MuSa

The experience manager
designed by Silvia Del Piano, Stefano Foti and Gianmarco Zizzo

THE PROBLEM

We projected MuSa thinking about many commons issues that a visitor has when he visits a museum, such as:

- limited time for take a full visit
- meet on his path some pieces of arts that he does not like for various reasons
- the desire of having more details about an opera, but at the same time he does not like to have a guide

At the same time, we thought about the curators: sometimes, they could need to have some statistics about the most appreciated artworks and about the visitor's paths, in addition to the fact of offering some services that tempt the visitors to come back another time

EXISTING APPROACHES

We thought that a possible solution is to propose personalized tours. The topic of personalized tours has been already experienced in smart museums, but we hardly found something modern and fully developed

The first problem is that most tour modalities contain a fixed list of artworks, which is the same for everyone or for visitors from the same predefined user groups (e.g. groups of tourists, students, experts)

The second problem is the lack of connection between online tours and on-site/multimedia tours, which are usually separated into two tours without any connections

These two problems became the main challenge in building personalized tours

OUR IDEA - WHAT MUSA IS

MuSa has been thought as an experience manager for the «Museo dell'Arte Classica della Sapienza». The basic concept of MuSa is to put the usar at center of the experience, satisfying his curiosity and offering him a customized way to visit a museum

MuSa collects and analyzes information about visitors and their tours and proposes custom journeys, to offer a new way to enjoy the art

MuSa provides, through the application, information about the artworks in the customized tour, like the author, the historical period and other details

Collected data could be used by the curators for several reasons, like to improve the artwork's disposition in the museum
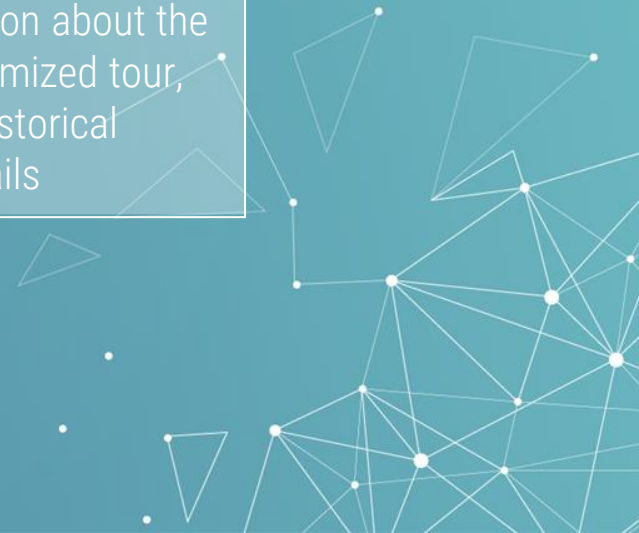
HOW MUSA CAME TO LIFE

Through an online questionnaire, we gave life to some personas

Some personas would have an experience with MuSa and moreover they allow to collect some data during their tour to help improving the service…

At the end of the tour, they can also leave a quality feedback

… other personas may not need the company of MuSa, but they want to help the service get better too, so they allow the data collecting

ARCHITECTURE
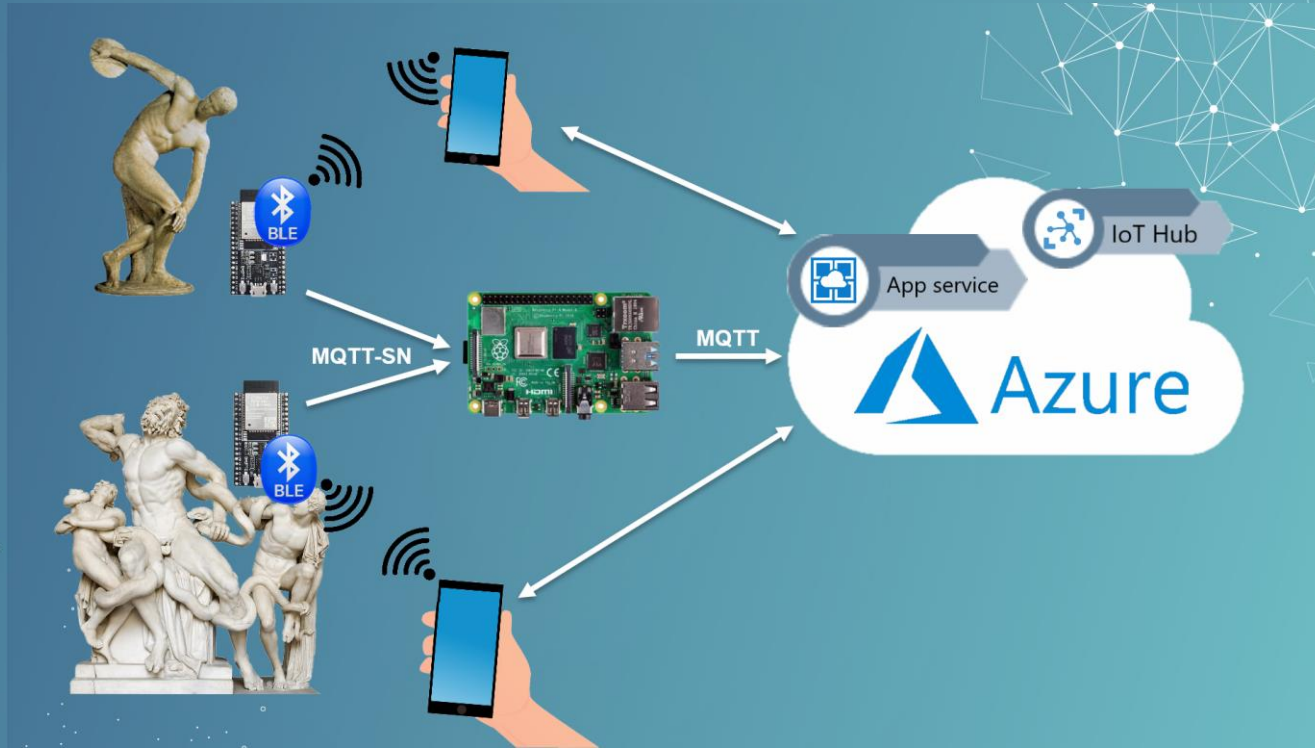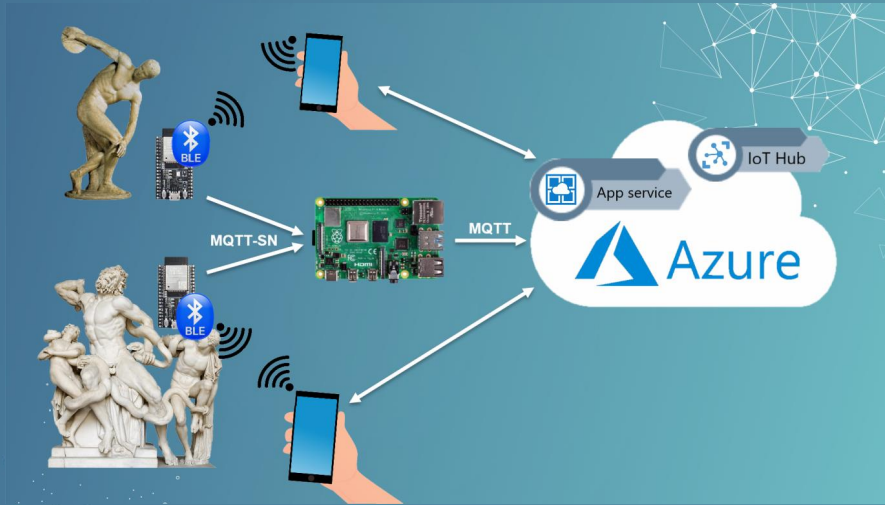
THE NETWORK DIAGRAM

THE NETWORK DIAGRAM

This is the network diagram of the whole system, including hardware and software.
The hardware part comprehend ESP32, Raspberry Pi, RGB Led and smartphones.
The software part comprehend MQTT, Bluetooth Low Energy, App Service and IoT Hub by Microsoft Azure, as well as the Android App and the backend

HARDWARE COMPONENTS

ESP32 board

Near each artwork or cluster of artworks, we have an ESP32 board that is connected to Wi-Fi and is capable to receive smartphone's beacons. In a variable time, these boards sends their reports to the gateway

Raspberry Pi

The Raspberry Pi is our gateway between the ESP32 boards and the Cloud Service. It receives the reports from all the boards in the room and every five seconds sends them to the Microsoft Azure

RGB Led

We use a generic RGB Led as actuator to light up the most appreciated artworks

SOFTWARE AND TECHNOLOGIES

Azure Cloud service

In Azure we can find the IoT Hub, the Event Hub, our application code, the backend, with its database and the (*not implemented*) machine learning algorithm. In our demo, the information are not stored in the database but in the DbContext in the backend, which takes care also to provide them to the application.

Bluetooth Low Energy (BLE)

We need to use BLE to understand the pieces of art the user is nearest to. So, what we really need to implement is proximity identification more than a complete indoor localization. The BLE technology met all our requirements: it's not expensive, compatible with mobile phones, has a wide enough range, can be used to do proximity identification. The only issue is the accuracy, but as we used techniques to mitigate this disadvantage like the consideration of RSSI and the application of a Kalman Filter.

SOFTWARE AND TECHNOLOGIES
MINOR MENTIONS

We involved also…

- **MQTT/MQTT-SN** for the messages exchange between the boards and the gateway
- **Android Studio** for the development of the Android App
- Several programming languages: **Java, C, C#, Python**
- … and a special mention to **Trello** that allowed us to organize our group working

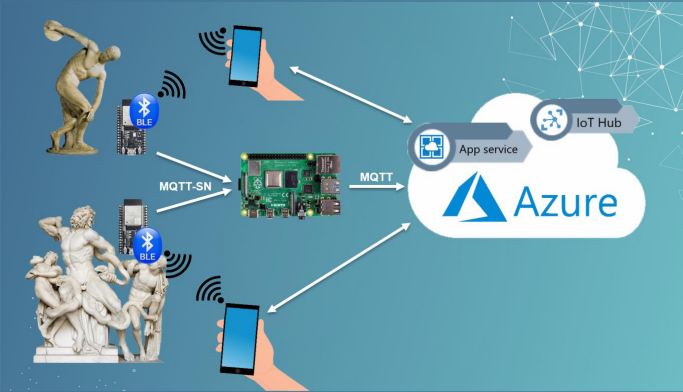# HOW THE WHOLE SYSTEM WORKS
TECHNICAL POINT OF VIEW
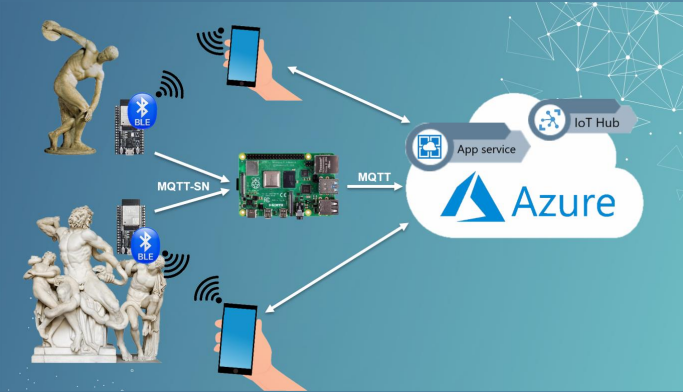
Let's have now a complete explanation of how the whole system works from a technical point of view!

» When a new user arrives at the museum and downloads the app, his smartphone starts sending beacons
When the user is next to a piece of art, the ESP32 board receives the beacon

» All the ESP32 boards send their report to the Raspberry Pi gateway, in a variable time, until to a maximum of five seconds, that depends on how many users there are in the museum

» Every five seconds, the Raspberry Pi gateway sends the message to Azure IoT Hub

» Through the Event Hub, the message reaches the backend, that stands in the cloud too

» The backend provides the responses to the received requests and sends them to the app, going backward in the IoT pyramid.
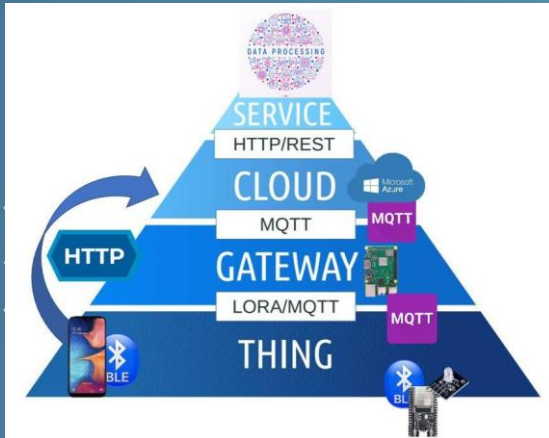
HOW THE WHOLE SYSTEM WORKS
USER'S POINT OF VIEW

Let's have now a complete explanation of how the whole system works from the user's point of view!

》 First of all, when a visitor comes to the museum, he has to decide if he wants to have an experience with MuSa or he wants only to help the data collecting: in both cases, he needs to download the MuSa Android App

》 After that a user downloads the App, he fills the profiling survey and decides officially the use of MuSa he wants to do

》 If a user wants only to help the data collecting, his work with the app is done until his visit reaches the end. The app itself will collect data about the visitor's path. At the end of the tour, the visitor will press the End button

》 If a user wants to have a customized experience with MuSa, he will receive a suggested tour. Then, along his tour, he will receive information about the artworks like the author, the year of creation, and some other details

》 At the end of the tour, the user will be asked to fill the feedback survey, helping to evaluate and to improve MuSa

EVALUATION AND PERFORMANCE

OVERVIEW

For the evaluation we used different methods for the user experience and the for the technologies used. In particular, the quality of the hardware and software components will be measured taking into account their peculiarities, but also the quality of the whole system.

In particular, we evaluated:
- the user experience
- the overall quality of the system
- the power consumption
- the complexity and the responsiveness
- the maximum number of users
- the backend and the frontend
- the price

EVALUATION

The user experience

Since that users are the core of MuSa, one of the main aspects to evaluate is the user experience. We thought about two evaluation methods: a Moment method and an Episode method.
For what concerns the Episode method, that is a method that gives a feedback about the whole experience, at the end of the visit the user will be asked to fill a short survey about his experience. This questionnaire can be implemented using also the AttrakDiff tool.
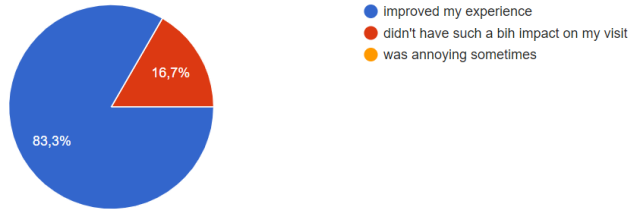
We didn't implement the Moment method evaluation for our demo, because it is suggested a commercial tool
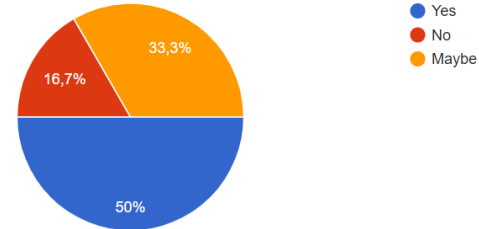
EVALUATION

The user experience

We presented to a group of people our demo and asked them to imagine the full product deployed in a museum. Considering this, we asked them to fill the survey the MuSa presents the user at the end of their customized tour. The response we got is quite good.

I think MuSa

- improved my experience
- didn't have such a bih impact on my visit
- was annoying sometimes

83,3%

16,7%

Would you recommend MuSa to a friend?

- Yes
- No
- Maybe

33,3%

16,7%

50%

The full results of our survey are available in our documentation.

EVALUATION

Overall evaluation of the system

Software quality is defined by a set of regulations and guidelines by ISO/IEC 9126-1. We used a criteria-based evaluation which gives a measurement of quality in several areas, including understandability, documentation, and portability. We did not use the criteria we did not need, so we produced a lighter customized version: MuSa Criteria. Again, we think that the results are very good and the complete analysis is available in our documentation.

Power consumption

Since that the number of boards could be quit high, we decided to evaluate the power consumption. Doing some measurements through an amperemeter, we discovered that each ESP32 board needs around 110mA at 5V to work, so it consumes 550 mW, which is a good value keeping in mind that we are using both Wi-Fi and BLE.

EVALUATION

Complexity and responsiveness

We need our system to be responsive to follow the user in real-time during his visit, but at the same time we also don't want to flood our network with messages, keeping the complexity low, also to save the board's power. Therefore, we did a brief analysis of the system and decided to use a Raspberry Pi board as a gateway, that allows as to save on a huge amount of messages.

We did some simulated calculations and the result is that without the gateway, Azure free plan will expire in five minutes, with the gateway the free plan allows as to be online for eleven hours! The explanation of this calculations is available in our documentation.

Notice that the choice of having a gateway brings benefits not only in messages saving, but also in the possibility of edge computing and data pre-processing.

EVALUATION

Maximum number of simultaneous users

We made some calculations and a stress test about the maximum number of simultaneous users with our current hardware and technologies.
Our ESP32 has 4MB of space,, and considering that we need some spaces for libraries, how much free space do we have?
We empirically found a good compromise for predefined sizes and we have space for around 94 iteration.

During our stress tests with beacon simulator applications, we caught 93 beacons out of 100.

Let's comment on it. If we have a big room with 20 artworks and 20 boards, if every board catches beacons of who is very far from the board, probably we will not be able to serve all the visitors, since 93 looks not to be such a big number.
What is our solution? Transmit the beacon with lower power. It's easy to set the transmission power of the and if we transmit the beacons with lower power, only the boards close to the visitor will catch his beacons, so we will be able to manage 93 visitors in a very small place around fewer artworks, but probably a very bigger number in the whole museum.

EVALUATION

Backend

When the backend was ready to be deployed on Azure, we made a last evaluation of the backend with CodeCity. The more the city is well structured, the higher the code quality is. These are our final results:

Frontend

The Android application works well, and it doesn't exibit delays or abnormal behaviour. We checked the most common metrics for software evaluation. These are the results:

UPDATED COST EVALUATION

Azure IoT Hub
Free until
8000 messages/day

Azure Database
Free for
the previous Azure
Database generation
(the 4th),
few GB of space

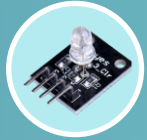Major drawback:
you can not have any
backup possibility

Azure App Service Plan
Free
complete solution to
deploy a full-stack
application with both FE
and BE.
1GB of storage
1GB of RAM
shared CPU

Please note that in such
a way you can not keep
your application always
running.

RGB Led
< 1 €

Raspberry Pi
> 20 €

ESP32
10 €

PRICE
For further details, please have a look at Microsoft Pricing Calculator.

FUTURE PLANS

MISSING PARTS & FUTURE PLANS

For several reasons we did not implement some parts even if they are clear in our mind

The main idea that we did not implement is the Machine Learning algorithm that should have to analyze the users' behavior for improving the customized tours.
The reason is that machine learning is not one of the principal topics of this course and it would have asked some specific kwnoledge bases

The second idea that we did not implement is a crowd-sensing logic in the Android App, through which we could have to improve the BLE positioning: even if it was very good for our purposes, we chose to follow the RSSI and Kalman Filter way.

Other minor improvements that we thought about are:
- the use of Azure database instead of a DbContetx, for storing the information about the artworks and the tours in a robust way
- the Moment method for evaluating the user experience during his use of the app, for which is suggested a commercial tool

THANKS FOR YOUR ATTENTION