movement of people”, collected their features and characteristics in order to know which benefits we can get or re-use from them. To review the types of studies that previous researchers have launched might be benefit in further developing a topic.

System Architecture is one of the most essential point in the system, so to describe the system in our mind we provide an example “Day at Uffizi” and the Architectural diagram of the system - Component diagram. System Architecture contains the deep description of each services used in the system separately, for what they are responsible, how they help to achieve the main goal of the project, and also the interaction among the services. The architectural diagram provides the short but very useful description to understand how the services will work, for example, how the data will be collected by using which component, then how it will sends to the DB through which components and ways the how the people can use the information, etc. Overall, the architectural description and diagram identify the software, hardware, connectors.

User stories are given immediately after the System Architecture. Actors that have interaction with the system are identified and to show this interaction we used an UML Use Case diagram and provide a little description of each actor. The main services in the Masaccio system have been made from the basic services which play role of the functional requirements.

Then in the next section was given an introduction to extra-functional requirements such as reliability, scalability, usability and the identification of their impact on our system and their importance.

Based on our description we identified list of stakeholders, concerns and the concern-stakeholder traceability. Firstly we paid attention to concerns that may have an impact on the stakeholders and the requirements. The list of the stakeholders almost the same with actors, that are given in the part of “User stories”.

Models give a point about the sequence diagrams that shows the interaction and connectivity among the services and all of them are parts of the system.

In the section “Design decisions” we mentioned the most essential architecture design decisions by using the QOC template. The ones which are given in the table have huge affect on our system and architecture. Both tabular and graphical templates show the concerns, criteria and the satisfaction of the criteria.

Appendix

Short interview

As we all know Museums are shared spaces where many people come to see rare and interesting thing. And surely there are many days where the number of people extremely increasing and also some days when just a few visitors. The density of people depends on the days of year such as weekends, holiday, vacations and other.

1. Do you want to know the exactly number of people in each day, who visit the museum? Is it important from your point of view?

2. Do you use cameras in museum? What are their responsibilities?

3. Do you save the data getting from the cameras? If Yes, so why?

4. Would you like to track all the movements of people and realize constant control automatically? Why?

5. Is it matter for you when someone get inside the room without permission? How do you avoid that kind of situations?

6. How do you think giving an authentication to people in museum (staff, visitors) will helpful to you in order to prevent access to the places which people shouldn't see? Would you like to have an access control system in the museum?

7. Do the people know everything about the things in museum (such as demonstrated things, the exit, restroom)? If yes, so which method do you use? Brochures, audio disk, website, app?

8. Would you like to use the User Event Information service instead of what you use now to give information about museum, its events and make it possible to people enroll there?

9. Is it happen when many people visit one exactly room but the other rooms are empty? How can you avoid this?

10. If you had a possibility to send information to visitors about density in one exactly room would you like to use it? Do you believe that it will help to avoid crowd of people?

11. Are you sure about the safety of each one who is inside the museum? Or someone needs help? How can you know this?

12. Do you think using an app and sending prepared text such as "I am safe" when going out from the museum will make the assurance?

13. What do you do in case of emergency? How many times it takes to call the police, ambulance, fireman?

14. Would it be better if they museum has an interaction with museum and you don't need to call them separately they just get notification in case of emergency?

Bibliography

References:

1. Accuware, 2017. Accuware Video Tracker. Retrieved on: November 22 2017, from: https://www.accuware.com/products/indoor-tracking-video/
2. Use of WiFi camera, 2017. Indoor device tracking using Bluetooth beacons, WiFi and/or Camera. Retrieved on: November 22 2017, from: https://www.youtube.com/watch?v=fhdZMZx3Dxc
3. RFID -system, 2017. RFID-system for access control and monitoring of movement of people. Retrieved on :November 22 2017, from: http://www.itproject.ru/otraslevye_resheniya/conference_exhibition_events/rfid_moving_people
4. TRAF-SYS people counting system. Retrieved on :November 22 2017, from: https://www.trafsys.com/people-counting/
5. FootfallCam People counting System. Retrieved on :November 22 2017, from: http://www.footfallcam.com/
6. Samsung Smart Guardian Entrance Doors. Retrieved on :November 22 2017, from: http://www.guardian.ru/blog/professional-solutions/6478/
7. Best smart locks with authorization. Retrieved on :November 22 2017, from: https://www.youtube.com/watch?v=dX8QMnnohfY
8. L. Bass, *et al*, 2013. Architectural tactics and patterns, Software architecture in practise. New Jersey, Person Education, Inc. P. 229 - 272
9. Kotaru, M. *et al*, 2015. Spotfi: decimeter level localization using WiFi.