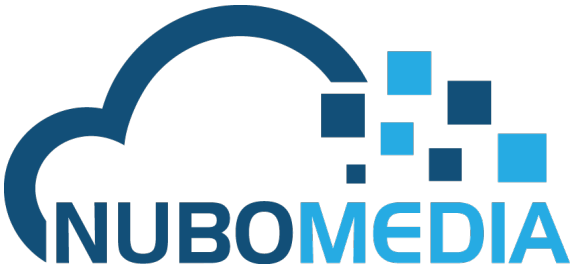
D6.3	
Version	0.3
Author	NAEVATEC
Dissemination	PU
Date	31/12/2016
Status	FINAL



D6.3: NUBOMEDIA Kurento Health Communications Demonstrator

Project acronym:	NUBOMEDIA
Project title:	NUBOMEDIA: an elastic Platform as a Service (PaaS) cloud for interactive social multimedia
Project duration:	2014-02-01 to 2017-01-31
Project type:	STREP
Project reference:	610576
Project web page:	http://www.nubomedia.eu
Work package	WP6: Demonstration
WP leader	ZED
Deliverable nature:	Report
Lead editor:	Teo Redondo / Paula Collazos (ZED)
Planned delivery date	30/09/2016
Actual delivery date	31/12/2016
Keywords	NUBOMEDIA, Demonstrator, Health Communicator.

The research leading to these results has been funded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement nº 610576

FP7 ICT-2013.1.6. Connected and Social Media

Contributors:

Teo Redondo (ZED)
Jorge Vinaches López (NAEVATEC)
Juan Ángel Fuentes (NAEVATEC)

Internal Reviewer(s):

Luis López Fernández (URJC)

Version History

Version	Date	Authors	Sections Affected
0.1	14/10/2016	Teo Redondo / Paula Collazos	All (Structure of the Content)
0.2	19/10/2016	Jorge Vinaches	Fill document with demonstrator information
0.3	29/11/2016	Juan Ángel Fuentes	Minor changes
0.4	27/12/2016	Jorge Vinaches	Minor changes

Table of contents

Executive summary

Kurento Health Communicator is an application, available on Android, iOS and WWW, which is focused on eHealth communication, providing services and capabilities such as multimedia instant messaging and real-time multimedia communications between, for e. g., doctors and patients. By this way, patients with home treatment could communicate with doctors (keeping his privacy) in the fastest way possible, or even establish a video call with the capacity of sending real time data from medical equipment to be evaluated by the doctor.

In relation with instant messaging, *Kurento Health Communicator* allows sending messages from doctor to doctor, doctor to patient, or creating a group with a patient and some doctors to share information. Instant messaging could attach multimedia information, such as images or videos to share medical information and improve user experience. This capability will use also the push notification service if needed.

In relation with real-time communication, this application allows creating a multi-conference video call where you can talk and see other patients or doctors in real time. Here you could, as a patient, send medical information (i.e. hearth rate and blood oxygen from a pulsioximeter) to other participants in the conference, which allow evaluate the patient health from a remote location without the necessity to move him to a hospital.

Introduction

After some study cases related to medics and hospitals in collaboration with Niño Jesus Hospital in Madrid, we have noticed that communication between patients and their doctors, or even between doctors, are quite poor. There are several cases where patients with home treatments have doubts or problems with their treatments, so they feel like they need to contact their doctors to handle this situation. That communication is not easy to be accomplished, because hospitals keep doctor's phone number private. Patients are required to call to the hospital first, and then, if needed, that call is redirected to the proper specialty department, and finally the patient must wait for the doctor, if available, to answer that call. All this sequence takes a lot of time, and is not always possible in the first attempt. When talking about health, time can be very relevant to solve the situation so that's why we want to provide the tools to route efficiently the patient to the doctors and accomplish this tasks in the best timing possible.

Thinking about this issues, we have found a way to communicate patients and doctors, keeping patient's and doctor's privacy and allowing patients to send data from the medical equipment they have at home, in a quickly and safely way. So, doctors are enabled to take smarter decisions based on the analysis of this data and the feedback provided by the patient. Making a big difference with traditional ways of communication.

With this demonstrator, we want to create patient to doctor chats, doctor to doctor chats, groups which contains patients and doctors, etc., so they could be able to send text messages between them using **instant messaging** and, in case of the need to communicate quicker or in a more interactive way, create a video conference using enhanced **real-time multimedia communication** and allowing to send data from their medical equipment, also in real-time.

Instant messaging has been implemented through a simple API REST. With this API you can send also media files, such as pictures or videos, attaching them to a specific message.

Nevertheless, real-time multimedia communication is much more complex than instant messaging, and here is where NUBOMEDIA offers a real solution. This demonstrator's feature is the most relevant because by this way a patient or a doctor can create a videoconference allowing then to talk and see each other in real time, so doctors could evaluate in every moment the patient's state. At the same time, the data sent by medical equipment (i.e. pulsioximeter) is represented in the same way doctors are used to get those readings, this feature is enabled by the KMS filters deployed in NUBOMEDIA. Thanks to NUBOMEDIA iOS and Android APIs, we have been able to develop all the real-time multimedia communication so much quicker and really easy, reducing the development time and increasing the quality of the demonstrator.

1 Technological objectives of the demonstrator

The demonstrator's objectives created by NAEVATEC are the following:

- **Multimedia instant messaging.**

Instant messaging is one of the main objectives of this demonstrator. We can send a text message to other user or a group and receive their answers. If I were a patient, I could send a text message directly to my doctor without knowing his personal phone number. Or, if I were a patient and were in a group with some doctors, I could write them asking something and anyone could answer me.

Also, we can attach multimedia files into a specific message, such as images or videos. For example, if a patient observed changes in his skin color, or an abnormal mass in the arm, he would be able to take a photo and send it to the doctor's group.

Push notification (GCM in Android, APNS in iOS) are used to notify users in case the application is not running in foreground and a new message arrives. These mechanisms are very energy efficient so the application can be running without affecting the stamina of the battery. And more important, even if it closed because a reboot or failure in the phone, it starts again when a new message arrives.

- **Real-time multimedia communication.**

This is one of the main objectives of this demonstrator. It consists in creating "rooms" where patients and doctors can join and talk privately with each other.

In the *Kurento Health Communicator* demonstrator, we use "rooms" when you want to talk directly with people inside a group, or if you want to establish a peer-to peer video-call from a one-to-one chat. Nobody can create or join a room with other person if they don't share a group or the initiator doesn't exist in the app contact list of the receiver. For this reason, you cannot create a room with a doctor if you don't have that doctor in your app contact list.

This objective was developed using NUBOMEDIA Android and iOS APIs. Without that APIs, this objective would be so much harder to develop.

- **Smartphone platforms.**

Kurento Health Communicator support Android and iOS devices.

In Android platform, the demonstrator has been implemented using the official Android APIs and NUBOMEDIA Android APIs. The application has been fully tested and took special attention to usability concerns.

In iOS platform, the demonstrator has been implemented using objective C APIs and NUBOMEDIA iOS APIs. The application has been fully tested and covers all the main objectives specified for the demonstrator including instant messaging, real-time multimedia communication…

- **WWW interfaces.**

NAEVATEC provides a web interface where a user identified by his username can join a room and communicate with other partners. It provides same functionality than real-time multimedia communication in mobile apps, you can join and leave a room, see all peers in the room...

- **Multisensory data integration.**

Other objective of this demonstrator is to integrate the data provided from medical equipment into the real-time communication. This way, a patient can send data from his medical equipment in real-time to be evaluated by his doctors.

Using data channels through iOS and Android APIs, we can send that medical information to other peers connected to the room.

At the moment, only Bluetooth has been used to retrieve information from medical equipment. But this could be enhanced with NFC or USB-C in futures versions if medical equipment vendors decide to offer this communication interface.

There are many other professional areas which demand a number of additional requirements such as:

- **Security and privacy guarantees.**

All communications using *Kurento Health Communicator* are completely secure and private, to achieve this goal, HTTPs connection are mandatory to use this service. The communication with the database can be done through an encrypted channel using certificate retrieval. The database can be stored in an encrypted store. And even security can be increased using VPN services available for mobile devices.

When relying on GCM and APNs, no content of the message is sent, It's just used to notify the application to connect Kurento Health Communicator service to retrieve messages.

- **Auditability and traceability of user behavior.**

All conversations established using *Kurento Health Communicator* instant messaging could be reviewed to get to know the evolution of the patient.

- **Storage and recovery of messages, pictures and clips exchanged through the application for external usage.**

All media transferred using *Kurento Health Communicator* instant messaging could be download into a pc or external storage for e.g. an SD card, in order to be analyzed in other devices rather than a mobile.

2 Requirements that the demonstrator must satisfy

Kurento Health Communicator contains a number of features or requirements that must satisfy, those are the following:

- **Scalability:**
  1. Scale up to 50 simultaneous users (hospital unit level).

     In relation with this feature, tests were run in collaboration with Hospital Niño Jesús in Madrid, which agreed with using a limited trial version and with a capacity from 20 to 50 concurrent users.

- **Communication capabilities:**
  1. Instant Messaging (IM).
     a. Attach pictures.
     b. Attach pre-recorded videos.

     This requirement has been developed through a simple REST API. Using HTTP POST request, we attach a JSON text in the body request with the message we want to send. If the message contains multimedia files, such as image or video, the "content-type" of the request must be "multipart/form-data".
     Here are the screenshots sending a simple text message.

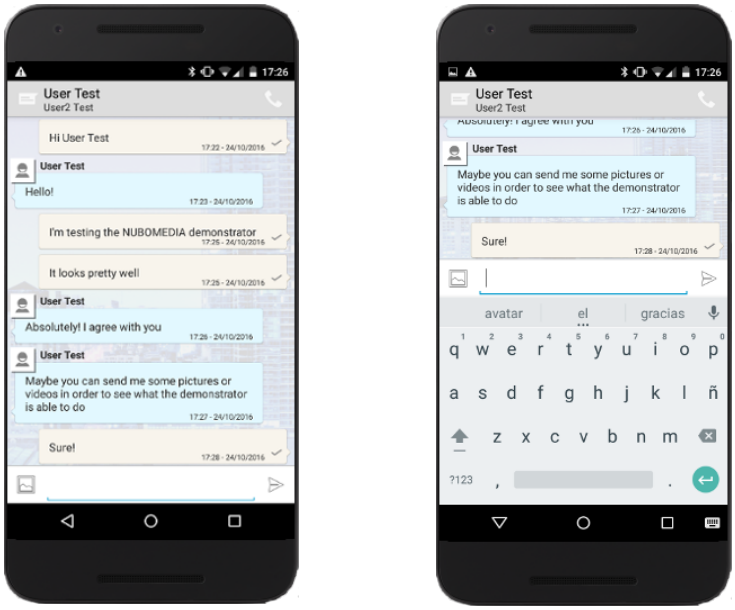


Figure 1 - Send text message

Other required capabilities are attaching files to a specific message, such as pictures or videos. Shown in the screenshots below, how to attach that type of files and the result after sending one.

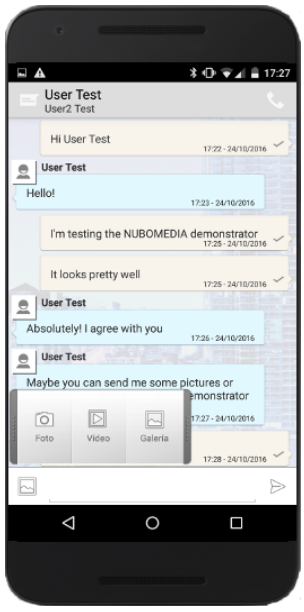


Figure 2 - Attaching button action

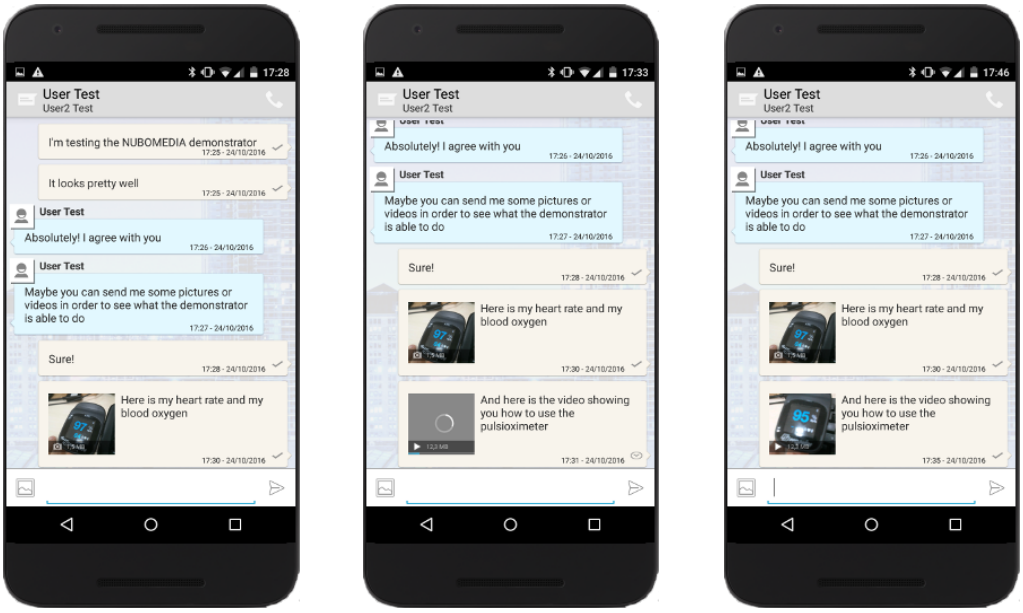


Figure 3 - Send picture and video

2. Real-Time multimedia Communications (RTC)
    a. QCIF RTC video al 15pfs

This requirement was the hardest to achieve, and here is where NUBOMEDIA has helped us through its iOS and Android APIs. NUBOMEDIA provides a simple API to build mobile application which needs real-time multimedia communications easily.

To develop this demonstrator, it was decided to create rooms to solve video communication between two participants or more from the group. Thanks to NUBOMEDIA APIs, you can join a room, leave a room, know who is joining…and send data to other peers using WebRTC Datachannels.

Next screenshot shows the result on an ongoing room with 2 participants.

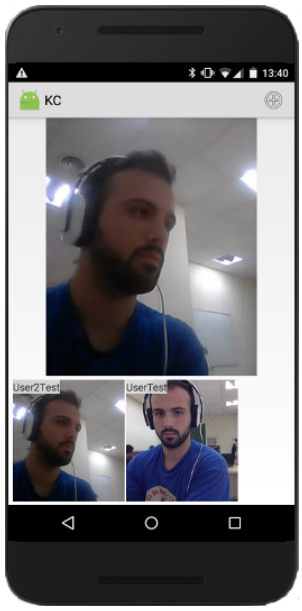


Figure 4 - Room

- **AR capabilities:**
1. Multisensory data coming from pulsioximeter.
2. VCA capabilities shown on top of videoconferencing flow.

To get the data from the pulsioximeter, we have used the BLE technology (Bluetooth Low Energy). NAEVATEC uses a pulsioximeter which is based in that technology, so we use the Bluetooth iOS/Android native API to get the data from the pulsioximeter.

Once we have retrieved the data, we send it through WebRTC Datachannel, using again the NUBOMEDIA APIs. By this way doctors can retrieve real-time data from medical equipment and analyze it.

Then, to show the data that we send into the room, Kurento Media Server merge the video stream through a filter to represent the data received through the datachannel in the output video stream.

Here we can see how to pair the pulsioximeter and the representation of the data in the video stream.

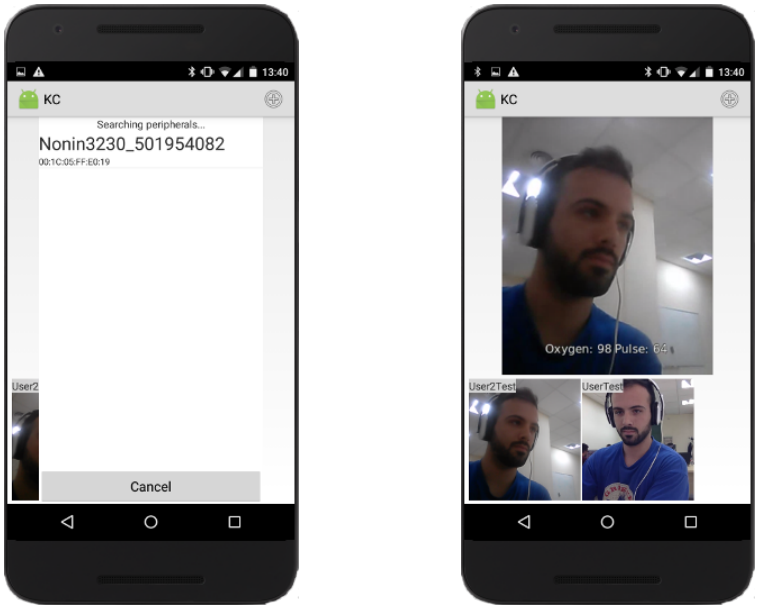


Figure 5 - Multisensory data and AR capability

- **Management capabilities:**
  1. User management (Registration, login, etc.)

To use the demonstrator you must be registered into the system. There are two ways to get registered. If the hospital allows to any person use the application there would be a *registration* button in the home screen, but if the hospital wants to use the application just patients and doctors, there are a manager web page to register users manually and to have the control about which users could use the application and could not.

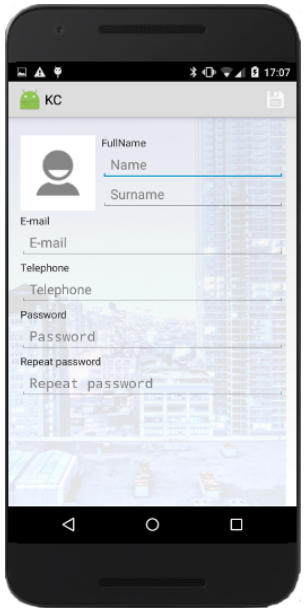


Figure 6 - Register screen

2. Group management (Creation, membership, etc.)

As in the *user management* capability, we can manage groups by two different ways. If the hospital allows creating, deleting and editing groups by anyone, then we will have buttons in the user interface to execute those actions. However, the hospital would want to take control about groups and members of those groups, so through the manager web page the hospital could also create, delete, edit groups and its members.

Figure 7 - Create group

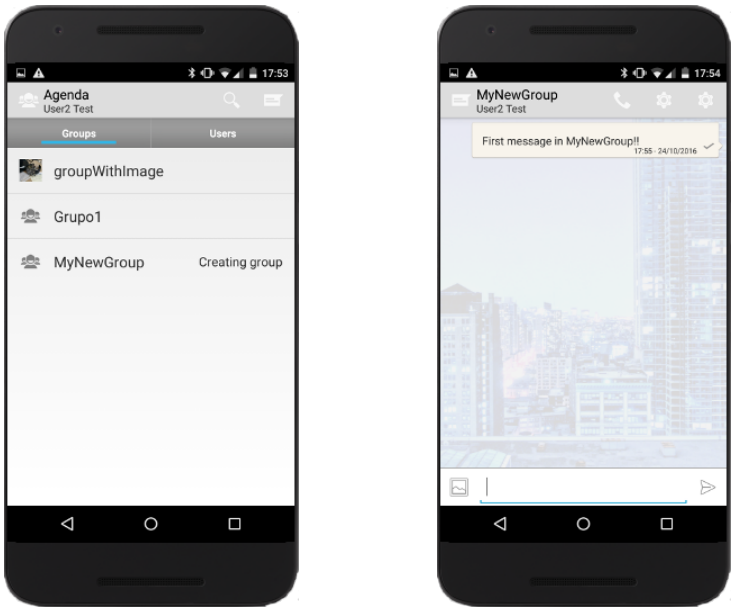


Figure 8 - New group created

- **Other features:**
  1. Privacy:
     a. Satisfaction of legal regulations.
     b. Protection of doctor personal information.
     c. Protection of patient information.
  2. Security.
  3. Multiplatform (availability on Android and WWW, iOS desirable)

- Android:

  Demonstrator on Android platform is fully implemented, all characteristics and features have been developed.

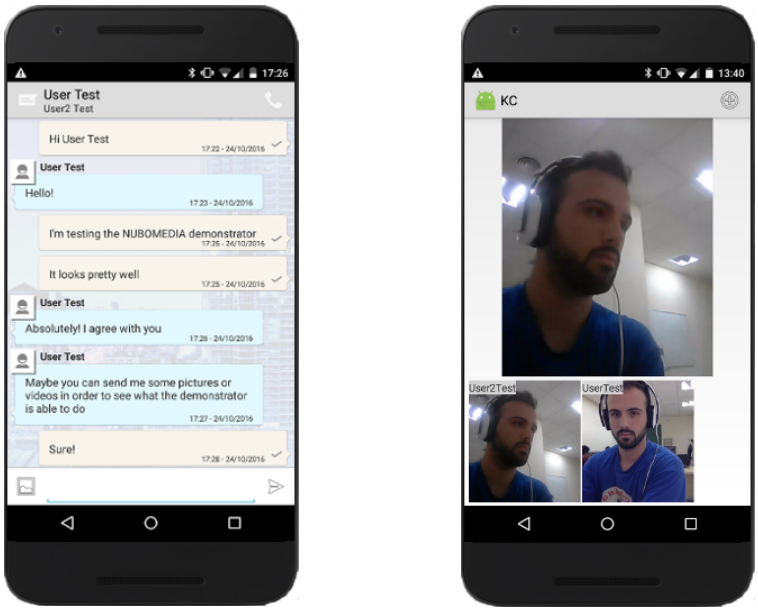


Figure 9 - Android demonstrator

- WWW:

  Developed room application to test NUBOMEDIA capabilities. Not developed instant messaging characteristics.

Figure 10 - WWW demonstrator

- iOS (desirable):
  Demonstrator on iOS platform has completed all NUBOMEDIA related features to test the proper performance of the API. Instant messaging has been developed, but without attaching media files to the message.

Figure 11 - iOS demonstrator

4. Multidevice (same user may connect to its account from different devices)

   As a demonstrator user (doctor or patient), you can log in and use the app at the same time in all devices as you want.
   All users have a unique *userID*, and all phones (both Android and iOS) have their own UUID. So, when you log in into the demonstrator, you will have your own *userID* that identifies the user logged in, and also a *phoneID* that identifies the device you used to log in.
   Because of that, in every moment we are able to know who is sending a message and from which device.

5. Application needs to be intuitive and simple to use so that non-experts can access it.

   Both, iOS and Android applications, have been developed using the *User Interface Guidelines*:
   - Android: https://developer.android.com/guide/practices/ui_guidelines/index.html
   - iOS: https://developer.apple.com/ios/human-interface-guidelines/

   Furthermore, as NAEVATEC has been collaborating with Niño Jesús Hospital in Madrid, their doctors have provided us the necessary feedback to improve the user interface and make it more intuitive and easy to use.

3 Architecture of the demonstrator (related to NUBOMEDIA architecture)

The following diagram shows the *Kurento Health Communicator* architecture and how it fits in NUBOMEDIA.

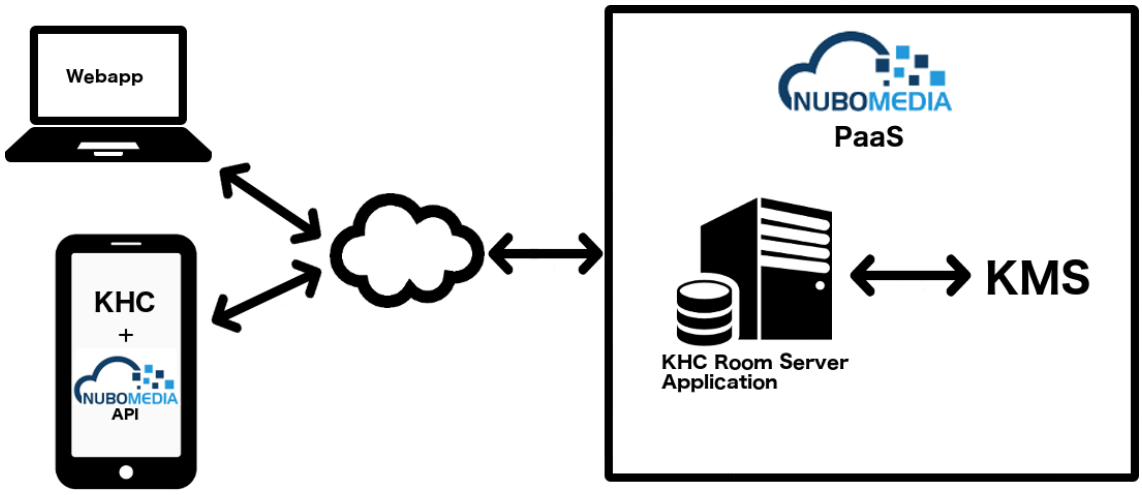


Figure 12 - KHC Architecture

As it is seen in the diagram, the NUBOMEDIA API it is used on the mobile-client side (iOS, Android) to communicate, through the Internet, to our KHC room server application deployed into the NUBOMEDIA PaaS. Inside the PaaS has been deployed our application and uses a Kurento Media Server.

The NUBOMEDIA API for mobile devices contains all the messages and notifications we can receive from the PaaS so it is really quick to develop a mobile application in the NUBOMEDIA infrastructure.

The following diagram is a chart showing the *Kurento Health Communicator* client architecture.

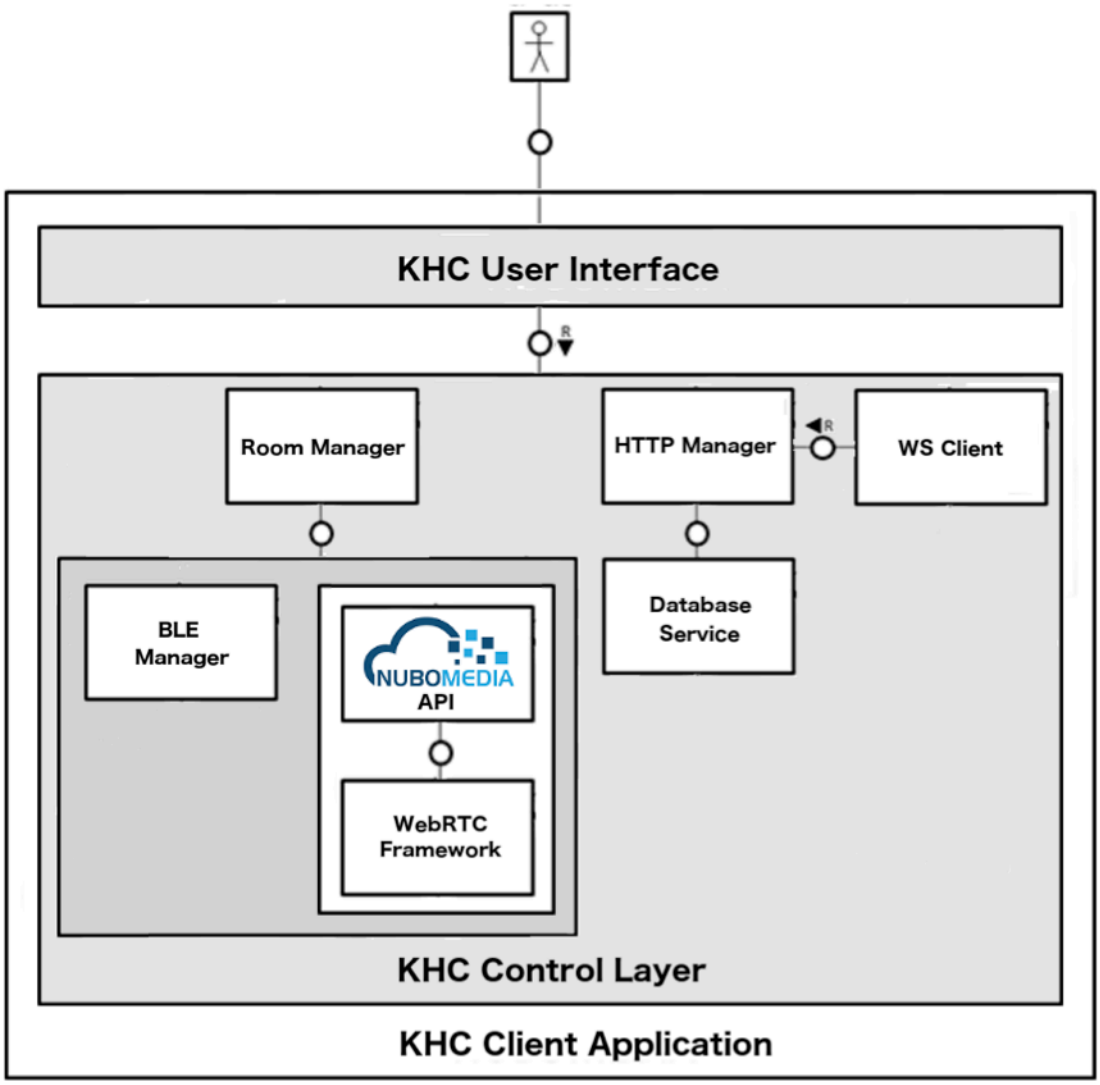


Figure 13 - KHC Client Architecture

The mobile client application is divided into 2 main layers. One is the user interface and other one is the control layer.

We are going to focus on the control layer. There it is important to see the *Room Manager* element that uses the NUBOMEDIA API to communicate to the NUBOMEDIA PaaS. It uses also a *BLE Manager* to control and retrieve information from every BLE device.

4 Relevant implementation details

GCM & WS

As mobile devices have limited battery life, it cannot stay polling to the server every second asking for new messages. Because of that it has been implemented push notifications in both platforms (Android, iOS) to know when there are new messages to retrieve and save battery.

Each platform has its own push system, Android calls it GCM (Google Cloud Messaging) and iOS calls it APNS (Apple Push Notification Service), but the concepts behind both systems are the same.

When a new message arrives to our server, it tells the push service "notify X device", then the GCM or APNS service sends a notification to that specific device, then the device knows that there are new messages in the server and try to recover it.

Using push notification technologies allows saving a lot of battery, but there are some problems related when using it. The main problem is that you don't have control over it, a notification could take 30 seconds or more to reach a device and there is no delivery guarantee, so to ensure the best user experience while using the app, it has been implemented also a websocket channel connection.

Websocket is a technology which keeps a channel opened and allows you send messages in real time with no delay, but consumes so much battery than push notifications.

So when the application is in foreground and it is needed to have the best user experience it is enabled the websocket channel and, when the application is in background it is enabled push services to save battery.

BLE (Bluetooth Low Energy)

Retrieving information from external gadgets is quite hard. There are no unified rules to retrieve information and the services provided from gadgets. In this way, to connect medical devices to the phone and retrieve the information, it has been used BLE (Bluetooth Low Energy) that has some services defined and consumes so much less battery than usual Bluetooth technology. Anyway, each gadget has its own way to send data, its own services and it is hard to create an unified code to manage all gadgets, so until there exists a consortium to unify BLE services and characteristics, and a unified way to send data, it is mandatory develop a manager class for each gadget integrated.

Room Manager (Managing NUBOMEDIA API)

This is the main class we use to communicate to the NUBOMEDIA API. From this class we init all the components related to the NUBOMEDIA API such as the room server URL, room name, username, use of datachannels, turn and stun servers…

From this class we also listen to receive all the events related to the NUBOMEDIA API. There are some important events that we must listen such as “room joined”, “peer joined”, “peer left”… By this way we are sure if we have been joined in a room successfully or not, if there are new peers joining or leaving the room…

When the Room Manager object receives an event from the NUBOMEDIA API, it parses it and tells the interface layer what is has to show. As an example, if a new peer has joined the room we must show a new peer in the user interface.

Also, we use the Room Manager to send data through datachannel. It has another module to manage BLE data, but when the data from the BLE device is retrieved we send it through datachannel using the Room Manager and the NUBOMEDIA API.

5 Testing mechanisms and methodologies

Demonstrator basic functionality (messaging, group management…)

Our tests have been made using testing mechanisms provided in the Android and iOS platforms.

Testing specific documentations are located here:

- Android: https://developer.android.com/training/testing/start/index.html
- iOS: https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/testing_with_xcode

Test we have done are the following:

Register user	✓
Login	✓
Create group	✓
Edit user profile	✓
Edit group	✓
Add/remove group member	✓
Add/remove group admin	✓
Send message	✓
Send message with photo attached	✓
Send message with video attached	✓
Logout	✓

NUBOMEDIA APIs

To test NUBOMEDIA APIs we cannot use automatic test as we have done to test other functionalities because it is impossible check if everything in a videoconference is going well. For example, we cannot check programmatically if our video is freeze, or the received video quality, or maybe if a remote peer doesn't send video/audio properly, is really difficult to know if there was a problem of the API, the internet connection or something else.

Because of that we have written a list of scenarios and we have tested manually each one to ensure the proper functioning of the API.

These are the scenarios tested:

Mobile iOS to mobile iOS	✓
Mobile Android to mobile Android	✓
Webapp to webapp	✓
Mobile iOS to mobile Android	✓
Mobile iOS to webapp	✓
Mobile Android to webapp	✓
Mobile (WiFi) to mobile (WiFi)	✓
Mobile (4G) to mobile (WiFi)	✓
Mobile (4G) to mobile (4G)	✓
Mobile (3G) to mobile	✓
Mobile (Bad internet signal) to mobile	✓

6 End-user's guide and tutorial: how to install and use

The best way to install the application in the mobile devices are downloading the source code and importing into the XCode or Android Studio as a project to build it and test it directly in physical devices.

As some libraries are just compiled for ARM architectures, there is a problem if we want to install and run the demonstrator in mobile emulator, so we need to have an Android and an iOS physical device to use it.

How to use:

Once we have the application installed in a physical phone, first time we enter the application we need to set our own account. To test the NUBOMEDIA environment, the account name will be ***nubomedia***.

Then we have to log into the application to send messages and start video rooms. To explain how to use the application, we provide some videos with basic functionality of both platforms (Android, iOS).

- Android login and messaging:
https://www.youtube.com/watch?v=RGiI3RyBQ10

- Android NUBOMEDIA room:
https://www.youtube.com/watch?v=gf6hGnX5sUU

- iOS login and messaging:
https://www.youtube.com/watch?v=nHsvOyRfMck

- iOS NUBOMEDIA room:
https://www.youtube.com/watch?v=DgIJMu-RjaI

7 Total effort consumed in creating the demonstrator

NAEVATEC real effort devoted to creating the demonstrator is about **4** Person/Month.

But all these effort is not related to develop the NUBOMEDIA part. There are other kind of efforts related to design, GUI development, instant messaging development, debug NUBOMEDIA APIs…

Because of that, to make a better approach, NAEVATEC is going to split the effort in 3 different ways:

- Development effort devoted in develop the NUBOMEDIA part is about **1.5** Person/Month.
- Development effort to optimizing and debugging the platform is about **1.5** Person/Month.
- Other development effort (i.e. GUI, design…) is about **2** Person/Month.

8 Developer's feedback in relation to using NUBOMEDIA

NAEVATEC's developer feedback has been quite positive when using NUBOMEDIA APIs.

As a company with expertise in video communication and WebRTC technologies, we already knew the difficulty behind the development of mobile APIs to manage room sessions with many audio and video streams at the same time. Managing SDP negotiation, ICE candidates and everything related to a video communication is hard, but when you want to scale it and make it work to many users at the same time, the difficult to accomplish that grow exponentially.

Because of that when we started using NUBOMEDIA APIs, we were really surprised about the simple use of it, saving a lot of time to NAEVATEC developers to achieve the same result.

Moreover, NUBOMEDIA APIs are easy to understand, so we don't need to spend a lot of time reading documentation to make it work, just after a couple of hours reading how to use it we started developing our demonstrator. First versions of the APIs didn't work as expected, but after reporting some bugs we found out, we noticed both APIs (Android and iOS) became really stable.

To sum up, NAEVATEC experience is really positive developing with NUBOMEDIA APIs, building a demonstrator really useful to other companies much more faster and safer than developing the same software from scratch.