Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

Realidad aumentada e IoT como sistema de información para usuarios y personal de hospitales

Autor: Pedro José Vacas Blanco

Tutor: Juan Manuel Miguel Jiménez

2019

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

Trabajo Fin de Grado

Realidad aumentada e IoT como sistema de información para usuarios y personal de hospitales

Autor: Pedro José Vacas Blanco

Director: Juan Manuel Miguel Jiménez

Tribunal:

Presidente: J. Antonio Jiménez Calvo

Vocal 1º: Alejandro Martínez Arribas

Vocal 2º: Sira Elena Palazuelos Cagigas

Calificación: .....

Fecha: .....

A mi familia y pareja, apoyos incondicionales en mis andanzas, sin ellos el camino hasta aquí no habría sido posible...

"Soy deshonesto, y siempre puedes confiar en que la persona deshonesta será deshonesta. Honestamente. Es a los honestos a los que tienes que vigilar de cerca, porque nunca puedes predecir cuando harán algo... absurdo"

Jack Sparrow

Agradecimientos

No importa quiénes seamos, sino cuál es nuestro plan.

Bane, El caballero oscuro

Un único trabajo para agradecer todo un final de ciclo en mi vida, un único trabajo que demuestre que todo el esfuerzo realizado por mi familia para que yo llegue hasta aquí tenía un sentido y posteriormente su recompensa. Gracias papá, gracias mamá, sin vuestra ayuda no estaría cerca de donde estoy ahora.

También merece mención Beatriz, mi pareja sentimental y de estudios, mi calendario y orden durante estos años en mucha medida, por no decir completamente, dependió de ella y su capacidad para ordenarse y ordenarme, de aguntar en las buenas y en las malas en esos momentos en los que necesitas un "algo más" si no quieres desesperarte.

Dar las gracias a Daniel, mi hermano pequeño, por encargarse de parte de mi crecimiento personal, por cosas que solo entendemos los que tenemos hermanos, y de forma especial, por ser el inspirador del tema de este trabajo, ya que está basado en él y en su experiencia "rompiéndose" en trozitos en las locuras de su edad. Sin ti no me conocería urgencias como me las conozco, sin bromas, gracias enano.

Por último, a todos esos profesores que "no" daban clase, o no solo daban clase, sino que enseñaban sus conocimientos al resto y no los que venían en el libro recomendado en la biografía. Los que sentían que su trabajo es más una vocación, permitiéndonos disfrutar de sus clases. Explicita mención entre ellos a mi tutor D. Juan Manuel Miguel Jiménez, gracias por aceptar este tema de trabajo y gracias por las materias impartidas.

Resumen

Ante la necesidad creciente de una gestión automatizada de la información, en cualquier ámbito, nace el siguiente proyecto. Siendo originaria la idea de aplicar un simple sistema de Realidad Aumentada a la información a clientes de un parque temático, evolucionó a la gestión de una estructura crítica como pueden ser las urgencias de un hospital, con la idea de liberar tiempos de espera y facilitar la información sobre el paciente a los familiares o conocidos sin necesidad de ocupar al personal del hospital.

Para conseguir este fin, se ha optado por la unión de tecnología basada en el Internet de las Cosas y Realidad Aumentada, en un aplicativo listo para usarse en dispositivos móviles y con una disponibilidad de 99,9 como debe ser en estos servicios, gracias al uso de otro de los avances de la última década como es la computación en la nube. Por lo tanto, durante el desarrollo de este proyecto, encontraremos trabajo que va desde el campo de OT, donde encontramos cableados, conexión de sensores, actuadores y otros, hasta IT con programas de gestión y visualización de la información, como aplicativos para móvil.

Palabras clave: Internet de las Cosas, Realidad Aumentada, Hospitales, Sistema de Información, Computación en la Nube.

+

Abstract

There is a growing need for automated information management in almost every field. and because of it, this proyect comes alive. Started time ago, with the idea of develop a simple Augmented Reality application, to give contextual information for customers in a theme park and evolving to an IoT proyect with some Augmented Reality inside. The target? Manage critical infraestructures like the emergency services of a hospital, making waiting time short and employees less asked about the userś relatives, which gives comfort to both, users (all avaiable information at their fingertips, include some information that employees are unable to have) and employees (more time to work in the serious bussines, less distractions and stops in the corridors).

For reaching this point, several technologies must join, from OT to IT, using wired connections to electronical devices such as sensors, mini-computers, cameras and other software solutions like neural networks for cameras, portable device applications, backends for serving the information. All of these, with a 99,9 percent of uptime, required in all of the critical services, so cloud computing takes an important placement here giving us the underlying infraestructure, a place where our code is safe to be running without worries of downtimes or updates.

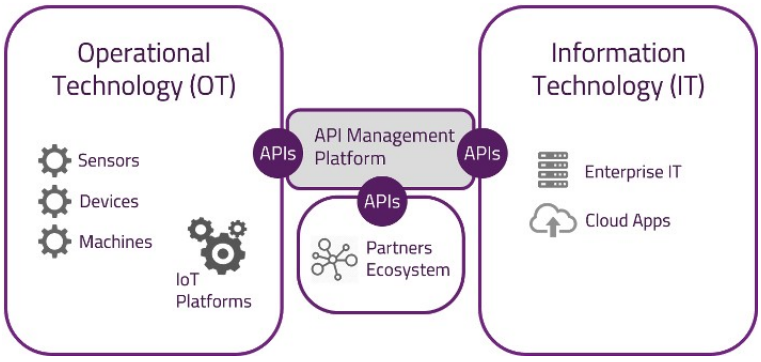
Keywords: Internet of Things, Augmented Reality, Hospital, Information System, Cloud Computing.

Resumen extendido

Actualmente nos encontramos en una era que quiere convertir todo en digital, donde la información obtiene cada vez un valor mayor en el mercado, siendo esta como dinero para las empresas, tanto por utilidad como servicios que permite desarrollar. La obtención de estos datos requiere siempre de un sistema de información, preferentemente automatizado, en el que no tener que incluir los datos de forma manual, evitando así gastos en personal que digitalice contenido analógico, como evitando la pérdida de datos por los momentos en los que el observador no estaba mirando. Un alumno que suspende el examen de maniobras de motocicleta porque lo hizo en +0.05 segundos del permitido, ya que el examinador se distrajo durante medio segundo al llegar a la meta, le supone un sobrecoste de 150 euros sobre su examen, un árbitro que no estuvo atento al último segundo del reloj y hace que cuente la última canasta del otro equipo que le sitúa un punto arriba en la final le ha costado unos cuantos millones de euros al equipo contrario, es decir, la información es mucho más poderosa, necesaria y valiosa de lo que se considera cuando no se trabaja habitualmente con ella.

Los grandes bancos de datos, tanto propio como externos, que usan las empresas para realizar tareas de machine learning y mejorar su producción aumenta día tras día, con cada vez más contratos de eficiencia energética, los contratos públicos para la recolección de datos para posterior analisis en algun E.D.A.R. por parte de empresas como Canal de Isabel II tampoco faltan, y esto es, porque esa información e inversión a futuro, a veces corto y a veces largo plazo, les supone un ahorro considerable o les provee de un servicio importante.

En este caso, se ideó un proyecto que incluía un sistema de Realidad Aumentada para parques temáticos. Estas tecnologías son conocidas por permitir la observación de la información de forma contextual, es decir, lee la sitúación en la que nos encontramos a través de cámaras y proyecta sobre ese contexto información que no podemos conocer a simple vista. Dando un pequeño giro de perspectiva se perfiló el proyecto que aquí se expone, un sistema de infromación para su uso en hospitales, que de forma económica permita un gran valor a los usuarios del hospital, como al propio hospital, dotando al sistema de una recolección de datos automátizada, tratamiento y muestra de la información y servicios añadidos al ciudadano.



Extrapolable a cualquier hospital, este sistema de información ha tomado como ejemplo nuestro querido hospital universitario, tomando como referencias sus instalaciones, posicionamiento y servicios que ya ofrece, simplemente porque siempre es más fácil analizar un sistema de este tipo cuando el lugar ya existe, y creo que además nos es conocido.

Queremos un sistema que, desde el momento incial en el que se produce una urgencía, se pueda trackear la misma y empezar cualquier preparativo. Para ello debemos considerar que la urgencia puede producirse en cualquier sitio, por lo que, el elemento avisador debe ser portátil. Por ello, decidí que cualquier dispositivo con internet debería poder realizar este aviso, es decir, debemos confiar en Internet, en la red, como infraestructura de datos, y además, la mayoría de avisos se producirán con un modelo de datos itinerante, es decir, no cableado y red móvil, por lo que las consultas que se realicen deben ser lo más livíanas posibles, evitando así que alguien que en el momento de la urgencia no tenga internet a alta velocidad, comunmente llamado sin megas o sin datos, pueda sin problemas envíar la urgencia, para ello, el tamaño de la petición estará por debajo de los kilobits por segundo que ofrecen normalmente las compañías.

Una vez hemos obtenido esos datos debemos extraer de ellos el conocimiento. Disponemos de dos aproximaciones, la primera de ellas es conocimiento para el usuario y la segunda es conocimiento para la empresa. Con la primera, en nuestro proyecto, conseguimos que el usuario tenga preferencia de paso sencilla y segura en las proximidades del hospital, ahorrandonos unos segundos, al darnos como información su matrícula y localización. También obtenemos una idea de qué se nos va a presentar en la unidad de urgencias, pudiendo establecer un intervalo de tiempo que tardará el paciente en llegar para preparar todo si fuera necesario una atención más inmediata y colocándole, desde el minuto 0 en la prioridad de urgencia correcta. Para el hospital, la información también es valiosa, encontrando entre las posibilidades de esa información, la confección de analíticas con el tipo de urgencia declarada, o la edad de los pacientes de urgencias e incluso mapas con las zonas que más utilizan el servicio tanto de dentro como de fuera de Alcalá, otros más banales como facilidad de acceso a las ambulancias también se pueden proveer.

Entre la infraestructura que aparece para proveer este servicio de control de tráfico, reconocimiento de matrículas y otros, encontramos dispositivos hardware como placas de desarrollo NodeMCU, especialmente pensadas para crear sistemas y dispositivos IoT y mini-ordenadores, en este caso se utiliza una Raspberry, cámaras y sensores de movimiento. De forma inicial, en las inmediaciones del hospital habría cámaras grabando permanentemente los diferentes accesos por los que se puede acceder al hospital, analizando de forma interna las matrículas que detecta, como las cámaras que se encuentran, por ejemplo, en los accesos a Madrid-Central, pero a efectos de la demostración teórico-práctica del proyecto, por el limitado presupuesto, se ha utilizado una webcam de bajo coste, sin la potencia suficiente para grabar y analizar todos los frames necesarios para algo en tiempo real, pero sí de realizar fotos y envíarlas a un ritmo relativamente rápido, por esto, se utilliza un sensor de movimiento que realizará la foto cuando el coche lo active al pasar por delante y será esta foto la que pase por nuestro algoritmo de License Plate Recognition, la matrícula detectada con mayor probabilidad será entonces envíada a una base de datos en memoria, conocida como Redis, donde se había guardado previamente la matrícula del coche que transporta al accidentado, aprovechando este tipo de tecnología para conseguir el tipo de respuesta más rápido posible de una base de datos, ya que tenemos un tiempo de respuesta limitado, porque el coche sigue avanzando hacia la intersección. Si la matrícula, por ende, ha sido registrada, cambiará los semáforos para otorgar prioridad a la vía por donde está acercándose el vehículo y cerrando el resto de vías para evitar choques y retrasos.

Cuando el paciente llega al hospital, deberá acceder a las máquinas de frontdesk del servicio de urgencias, e introducir en ella el código numérico que le asignó el aplicativo nada más comunicar su urgencia. Este número por sí solo no dice nada, pero está internamente relacionado con los datos que

ha comunicado y otros propios del hospital. Este frontdesk imprimirá y dispensará una pulsera con un sistema RFID pasivo en su interior, su dirección será grabada y asociada al paciente en el sistema para poder utilizarla como medio de obtención de datos en otros servicios. A continuación, pasará a la sala de espera, donde a través de sistemas de localización por RFID se podrá conocer la posición de este dentro del hospital para facilidad del personal si fuere necesario y de los familiares y conocidos que tengan permitido el acceso a esos datos de forma explicita por el paciente, con sistemas de barrera establecidos a través de asociaciones en las bases de datos del hopsital. Para la localización por RFID se han estudiado diversos sistemas que permitirían este trabajo de forma no invasiva, siempre respetando las zonas sensibles a interferencias, de forma pasiva y de forma activa, presentando cada uno sus ventajas e inconvenientes.

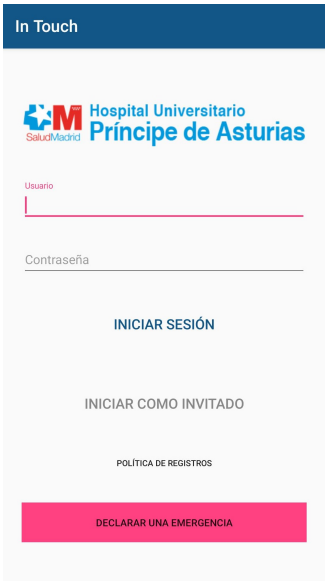
Para el RFID activo, debemos saber que son dispositivos más grandes, que requieren de batería con una duración de unos 5 años, pero que permiten la localización con un error de centímetros sobre el plano 3D real de una instalación, por lo que puede ser recomendado su uso en mobiliario del hospital que esté en movimiento, carros de parada, sillas de ruedas, camillas. Este sistema supondría, sin embargo, un coste de mantenimiento e inversión notablemente más elevado, que habría que valorar.

Para los sistemas RFID pasivos destacan entre sus características la ausencia de baterías, ya que la energía que necesitan la consiguen de las antenas del entorno, un rango de hasta 8 metros para su detección, por lo que las hace suficientes para el interior de un edificio, y un coste y mantenimiento nulo o muy bajo, además de disponer de formatos más cómodos al poder ser tan finos como un folio. Por todo ello, junto con un coste mínimo, son los elegidos para el proyecto.

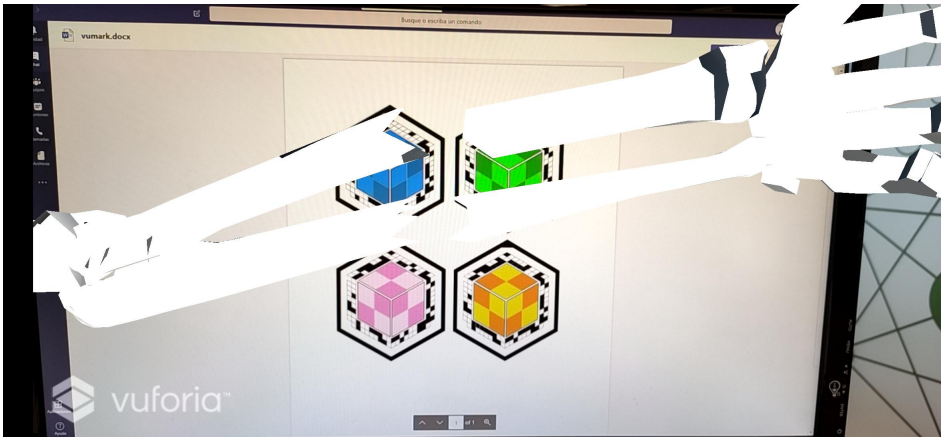
A través del aplicativo, esta información será visible, dando una información bastante fiable e importante para familiares del paciente. Muchos más sensores pueden establecer otro tipo de métricas y conectar con el aplicativo de la misma forma en tiempo real, para proveer servicios de constantes vitales, temperatura de la habitación, humedad, etc... Para realizar esta actualización en tiempo real, pero evitar consumos ineccesarios de batería y datos, se realizará mediante notificaciones push que realizan los cambios en el aplicativo únicamente cuando llegue. Esto supone la subscripción a un canal por parte del aplicativo, y dentro de ese canal, subcanales que diferencian el tipo de evento que se recibe y los campos a actualizar. La tecnología de notificaciones push permite que sea el servidor el que inicia la comunicación con el cliente, además, el usuario solo recibirá estas notificaciones si está conectado al aplicativo, impidiendo mensajes indeseados o en segundo plano.

Esta aplicación también posee otras vistas para consultar diferentes datos. Comenzamos con un menú inicial que permite el login o la declaración de una emergencia directamente, sin necesidad de usuarios. Para la parte del acceso con usuario, deberá registrarse, y ese registro debería ser de forma presencial en el hospital e incluir un sistema en la aplicación que permita el uso de certificados y/o DNI-e para el mismo, asegurando que la persona que se registra es quien dice ser, debido a la sensibildiad de los datos y leyes sobre los mismos. Una vez obtenemos ese usuario y contraseña accedemos a la navegación interna de la aplicación, aterrizando en primer lugar en la pestaña de hospitalización, que nos dará toda la información básica sobre nuestro estado como paciente si hemos sido hospitalizados, entre estos datos aparecen nuestro estado actual, el motivo por el que se nos ha hospitalizado o hemos acudido a urgencias, la habitación si requerimos ingreso con asignación de la misma, nuestra localización actual por el hospital, el doctor que está a cargo de nuestro caso en ese momento y la fecha con la que se ingresó. También disponemos de 2 botones, uno para la selección de dietas del día siguiente si las necesitaramos, evitando así gastos en papel y aumentando la centralización de la información y un botón para activar el modo de realidad aumentada, que activará la cámara y nos otorgará información sobre el contexto en el que nos encontramos, como por ejemplo, infromación sobre el hospital, mediante marcas especiales, en los casos que se considere, se podrá ver en 3D lo que nos pasa. Por ejemplo, cómo tenemos el hueso roto, dónde se encuentra la úlcera de estómago y cómo va a ser la operación, proporcionando un sistema de ayuda

para que los doctores expliquen a sus pacientes en que consistirá la intervención y que estos entiendan con mayor facilidad qué va a ocurrir, como dicen, una imagen vale más que mil palabras. En otros casos, fuera de hospitalizaciones, este modo también será accessible, proporcionando infromación sobre citas, como llegar al destino que deseemos y otros.



El modo de realidad aumentada citado anteriormente se ha conseguido mediante la programación en Unity con la librería Vuforia y Google AR Core, que permiten una forma sencilla de asignar ciertas imágenes a nueva información de la que no disponen. Para complementarlo se ha utilizado también programas de modelado 3D como Blender para, por ejemplo, huesos del brazo rotos, un sistema digestivo completo e incluso un feto, entendiendo que este tipo de casos serían los más facitbles para usar este tipo de tecnología ya que permite texturizados, grandes definiciones de detalle, etc... Para otro tipo de información, como puede ser la posición en el hospital de un departamento o consulta, se ha utilizado MagicaVoxel, que permite el modelado de arte voxel, basado en cubos, de prácticamente cualqueir cosa pero con menor nivel de definición, ya que nos basamos en que la unidad minima siempre será un cubo a diferencia de los vértices de Blender, aunque mismo nivel de detalle, si quisiéramos. Por último, se ha utilzado Gimp2 para mostrar información que no precisa de un entorno 3D para ser completamente eficaz o que incluso sería contraproducente si no fuera plano.



Otro de los factores a tener en cuenta sobre esta arquitectura es la aparición de computación en la nube, conocido también, del inglés, como Cloud Computing. La computación en la nube nos ofrece un sin fin de ventajas, con las cuales podemos cumplir los retos de las arquitecturas críticas de, con la

aparición del Edge Computing me atrevería a decir, cualquier empresa. Una infraestructura de este estilo necesita estar siempre disponible, esto es, tener un downtime cero o lo más cercano posible. Para ello, el primer paso es elegir un proveedor de servicios, en nuestro caso utilizamos Google Cloud Platform simplemente por el paquete de bienvenida inicial de 300 dólares para gastar durante 1 año en servicios, sumado a que los primeros X usos de cada servicio son gratuitos y no incurren en ningún tipo de gasto, ni siquiera de ancho de banda, X depenende del tipo de servicio. A continuación, si fuera posible, utilizar siempre los servicios en la nube que sean serverless, esto es, que no tengamos necesidad de gestionar, actualizar o revisar periódicamente, dejando esas tareas al equipo de mantenimiento y desarrollo del proveedor, ocupandonos únicamente del código. En el proyecto, se utiliza el servicio de Google Functions, que permite la ejecución de código, previámente subido a la plataforma, disponible esta subida en forma de zip, de editor de código en la nube o incluso directamente desde repositorios git. De esta forma, todas nuestras preocupaciones son relativos al código y a una configuración inicial de cómo queremos que se ejecute o a quién dejar acceso, el resto, relativo a memorias, disco duro, uptime y demás, solo será problema del proveedor. Otro de los servicios en la nube utilizados en MongoDB, una conocida base de datos NoSQL, que hace relativamente poco ha comenzado a ofrecer un servicio en la nube de clusters con MongoDB, dando toda la infraestructura y preocupando a sus usuarios únicamente de copiar la cadena de conexión, crear colecciones pero no de mantenerlo encendido, actualizado y otros trabajos de mantenimiento. El tercer servicio en la nube viene de la mano de Redis, que también ha empezado este año a ofrecer, desde su propio sistema, un servicio de clusters Redis en los que simplemente obtenemos una cadena de conexión a nuestra base de datos y jugamos con ella con el API y librerías de Redis, como si lo hicieramos en local. Por último, pero no menos importante, Vuforia, librería de trabajo con realidad aumentada, también nos ofrece un servicio en la nube para almacenar nuestros modelos y asignarlos a las imágenes correspondientes, que es como lo hemos utilizado durante el proyecto para poder actualizarlo desde fuera del aplicativo.

Índice general

Índice de figuras

Índice de tablas

Índice de listados de código fuente

Índice de algoritmos

Capítulo 1

Introducción

Nos estamos ahogando en información, pero estamos hambrientos de conocimiento.

John Naisbitt, Autor y Orador

1.1 Definición del problema

En un mundo donde los datos cada vez están cobrando más importancia, aún se encuentran servicios y empresas que, generando gran cantidad de datos, los desechan por varios motivos. Entre estos motivos encontramos: la incapacidad de usarlos, bien por no tener las herramientas o por no tener los conocimientos necesarios para ello, el desconocimiento del valor de empresa que contienen estos datos una vez se convierten en infromación y la cultura "si funciona no lo toques". [1]

Figura 1.1: Distintos niveles de una empresa según la etapa de digitalización en la que se encuentren.

De entre las posibilidades de ejemplo de empresas o servicios que tienen este problema, para ser usados durante el proyecto, se valoró el uso de un parque temático o el uso del Hospital Universitario Principe

de Asturias, ya que analizándolos, ambos generan y poseen una gran cantidad de datos que no utilizan o no aprovechan, además de tener sitios privilegiados donde poder añadir nueva tecnología que les retorne beneficios. Por ejemplo, para el parque temático se pueden establercer métricas del físico sobre el tipo de gente que visita el parque, para en un futuro poder invertir en atracciones que se ajusten al mayor de los perfiles que aparezcan, o establecer mapas con el lugar de origen de los visitantes para orientar mejor la información que ofrecen respecto a idiomas en los empleados y cartelería en visitantes internacionales y mejorar la comunicación con el parque para los visitantes nacionales, entre otras muchas mejoras posibles. Para el hospital, el despliegue de tecnologías nuevas en las cercanías del mismo permitiría obtener un tráfico más fluído de acceso al hospital tanto para pacientes con urgencias en vehículo privado como para la entrada y salida de ambulancias en servicio de urgencia. También entraría en juego la generación de mapas del entorno, con el lugar de origen de los pacientes, y métricas sobre cuáles son las intervenciones y urgencias que más aparecen, entre muchas otras características.

Finalmente, se ha optado por utilizar el ejemplo del HUPA, por ser más cercano y conocido para todos, lo que supone una facilidad en la detección de mejoras, posibilidades que ofrece y el hecho de ser universitario. Con este lugar en mente, se han detectado los siguientes puntos, a tratar como problema en el proyecto que nos ocupa:

- `accesos`, dependiente de la hora en la que nos acerquemos al hospital, encontraremos una situación u otra. En el caso de tener una urgencia a horas de salida y/o entrada de la universidad, estas son, por la mañana, a la hora de comer y a media tarde en menor medida, encontraremos esperas de unos 10 minutos de media para completar el recorrido desde la rotonda exterior al recinto universitario, hasta el HUPA.

- `recepción`, a veces ocupada por personas que usan el mostrador como información, bien porque no hay alguien en información o no han leído el rótulo, o por otros motivos, pero que hacen más lento el registro en el hospital.

- `localización`, en este caso del paciente, dentro del hospital. En el servicio de urgencias muchas veces se causa ruido y revuelo por personas que reciben llamadas al teléfono, que contestan allí y en voz alta, ya sea por edades avanzadas, estrés por la situación en la que se encuentran, pero que posiblemente cause molestia a otros usuarios. En el caso contrario también aparece la problemática de llegar al servicio de urgencias y no saber donde se encuetnra nuestro conocido porque tiene el móvil en completo silencio y no nos responde, por lo que nos paseamos por el servicio de urgencia pudiendo molestar a otros pacientes o al personal médico con preguntas que posiblemente no sepan.

- `estado del paciente`, dadas las limitaciones de acceso a urgencias para que sea un lugar tranquilo de atención a pacientes, muchas veces críticos y otras veces que necesitan tranquilidad, no siempre vamos a poder estar al lado de nuestro conocido o familiar, pero siempre queremos saber en todo momento como están, dando lugar a varios paseos hasta el box de urgencias cada poco tiempo, pudiendo causar molestias y falta de descanso a otros, así como que el personal de UVI esté también pendiente de nosotros y no solo de los pacientes.

- `mapas` y `cartelería`, quien no se haya perdido en un hospital es que no ha estado nunca en uno, o ha tenido mucha suerte. Muchas veces, debido a la gran cantidad de servicios que nos ofrecen, la cartelería no puede aumentar o estar presente en todos los lugares, lo que provoca llegadas retrasadas a nuestras citas, equivocaciones de sala de espera y otros inconvenientes.

- `otros`, o todo aquello que le podemos añadir a los servicios del hospital para mejorar la comunicación o la información que reciben empleados, alumnos al ser univesitario y pacientes.

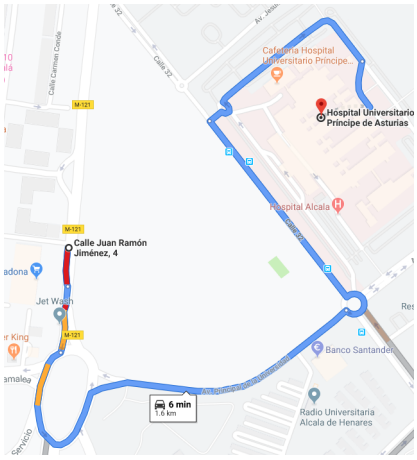


Figura 1.2: Recorrido hasta el HUPA desde rotonda de entrada al recinto universitario. Marcando ya 10 minutos a las 14:30 del 23 de Agosto

1.2 Objetivos

Enmarcamos el proyecto en el área de digitalización de la información y de Servicios 4.0, un concepto tanto nuevo, acuñado el año pasado, 2018, por empresas pertenecientes al sector de servicios en España, como antiguo, ya que es la aplicación de los conceptos de la Industria 4.0, concepto que apareció en el año 2011 durante el salón de la tecnología industrial y que indica el cambio a la cuarta revolución industrial.

El objetivo principal del proyecto es, el estudio y desarrollo de un posible sistema de información para su uso en hospitales. Este sistema, al tratar con información, siendo gran parte de ella sensible y de salud, está regulada por el Reglamento General de Protección de Datos y la Ley Organica de Protección de Datos, ambas Europeas. Por lo tanto, los datos que se introduzcan a través de la aplicación o de cualquier otro método, debe gozar de un consentimiento y de forma explícita y recogida por escrito, por ello, se estblece un registro en el aplicativo de forma presencial en el hospital para poder optar a una cuenta donde almacenar los datos, si se utilizara la parte de aplicativo sobre el servicio de urgencias de forma anónima, no se almacenarían los datos de la aplicación de forma que se tratarán únicamente los datos recogidos en el hospital. Entre otras normas, se recoje la permanencia de los datos y la custodia clínica, por al menos 5 años, es decir, que los datos no deben cancelarse, sino bloquearse, facilitándose los derechos ARCO, que son de acceso, rectificación, cancelación y oposición de la misma forma que el registro.

Además, debe ser un sistema que permita su facilidad de uso por parte de cualquier persona, así como ofrcer la misma utilidad para el tipo de persona que lo utilice, ya sea empleado, visitante o paciente del hospital. Para ello se deben estudiar las necesidades de cada uno de los usuarios, que serán:

- `paciente`, entre las posibilidades destaca la posibilidad de declarar una urgencia en cualquier momento y de forma sencilla, ver su estado actual, decidir las dietas del día siguiente si estuviera ingresado durante un periodo superior a un día y acceder a un modo de información de su intervención.

- `visitante`, si el paciente le permite compartir sus datos, podrá ver el estado en el que se encuentra, esto es, si ya ha sido operado, si está en espera de ser atendido en urgencias o cualquier otro estado posible, también podrá saber donde se encuentra el paciente dentro del hospital para facilitar su acceso hasta él, en los momentos permitidos bajo la dirección del hospital. En esta facilidad para moverse, se encuentra un mapa de ubicación con la cartelería que encuentre por el hospital, de

forma cómoda y contextual.

- `profesional`, que, a parte del acceso a la historia clínica que ya tiene por parte del hospital, podrá utilizar la aplicación para mostrar a sus pacientes a qué intervención será sometido y cómo, de forma mucho más sencilla mediante modelos 3D, la posibilidad de encontrar la posición del paciente en cualquier momento, por si fuera requerido de urgencia. La gestión del servicio de urgencias de una forma más rápida, recibiendo menos molestias de los visitantes y pacientes, junto con la generación de modelos de machine learning con los datos que se extraigan digitalmente de los pacientes.

Para llegar a completar estos objetivos, es necesario el uso de diferentes tecnologías disponibles actualmente de forma económica pero eficaz y fiable, así como servicios en la nube de alta disponibilidad y plataformas móviles. En nuestro caso Google Cloud Platform es la elegida para proveer estos servicios, tanto para los backend del aplicativo, como para la parte IoT, mediante IoT Core.

1.3 Alcance

El alcance del trabajo incorpora, principalmente, la creación de un servicio de gestión de urgencias, que deje provisto al servicio de urgencias del hospital de la facilidad para declarar emergencias en su servicio de forma sencilla desde cualquier lugar, por medio de un dispositivo con Android y en programación Java-Android para el API mínima 26, por otro lado, se añaden servicios de realidad aumentada al mismo aplicativo, creando librerías AAR, que pueden acoplarse a la aplicación principal, en Unity en su version 2018.3, que utiliza el lenguaje C#, junto con las librerías AR Core de Google y Vuforia Engine, de Vuforia.

Para que el sistema funcione, debe crearse un sistema de backends que asimile y reparta la información a los diferentes sumideros de información, esto es, que sea capaz de guardar la información y mostrarla cuando se solicita. Esta tarea se realiza mediante programación en javascript, y la versión 10 de nodejs. Además, tenemos como objetivo un sistema crítico, por lo que la información debe estar presente y disponible en todo momento, esto es conocido como cero downtime, para ello usaremos tecnología de computación en la nube, utilizando a Google Cloud Platform como proveedor de servicios. Dentro de estos servicios, se utilizan las Google Functions para que se ejecute el contenido únicamente cuando la funcionalidad es llamada, y no a todo momento, ahorrando costes en la implementación, siendo otro de los objetivos conseguir el menor coste posible junto con la mayor efectividad.

1.3.1 Prerrequisitos

Para la ejecución del proyecto, es necesaria una serie de prerrequisitos como base sobre los que empezar a trabajar. Estos prerrequisitos son:

- `NodeJS`, para el desarrollo de programas en JavaScript, es necesaria la instalación de nodejs y npm, para la compilación y obtención de paquetes node. Se utiliza la versión 10.16.3 de nodejs y 6.9.0 de npm, ambas LTS.

- `Android Development Kit`, para el desarrollo de programas en Android necesitamos tanto el NDK (Native Development Kit) como el SDK de Java, junto con la herramienta de gestión de tareas y repositorios Gradle y Maven, que nos permiten acceder a los distintos repositorios y librerías de terceros.

- `Integrated Development Enviroments`, para la facilidad en el desarrollo del proyecto utilizamos 4 entornos disintos, Unity para Vuforia y Google AR Core, Android Studio para Android, Visual Code para C# y Visual Studio Code para Javascript.
- `Suscripción Google Cloud Platform`, para poder utilizar la tecnología de computación en la nube de Google necesitamos una suscripción en el proveedor. Actualmente nos ofrece 300 dólares a gastar durante un año, por lo que no supone un gasto para el proyecto.
- `Computador`, necesita capacidad para la compilación de los anteriores programas, para la renderización y escultura de modelos 3D, es decir, con capacidad gráfica, y con Internet.
- `Blender`, para el modelo y renderizado de objetos 3D con un detalle elevado.
- `MagikaVoxel`, para crear modelos 3D con arte Voxel, que son dibujos cuya unidad de medida mínima son cubos.
- `Raspberry Pi 3B`, como dispositivo de Edge Computing para tareas de IoT y ejecución de NodeRed.
- `ESP8266`, para convertir dispositivos desconectados en dispositivos IoT, al ser módulos WiFi con capacidad de computación de independientes.
- `Sensores`, para obtener datos del entorno con los que proveer datos a los aplicativos.

1.3.2 Limitaciones

Debido a la naturaleza de un proyecto de fin de grado, aparecen limitaciones en cuanto al tiempo y a los recursos a dedicar. Por otro lado, al ser dedicado a hospitales, donde los procedimientos son en su mayor parte privados, no se utilizan los procedimientos reales del hospital por desconocimiento de los mismos. Para subsanar estas limitaciones se han tomado algunas decisiones que evitan un alto impacto sobre el proyecto, por tanto, se han reducido aspectos que son relevantes para un entorno productivo pero no para una PoC (Prueba de concepto), como pueden ser, entre otros, el sistema de login, estando ese falseado creando las consultas de usuario y contraseña en la misma aplicación, a diferencia de un entorno productivo donde hay una base de datos detrás. Para reducir más tiempos se ha optado, en algunos momentos, por usar algoritmos consolidados para embeber en la aplicación en vez de programar nuevos desde cero, siendo por otro lado el diseño y programación original. Por último, algunas de las funcionalidades que se han propuesto, pero que quedaban más lejos de los objetivos, no se ha programado la lógica, pero sí se ha tenido en cuenta el diseño sobre la interfaz, apareciendo donde podrían colocarse estas funcionalidades.

Por otro lado, para subsanar los recursos limitados, se ha optado por el uso de Google Cloud Platform, ya que ofrece 1 año de uso gratuito sin exceder 300 dólares, junto con una capa permanente de uso gratuito. Si el recurso monetario no fuera limitado, se recomienda el uso del proveedor Amazon Web Services, por la amplia documentación y gran cantidad de funcionalidades que ya ofrece, además de ser más user friendly.[2] También se han limitado características hardware como puede ser la cámara de reconocimiento de matrículas a una simple webcam, utilizando una raspberry como procesador, y cambiando el diseño inicial de streaming de imágenes constante por una toma de imágenes en el momento que ocurre un movimiento, para lo que se añade un sensor que detecta el mismo.

Por último, para subsanar los problemas de falta de datos respecto a los procedimientos oficiales que se realizan en un hospital, y más concretamente en el Principe de Asturias, se ha utilizado la experiencia como usuario del servicio de urgencias, desde lo que se ve desde fuera, así como de mi experiencia como

voluntario del Samur-PC del distrito 8 en la Unidad de Gestión de Redes Sociales y del distrito 21 en la Unidad de Gestión de Prevención Activa. Otros datos se han intentado extraer del contacto al ciudadano y prensa que ofrece el hospital, pero no se ha recibido nunca respuesta a ninguna pregunta.

1.4 Organización de la memoria

Esta memoria se organiza en seis capítulos principales, con sus correspondientes subdivisiones y un final que incluye diversos anexos relevantes en el desarrollo del trabajo.

Durante el primero se ha podido ubicar el proyecto en un marco contextual sobre cuál es el problema, cuáles son las soluciones y una vez establecidas, definir un alcance, dicho de otra forma, lo que se espera del proyecto una vez terminado. También da una amplia idea de cuáles son las diferentes tecnologías sobre las que se va a poder leer en el resto de la memoria, de las que se han hecho uso y de los retos que se han presentado en la ejecución del mismo.

A través del segundo capítulo se desarrollarán a fondo las ideas y retos que han surgido del proyecto, las comparaciones entre unas y otras tecnologías que han hecho descartar unas para implementar otras, y una vez elegidas estas, un desarrollo más extenso sobre las cualidades que otorgan al proyecto, ya que añadirán ventajas y desventajas frente a otras implementaciones, incluyendo un apartado para la tecnología en la nube, los distintos proveedores y como ha cambiado el modo de trabajo de empresas y servicios gracias a esta tecnología. Además, podremos entender mejor la idea de IoT, Internet of Things o Internet de las Cosas, un concepto que se ha puesto de moda últimamente y que forma parte de nuestra vida mucho más de lo que pensamos.

Siguiendo el orden, alcanzaremos el tercer capítulo, donde pondremos en marcha todos los conocimientos y estudios expuestos en el anterior. Es un capítulo dedicado al desarrollo práctico de la idea, mostrando los aplicativos desarrollados, el flujo de trabajo y las funcionalidades que proveemos tanto a médicos como pacientes y visitantes.

Pasando al cuarto capítulo haremos una valoración de resultados, en la que de forma más o menos empírica, mediante observación de operaciones realizadas en el servicio de urgencias como serán el registro en el hospital, la localización de un paciente junto con otros que sean posibles sin ser una molestia para el servicio y la simulación del mismo mediante el aplicativo desarrollado.

Como buen trabajo de fin de grado, debe ofrecer unas conclusiones y unas líneas de investigación futura, que demuestran que el aplicativo es ampliable en el tiempo a ofrecer nuevas funcionalidades y no se limita a lo estudiado, sino que es un campo extenso de investigación y desarrollo, podremos encontrarlo en el quinto capítulo.

Para terminar nos toparemos con el sexto y último capítulo, que es algo más orientado a gestión del proyecto que a su desarrollo, mostrando el mal llamado presupuesto, mejor llamado coste del mismo, ya que se establece a posteriori de la compra y ejecución del mismo.

También se añaden algunos anexos al trabajo que tocan y ofrecen aspectos relevantes del proyecto que se enmarcan más como entregables del proyecto que como parte de la memoria.

Capítulo 2

Estudio teórico

Siempre que una teoría aparece como la única posible, tomarla a rajatabla es señal de que ni se ha entendido la teoría ni el problema que se pretende resolver.

Karl Raimund Popper

2.1 Introducción

Durante el desarrollo de este capítulo se podrán comprender mejor los conceptos teóricos que van ligados al proyecto, desde como funciona el servicio de urgencias de un hospital al funcionamiento de la tecnología de realidad aumentada, pasando por el estudio de sensores, Internet de las Cosas, Cloud computing y otros, dividiendo el capítulo en 6 apartados que profundizan en estos temas.

2.2 El servicio de urgencias

La OMS define una urgencia como «la aparición fortuita, en cualquier lugar o actividad, de un problema de salud de causa diversa y gravedad variable, que genera la conciencia de una necesidad inminente de atención por parte del sujeto que la sufre o de su familia» [3] y por tanto, en el servicio de urgencias de un hospital, nos darán la atención que necesitemos conforme a esa necesidad. En España se atienden una media de 800.000 personas por día en este servicio (hospitalario y extrahospitalario), un servicio de vital importancia para la sociedad, puesto que la vida de mucha gente depende de que su mecanismo funcione perfectamente, disminuyendo los tiempos de respuesta cada vez más, con un único objetivo principal, salvar vidas. [4]

2.2.1 Qué es el servicio de urgencias. Servicio y características

A grandes rasgos, el servicio de urgencias es el encargado de atender las urgencias y emergencias hospitalarias y extrahospitalarias que suceden en nuestro país. La historia de este servicio surge, de manera primitiva y básica, inicialmente en Europa durante 1789 en las guerras de la época donde se asistía a heridos de guerra mediante lo que ahora serían ambulancias rudimentarias a caballo. Como todo, fue especializandose y mejorando conforme avanzaban los años, comenzando por dejar de ser un servicio exclusivamente militar, muy usado en las guerras mundiales por ejemplo, para pasar a ser también civil,

siendo Sir Robert Jones durante 1914 el primero en establecer un servicio civil de urgencias, pero destinado a fracturas. Por supuesto siguió evolucionando hasta convertirse en lo que es ahora, una organización de profesionales sanitarios que ofrece asistencia multidisciplinar, ubicada en un área específica del hospital que cumple con unos requisitos funcionales, estructurales y organizativos, que garantizan condiciones de seguridad, calidad y eficiencia para atender a la urgencia y la emergencia. Este concepto se amplia con el concepto de urgencia extrahospitalaria, donde son los profesionales los que acuden en la ayuda del sujeto para ofrecerle una atención primaria o avanzada según las necesidades del mismo.

Entre las características del servicio, aparece una guía de servicios y capacidades mínimas de la que todo servicio de urgencias debe disponer. [5] Los más relevantes a nuestro proyecto aparecen en el primer nivel, mientras que su relación con el proyecto se explica en el segundo, estos son:

- Debe estar físicamente diferenciada del resto de áreas asistenciales del hospital.
  - Al estar diferenciada físicamente, siempre tendremos un acceso directo a urgencias, por lo que nuestro flujo de trabajo siempre empezará por el mismo sitio.
- Su programa funcional debe tener en cuenta las necesidades de todos los usuarios de la unidad, esto es, debe ser accesible por el resto de unidades del hospital, por la seguirdad y por los requerimientos de los sistemas de información.
  - Al deber sitúar su funcionalidad en un lugar donde se puedan desplegar sistemas de información, no tendremos problemas a la hora de generar uno nuevo, pero también implica que el sistema de información deberá tener en cuenta a todos los usuarios.
- En concepto de recursos humanos, se recomienda el cálculo de tiempo medio por nivel de gravedad de cada usuario.
  - Este cálculo es dificil de realizar desde que el paciente llega a emergencias hasta que va pasando por todas sus fases. Sin embargo, con el aplicativo automatizado y con control de eventos como la ubicación en la que se encuentra, podemos calcular con precisión cuanto tiempo esta en cada una de las zonas de urgencias, con que profesional, etc. . . y además aplicar modelos de machine learning sobre los datos.
- Puede definirse como una unidad intermedia que presta servicios desde la estabilización del paciente hasta su ingreso en el hospital o como un servicio final para pacientes que reciben el alta en el mismo servicio.
  - Por esto deberemos tener en cuenta que la trazabilidad del paciente de urgencias puede seguirse hasta ser ingresado en el hospital y no simplemente al servicio de urgencias, debiendo llevar la información hasta el ingreso y solo liberarse en el posterior alta.
- Los pacientes son atendidos por el tipo de necesidad asistencial que demande, no por su orden de llegada y no por los recursos físicos de los que disponga la unidad.
  - Por lo que un sistema de información parecido al que realiza las ambulancias pero sin necesidad de ellas ayuda a establecer el sistema de triaje más fácilmente, otorganzo al sistema de organización una capacidad de visión más amplia respecto a los pacientes que le lleguen.

En un hospital se debe ofrecer una cartera de servicios mínima para contar con una unidad de emergencias. Entre los que se encuentra la medicina aguda, el nuvel 2 de asistencia hospitalaria, una unidad

coronaria no invasiva, un laboratorio básico de análisis clínicos y una unidad de diagnostico por imagen en el propio hospital, y ayudado por otros hospitales, o añadido al propio hospital si es posible, un servicio de cirugía urgente, traumatología, pediatría, obstetricia y ginecología, salud mental, radiología intervencionista y cirugía especializada.

Esta cartera mínima podrá verse ampliada por el volumen de urgencias que atiende la unidad así como las características inherentes al hospital.

2.2.2 Organización del servicio de urgencias. Dependencias y roles

Como cualquier servicio organizado en niveles y por los diferentes roles de empleado que hay, la unidad de urgencias debe estar divida y gestionada por grupos más pequeños según titulación y funciones. [6]

Los responsables de la unidad, que deberá tener al menos 5 años de experiencia en medicina de urgencias y emergencias. Este debe existir las 24 horas del día y ser conocido en todo momento quien es, como máximo responsable.

Los médicos, siendo estos facultativos que prestan servicio en la unidad estando directamente en la unidad contratados o especialistas contratados en el hospital que se encuentran de guardia y acuden al servicio cuando se les requiere. Su función es evaluar, estabilizar y atender a los pacientes que están o pueden estar sufriendo una enfermedad o lesión grave o urgentes. Estos médicos deben tener formación específica en urgencias.

Responsables de enfermería, que por debajo de los reponsables de unidad disponen de las capacidades y mando para organizar a los enfermeros que prestan servicios en la unidad.

Los propios enfermeros, cuya funcion es evaluar, dar prioridad a la atención de pacientes y proporcionarles los cuidados de enfermería necesarios, identificando las necesidades de pacientes y familiares.

Los auxiliares de enfermería, celadores y personal auxiliar administrativos, con roles derivados de las funciones anteriores.

En materia de diferenciación de estancias de la unidad de urgencias, encontramos diferentes entornos que condicionan el funcionamiento efectivo del hospital:

- Entorno exterior. Debe estar protegido frente a las inclemencias del tiempo, facilitar la descarga de pacientes y con una buena iluminación e indicativo de entrada, con espacio para al menos 2 ambulancias en paralelo y con la posibilidad de realizar el giro sin útilizar la marcha atrás. Se recomienda disponer de un aparcamiento exterior si cumple un volumen amplio de asistencias así como lugares donde poder instalar un área de descontaminación fija o portatil y entradas diferenciadas de pacientes urgentes y sitúaciones de emergencia. Si el hospital dispone de infraestructura para el transporte aéreo, este tendrá una fácil comunicación con la unidad de urgencias.

- Entrada. Las entradas deben estar protegidas para facilitar la entrada del paciente en la camilla, con puertas automáticas y cortavientos en las mismas para evitar las corrientes de aire en el interior de la unidad. Este vestíbulo también debe de disponer de alfombra que recoja la suciedad que pueda introducirse del exterior y las paredes deben ser acristaladas para permitir la visión de la entrada desde el mostrador de recepción.

- Recepción y admisión. Siendo la parte de la unidad que atiende en primer lugar a los pacientes que acuden a la unidad. Desde el mostrador se controla la entrada de pacientes y contará con el equipamiento necesario para desarrollar las funciones administrativas que se requieran eficazmente.

- Box de triaje. Contiguo a la admisión, y la sala de espera, y disponiendo de acceso a las salas especializadas, son pequeños despachos médicos destinados a la clasificación de la gravedad de cada paciente, determinando el recurso más adecuado a su sitúación y el tiempo estimado de atención. Para realizar este triaje se realiza una valoración preliminar del motivo de la consulta junto con otros sintomas evidentes a simple vista.

- Sala de espera general. En este espacio se encontrarán los pacientes que aún no han sido atendidos junto con los familiares que le acompañen. Por su naturaleza debe ser amplia y disponer de indicativos de llamadas a las salas de triaje. Por comodidad de los paciente y familiares, se recomienda disponer de máquinas expendedoras de alimentos y bebidas envasadas, fuentes y aseos públicos y dispositivos de ayuda al movimiento como sillas de ruedas o muletas.

- Zona de consultas. Siendo estos los lugares para pacientes que durante el triaje se les ha asignado un nivel de atención bajo, facilitando la privacidad del paciente y con mobiliario suficiente para que este sea atendido como camillas de exploración, tomas de oxígeno y vacío, intrumental de exploración...

- Zona de reanimación. Donde entran los pacientes críticos en parada o politraumatismos, correspondientes al nivel 1 de asistencia, estos son, los que no pueden demorarse en su atención por suponer un riesgo para su vida. Estos lugares disponen de instrumentación especial para reanimar y sacar de zonas críticas a los pacientes.

- Boxes de urgencias y salas de observación. Siendo estos para atención y cuidados de un paciente que conforme al resultado de triaje no es enviado a la consulta porque requiere de un nivel de atención mayor. Entre los boxes aparecen los de tipo polivalente, de traumatismos e incluso de psiquiatría. En el caso especial del box de trauma, normalmente esta acompañado de una sala de yesos. El resto de boxes también estarán cerca de una sala de curas y suturas.

- Otras zonas. También existen gran cantidad de zonas como pueden ser de servicios especializados, como radiología, quirófanos y otras orientadas a los empleados del hospital, como salas de espera de personal, puestos de control de enfermería, almacenes, aseos y despachos de responsables y farmacias.

Figura 2.1: Plano servicio de urgenicas del hospital de Jaén.

2.2.3 Funcionamiento general del servicio de urgencias

Para terminar con esta sección estudiamos el flujo de pacientes que se da en un hospital cuando se recibe una urgencia. La unidad de urgencias debe disponer de protocolos, procedimientos específicos y guías de práctica clínica. De forma general, el proceso de atención al paciente es el siguiente.

El paciente accede al servicio de urgencias del hospital de forma directa, por sus propios medios, derivado dede atención primaria por la imposibilidad de ser atendido en este servicio o mediante un traslado en ambulancia hasta el mismo hospital. Una vez llegan al hospital, pasarán por un mostrador de recepción en el que, normalmente a través de únicamente la tarjeta sanitaria se procederá a la adminisión del paciente y a la entrega de una documentación interna al hospital, que le otorga un número de identificación y espera respecto al resto de pacientes.

Una vez admitido, es turno del triaje o triage [7], que basándose en la información remitida por admisión, irá llamando a los diferentes pacientes para su evaluación profesional y asignación de un nivel de asistencia, que les otorgará una prioridad u otra respecto al resto, dependiendo del nivel de gravedad que muestren además de asignar a cada paciente un recurso correspondiente y un tiempo de espera máximo, por lo que el paciente entrará en cuanto el recurso quede libre o cuando empiece a peligrar su condición con el aumento de espera.

De este triaje, se accede a uno de los diferentes servicios que ofrece urgencias, pudiendo ser mandado a una consulta, normalmente destinada a los casos de más baja gravedad, normalmente niveles 5 y 4, recordando que niveles con número más bajo representan mayor gravedad. En esta consulta se realizan los servicios que podrían darse en un consultorio o servicio de atención primaria, es decir, servicios que realmente no son urgentes pero a veces son necesarios por limitaciones de horarios de zonas sin servicio médico local o fin de semana, donde permanecen cerrados, de toda formas, se debe tener en cuenta que tras una consulta se puede reasignar el nivel de triaje a uno superior y que el mismo paciente resulte en una complejidad mayor de la que fue valorada inicialmente y derivado a otros servicios de urgenicas. Los otros dos servicios de exploración y observación se reservan a niveles más altos de gravedad, el 2 y el 3, donde el paciente requiere una atención mucho más rápida o de técnicas que no pueden aplicarse en consulta. Durante la estancia en estos niveles, se realizarán diferentes tipos e pruebas de diagnóstico y laboratorio, curas o tratamientos de urgencia, que posteriormente derivarán en un tratamiento final o alta.

Number	Name	Colour	Max time
1	Immediate resuscitation	Red	0 minutes
2	Very urgent	Orange	10 minutes
3	Urgent	Yellow	60 minutes
4	Standard	Green	120 minutes
5	Non-urgent	Blue	240 minutes

Figura 2.2: Niveles de gravedad de urgencia según Triage Manchester.

Una vez valorado y explorado el paciente, este puede necesitar pasar por el servicio de cirujía, muy normal en caso de traumas y tras esto ser ingresados en los servicios de observación y cudiados del hospital como puede ser la Unidad de Cuidados Intensivos antes de recibir el alta. Por otro lado, puede ser que el servicio de urgencias al que se esta acudiendo no pueda curbir el servicio que requiere, como puede ser un hospital sin ala de maternidad en una urgencia por obstetricia, que terminará en una derivación a otro servicio de urgencias, normalmente usando el traslado por ambulancia. El último camino posible es que el paciente quede estabilizado y recuoperado dentro de unos límites aceptables para el alta del

mismo, aunque quede derivado a seguir unas consultas con la atención primaria. En caso de paciente irrecuperable, se realizará un alta por éxitus Letalis, siendo este el peor de los casos.

Por supuesto, los pacientes que entren directamente en el servicio de urgencias en estado de parada cardiorespiratoria o la haya sufrido antes o durante el traslado al hospital, irán directamente a la sala de recuperaciones y cuando se considere oportuno, pasarán a observación y seguirán el flujo normal de urgencias con el nivel más alto de gravedad.

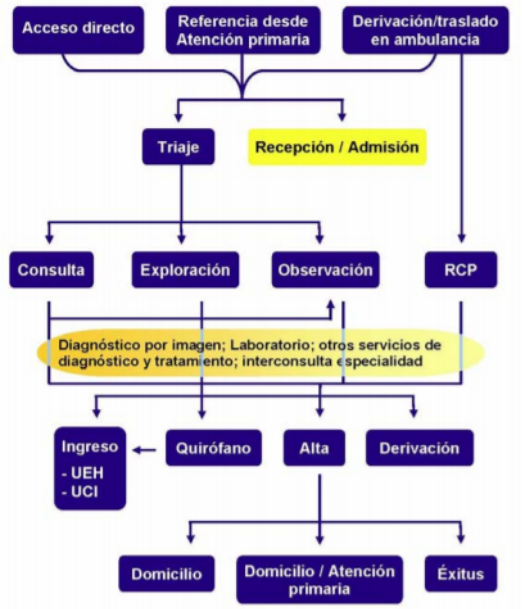


Figura 2.3: Flujo de atención a pacientes de urgencias.

2.3 El Internet de las Cosas

El Internet de las Cosas, del anglicismo Internet of Things y conocido bajo el acrónimo IoT se está expandiendo cada vez más rápido por prácticamente todas las áreas de negocio, desde el profesional técnico que se dedica a desarrollar estos dispositivos hasta la secreataría de una empresa que útiliza un dispositivo para establecer agendas o el expendedor de números de espera automáticos que encontramos en los supermercados. Esta expansión del negocio IoT se debe a la cantidad de beneficios que está otorgando a las empresas desplegar estos dispositivos ya que ofrecen mucha información e impacto a un precio hardware normalmente mínimo. Durante esta sección se explicará que es el IoT, para poder entender como funciona y por qué está ganando este poder.

2.3.1 Historia y Concepto

Comenzamos descubriendo cuándo apareció este término en la historia. Normalmente todo este tipo de tecnología es desarrollada inicialmente en entornos industriales, y este concepto no iba a ser menos. Un concepto que nació en los 70, con la idea de interconectar dispositivos, pero que no fue hasta 1999 cuando este concepto se especializó y nombró como IoT.

Fue Kevin Ashton [8], trabajador de una empresa de montaje en cadena optimizando los procesos, el que por primera vez útilizó el término. Ashton comenzó a llamar la atención de sus superiores con la actualemente de sobra conocida tecnología RFID, parte importante de nuestro proyecto, exponiéndola

bajo una presentación llamada «Internet of Things». Esta presentación llamó la atención de los directores de la cadena, pero paso desapercibido para el resto del mundo hasta el 2010.

A partir de 2010 varias empresas empiezan a querer conectar el mundo de Internet con el mundo real, muestra de ello aparece Google StreetView, actualmente implementado en Google Maps, ofreciendo a través del buscador y un mapa, imagenes del mundo real, también China introduce un plan nacional de estrategia para el desarrollo e implementación de dispositivos IoT y en Europa el concepto IoT cobra más fuerza que nunca, con charlas específicas sobre ello e incluso apareciendo las primeras estimaciones sobre el mercado que supondría el desarrollo.

Figura 2.4: Búsquedas en Google del término IoT desde 2004 hasta hoy. El valor 100 corresponde al pico de búsquedas mundiales.

Actualmente un amplio porcentaje de empresas implementa o tiene implementados sistemas de IoT en sus lugares de trabajo. En una encuesta realizada por Fujitsu Limited JP el 99 % de los preguntados respondió que ya habían implementado dispositivos IoT o tenían un plan de despliegue a ejecutar, por otro lado, el 47 % de los encuestados implementaba estos dispositivos por seguridad y el 45 % para mejorar y renovar sus infraestructuras. A nivel de ocio, fuera del trabajo, muchas son las casas que ya cuentan con dispositivos Nest de Google para automatizar o acceder a la domótica de su casa desde cualquier lugar, obtener información con los Home Assistant o limpiar la casa mientras no están dando ordenes desde su móvil al robot de limpieza.

Por lo tanto, podemos decir que la tecnología IoT ha madurado lo suficiente y se encuentra ya lejos de las fases de testeo o de marketing, forma parte de nuestro día a día mucho más de lo que creemos, pero no nos damos cuenta. Actualmente los dominios del IoT son la industria con un 40 % y la sanidad con un 30 %, dejando la seguridad y el pequeño comercio en un 8 % cada uno. [9]

Pero conozcamos más qué es el Internet de las Cosas. El concepto se define, a grandes rasgos, como «conectar todos los dispositivos a Internet y permitirles comunicarse entre ellos». Por tanto, IoT no es más que una red gigante de dispositivos interconectados a través de Internet, que además son capaces de recoger y compartir los datos que recolectan, ya sea sobre su uso o sobre el entorno en el que se encuentran, con el usuario, con otros dispositivos con los que interopera o con grandes bancos de datos para el estudio posterior de los mismos. Tras el estudio de estos datos, se consigue mejorar estos dispositivos para que cada vez sea más fácil y disimulado su uso y tras varias iteraciones un producto recibe varias versiones y revisiones hardware que lo convierten en un dispositivo cada vez mas dependiente de la interación humana, consiguiendo así que realice los cambios de forma automática como lo haría una persona.

Figura 2.5: El mundo está interconectado por medio de dispositivos IoT. Las smarticities estarán presentes en este proyecto.

Asimilándolo como un ejemplo, tenemos un sensor de temperatura en casa, al igual que un sensor de geolocalización en el transporte. Con estas interconexiones, el controlador de ambos sensores puede indicar que cuando el sensor de GPS detecte que estamos a 10 minutos de casa, y la temperatura está por debajo de 19 grados, actúe sobre la calefacción para que esta se encienda y obtengamos una temperatura más agradable al entrar en casa, sin que nosotros hayamos realizado ninguna acción salvo la de conducir a nuestro domicilio. Por tanto, la tecnología IoT nos reporta gran cantidad de beneficios con solo interconectarlos entre ellos y con Internet y dejando que ciertos datos, por supuesto este proyecto da por hecho la buena fe del uso de los datos que compartimos, la anonimidad y el respeto por la privacidad del usuario, sean compartidos a bases de datos de estudio que mejoren y actualicen los sistemas inteligentes, entre las ventajas[10], aparecen algunas como:

- útilización eficiente de recursos. Puesto que con la recogida de datos de las cadenas de producción se pueden detectar donde se encuentran las fugas de energía, de material sin útilizar que va a la basura o, por ejemplo, del tiempo medio de vida que ofrecerá una pinza de soldadura y de qué rasgos eléctricos va a mostrar antes de funcionar de forma irregular, evitando que las soldaduras sean defectuosas y deba desecharse la pieza soldada.

- Datos de calidad, ya que se instalan estos dispositivos allí donde necesitamos información útilizando sensores específicos para recuperar un determinado dato, termómetros para temperatura, caudalímetros para corrientes de agua,... mapeando estas características del mundo real en datos digitales.

- Análisis de decisiones, ya que gracias al estudio de estos datos podemos saber hacia qué está derivando un sistema cuando realizamos un cambio y estos dispositivos nos devuelven la variación que ha causado en los datos ese cambio, permitiendo revertirlo si mejoran o aumentar los cambios si ofrecen mejora en el proceso.

- Mejor experiencia de usuario, pues no es lo mismo tener que ir a ajustar el termostato a la centralita que desbloquear el móvil y aumentar la temperatura o sin que se haya automatizado un aumento de la temperatura y ni siquiera debamos preocuparnos. A rasgos industriales no es lo mismo que un dispositivo te indique que hay fallo en un componente del armario eléctrico directamente sobre una interfaz web que tener que revisar todos los días y a todas horas los componentes de todos los armarios.

- Bajo precio y grandes prestaciones, normalmente estos dispositivos son pequeños y baratos, por lo que pueden acoplarse en cualquier lugar sin ser invasivos, sin embargo ofrecen una capacidad de información muy superior a otros dispositivos y sistemas mayores.

Con todas este tipo de ventajas se entiende que su uso se este extendiendo en las empresas y servicios del mundo como un claro ejemplo de digitalización del entorno y de mejora de procesos mediante el análisis de datos.

2.3.2 Aplicaciones

Las aplicaciones del IoT son incontables, debido a que cada vez mejoran más las conexiones inalámbricas ofreciendo mayores velocidades y capacidades, a la vez que los sensores útilizados son más sensibles y ajustados mientras que los dispositivos de computación aumentan o grandes dispositivos de computación se ponen al alcance de todos mediante el cloud computing para el análisis de datos mediante técnicas de machine learning. Pero aclaremos, ¿qué puede ofrecer el IoT a nuestro proyecto?, a lo que la respuesta es, básicamente todos los datos que necesitamos para automatizar procesos de urgencias pasan por estos sensores.

Imaginemos un dispositivo inteligente como unas cámaras de tráfico, colocadas en varios puntos de acceso al hospital puede realizar un reconocimiento de vehículos y matrículas. Por cada reconocimiento exitoso se puede realizar una consulta a una base de datos en Internet, que posea diferentes matrículas que tienen permitido un paso prioritario, por tanto, al detectar una de estas mátriculas o un vehículo de tipo ambulancia, un controlador de tráfico automático cambiará el color de los semáforos para dar un paso prioritario a estos, dándoles una salida rápida y segura hacia su destino, sobre todo en intersecciones o lugares donde el tráfico está congestionado.

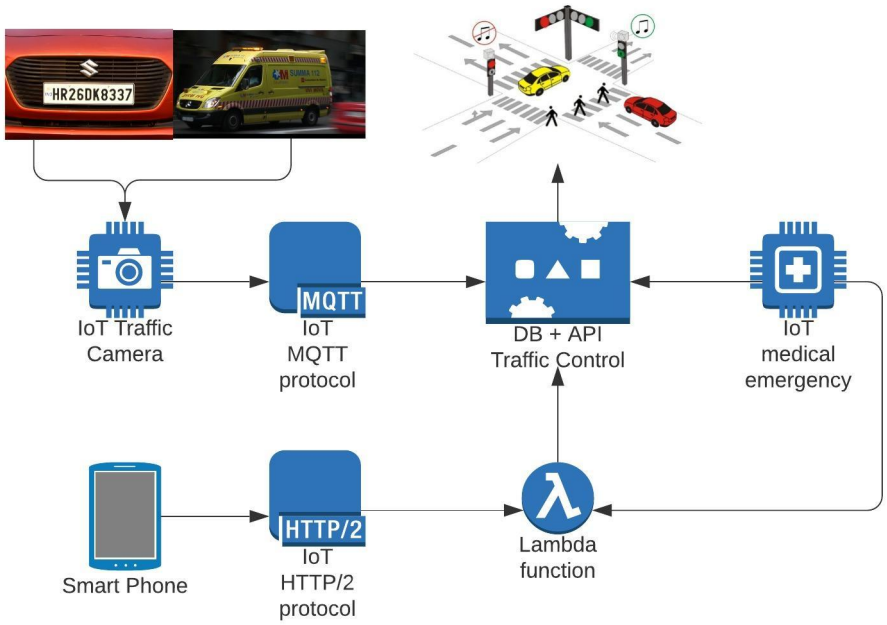


Figura 2.6: Arquitectura de control de tráfico autónomo para sitúaciones de urgencia médica.

Tras esto, contemplemos otro sistema con más componentes, una antena RFID, un tag RFID y un dispositivo de control. Colocando diversas antenas a lo largo de un edificio podremos detectar cuándo

un paciente que lleve en la pulsera de identificación un tag RFID, dónde se encuentra este paciente en cada momento, recogiendo información de su localización podemos proveer a familiares y médicos de su ubicación en caso de necesidad, así como ver el camino que sigue un paciente desde que ingresa a urgencias hasta que recibe el alta, pudiendo crear mapas de trazabilidad y mejorar los procesos actuales, así como para evitar molestias por parte de los familiares al personal sobre el lugar donde se encuentran los familiares. También podremos recoger tiempos de estancia en cada lugar mediante la generación de eventos en un timeseries para luego generar métricas y modelos de machine learning sobre esos datos que nos permitan saber dónde se necesitan más recursos y dónde menos por ejemplo.

Añadiendo sensores podemos incrementar la cantidad de datos que se obtienen del lugar, podríamos medir humedad y temperatura que se alcanzan en las salas de espera por número de personas en ella, variando automáticamente así los sistemas de refrigeración o calefacción, el porcentaje de ocupación del hospital para generar avisos de falta o liberación de recursos más rapidamente, y muchos otros, ya que las posibilidades son prácticamente infinitas.

2.3.3 El protocolo MQTT

Con la aparición del IoT se requierió de una nueva forma de establecer comunicacaciones, un estándar para los dispositivos que quisieran ser parte de este nuevo mundo. Para ello se estudiaron los protocolos y corrientes existentes hasta ahora y fue el protocolo MQTT, Messaging Queue Telemetry Transport, el elegido para representar el estándar de comunicaciones de IoT.

Este protocolo se creo en 1999 por Andy Stanford y Arlen Nipper, en su necesidad de conectar las tuberías de transporte de aceite contra redes satelitales. Para esto necesitaban un protocolo ligero y de un bajo consumo de ancho de banda, con soporte para varios QoS (Quality of Service) y por supuesto, independiente del tipo de dato telemétrico que se quisiera enviar. Casualmente, estos conceptos son los mismos que se buscan en las conexiones entre dispositivos IoT, por lo tanto se ha popularizado su uso para conectarlos siendo el preferido por la mayoría de las plataformas IoT actuales. [11]

Este protocolo únicamente dispone de 5 paquetes, diseñados para conectar dispositivos punto a punto, es decir, no funciona como middleware. Este API limitado consiste en algoritmos de publicación-suscripción, de forma que el publicador envía un mensaje a un tema central que puede tener varios clientes suscritos. Gracias a este modo de comunicación, cada uno puede funcionar sin necesidad de saber si el otro está presente, de forma autónoma. Por tanto en este protocolo de comunicación, se diferencian:

- Cliente. Siendo este cualquier dispositivo que esté conectado al broker de mensajes para consumir los mensajes del mismo o para enviarle mensajes, esto es, tanto publicadores como subscritos son clientes. Además existen dos modos de conexión, uno permanente y otro transitorio.
- Broker. Este es el software que recibe todos los mensajes de los publicadores y los reenvia a todos los subscriptores, además maneja las conexiones permantentes. Por otro lado forma el cuello de botella de la aplicación, por lo que se recomienda que sea escalable y replicado.
- Topic. Conocemos a los topic como los diferentes temas a los que un cliente se puede conectar, formando cada topic un endpoint del API diferente. Cumple las normas de codificación UTF-8, soporte de wildcards y barras inclinadas para las jerarquías.
- Conexión. Puede usarse de dos formas, a través de TCP/IP por el puerto reservado 1883 o por TLS/SSL a través del puerto 8883, donde podremos cargar certificados para generar seguridad en nuestra aplicación a través de encriptados y algoritmos obteniendo certificados de tipo x.509.

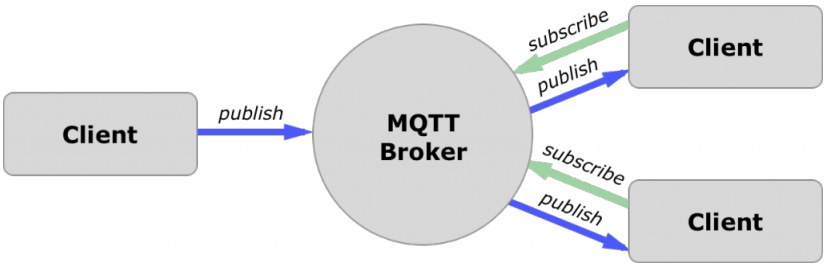


Figura 2.7: Arquitectura del protocolo MQTT.

Para este último punto en el que hablamos de certificados, posteriormente vamos a explicar en el apartado de seguridad como se generan y obtienen esos certificados x.509

Como punto importante a remarcar, algunos dispositivos también son capaces de soportar el protocolo de comunicaciones HTTP que facilitan las conexiones contra algunos de nuestros desarrollos. El conocido protocolo basado en texto está presente de forma global, por lo que muchos fabricantes prefieren crear procesadores que también sean compatibles con el crafteo de paquetes HTTP para realizar conexiones persistentes, grandes tranmisiones de datos que pueden ser mandado como chunks o bloques fragmentados en vez de las limitaciones de 1 mensaje de MQTT o para conexiones que requieren de una respuesta por parte del servidor.

2.3.4 Seguridad

Si tiene una desventaja el IoT es la seguridad, pues posiblemente sea el punto de todas sus características donde más flojea, por dos motivos principales, lo que más usan los dispositivos de IoT es el envío de datos de todo tipo, por lo cual es objetivo de robos y por otro, actualmente seguimos sin tener una cultura en seguridad implementada en niveles ni personales ni empresariales a no ser que sean empresas con gran cultura tecnológica como multinacionales, donde aún así el punto más débil de la cadena sigue siendo el eslabón humano. Dicho esto, debemos tener especial cuidado a la hora de establecer conexiones con dispositivos IoT, por lo que siempre se recomienda, aunque debería ser obligado en entornos productivos, realizar conexiones seguras mediante cifrados, en este caso usando certificados x.509, que son los soportados por MQTT.

Un certificado x.509 es aquel certificado que se ha creado siguiendo el estándar establecido por la UIT-T para certificados de claves públicas y validación. Para ello, el sistema x.509 cuenta con diversas partes para su obtención.

Una autoridad de certificación, que nos otorgue un certificado raíz para avalarnos en las comunicaciones con el resto del mundo. Este certificado forma la clave pública de la compañia y pueden ser tanto certificados autofirmados, donde nosotros mismos somos la certificación raíz, útil cuando los proyectos son realizados para un cliente que nos conoce y permite instalar nuestros certificados en sus dispositivos, y certificados firmados por la AC, que forma el tercero de confianza como dijimos antes. Estos certificados, tanto autofirmados como proporcionados por la AC tienen una estructura igual que incluye desde metadatos del certificado, como son numeros de versión, serie, emisor, etc. . . como información del sujeto al que representan, con su nombre, localización y contacto. Finalmente añaden las diferentes firmas y claves públicas del certificado de forma que sean únicas, esto es, no ofrecen colisiones, y están certificadas por el AC o uno mismo, mediante la firma digital. Estos certificados pueden pasar por diversos otros módulos o entidades intermedias, que se firman entre ellas para generar un certificado cadena, siendo este la suma del certificado raiz con el certificado de expedido por una autoridad intermedia, o, de nuevo, autofirmado.

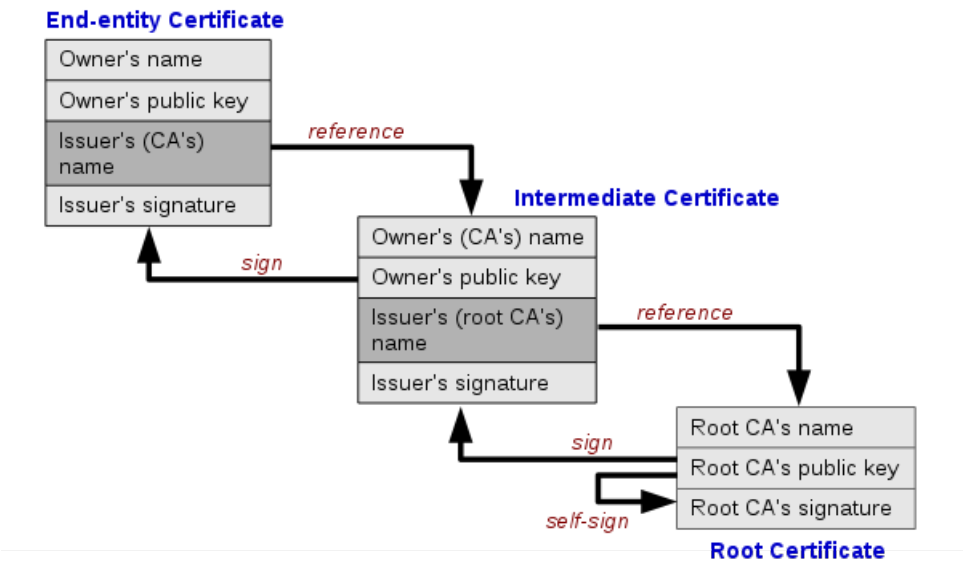


Figura 2.8: Flow de firma y referencia de certificados x509.

A partir de aquí ya tenemos la parte de certificación para el servidor, pero ¿qué ocurre con los dispositivos IoT?. Para esto lo que hacemos es generar certificados nuevos, autofirmados o no, con los datos del dispositivo, como pudiera ser el nombre que le queremos dar y a continuación firmamos y ciframos este dispositivo con el certificado raíz. Esto provoca que solo los dispositivos autenticados con un certificado que haya sido encriptado por la autoridad raíz sean autenticables al sistema, y por tanto se acepten sus paquetes para realizar acciones sobre otros actuadores dependiendo de la carga útil del mensaje. Por tanto, debemos generar 2 certificados de diferente tipo, un SSL para cliente y otro para servidor. A continuación podemos ver un ejemplo del archivo PEM generado por un certificado x.509 autofirmado de tipo raíz. [12]

-----BEGIN CERTIFICATE-----
MIIFOjCCAyKgAwIBAgIJANdmIg8qHXusMA0GCSqGSIb3DQEBCwUAMCoxKDAmBgNV
BAMMH0F6dXJlIElvVCBIdWIgQ0EgQ2VydCBUZXN0IE9ubHkwHhcNMTkwODEzMTUz
ODMwWhcNMTkwOTEyMTUzODMwWjAqMSgwJgYDVQQDDB9BenVyZSBJb1QgSHViIENB
IENlcnQgVGVzdCBPbmx5MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA
o8KyJ5tmj+K4ImYeWD6PkwNC0g2pGjVMIh6uJsxGbEkIz2w/xeXImwErcjxhSpZV
9TpqK5LHsUc73m7OZdIIHMw1K0/HInLc2GXTepu0opABXgaoRYH/u8SOEfAFzzX5
sVXSKP+7V2qNd4/ppk7ZV9EUY3mkQRl7qGvQZNnlRZCF93YKxqKEAvwUOB7ixmly
HLUKrpuvKO/BmL1080Qrnz4naoY4mz1nkGGbmNWhPn/XQ2P/2gzQQwPx0ps3viwV
cP75hQNA3sTX9tLCm+/ADzIwA9qAFULhBp8pqiJ5pJbFm55ljXSFv7if7ZDYraUM
8ActHrOw7NfHYMz/LIgHqHLOr/PlcIdc/nmul2sGFyqCqr9TJkHUNy2LK9selRfr
4nzCXoBuyI5Hhyilyw7d7KyXCsuLy0jQlY+k8hQC9mCh8pT0hqLhCAfY037EK+0n
Smfp012ZuF52hdtPNnFM1voqAeZzU9bo9R3oyH2ctqSHKrpZty/PNHtBKesiQ39P
BEljy8v7jrwJTK4Klm9gxPHcbJ64F8EI8AFysemTXaRNxr1TgqBv87kKhNQKaLx9
BQ5gruuSVMGe4ILhgOj1dcsQLF5EatEmILZOz88pIZtrecdA1NdrowCq8k9iCgZi
HSMafP6ONG+qzipF9WWr5ZiaLkgcb6yo7p/V7XZzTwsCAwEAAaNjMGEwHQYDVR0O
BBYEFJCmckesOZgKT2Ie02E8RjuG8F5VMB8GA1UdIwQYMBaAFJCmckesOZgKT2Ie
02E8RjuG8F5VMA8GA1UdEwEB/wQFMAMBAf8wDgYDVR0PAQH/BAQDAgGGMA0GCSqG
SIb3DQEBCwUAA4ICAQBoGkoikA7wCwgWgBbAVsIOgjm6usd2dkJnbpW1MARRi2Pq
J187hmOGnExjTni7woQ7pc120jwAOpo5KFitV6d42q9JS3vzfEZmx7EoyNrhfhAM
TiT9ibfrgLftGWx1AxjGIfNV9TE3eSILh/D52mvawG8KCNT62I2G04QcuKFs318W
tJCyjclJlOzfZJfW5ufDPcGgsunGCGtKHeukTQWgf1k7AWGOqT1zZhykGFFTUzcy
rj4Ct/lxB/jDkb5pFOcgmWFpvaR86hVpGdt10gwBLvsuUK0G00t3dG5hrnveDA1r
vjxSqVls92+02T+jiHVvf0clNGcIu/KFnJxFtnSe74X9vPub7DYN2Vc04kvoy1i4
HY6A6B6TyGM+aio9TgyTBt81XoywuXhDTvGOq7XMuFqnM3G3N3DNjjXw2TuF1khq
0NAX98mZ55eRIc87nsXHUJFZTBTR2KnnJIb12XKAZUwbBvWEas/5nKfcFXbZgRaI

fyhH+661Dv33A9L5V2prC4HxqNK74icQPLTKHFJc3/UljQZ1cf/UGA2rwl/xAQT7
lvmTQJTHhxNme/8MbEU+SYEi6wSqBx5KzArs6Uf98rcQ7r0ZEcFvCLCfegWTh/4L
z72sFxcjVd0Uhz5xt68aW9FLexnUCjuD0WF81m+0fIgFgwUT6hgP06guvU6N5g==
-----END CERTIFICATE-----

2.4 Sensores

Inicialmente solo los procesos industriales, la ciencia y la tecnología necesitaban generar y medir magnitudes con cierta precisión y fiabilidad, esto suponía una alta inversión en electrónica ya que cada sensor suponía un alto coste. Sin embargo, actualmente el precio de los sensores ha disminuido drásticamente, incluyedo gamas de uso doméstico que no requieren tantan unidades de precisión, por lo que la adquisición de estos datos se ha generalizado para grandes cantidades de dispositivos que antes no hubiera salido rentable su construcción. Además, estos van completamente de la mano de los dispositivos IoT, siendo los sentidos del mismo, los que son capaces realmente de recoger la información para que el dispositivo IoT la transmita, generando un gran dispositivo IoT.

2.4.1 Definición y características

Los sensores son la parte de instrumentación de un proceso que se útiliza para determinar el estado del mismo mediante la medición de una o varias magnitudes, que son captadas y trasnformadas a una señal eléctrica y esta última enviada a un destino que almacena y/o muestra el valor leído. Por tanto, un sensor es aquel dispositivo capaz de detectar acciones o estímulos externos, como cambios de temperatura, movimiento en su zona de recogida de datos u otros y mandar esta información a otro, para esto realizan una conversión desde una magnitud física o química a una señal eléctrica que otro sistema puede entender y procesar.

Distinguimos en el sensor el elemento activo o transductor de un elemento pasivo llamado acondicionador de señal, la unión de ambos crean los sensores inteligentes, pero todos deben ser calibrados para ofrecer mediciones fiables, por lo que parametros comunes y estáticos como intervalo, extensión y error deben ser documentados y tenidos en cuenta. [13]

- Intervalo. Es el conjunto de valores que puede tomar un transductor, sin detectar un valor por encima o por debajo de este rango como un estímulo.
- Error. Como toda medida, existen 2 valores, el medido y el verdadero, la diferencia entre ambos forma el error.
- Exactitud y Precisión. Esto es, cuál es la diferencia entre el valor real y el valor medido, tomando para ello la uma de todos los errores obtenidos junto con el error de exactitud de la calibración del propio transductor, mientras que la precisión es de forma estadística, la dispersión de los valores obtenidos de mediciones repetidas de una misma magnitud.
- Sensibilidad. Corresponde a la relación de información de salida por unidad de entrada, conocido en informática como I/O. Por ejemplo, cuando hablamos de ratones de ordenador, la cantidad de dpi que el sensor del ratón es capaz de detectar, siendo más sensible a mayor cantidad.
- Reproducibilidad y estabilidad. O cuánto de constante y fiable es un sensor al estudiar la salida que se obtiene al realizar mediciones del mismo valor de entrada repetidas veces. Si se realiza de forma continuada, es decir, mediante una entrada de dato constante, se denomina estabilidad.

- Banda. Aquellos valores que no producen un estímulo en el sensor y por lo cuál para ellos no hay una salida.
- Impedancia. Los sensores son agregados a circuitos eléctricos y por tanto deberemos protegerlos ya que la impedancia del mismo puede variar el comportamiento del sensor.

Existen por otro lado, características dinámicas, siendo estas las que se refieren al comportamiento entre el momento de la lectura y el procesado posterior hasta convertirlo en una señal eléctrica.

- Tiempo de respuesta. Conocido como la cantidad de unidades de tiempo que ocurren desde que provoco un estímulo cosntante en el entorno hsata que el transductor produce una respusta acorde.
- Constante de tiempo. Que mide la inercia del sensor, esto es, el tiempo que tarda en reaccionar al cambio de la entrada.
- Tiempo de levantamiento. O cantidad de unidades de tiempo que requiere un trasnductor para producir una salida de forma estable.
- Tiempo de asentamiento. Que tiene en cuenta cuanto tarda el transductor en ir alcanzando porcentajes de salida determinados.

2.4.2 Tipos de sensores

Existe una inmensa cantidad de sensores, prácticamente tantos como magnitudes hay en el mundo, pero no todos funcionan internamente igual, puesto que diferentes materiales reacionan de forma diferente a los estímulos del entorno. Vivimos en un mundo lleno de sensores, en el teléfono se concentran de hecho gran cantidad de ellos, pero también en la nevera, en el coche e incluso en las propias puertas. Para suplir las necesidades de cada uno de estos entornos, se fabrican diferentes tipos de sensores.

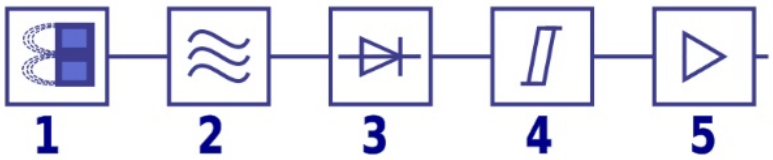


Figura 2.9: Diagrama de un sensor inductivo sencillo. En orden, sensor de campo, oscilador, demodulador, flip-flop y salida

2.4.3 Sensores químicos

Los sensores químicos son aquellos transductores que producen una señal eléctrica proporcional a un parámetro químico. Entre los sensores químicos encontramos los sensores de PH, sensores de concentración y de composición. [14]

Los sensores potenciométricos, que contienen transductores que determinan la diferencia de potencial entre un electrodo de trabajo y uno de referencia. Para la conversión a señal eléctrica se útiliza la ecuación de Nersnt y forman el grupo de sensores más estudiados y útilizados, siendo el electrodo de vidrio el más útilizado. Se distinguen los transistores de efecto de campo sensibles a iones ISFET y los que útilizan membranas selectivas CHEMFET.

Los sensores amperométricos, que contienen transductores capaces de determinar la proporcionalidad que existe entre la concentración de una especie electro activa y la corriente que genera al oxidarse o reducirse. En este caso se útiliza la ley de Fick para la conversión.

Los sensores conductimétricos, que son aquellos que útilizan como medida los cambios de conductividad que aparecen en una solución. Estos sensores disponen de electrodos de metales nobles que detectan cambios de concentración en soluciones a los que son sensibles, como por ejemplo, el oro con el ácido sulfhídrico.

Los sensores de gases pueden ser de muchos más tipos generales, como catarometros, catalíticos, piezoeléctricos, resistivos, etc...

Todos estos sensores se basan en el mismo funcionamiento lógico, pero es la membrana activa, los ánodos y cátodos o la sustancia interna, los que provocan que el sensor de como salida diferentes señales eléctricas con sentido.

2.4.4 Sensores físicos

Los sensores físicos, por su parte, son capaces de producir una señal eléctrica proporcional a una magnitud física cambiante. Entre estos sensores aparecen diversas posbilidades, posiblemente más conocidas y usadas en el día, por lo que estos sensores son más conocidos que los químicos. Entre los sensores encontramos de temperatura (pirómetros, termistores), de esfuerzo y deformación, de movimiento a grandes y pequeñas distancias (LDT, VDT, Radar, Láser) de posición lineal o angular, de presencia o proximidad (infrarrojos, de efecto Hall, fotoeléctricos), de humedad y punto de rocío, táctiles (matrices piezoeléctricas o capacitivas), y muchos más, porque como ya dijimos, prácticamente existen sensores para todas las magnitudes posibles.[15] En el siguiente punto, vamos a centrarnos en los dos sensores más importantes para nuestro proyecto, sensores de radiofrecuencia y sensores ópticos de visión artificial como las cámaras.

2.4.5 Sistemas RFID y Sistemas de visión artificial

Los sistemas RFID útilizan sensores inteligentes junto con middleware para la detección e interpretación de la información que recojen. Un sistema RFID pasivo se compone de 3 módulos. El primero es una etiqueta RFID, del inglés tag RFID, con un número de serie MAC y con otros datos que queramos atribuirle a través de un lector. Esta etiqueta necesita una cantidad ínfima de enerfía para funcionar, tanto es así que la obtiene de otro del segundo de los módulos del sistema, las antenas de radio. El tag RFID almacena energía de las ondas de radio de las antenas que hacen un polling preguntando por la información de los dispositivos que tienen cerca.

Una vez adquieren esa energía, son capaces de emitir una onda con la información que contienen y esta onda es recogida por las antenas de radio que transforman la señal electromagnética en señal eléctrica que envían directamente al tercer módulo, un sistema middleware que es capaz de decodificar esa información y obtener el contenido de la misma en caracteres legibles, para, por último, actuar con esa información contra algún sistema de información actualizando datos o generando nuevos eventos.

El uso de la radiofrecuencia está reservado y regulado por diferentes entidades [16] que limitan qué banda de radio puede usarse, siendo la misma banda la que limita la distancia a la que un tag puede leerse. En el caso del sistema de localización de hospitales que queremos llevar a cabo, el sistema será el mismo que el de las tiendas de ropas, con la frecuencia de 13.56 MHz que permite una lectura a una distancia de 10 cm a 1 metro, siendo además los tipos de tag más baratos a la hora de la compra y los más variados a la hora de darles forma o útilizar materiales donde embeberlos, como por ejemplo pulseras de papel.

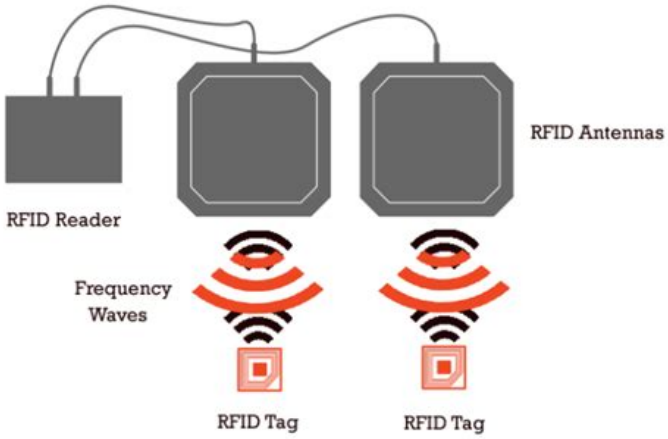


Figura 2.10: Diagrama arquitectura general de un sistema RFID

Por otro lado, los sistemas de visión artificial normalmente usan cámaras de tipo CCD, las llamadas cámaras para visión de máquina son cámaras con una alta resolución y muy bajo ruido que permiten la extración de información mediante la obtención de imagenes. Estos sensores suelen estar provistos de un interfaz de datos amplio, por ejemplo de tipo Gigabit a través de puertos Ethernet o USB 3.0.

Muchas de estas cámaras ya tienen integrado un pequeño computador que les permite procesar gran cantidad de imágenes por segundo en la misma cámara sin necesidad de enviar los fotogramas a un computador externo y tomar decisiones conforme a la información extraída. Para realizar la extración de esta información se útilizan técnicas de postprocesado de la imagen como los filtrados morfológicos, el conteo de píxeles de determinado color, la detección de bordes y colores o la segmentación de la misma. Existen otros métodos mucho más avanzados pero que también requieren de capacidad de computación extra, como puede ser el uso de redes neuronales o deep learning, metrologías, reconocimiento de patrones o lectura de códigos como los conocidos QR. Este postprocesado resulta en una salida de información contextual, ya sin imagen, donde se enumera todo lo detectado en la imagen postprocesada, pudiendo realizar acciones respecto a los valores obtenidos.

2.4.6 Implementaciones. Smartcities y Smartbuildings

Entre las implementaciones, más solicitadas en los últimos años, que podemos encontrar en los sensores destaca la creación de smartcities y smartbuildings, estos son, ciudades y edificios que son capcaces de tomar decisiones por sí mismos que benefician al conjunto, a través de los datos recogidos por los sensores que hay repartidos por toda la arquitectura.

En el momento de la escritura de este documento, son muy pocas las ciudades que son completamente inteligentes, sín embargo, todo lo que se construye nuevo si se realiza con dispositivos inteligentes, lo mismo pasa con las remodelaciones, donde se implementan cada vez más sensores y programas de gestión de la información que se recopila. Muestra de esto son los diferentes parkings de los nuevos centros comerciales, que ya equipan sensores de proximidad para detectar si hay un coche ocupando una plaza y mostrar el número restante de las mismas en la entrada de las calles del aparcamiento, o el uso de sensores que detectan la polución de Madrid a través de los cuales se activan escenarios de contaminación, incluso algo que si está completamente implementado, la posbilidad de pago mediante tecnología NFC.

El aumento y proliferación de los sensores, permitiéndose que sean más reducidos y baratos, nos llevará a un futuro donde todos los edificios sean inteligentes, que es el primer paso para permitir ciudades

inteligentes, donde las proximidades estarán controladas automáticamente, la temperatura de los edificios se mantendrá estable según el momento del día, las iluminaciones serán adaptativas a los cambios de luz exteriores, para permitir el mejor aprovechamiento diurno sin provocar reflejos o falta de luz, implementar el mismo sistema de parking inteligente que se usa en los centros comerciales para la calle evitando contaminación y consumos de gasolina innecesarios y un largo etcétera.

Figura 2.11: Cámaras con reconocimiento de matrícula de acceso a Madrid Central

En nuestro caso vamos a aprovechar la tecnología de sensores junto con la tecnología IoT para crear un espacio hospitalario inteligente, desde que el usuario sale del lugar donde ocurrió la urgencia hasta que recibe el alta en el hospital, para eso se propondrá y desarrollará un sistema de visión artificial, sensores de identificación por radiofrecuencia, control de tráfico a través de la información que recoja estos sensores así como generación de eventos desde los sensores anteriores. Siempre se pueden añadir otros sensores durante el desarrollo que permitan añadir nuevas funcionalidades para aumentar la inteligencia del entorno hospitalario.

2.5 Cloud Computing

La computación en la nube forma parte de un nuevo paradigma computacional donde grandes granjas de servidores proveen de servicios a terceros que las alquilan, teniendo accesos a infraestructuras fácilmente escalables a un bajo coste en comparación con la compra y mantenimiento de los mismos por parte personal.

2.5.1 Definición e historia

Definimos la computación en la nube como el suministro de servicios informátiocs a través de Internet, de forma que podemos tener acceso a recursos flexibles y escalables de forma económica y rápida, y facturando solo el tiempo que estemos útilizando el recurso.

Para permitir costes bajos para usar tecnologías y servicios como la que los proveedores nos ofrecen en la nube nos basamos en el principio de reusabilidad IT o principio de reusabilidad, técnicamente, un principio de diseño en el que las aplicaciones se diseñan orientadas a servicio, por lo que cualquiera puede usar en su negocio ese servicio, al no estar especializado, por lo que puede ofrecerse a todos los clientes por igual y no tiene que ser desechado cuando no se útilice. Por ejemplo, el aprovisionamiento

de máquinas virtuales en la nube por parte de un cliente puede no ser permanente, de hecho, lo normal es que no lo sea, y en el momento de que se desaprovisionan esas capacidades por parte de un cliente, otro puede pasar a útilizarlas directamente, por lo que no se desecha y el equipo pasa toda su vida útil funcionando entre varios clientes. Gracias a esto, las empresas pueden lanzarse a útilizar grandes capacidades computacionales para proyectos que lo requieran, sin tener que invertir en la compra e instalación de las mismas, ampliando los proyectos a los que una empresa sin demasiados recursos puede solicitar.[17]

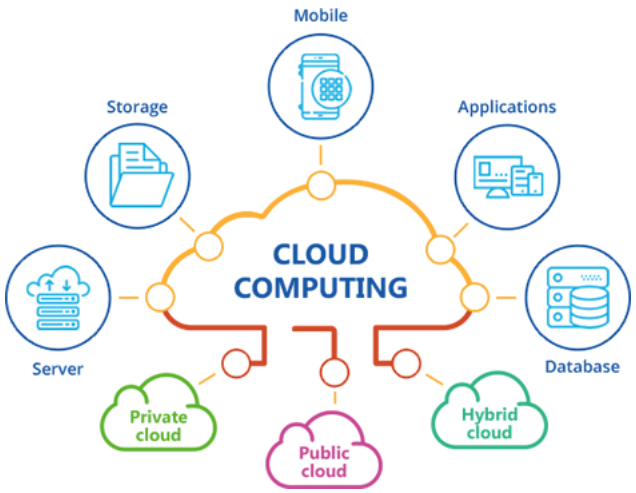


Figura 2.12: Servicios principales y tipos de nubes.l

Una vez entendemos este paradigma un poco más podemos hablar como se alcanzó el nivel de servicios que tenemos actualmente, y es bien sencillo, todo ocurrió gracias al desarrollo de la virtualización. La virtualización, cuyas primeras andaduras fueron allá por los 90, fue un termino anterior en el que se definia como podría crearse un recurso computacional virtual que funcionara de cara al exterior como si de un recurso físico se tratara. Por supuesto, este concepto ha ido evolucionando, desde las primeras VPN que nacieron como un negocio de alquiler de redes de Internet a los primeros despliegues de programas a través de Internet, en vez de usando soportes físicos.

Por supuesto, el concepto actual de cloud computing ha evolucionado muchísimo, tanto que parece un concepto completamente distinto de lo que en un principio fue, pero no es tanto así. Fue en 2002 cuando Amazon comenzó a pensar qué hacer con toda la infraestructura que le sobraba en algunos centros logísticos, pero que no podían usar para aprovisionar otros, fue entonces cuando se empezaron los desarrollos de los modelos de infraestructura que permitían el uso de estas capacidades por otros en la nube. Posteriormente, ya en 2006, aparece Amazon Web Services, que ofrece servicios en la nube de forma externa y como modelo de negocio, en vez de como aprovechamiento de sus servidores.

En los años siguientes, otros proveedores como Google, Microsoft, Oracle o IBM, incluso se desarrolla software destinado a la creación de recursos en la nube, como la NASA hizo con OpenNebula

2.5.2 Modelos

Por supeusto, la diferencia de necesidades sobre los servicios que se ofrecen en la nube crea también una diferencia de modelos en la infraestructura que ofrecen los proveedores, que debemos tener en cuenta para el desarrollo de nuestro proyecto.

- SaaS. De «Software as a Service», donde un aplicativo es ofrecido a los clientes a través de Internet según la demanda. De esta forma uno o más servidores corren la aplicación y son capaces de

servirla de forma individualizada a todos los clientes, por lo tanto el cliente solo tiene que pagar por el servicio sin contar con los costes de mantenimiento, actualizaciones o servidores que se necesiten para correr el programa, simplemente lo usa.

- PaaS. De «Platform as a Service», donde el software desarrollado es encapsulado de forma que la plataforma donde va a ser lanzado lo entienda, como quien crea un contenedor Docker y corre la imagen. De esta forma es el cliente el que puede crear un software y servirlo en la nube, por lo que a efectos de otra persona estará usando un servicio SaaS, mientras que para el cliente que ha creado el software es PaaS. Otro ejemplo de capacidades computacionales que se lanzan en este modelo son los stacks como LAMP, XAML, o incluso ELK para Elastic.

- IaaS. De «Infraestructure as a Service», este modelo es otro de los que más nos interesa a nivel de proyecto y de los más usados a nivel empresarial. En este modelo, el proveedor nos ofrece la infraestructura como puede ser una base de datos, una máquina virtual o discos duros y nosotros hacemos el uso que deseemos sobre ellos, dentro de los límites de configuración interna que realice el proveedor.

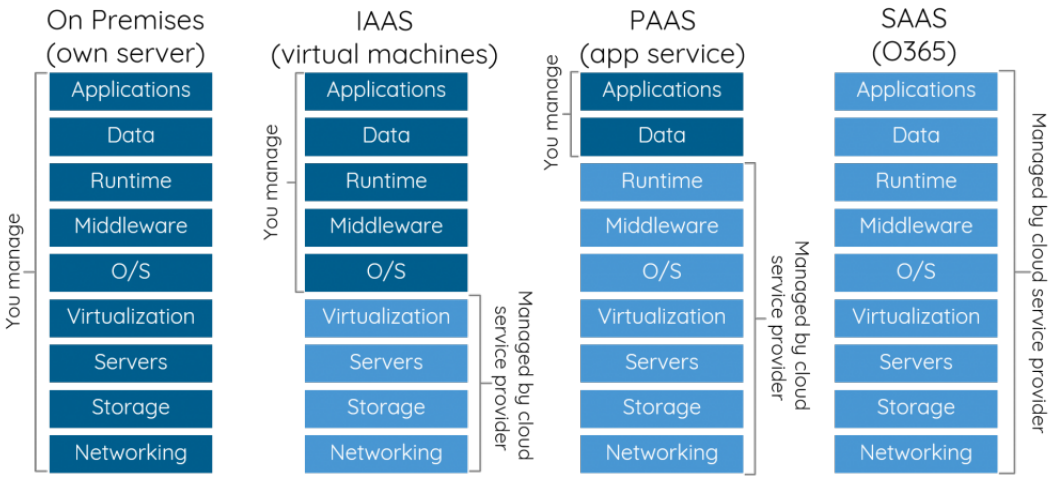


Figura 2.13: Capas que maneja el usuario por cada tipo de servicio

Además, estos modelos de nube, son aplicables a todos los tipos de nubes que se desplieguen, que vienen a ser tres:

- Nube Pública. Son aquellas nubes que están dirigidas por terceros para ser usados por clientes, mediante suscripción. Normalmente esta suscripción es de tipo Pay-as-you-go o Pago por uso, donde solo pagamos por las capacidades computacionales que hemos útilizado a final de mes, sin costes por tener una suscripción activa y a veces incluso por hacer un uso mínimo del mismo, ya que también existen y ofrecen capas de uso gratuito cada mes. Ofrrecen tanto recursos computacionales compartidos como la posbilidad de contratar recursos dedicados a un mayor coste que los anteriores

- Nube Privada. Construidas exclusivamente para una empresa, pudiendo ser usada por todas sus filiales pero no por terceros externos a la misma. La mayor ventaja de este tipo de nube es el control, ya que todos los datos y accesos se realizan desde dentro de la empresa, manteniendo el control completo de la infraestructura. No debemos confundir el concepto de nube privada con que la infraestructura donde se despliegue nuestra nube privada lo sea. De hecho, pueden exisitr nubes privadas creadas por proveedores externos de servicios cloud y nubes privadas creadas por data centers propios de la empresa.

- Nube Híbrida. La unión de los dos tipos anteriores da forma a las nubes híbridas. Normalmente se usa esta infraestructura cuando existe la posibilidad de que la nube privada se quede corta de recursos en algún momento de gran carga, para ello se hace uso de las capacidades en la nube de un tercero.

2.5.3 Servicios en la nube. Beneficios y retos

La nube ofrece cientos de servicios distintos, entre ellos los más útilizados son el aprovisionamiento de máquinas virtuales (Amazon EC2 o el servicio de Máquinas Virtuales de Microsoft Azure), aprovisionamiento de discos de datos para almacenamiento (Amazon S3, Azure Blob Storage), bases de datos como servicio, donde no necesitamos instalar nada sobre ninguna máquina, ya que nos ofrecen el recurso exclusivo de bases de datos, más baratas que un recurso de máquina virutal completo del mismo nivel, tanto SQL como noSQL (Amazon RDS, Dynamo DB, MongoDB Atlas) y por último, sistemas de mensajería que hace las veces de Kafka o ActiveMQ con accesibilidad 24/7 desde cualquier lugar (Amazon SNS, Google Pub/Sub).

Como hemos dicho, existen muchos servicios más en estos proveedores, y solo hemos tenido en cuenta los de tipo IaaS, por lo cuál hay un mundo inmenso de posibilidades. Este aumento del uso en cloud computing es posible, debido a las ventajas que presenta respecto a un sistema on-premises.

- Bajo Coste. Ya que, como indicamos, el contrato de uso suele ser de pago por uso, por lo tanto solo se paga mientras el recurso esté encendido, por otro lado no se realiza la inversión inicial de los dispositivos y la configuración y cableado ya esta realizado, junto con el mantenimiento muy importante para no tener tiempos de downtime, esto es conocido como contratos 99,99 %, donde se comprometen a que no van a tener más de un 0,1 % de dowtime sobre tus recursos, bonificandote si es incumplido.

- Altas capacidades. Ya que el uso compartido de ciertos recursos permiten que se invierta en dispositivos de muy altas prestaciones para el alquiler, tanto de máquinas virtuales, como de cuentas de almacenamiento o entornos de trabajo que requieran grandes cantidades de computación y servicios, como por ejemplo renderizados de videos o experimentos científicos.

- Escalabilidad. Siendo una de las características más importantes, puesto que es la que permite que en caso de necesidad nuestro aplicativo crezca con una simple instrucción, sin necesidad de comprar o instalar nuevas máquinas como pasaría en un modelo on-premise, y lo mismo al contrario, reducir la capacidad operativa cuando ya no sea necesesaria sin perder el valor del dispositivo.

En la otra mano, tenemos las complicaciones propias del paradigma del cloud computing. Entre ellos aparece como principal la protección de datos, estos normalmente de carácter privado, deben estar completamente securizados y que en un recurso compartido no puedan ser accedidos por un tercero no autorizado que casualmente tenga asignado el mismo servidor que yo. Por otro lado, sobre el tema de la posesión de datos, debemos tener en cuenta que todo lo que almacenemos en la nube sigue estando bajo nuestro mando, es decir, somos los dueños de esos datos[18] y somos los responsables de que esos datos no sean alojados en lugares ilegales o que el propio dato no deba ser almacenado o no sea de contenido legal, mientras que el proveedor de servicios es el responsable de que ese dato no se pierda y de que el almacenamiento físico del dato este protegido. Por tanto existe una relación de confianza empresa-proveedor. El otro reto tiene que ver con los contratos 99,99 % uptime que hablamos antes, pues los datos deben estar permanentemente accesibles y ser recuperables siempre que no se borren por nuestra parte. Para esto, los proveedores deben establecer planes detallados de replicación de datos y mantenimiento

durante los cambios de dispositivos por antiguedad, frente a fallos del sistema que puedan darse en el data center. Por ser más concretos, el total anual que supone este contrato es de menos de 1 hora de servicios no disponibles al año.

2.5.4 Machine Learning

Otro de los servicios más útilizados actualmente en la nube, ya que provee de grandes beneficios pero a la vez su infraestructura es demasiado cara para mantenerla on-premises, es el machine learning. [19]

En un mundo donde la siguiente revolución industrial y de servicios vendrá marcada por la implementación de inteligencias artificiales reales, el machine learning tiene mucho que decir, pues es la capacidad de crear modelos analíticos a través de reursos de computación y que estos modelos sean capaces de predecir con la mayor exactitud posible la derivación de una serie de datos a través de la entrada de los mismos. Por ejemplo y exageradamente, si tengo consumos de electricidad de una pinza de soldadura de automoción de los últimos 4 meses, y siempre que el consumo pasa de X kW la pinza comienza a hacer soldaduras deficientes, podremos adivinar cuando va a llegar el resto de pinzas de la fábrica a ese X antes de que llegue, permitiendo la planificación de manteimientos sin afectar a la producción, tanto por la parada de mantenimiento como porque se llegue a realizar una soldadura defectuosa.

2.5.4.1 Obtención de datos

La obtención de datos se realiza a través de sensores que se colocan en puntos estratégicos del lugar donde se quiere realizar el análisis. Durante un tiempo determinado, depende de la naturaleza de la magnitud a medir, e sensor ira recogiendo y almacenando los datos al enviarlos a algún dispositivo de almacenamiento, y pasado este tiempo se obtendrán diversos datos como para crear un dataset.

Una vez obtenido el dataset, se envía al software de machine learning para que empiece a establecer modelos matemáticos, relacciones y detección de eventos anómalos en las series.

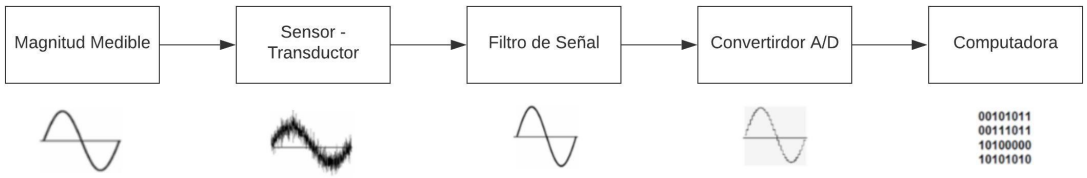


Figura 2.14: Conversión desde el dato real al dato virtual

Pero las novedades azotan este campo continuamente, y en los últimos años el machine learning ha conseguido un campo de aplicación nuevo, el Deep Learning, capaz de conseguir información y estudiarla de forma autónoma a través de funciones de recompensa, que «bonifican» al software cada vez que realiza un movimiento correcto, con mayor bonificación cuanto más acertado es el movimiento incluso penalizándolo si es erróneo, tomando este tipo de aprendizaje como un juego donde el algoritmo busca el conjunto de decisiones que le ofrecen la mayor puntuación, por lo que el paso de la obtención de datos se hace literalmente al mismo tiempo que la creación del modelo.

2.5.4.2 Modelo y Entreno

El entreno se realiza a través de 2 técnicas distintas, aprendizaje supervisado o no supervisado. Para el primero requerimos la generación de un modelo por el que pasan los dataset que hemos creado y donde

hemos indicado que significa cada uno de los datos recogidos, para los que la red neuronal generada asignará pesos y salidas, creando clasificaciones (vectores, algoritmo de Bayes, vecino más cercano) y regresiones (SVR/GPR, árboles de decisión, redes neuronales). Una vez entrenado, cualquier otro dato que le pasemos, atravesará esta red neuronal y saldrá clasificado según la posición que haya tomado respecto a los otros datos, por ejemplo, si la imagen de un león sale por el mismo punto que en el modelo supervisado le dijimos que era un pulpo, dirá que es un pulpo, pero si dijimos que era un león, dirá que es un león.

El otro modo, no supervisado, requiere que los datos que le pasemos estén etiquetados, y estos datos se agruparan mediante diversos algoritmos en una matriz de puntos o clusters, creando zonas que determinan que todo dato caído en esa zona, pertenece al mismo cluster y por tanto a la misma clasificación que el el resto de puntos de ese cluster. A esta técnica se la llama clustering, K-Means, Mezcla Gaussiana, Redes Neuronales o modelos ocultos de Markov entre otros son ejemplos de algoritmos basados en clustering.

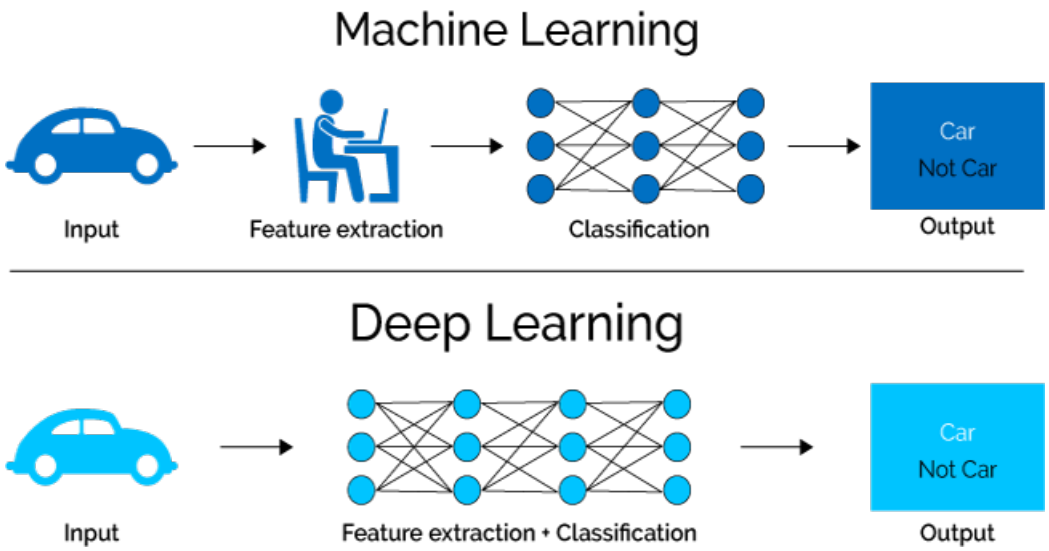


Figura 2.15: Diferencia entre machine learning y deep learningl

fc

Capítulo 3

Desarrollo

El principio de la ciencia, casi la definición, es el siguiente: «La prueba de todo conocimiento es el experimento». El experimento es el único juez de la verdad científica

Richard Feynman, Seis piezas fáciles

3.1 Introducción

Tras el estudio teórico comenzamos con el desarrollo práctico de los conocimientos adquiridos sobre el problema real. El capítulo está separado en 2 secciones diferentes, en la primera trataremos sobre la descomposición del problema general en pequeñas fases, explicando como se ha resuelto cada uno de esas fases, si se han valorado alternativas y la arquitectura que hay bajo ellas. Por tanto, se tocarán todos los conceptos desde la programación en Javascript de funciones orientadas a servicios serverless hasta el desarrollo del aplicativo final que corre sobre los dispositivos Android como enlace entre la lógica del producto y nosotros.

En la otra parte hablaremos sobre los servicios en la nube que se han utilizado, por qué se han utilizado en vez correr contra un servidor físico como pudiera ser nuestro ordenador y que ofrecen estos servicios en la nube, su gestión y entornos.

Por supuesto, todo el desarrollo se ha realizado con mecanismos de bajo coste para abaratar los gastos del desarrollo, pero posteriormente, en el cálculo presupuestario del mismo se podrá abordar tanto gasto de los componentes recomendados para un sistema en entorno productivo como los utilizados para la prueba de concepto que aquí se muestra.

Este capítulo solo muestra como se ha llevado a cabo el desarrollo, sin trozos de código molestando entre medias. Para poder ver ejemplos acceda a los anexos del proyecto donde sí podrá ver fragmentos de algoritmos y código utilizado.

3.2 Descomposición del problema

Planteada la necesidad de un sistema de información para hospitales, dada la amplitud de un complejo hospitalario, hemos decidido centrarnos en una unidad específica como es la unidad de urgencias hospitalarias del Hospital Universitario Principe de Asturias. Para ello consideraremos la necesidad de un

sistema de información que cubra desde el mismo momento que se produce una incidencia por parte de un ciudadano, ya sea en la calle, en su casa o realizando cualquier actividad, hasta que el mismo recibe el alta hospitalaria, es decir, sale del hospital al que no tiene que volver a ser ingresado, pero puede seguir teniedo tratamientos de rehabilitación u observación en los que tiene que acudir al mismo, o lo que es lo mismo, terminamos el flujo de urgencias antes del alta médica.

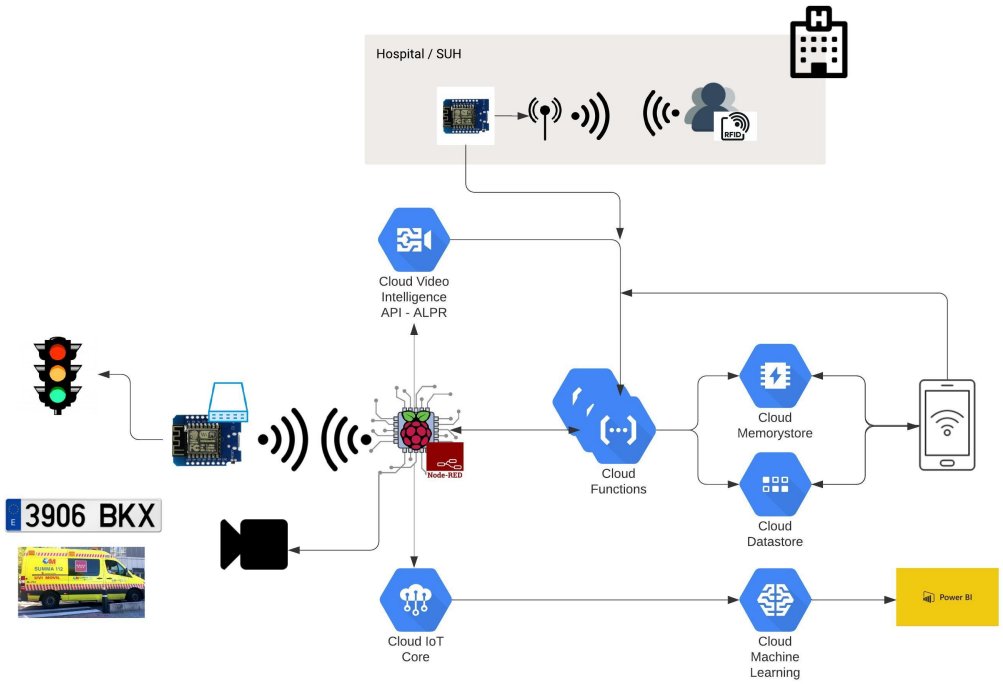


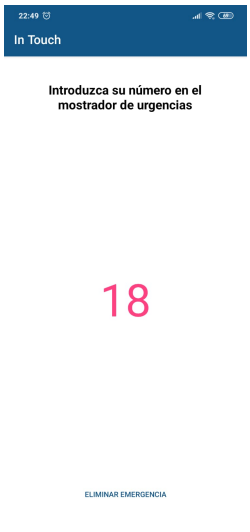
Figura 3.1: Vista general de la arquitectura del desarrollo

3.2.1 Declarando una emergencia

Supongamos que Iris Sakura ha tenido un accidente con un monopatín. En el accidente Iris cayó del mismo y para no aterrizar con la cabeza puso la mano, con tan mala suerte que dejó caer el peso con demasiada fuerza sobre la misma, provocándole la fractura del antebrazo. Los amigos de Iris, que están presentes llaman a los padres, que se presentan inmediatamente con el coche en el lugar de los hechos, ven el estado de su hija y deciden llevarla al servicio de urgencias del Hospital. Nos encontramos en Daganzo, una población cercana a Alcalá de Henares, por lo que nuestros protagonistas tienen un tiempo antes de llegar al hospital.

Ante esta emergencia se nos plantean los siguientes problemas, el primero es que el hospital no sabe que ha ocurrido esta emergencia, por lo que no esperan atenderla. El segundo es que la entrada al hospital cercana a Daganzo suele estar colapsada a la hora en la que los protagonistas quieren acudir ya que coincide con la salida de cientos de universitarios y trabajadores que bloquean las salidas de las rotondas, añadiendo al viaje 10 minutos o la asunción de peligros innecesarios en la conducción por intentar llegar antes.

Para solucionar ambos problemas, se plantea la creación de un aplicativo que sea accesible desde cualquier sitio, y esto se traduce en el desarrollo de un aplicativo Android para móviles. Desde el mismo se piden 2 formularios sencillos, ya que solo pedimos la información necesaria para tramitar la emergencia de la forma más rápida posible. En el primer formulario pedimos los datos de la urgencia, DNI del paciente



(a) Pantalla con el número de identificación en urgencias

(b) Disclaimer de condiciones de uso

Figura 3.2: Pantalla de información sobre el paciente hospitalizado

al que nosotros asociaremos por base de datos el numero de la tarjeta sanitaria ya que es normalmente común que los ciudadanos sepamos de memoria nuestro número de identidad pero no el de la tarjeta sanitaria, también se pedira nombre y apellidos del mismo y la selección en el desplegable del tipo de urgencia ocurrida de entre las mostradas.

En el segundo formulario se piden datos relativos al vehículo que va a transportar al paciente, que serán matrícula del vehículo y DNI del conductor, este segundo a efecto más disuasorio de usar el servicio a la ligera o para motivos ajenos a una urgencia, además de la ilegalidad de hacer gasto de recursos para urgencias de forma ficticia recogido en la ley.

Por tanto los padres de Iris escribirán en el formulario, DNI: 01234567A, Nombre: Iris, Apellidos: Sakura, Urgencia: Fracturas, a continuación, en la parte inferior, escribirá Matrícula: 4676NAH y DNI Conductor: 35146418A para terminar pulasndo sobre el botón de «Declarar una emergencia».

Tras pulasr el botón de enviar, aparecerá un diálogo más que nos recuerda que estamos declarando una emergencia al usar este servicio, con todas las responsabilidades legales y derechos que conlleva, además nos indica cómo será el tratamiento de nuestros datos. Aceptamos las condiciones del cuadro pulasndo sobre «Continuar» y nuestra aplicación actualizará la pantalla para mostrarnos un número de registro, que debemos tener presente en la llegada al hospital, pudiendo cerrar la aplicación ya que persistirá aunque se cierre pudiendo ser consultado en cualquier momento.

Esto es lo que ocurre por arriba, lo que ven nuestros pacientes, pero por debajo estamos encadenando llamadas, obteniendo más información del dispositivo y generando identificadores en la nube, concretamente el flujo de backend es el siguiente:

- Durante el formulario. Mientras se rellena el formulario se están accediendo a 2 variables internas del teléfono que también se envían junto con los datos introducidos por el usuario. Estos datos corresponden al IMEI del teléfono y a la localización del mismo. Con el IMEI conseguimos un valor único que se usará para ser pareja clave-valor con la matrícula y con el hecho de que se haya declarado una emergencia con ese dispositivo. Con la localización conseguimos que el hospital pueda saber el tiempo que va a tardar aproximadamente en llegar el paciente y conforme al tipo de problema que urgencia que haya marcado reservar recursos para una atención orquestrada.

Siguiendo el flujo, el aplicativo inicialmente lanza a un API REST en forma de Google Function, bajo el nombre de EmergDialer, una petición con los datos de registro del paciente. Google detecta que esa función ha sido llamada y le pasa los parámetros de entrada. Estos parámetros son parseados por la función y formateados como un nuevo JSON entendible para la base de datos en la nube, no transaccional, MongoDB Atlas, donde se almacenan los datos del paciente.

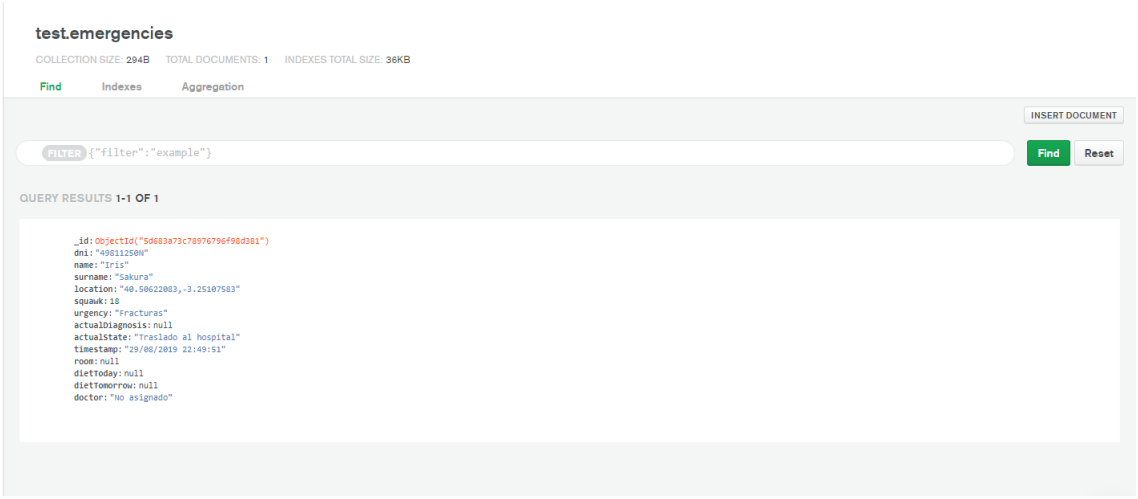


Figura 3.3: Vista de documentos desde MongoDB Atlas

- Tras el formulario. Los valores de posición e IMEI que hemos recolectado se utilizan en parte de los siguientes pasos, para ello se generan nuevas peticiones desde el dispositivo a otros endpoints de GFunctions.

Una vez la llamada a la anterior petición ha sido existosa, y esto se conoce porque el dispositivo devuelve el código de estado 200, 2 peticiones más son lanzadas, una a la GFunction llamada Squawker, conteniendo únicamente el IMEI y otra a la GFunction TRACON que contiene el IMEI y la matrícula del vehículo que se haya escrito. Estas funciones tienen un único objetivo, actuar lo más rápidamente posible, para ello todos los datos que se introducen en la petición pasan a ser mapas clave-valor de una base de datos Redis colocada en la nube y Redis, al ser una base de datos en memoria, actúa de forma inmediata indexando y encontrando valores, gracias a esto podemos garantizar un tiempo de reacción mínimo que será vital para los siguientes pasos.

Y así termina la declaración de una emergencia.

3.2.2 Controlando el tráfico

Nuestros protagonistas ya están acercándose al hospital, pero antes tienen que pasar por una intersección y rotonda que parece obstruída por la cantidad de coches que circulan por ella. Afortunadamente, según se están acercando, los semáforos cambian a rojo en todas las incorporaciones menos la suya y el tráfico comienza a fluir, por lo que no tienen que detener el vehículo y alcanzan el hospital antes de lo esperado.

En este momento ha entrado en juego el IoT puro, ayudado por los datos que han introducido antes en la aplicación. El semáforo no cambió de color mágicamente, sino porque una cámara situada a varios metros de la rotonda detectó la matrícula del vehículo en el que viajaba Iris, el dispositivo IoT la comparó con su base de datos y esta le informó de que era un vehículo prioritario, por lo que procede a cambiar los semáforos para facilitarle el traslado.

En primer lugar, un sensor de movimiento detecta que hay un coche pasando por delante del mismo lo que provoca el comienzo de un flujo de operaciones en una Raspberry Pi 3B a través de NodeRed, una herramienta de programación de flujos de conexión para dispositivos hardware con otras herramientas computacionales como APIs u otros dispositivos hardware, especialmente inventado para IoT. Node-Red empareja este sensor con el comienzo de un flujo cuando la salida del mismo es de alto nivel ó 1, lo que indica al flujo que debe activar la cámara que posee y tomar una foto, esta foto se convierte a base-64 y se envía a un algortimo de detección de matrículas también conocido como ALPR. En nuestro caso, se ha optado por utilzar, como siempre que aparecía opción, un servicio en la nube que dispone del algorítmo. Al estar optimizado para realizar solo este cometido, es capaz de realizar el análisis de visión artificial bastante rápido, devolviendo cuál es la matrícula que más posibilidades tiene de ser la que aparece en la foto. Con esta matrícula se realiza una consulta al TRACON, la GFunction que vimos antes, pero por otro método, que devuelve si se encuentra en Redis o no, siendo el tiempo de actuación de Redis como dijimos antes de milisegundos. Si la matrícula no aparece, el flujo termina y no cambia nada, por el contrario, si apareciese, el flujo continuará con la conexión de la Raspberry al centro de control de tráfico de la rotonda, conectada a Internet a través de otro dispositivo IoT, un ESP8266 que recibirá la orden de dar prioridad a una de las incorporaciones, y será este el que cambie los semáforos.

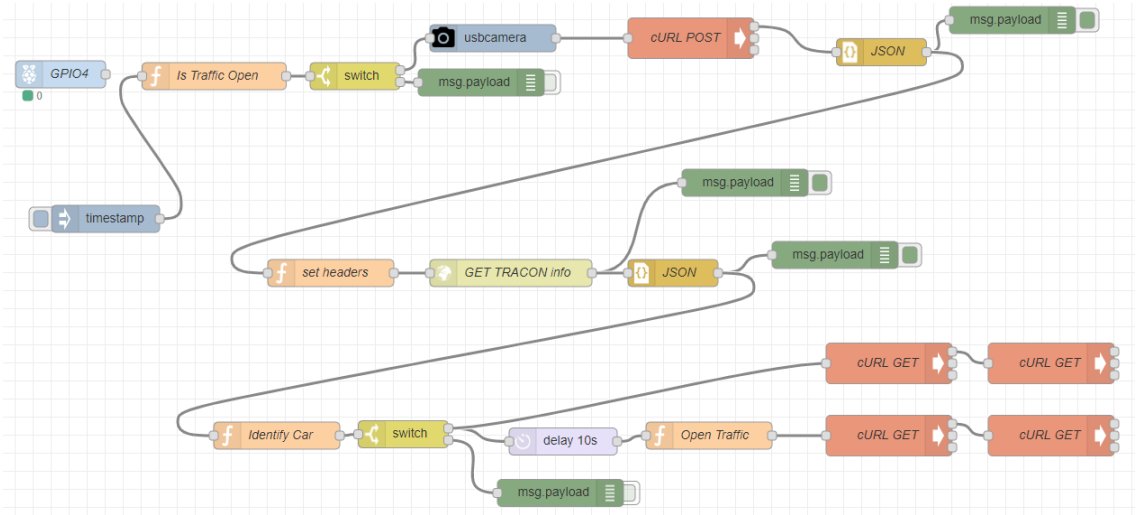


Figura 3.4: Vista del flujo programado con NodeRed

3.2.3 El registro y espera

Iris acaba de llegar al hospital, se baja del coche con su madre mientras su padre lleva el coche a un sitio libre del aparcamiento. Iris se acerca al mostrador e introduce el número que la aplicación le otorgó en la pantalla táctil de registro, esta le devuelve una pulsera que se pone inmediatamente y pegatinas de identificación para las necesidades internas del hospital.

Iris accede a la sala de espera, pero ya contaban con su visita, así pues, tras esperar un par de minutos es atendida en los box de triaje. Mientras la atienden y valoran profesionalmente, el padre llega al vestíbulo después de aparcar, pero no encuentra a sus familiares, sin embargo, en vez de ocupar a la persona de registro preguntándola o pasearse por la sala de espera pudiendo molestar a otros, que ya sabemos todos como nos ponemos de sensibles cuando estamos malos o tenemos algún problema, decide abrir la aplicación y esta le dice que su hija ya está en el sistema de Triaje, por ende, se sienta a esperar tranquilo.

Durante el registro, el frontdesk donde se introduce el número de la aplicación está conectado con la

base de datos de urgencias, donde está la información que la familia ha introducido antes, junto con otros datos extra como los tiempos de llegada, el número squawk que es el introducido y el más interesante para este proyecto, la localización.

El campo localización aparece inicializado al declarar la emergencia con las coordenadas geográficas que obtenemos del dispositivo móvil a declarar la emergencia, las mismas que utilizarían los servicios de urgencias para saber el tiempo de llegada al hospital. Este campo se va actualizando conforme se lanzan eventos en la GFunction de EmergDialer. En primer lugar, este campo pasó de tener las coordenadas a decir que ya está presente al alcanzar la entrada del hospital cambiando la localización por la frase «En el recinto», a continuación se imprime la pulsera, lo que provoca un nuevo evento, el de registro y por lo tanto, el de estadía en el hospital, cambiando la frase de nuevo y por último, la frase cambio 2 veces más, una al entrar en la sala de espera, escribiéndose «Sala de Espera» en la localización y a continuación por la frase «Triaje - Primera atención», indicando que ya estaba siendo valorada por un profesional.

Pero cómo saltaron estos dos eventos finales. La respuesta aparece tanto en la pulsera como en las paredes del hospital. La pulsera, aunque sea de papel o de un compuesto entre papel y plástico, contiene un dispositivo tag RFID en su interior, que ha sido grabado antes de ser entregado con la información de la persona a la que se le ha entregado. Mientras que en las paredes aparecen, en las entradas a las diversas salas, antenas RFID embebidas, que activan la pulsera mediante la onda de radio y leen del mismo el identificador. Cada antena se situa en un lugar y tiene un identificador de antena, cuando la antena envía los datos a un middleware que traduce la dirección de la antena por su posición, por ejemplo, Triaje, y el identificador del tag por la persona que lo lleva, en este caso 01234567A, y este los transmite a MongoDB a través de la GFunction, formando otra arquitectura IoT, esta vez de RFID.

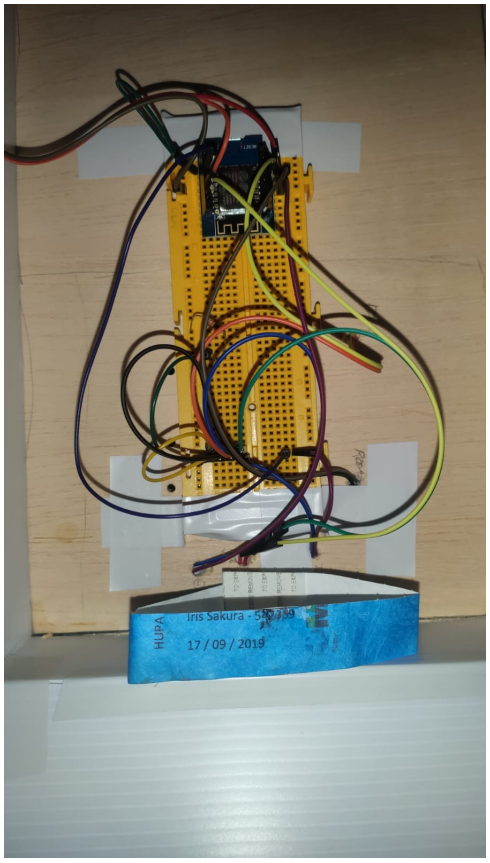


Figura 3.5: En azul, tag RFID para paciente, detrás las conexiones de la maqueta para el sistema IoT - RFID

Por otro lado, para que los valores lleguen a la aplicación, se ha dispuesto de otro servicio orientado a IoT, el de notificaciones push. Este servicio permite publicar mensajes sobre canales y temas, y en efecto, es lo que hace también la GFunction, publicar en estos canales la información actualizada, en este caso la localización. Los dispositivos, que tienen las credenciales para conectarse a estos canales son preguntados por el servidor, siendo esta la característica principal de la tecnología de notificación push, que el mòvil no necesita estar consumiendo datos ni batería preguntando por actualizaciones, porque es el servidor el que avisa al dispositivo de que debe leer un campo nuevo cuando toque.

3.2.4 El ingreso médico

Iris, tras pasar por rayos X es llevada a la sala de curas, donde se le explica que se ha fracturado tanto el cúbito como el radio y que deben colocarle el brazo lo primero, posteriormente dejarla en los boxes de urgencias en observación para ver si baja la inflamación e ingresarla cuando este fuera de peligro para realizarle una operación quirúrjica donde se le colocará correctamente el brazo y se colocarán placas, por lo que pasará unos cuantos días en el hospital.

Durante el ingreso médico, el resto de campos de la apliicación en el navegable de hospitalización irá actualizándose continuamente según se sucedan eventos. Si es necesario se le asignará un médico y este aparecerá reflejado en la aplicación, si en algún momento ha sido asignado a planta y a una habitación, lo mismo ocurrirá, el campo de habitación se actualizará con el número correspondiente. Todas estas actualizaciones se realizan a través del GFunction EmergDialer y del servicio de notificaciones push.

Pero en un ingreso hospitalario aparecen más factores que pueden interesar inlcuir en un sistema de información, como pueden ser las dietas, y que mejor forma de dejar de consumir papel para selecionarlas, que ofrecer a través de estas notificaciones los platos y menús correspondientes al día y que sea desde aquí desde donde se seleccionen estos. Normalmente, es también en este momento cuando se plantean las intervenciones al paciente o cuándo los familiares realizan las visitas. Para estos casos, es también durante el ingreso médico cuando se utilizará el modo de realidad aumentada del aplicativo, pero dejamos la explicación de este modo para un apartado único.

3.2.5 La aplicación

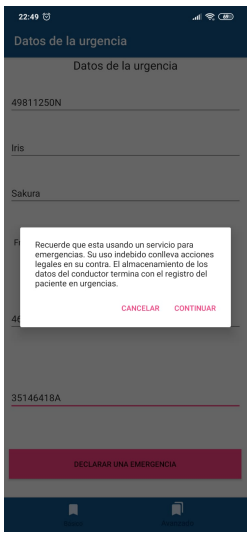
El aplicativo desarrollado también ofrece otra información, fuera del conenido de hospitalizaciones de urgencias. A través de ella, según el número de pacientes que se estén atendiendo podemos mostrar una pantalla de información con el nivel de saturación del mismo, así como de su posición en el mapa para facilitar la búsqueda o la llegada por parte de familiares que no conocen el hospital, por ejemplo. Por otro lado, también estaría la implementación que permite seleccionar que usuarios podrán ver nuestro estado de hospitalización, es decir, habitación, doctor, localización y estado actual y demás datos o por otro lado ver el estado de hospitalización de los que nos han permitido acceso con la opción anterior.

Por último también aparece una pestaña con la información de nuestro perfil, que nunca esta de más tener estos datos médicos a mano por si los necesitamos en algún momento para alguna gestión en el hospital, conteniendo información como el médico de cabecera, caducidad de la cuenta y más.

3.2.6 Información por Realidad Aumentada

Iris es una chica muy espabilada y cuando la doctora le explicó la intervención la comprendió sin problemas, sin embargo, su padre, bastante más mayor no comprendía del todo la situación, entonces ella, con su tablet Android y la cámara, coloca sobre la mesa un marcador de la historia clínica de Iris y de

(a) Formulario de aplicación con los datos de la urgencia y conductor



(b) Disclaimer de condiciones de uso

Figura 3.6: Proceso de declaración de emergencia sobre la aplicación

él aparece una representación tridimensional del estado del brazo de Iris, explicandole de nuevo al padre la intervención que viendolo sobre el modelo lo entendió sin problemas.

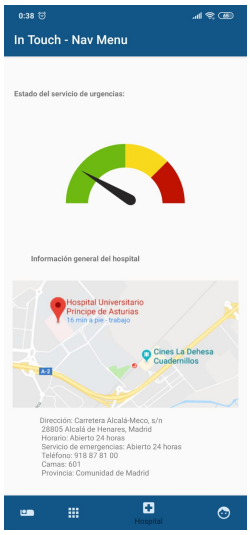
Para este tipo de sistema de información por realidad aumentada, hemos requerido el uso de herramientas de modelado 3D como Blender para la escultura y texturización del brazo, y quien dice del brazo dice de cualquier cosa imaginable, donde un artista 3D, desde un modelo básico puede modificarlo para representar lo mismo que se aprecia en una radiografía o se puede obtener directamente el modelo 3D utilizando instrumentación especial que permite esto, como las famosas ecografías 3D que devuelven el modelo del feto aún en el útero materno. Estos modelados se adjuntan a una imagen en 2 dimensiones a través de la librería Vuforia para Unity y junto con algún sencillo script y una conversión a través de Android Studio a librería del formato AAR, obtenemos una nueva aplicación que puede ser embebida en la nuestra de urgencias.

Los modelos 3D son actualizables además sin necesidad de reinstalar la aplicación, ya que Vuforia posee una nube con capacidad para almacenar estos duos marcador-modelo y que el dispositivo, configurado para trabajar contra la nube, pueda descargarselos automáticamente al iniciar la aplicación o al detectar el marcador.

Para los visitantes, también existe la posibilidad de recibir cierta información a través de la realidad aumentada. En este proyecto se ha replicado una planta del hospital mediante arte pixel con MagicaVoxel que permite ver, enfocando a un plano de los que se encuentran por el hospital, el mismo en 3 dimensiones, facilitando la orientación de nuestra localización respecto del punto donde queremos ir.

3.3 Los servicios en la nube

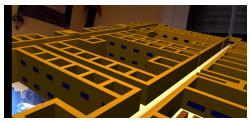
Para la confección de este proyecto, debido a estar orientado a una infraestructura crítica y puesto que por mi empleo tengo muy presente la tecnología de computación en la nube, se ha optado por utilizar esta en todo lo posible. Entre los motivos se encuentran los contratos de uptime 99,99 %, es decir, la aplicación siempre va a estar funcionando aunque apague mi ordenador, ofrecen servicios siempre actulizados, donde puedo desentenderme de la gestión de los mismos y centrarme más en su uso y en el código. Y posiblemente es el futuro de toda arquitectura productiva a la que queramos acceder desde cualquier lugar.



(a) Pantalla de información sobre el servicio de urgencias

(b) Disclaimer de condiciones de uso

Figura 3.7: Proceso de declaración de emergencia sobre la aplicación



(a) Modelado de planta hospital en 3D mediante MagicaVoxel mostrado a través del aplicativo

(b) Modelado de fractura antebrazo en 3D mediante blender mostrado a través del aplicativo

Figura 3.8: Muestra de infromacón mediante realidad aumentada

3.3.1 Google Cloud Platform

En GCP hemos utilizado las famosas Functions. Las Functions son servicios de tipo PaaS en las que nosotros subimos nuestro código, con una estructura especial que requieren en la que se define el punto de entrada a la función o manejador y se establece un tipo de trigger o disparador del mismo. Cuando el disparador se activa, la petición es redireccionada hasta la function correspondiente y por tanto al código, que comienza a ejecutarse, estando parado hasta ese momento. Esto nos permite ahorrar en costes de tener un servidor corriendo continuamente, porque además, pagamos solo por el tiempo que se haya estado ejecutando la función y en fracciones de milisegundos, por lo cuál si nuestra función tardara en ejecutarse 300 milisegundos, solo pagaríamos esos 300 milisegundos.

Por otro lado estas funciones están siempre listas y no fallarán, a nivel infraestructura, puesto que se replican en varias zonas de disponibilidad, que en el mundo real son diferentes data centers, sin necesidad de cambiar el nombre del endpoint, puesto que balancea automáticamente y siempre bajo las últimas versiones y por https, evitando fallos de seguridad o vulnerabilidades por dependencias anticuadas.

3.3.2 Push Notifications

El servicio de notificaciones push, como ya hemos indicado antes consta de dos conexiones y siguen el mismo principio que el protocolo MQTT explicado durante la teoría. Existe un broker de mensajes, que será el servidor de notificaciones, un cliente de publicación que corresponde a uno de los métodos de la GFunction EmergDialer y un cliente de suscripción, que corresponderá al dispositivo móvil.

La diferencia con MQTT es que el broker de mensajes establece una comunicación con el dispositivo

móvil para saber si este se encuentra escuchando, y si lo está le otorga la notifiación, siendo este el que despierta al dispositivo, o al revés, el dispositivo no necesita hacer un polling de información porque no es el que inicia la conexión.

3.3.3 MongoDB Atlas y Redis Cloud Labs

Las bases de datos, cuanto menos se toquen, más replicadas estén y más actualizadas, mejor. A través de servicios en la nube conseguimos esto y mejor aún si son las propias empresas dueñas de estas tecnologías las que nos ofrecen el mismo servicio pero en su nube.

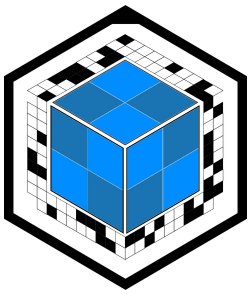
MongoDB Atlas nos permite establecer un lugar donde inicializar nuestros clústeres replicados y no preocuparnos más que por establecer roles, normas ACL y nombre a nuestras bases de datos y colleciones, siendo ellos los que se preocupan de las métricas, de mantenerlo actualizado y encendido y de no perder nuestros datos. Exactamente la misma situación ocurre con Redis, con la única diferencia que el almacenamiento aquí se realiza sobre memoria y no sobre discos, por lo que es un servicio algo más caro pero igual de ventajoso y necesario. Toda la operativa se realiza a través de endpoints y librerías diseñadas para establecer conexión con estos servicios sin problema alguno.

Figura 3.9: Panel de métricas de Redis Labs Cloud

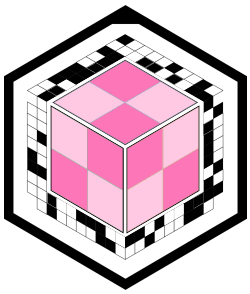
3.3.4 Vuforia Cloud

Vuforia funciona con archivos que forman bases de datos propias, como los antiguos archivos de Microsoft Access pero especialmente codificados para este servicio. Estos archivos pueden ser generados en su propia nube y guardados en el dispositivo o marcados como archivos de acceso a través de Internet, para los cuales vuforia habilita endpoints bajo autenticación, que es lo único que necesitaría nuestro aplicativo de Unity.

En estas bases de datos podemos almacenar los pares marcador-modelo, permitiendo actualizar nuestra aplicación directamente cambiando archivos en la nube y no necesitando que el usuario vuelva a descargarse el aplicativo o tenga que actualizarlo desde la store o parecidos.



(a) Objetivo de Vuforia utilizado para traumatismos



(b) Objetivo de Vuforia utilizado para maternidad

Figura 3.10: Dos objetivos para la cámara de realidad aumentada de Vuforia

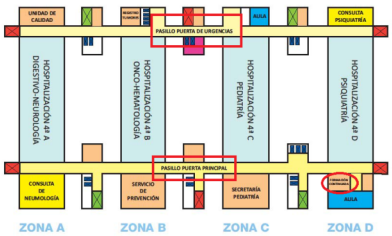


Figura 3.11: Objetivo de Vuforia utilizado para mostrar el plano del edificio

Capítulo 4

Resultados

Me ha costado ponerlo a punto, pero el resultado ha superado todas mis esperanzas.

Boris Vian, La espuma de los días

4.1 Introducción

Una vez realizado el desarrollo se dedeben establecer unas bases de experimentación. Estas vienen dadas por la naturaleza del proyecto, en la que no es posible implementarlo en un hospital real por motivos obvios. Por tanto, se ha creado un entorno de experimentación con simulaciones de tiempo reales de atención a un caso personal y se comprobará el mismo contra el mismo caso real pero usando la aplicación, simulando las características del hospital que no puedan ser comprobadas físicamente.

4.2 Entorno y resultados experimentales

Para la experimentación se utiliza cómo marca de referencia actual, los tiempos de espera de 2 casos reales personales, uno por un accidente de tráfico con alta médica en el mismo día y otro por una rotura de antebrazo completa de un familiar cercano con hospitalización de 3 días antes del alta hospitalaria

4.2.1 La atención en urgencias. Resultados y metodología

Durante el primer caso, la atención por accidente de tráfico, sucedió por la mañana a las 9:00 horas, de camino al trabajo, siendo de prioridad alta, de nivel 3 por la naturaleza del mismo, la llegada al hospital fue aún más larga de lo habitual debido a la entrada de todo el mundo a la universidad y al trabajo, tardando 7 minutos en recorrer la salida desde la A2 hasta la entrada en el recinto universitario del puente sobre las vías de la estación de Renfe, una vez allí, esperé la cola de registro e ingreso de 4-5 personas y en la sala de espera tardé un total de 12 minutos en entrar al servicio de triaje, a cotninuación 30 minutos para ser atendido por la unidad de radiología, con solo un instrumento de rayos operativo. Y otros 25 minutos en ser atendido por una especialista de trauma tras recibir las placas. Obtuve el alta del servicio de urgencias con un montón de recetas para pastillas antiinflamatorias y un collarín, junto con la necesidad de pedir una cita para rehabilitación y fisioterapia por el seguro debido a la saturación actual del servicio público. En este caso gestioné solo la urgencia avisando a mis padres tras confirmar

en el servicio de urgenicas que no había sido grave más allá de algúna contusión y contracturas de cuello y espalda, asi que no hubo acompañantes.

En el segundo caso la situación cambia, mis padres se encontraban en casa, en un pueblo cercano a Alcalá, yo en el centro de Alcalá y mi hermano en la calle del pueblo con un longboard. Mi hermano se cayó del mismo apoyando la mano y se fracturó completamente cúbito y radio. Mis padres recogieron a mi hermano y se lo bajaron al hospital, sin tráfico ya que era ya de noche. Sin embargo, al llegar al hospital, siendo una atención más grave que la mía, tardó incluso un poco más que yo en ser atendido en triaje, unos 15 minutos, debido a que acababa de pasar otro accidentado por fractura y no tenían recursos en ese momento para atender otra simultáneamente. Mientras estaban de camino al hospital, recibí la llamda de mi madre con la noticia y puse rumbo al hospital. Una vez allí no los encontré por ningún lado que busqué de vestíbulo y salas de espera, por supuesto no respondían al teléfono por estar en silencio y ocupados con mi hermano, hasta que en determinado momento salió mi padre por la puerta y ya me contó qué estaba pasando, dónde estaba mi hermano y cómo estaba. cómo hemos dicho, aunque había poco tráfico, ese día el servicio de urgencias tenía alta ocupación, y que justo antes acudiera otra persona con la pierna rota por un segway no ayudaba. Tuvimos que esperar después de rayos, a que un traumatólogo de guardia acudiera al servicio de urgencias a colocarle el brazo para volver a radiología, en total 40 minutos, de todo esto me enteraba mediante comunicaciones esporádicas de mi padre que de algúna forma se las arreglaba para entrar y salir del área de urgencias. Por últmo, antes de dejarle sedado en los boxes de urgencias y poderle ver por primera vez por una enfermera a la que parece que le di pena al verme preocupado, le pasaron a la sala de yesos, otros 40 minutos.

Ahora comprobemos rápidamente los tiempos utilizando el desarrollo. En primer lugar el desplazamiento desde la zona cercana al hospital y el hospital, reducimos en el caso A, 7 minutos el acceso a urgencias, mientras que no reducimos nada en el caso B ya que no existe tráfico, simplemente aportamos seguridad en la rotonda de acceso. Por otro lado y utilizando Google Maps para la rotonda del ensanche, en el momento de más tráfico registrado, sin accidentes ni obras, el recorrido desde el supermercado Mercadona, el más cercano al hospital, hasta el mismo hospital es de 24 minutos, mientras que en el de menos tráfico son 5, reduciendo en casi 20 minutos el tiempo de llegada al hospital de forma segura.

Por otro lado, el ingreso y la recepción de la pulsera y pegatinas, junto con la documentación relativa al accidente, debido a las reclamaciones que luego realizan a la aseguradora desde el hospital, fue de unos 2-3 minutos por persona, a 4-5 personas delante de mi, hace un total de 10 minutos de espera y 3 de mi propio registro. En el caso B la espera fue mínima, 2 minutos del propio registro, puesto que no había nadie. Con el frontdesk, la espera hubiera sido menor en el caso A, 2 ó 3 máquinas ya habían reducido los tiempos a la mitad suponiendo que tardaran lo mismo que el registro con una persona, pero es que además ya tienen los datos preparados, por ende el tiempo es igual a lo que tarde el usaurio en introducir un número no mayor a 3 cifras y que la máquina imprima los dispositivos RFID, documentación, etiquetas, o lo que necesite, por lo tanto, le quitamos a mi caso 10 minutos más de espera.

Ya para terminar, al estar avisado el hospital de la llegada del caso B especialmente, no habría que esperar los 40 minutraumtos o al menos no completamente, puesto que ya podrían haber reorganizado recursos desde que la persona con la pierna rota aparece delante, pudiendo llamar al especialista de guardia antes, para que la atención sea prácticamente inmediata en nuestra llegada o al menos la mitad de los 40 minutos en el peor de los casos, recuperando 20 minutos más en la atención tras radiología y otro tanto en el enyesado.

Sumando estos tiempos, obtenemos que en el caso ideal en el que unamos las ganancias de ambos casos obtenemos una atención de 1 hora más rápida, o lo que es lo mismo, visto desde la perspectiva del hospital, liberan recursos 1 hora antes que actualmente.

Tabla 4.1: Comparativa tiempo real y tiempo experimento.

Entorno	Real (Máx. A/B)	Proyecto
Llegada al hospital	20 minutos	3 min
Registro/Ingreso	10 min/3 min	3 min
Sala de espera general	15 min	10 min
Sala de espera específica	40 min	20 min
Box urgencias	40 min	20 min

Debemos tener en cuenta que, a la ganancia de estos tiempos, también le quitamos la preocupación de saber cómo y dónde está nuestro familiar, gracias al sistema de localización y estado del mismo, que nos indica el tiempo real de la intervención que están realizando y dónde se encuentra a través de las notificaciones push, algo que tranquiliza a todo el mundo, ya que el miedo y los nervios vienen del desconocimiento, es decir, de la falta de información.

4.2.2 El entorno de realidad aumentada

Para probar el servicio de realidad aumentada se ha seguido un experimento de tipo más social. Aprovechando la participación en una organización social que existe en mi lugar de residencia, donde hacemos diversas actividades para todos los rangos de edad, he hecho un pequeño experimento, explicando a dos grupos, de 6 personas con varios rangos de edad, desde los 12 años hasta los 70 años, una operación un poco compleja de ojo, posteriormente al otro grupo se le explicó la misma operación, pero mostrando por pantallá el modelo 3D del ojo pudiendo mostrar cada parte, por dentro y fuera y cómo sería la intervención paso a paso sobre el modelo.

Después de la explicación se les realizaron unas preguntas relacionadas con la intervención y con la explicación dada. El primer grupo, en las edades adultas la intervención se entiendió correctamente, mientras que en los otros dos grupos de edad, las respuestas no eran del todo acertadas, con falta de detalle en el tipo de intervención del que se hablaba y sin entender realmente cómo se iba a realizar la intervención, cómo por ejemplo que el ojo se sacaba para operarlo. En el segundo grupo, con el modelo 3D delante, todos los grupos de edad entendieron la operación correctamente salvo uno de ellos que no comprendía algún detalle concreto, pero en definitiva muestra cómo es más sencillo y rápido la explicación con este sistema por realidad aumentada.

4.2.3 Métricas de calidad

Los indicadores de calidad del servicio de urgenicas vienen definidas por diferentes estudios que realizan desde el Ministerio de Sanidad, desde el servicio SEMES, SUMMA y los diferentes SAMUR-PC. Debido a la complejidad del servicio de urgencias, debemos atender solo a las métricas que tratan con el proyecto, entre estas, destacan las de tiempos me permanencia e información a pacientes y familiares. [20]

4.2.3.1 Tiempo medio de primera asistencia facultativa

Esta métrica mide el tiempo medio que tardan los pacientes en recibir la primera asistencia por un médico desde que accede al servicio de urgencias, excluyendo el contacto no asistencial con el médico que realiza

un triaje inicial en el registro. El tiempo se expresará en minutos de media y no contarán aquellos casos en los que la consulta se anule o interrumpa por parte del usuario o facultativo.

La fórmula de cálculo de esta métrica corresponde a

$$m = \frac{\sum x}{y}$$

con x correspondiente al tiempo de asistencia de la primera asistencia e y el número total de pacientes atendidos en el periodo de tiempo correspondiente al sumatorio.

4.2.3.2 Tiempo medio de permanencia en urgencias

Correspondiente al tiempo total que permanece el paciente en el servicio de urgencias desde que llega hasta que abandona el mismo por algún motivo. Esta medida tiene cómo unidades minutos de permanencia de forma media, sin contar las mismas exclusiones anteriores.

La fórmula para esta métrica corresponde a

$$m = \frac{\sum x}{y}$$

con x correspondiente al tiempo que un paciente gasta en emergencias e y el número total de pacientes atendidos en el periodo de tiempo correspondiente al sumatorio.

Con estos datos se propone además realizar dos indicadores de tiempo, uno correspondiente a pacientes con más de 3 horas y otro de 6 horas de atención o más.

4.2.3.3 Información a pacientes y familiares

Mientras que las medidas anteriores eran cuantificables, debemos conocer que también existen otro tipo de medidas no cuantificables que deberán ser recogidas por el propio hospital y afectan directamente al comfort e información de la que disponen los usuarios. Por ejemplo, con estos puntos medimos el hecho de que el paciente y sus familiares han sido informados acerca de la identificación del médico responsable, el motivo de la consulta y el plan terapéutico si existiera. Para realizar esta medición, al no ser matemática contaremos con los siguientes puntos:

- Existencia de un plan de información del servicio de urgencias. Con ello, el usuario puede conocer cómo se asigna la prioridad de atención en el hospital y por qué fases va a pasar hasta ser dado de alta.
- Existencia de información previa sobre el fucionamiento del servicio de urgencias. Entre otros, cómo solicitar asistencia de urgencias extrahospitalaria, cómo es el uso de ambulancias o asistencias en casa, qué información debe poseer antes de solicitar la asistencia y otros.
- Existencia de formularios para recoger la satisfacción ante diversos eventos que puedan sucederse en el hospital. A modo de feedback para el hospital, permitiendo detectar los puntos débiles del mismo para cubrirlos y seguir retroalimentándose.

La publicación de este tipo de información revierte directamente en la facilidad de movimiento dentro del mismo hospital, junto con la gestión de citas en centros de especialidades y otras necesidades hospitalarias, por lo tanto influye en la calidad del hospital y de los sistemas de información tanto como los otros.

Capítulo 5

Conclusiones y líneas futuras

En este apartado se resumen las conclusiones obtenidas y se proponen futuras líneas de investigación que se deriven del trabajo.

5.1 Conclusiones

Tras el análisis del proyecto podemos realizar diversas conclusiones respecto a los diversos campos que se han tocado durante el estudio teórico y el desarrollo.

En primer lugar, podemos concluir que el valor de los dispositivos IoT, así como de los sensores que los acompañan, están cobrando cada vez más valor y esto a su vez permite avances con dispositivos más pequeños y precisos. Estos dispositivos permiten la interconexión de prácticamente cualquier cosa con la red, por lo que también debemos tener en cuenta las implicaciones que esto puede tener frente a la privacidad y control sobre nuestros datos. Entre los ejemplos de privacidad violada aparecen los distintos contratos de uso abusivos sobre los datos o incluso actualmente se encuentra de moda el uso de pequeños dispositivos IoT escondidos en paquetes que llegan a empresas y se encargan de realizar un espionaje empresarial, intentando explotar vulnerabilidades automáticamente desde dentro de la empresa.

Por otro lado, según el ritmo de mejora de conexiones que llevamos, imaginamos que la tecnología IoT irá ampliándose y mejorando también en concepto de comunicaciones, pudiendo dejar atrás en algún momento protocolos livianos y básicos como MQTT y estandarizando otros que permiten muchas más funcionalidades, como es actualmente AMQP, permitiendo arquitecturas mucho más complejas, pero que a su vez permiten un mayor avance y el desarrollo de tecnologías nuevas. [21]

Tabla 5.1: Tecnologías alternativas a MQTT. Cuadro comparativo

Protocolo	Patrón	QoS	Seguridad	Direccionamiento	RESTful
MQTT	Publish/Suscribe	3 niveles	TLS/SSL x.509	Por Tema	No
CoAP	Request/Response	Opcional	DTLS	URL	Sí
AMQP	Publish/Suscribe	Personalizable	TLS - SASL	Por dirección	No
HTTP/2	Request/Response	TCP QoS	TLS/SSL	URL	Sí
STOMP	Publish/Suscribe	Simple	Ninguno	Servidor	No

En el campo de la computación en la nube, son cientos los servicios que ya se nos permite usar sin disponer físicamente de los recursos computacionales, entre ellos el renderizado de videos para grandes películas, el uso de GPUs dedicadas a análisis experimental, mucho más barato y rápido que realizar los experimentos físicamente, o la ampliación de los sistemas informáticos de una empresa que comience con pocos recursos pero quiera optar a mayores proyectos para crecer más rápidamente sin realizar una fuerte inversión inicial. Sin embargo, son muchos más los servicios que aún no se permiten en la nube pero que poco a poco van siendo posibles gracias a la investigación y desarrollo, nosotros hemos dado ejemplos de uso en este proyecto como ejecución de código en funciones sin manejo de servidores, bases de datos no transaccionales y en memoria, sistemas de notificaciones con broker centralizado y conexión de dispositivos IoT mediante hubs en la nube como Amazon Greengrass o Azure IoT Hub. Ejemplos de tecnologías que están en los últimos momentos de desarrollo y que antes no existían es por ejemplo, el streaming de videojuegos por suscripción, que saldrá bajo la nube de Google y con el nombre de Stadia, permitiendo que todos puedan jugar a los últimos lanzamientos solo con una pantalla y un dispositivo IoT conectado a la misma y al servicio cloud de Google.

Por último, hablamos de realidad aumentada, un concepto que últimamente vuelve a cobrar fuerza ya que están apareciendo proyectos en plataformas de kickstarter que tienen como tecnología principal esta tecnología y en la integración de la misma en las gafas. Esta tecnología, que aparece en todas las películas de ficción, pues quién no se sorprende con la imaginación depositada en el diseño de los dispositivos de Batman o Iron Man, está cada vez más cerca de ser un habitual, pero para ello se requieren sensores mejores y pequeños, como cámaras, micrófonos o GPS y sobre todo, una implementación de machine learning muy reducida, que permita la identificación del contexto rápido para poder llevarse a cualquier campo sin necesidad de hacer una programación ad hoc para el motivo que se cree el sistema, es decir, crear un sistema de realidad aumentada comercial. Aún así, considero que es una de las tecnologías que más desaprovechadas están ahora mismo ya que es más fácil la inmersión, o realidad virtual donde todo está controlado por un programa lo que limita el contexto a lo que nosotros hayamos programado, sumado a los problemas enumerados anteriormente sobre los sensores y capacidad de inteligencia que debe tener esta tecnología.

5.2 Líneas futuras

Como cualquier proyecto de investigación, este posee gran cantidad de líneas futuras y presentes por donde ampliar el trabajo o desarrollo del proyecto ya que se utilizan tecnologías en pleno desarrollo, algunas de estas son:

- El desarrollo de dispositivos IoT específicos para hospitles que puedan integrarse en las camillas, habitaciones o instrumentación médica. Dicho de otra forma, ser capaces de integrar el llamado Edge Computing a los dispositivos de campo de un hospital.

- Una propuesta de migración de los servicios informáticos de una empresa para ser completamente de filosofía cloud, para ello se deberá tener en cuenta la naturaleza de la empresa y comprobar que todos los servicios que necesita pueden ofrecerse desde la nube. Sabemos que una empresa que esté presente en la nube, está localizada de forma mucho más fácil en el mundo, lo cual le da ciertas ventajas frente a otras.

- Realizar una gestión más extensa con realidad aumentada, creando por ejemplo un sistema de guía para interiores de edificios, la ampliación de cartelería de forma automática mediante búsquedas

sobre el contexto analizado, esto es, si la cámara lee un cartel de «Urgencias - Sala de espera», muestre también cómo se encuentra en ese momento la lista de llamadas del mismo.

Bibliografía

[1] V. CEOE, *Plan digital 2020: La digitalización de la sociedad española*, CEOE, 2016.

[2] S. Bail, "Comparison of the 3 leading cloud players," no. 1, Marzo 2019.

[3] "Organización Mundial de la Salud: Emergencias," https://www.who.int/topics/emergencies/es/ [Último acceso 13/Julio/2019].

[4] I. N. de Estadística y Ministerio de Sanidad y Consumo, "Población que ha utilizado algún servicio de urgencias en los últimos 12 meses por sexo, nivel de estudios del sustentador principal y número de visitas a urgencias," INE, Tech. Rep., 2019.

[5] S. E. nola de Medicina de Urgencias y Emergencias, "Criterios de acreditación de servicios de urgencias de hospitales," SEMES, Tech. Rep., 2017.

[6] G. de expertos de hospitales espa noles, *Unidad de urgencias hospitalarias. Estándares y recomendaciones*, ser. Unidad de urgencias hospitalarias. Ministerio de Sanidad y Política Socia, 2010.

[7] I. E. Nursing, "Efficacy of the Manchester Triage System: a systematic review," no. 23, Abril 2015.

[8] S. B. y. O. B. Pradyumna Gokhale, "Introduction to iot," no. 5, Enero 2018.

[9] "Introduction to iot," https://geeksforgeeks.org/introduction-to-internet-of-things-set-1/.

[10] D. Evans, "How the next evolution of the internet is changing everything," Abril 2011.

[11] P. A. N. y. S. P. S. Shubhangi A. Shinde, "MQTT-message queuing telemetry transport protocol," no. 3, Enero 2016.

[12] T. K. y. T. M. C. Adams, S. Farrell, "Internet x.509 public key infrastructure certificate management protocol (cmp)," in *RFC 4210*, 2005.

[13] "Characteristics of sensors," https://www.electrical4u.com/characteristics-of-sensors/.

[14] J. W.Gopel, J.Hesse, *Sensors, A comprehensive Survey, Vo.l 3.* VHCl, 1991.

[15] T. G. y W.H. Ko, *Sensors, A comprehensive Survey Vol. 1.* VHCl, 1991.

[16] E. y CEPT, *THE EUROPEAN TABLE OF FREQUENCY ALLOCATIONS AND APPLICATIONS IN THE FREQUENCY RANGE 8.3 kHz to 3000 GHz (ECA TABLE).* Electronic Communications Committee (ECC) within the European Conference of Postal and Telecommunications Administrations (CEPT), 2019.

[17] Y. D. y. L. G. Ling Qian, Zhiguo Luo, "Cloud Computing: An overview," 2009.

[18] "Regulation (eu) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (General Data Protection Regulation) (text with EEA relevance)," https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC [Último acceso 13/Julio/2019].

[19] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[20] G. de Trabajo SEMES-Insalud, "Calidad en los servicios de urgencias. Indicadores de calidad," 2001.

[21] B. Moyer, "All about messaging protocols," Abril 2015.

[22] "Raspbian, the Foundation's official OS for Raspberry Pi," https://www.raspberrypi.org/downloads/raspbian/.

[23] "Windows 10," https://www.microsoft.com/es-es/windows.

[24] "Visual Studio Code. Code editing. Redefined," https://code.visualstudio.com/.

[25] "Unity editor 2018, via Unity Hub," https://unity3d.com/es/unity/editor.

[26] "NodeRed, the programming tool for wiring things together." https://nodered.org/.

[27] "Android Studio 3.5: The official IDE for android," https://developer.android.com/studio/index.html?hl=es-419.

[28] "Visual Studio 2019, community edition," https://visualstudio.microsoft.com/es/vs/.

[29] "MagicaVoxel, voxel art editor and GPU based path tracing render," https://ephtracy.github.io/.

[30] "Blender Editor for Windows. open source 3d creation," https://www.blender.org/.

[31] "Plantillas para TFG de la UAH," https://www.depeca.uah.es/index.php/docencia-menu.

[32] "Nodejs 10.16.3 LTS, javascript execution context," https://nodejs.org/es/.

[33] "Azure DevOps, the collaboration tool for teams," https://azure.microsoft.com/en-us/services/devops/?nav=min.

[34] "Gradle Management and Build tool," https://gradle.org/.

[35] "Google Cloud Platform, your cloud computing provider," https://cloud.google.com/gcp/.

Apéndice A

Programación y Algoritmos

A.1 Introducción

En este apéndice se pretende mostrar algunas porciones de código que se han desarrollado, como ejemplo de partes funcionales del trabajo, así como ciertos algoritmos. No podemos mostrar todo el código debido a que la cantidad de líneas programadas haría de este imposible de leer y entender. Se realizará una explicación mejor del código a través de los comentarios que se encuentran en los fragmentos, mientras que se explicará la función que realizan en el escrito que se encuentre sobre ellos.

A.2 Programación

A.2.1 Dispositivos IoT

Las programación contenida en esta subsección se ejecuta en pequeñas placas NodeMCU, que tienen un procesador compatible con los protocolos y lenguaje de Arduino y se programan mediante el IDE del mismo.

En el siguiente fragmento de código, podemos ver como se utilizan las diferentes entradas del dispositivo IoT NodeMCU o Weemos D1 para la gestión de los diferentes semáforos, colocando las salidas del mismo en 1 ó 0.

En este primer fragmento, indicamos al procesador con WiFi como debe iniciarse, qué salidas vamos a usar y cómo conectarse a nuestra red.

Listado A.1: Inicialización NodeMCU para control de tráfico

```
#include <ESP8266WiFi.h>

WiFiServer server(80);

...

void setup() {
  //Indicamos la velocidad de escritura por puerto serie que vamos a utilizar, nuestro
      dispositivo Weemos D1 soporta 115200
  Serial.begin(115200);

  // Indicamos que salidas vamos a controlar desde el programa
  pinMode(output5, OUTPUT);
  pinMode(output4, OUTPUT);


  // Dejamos las salidas anteriores sin corriente
  digitalWrite(output5, LOW);
  digitalWrite(output4, LOW);

  // Conectamos al WiFi con ssid y password, que se han omitido en el fragmento de c\'
      odigo
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println(WiFi.localIP());
  server.begin();
}
```

En esta segunda parte, se observa como se manejan los datos que son enviados desde el cliente hasta el dispositivo, para posteriormente activar o desactivar en consecuencia las diferentes incorporaciones a la rotonda.

Listado A.2: Tratamiento de los datos enviados por el cliente al dispositivo IoT

```
void loop(){
  //Abrimos el servidor y manejamos las conexiones en la variable client
  WiFiClient client = server.available();
  //Si client deja de ser null, un cliente se ha conectado al dispositivo
  if (client) {
    //creamos una variable para almacenar los datos que el cliente nos env\'ie
    String incomingData = "";
    currentTime = millis();
    previousTime = currentTime;
    //Establecemos un tiempo de timeout y comprobamos si el cliente esta ese timepo sin
        enviar mensajes para cerrar la comunicaci\'on
    while (client.connected() && currentTime - previousTime <= timeoutTime) {
      currentTime = millis();
      if (client.available()) {
        //Si el cliente nos ha enviado datos, los leemos y almacenamos
        char c = client.read();
        header += c;
        if (c == '\n') {
        //Pasamos las cabeceras hasta que llegamos a un salto de l\'inea que...
        //indica el cuerpo del mensaje o el final del mismo.
            if (header.indexOf("GET /allowPreference/1") >= 0) {
                //Ejecutamos acciones dependiendo de la petici\'on recibida
                ...
            }
```

Otro de los dispositivos IoT que tenemos se encarga de controlar las distintas antenas RFID que habría en el hospital, así como de mandar estas actualizaciones a las bases de datos a través de las Google Functions. Debido a la falta de antenas RFID que podía instalar sin aumentar el presupuesto, se ha instalado un demodulador de infrarrojos que a través de un mando hará las veces de antena RFID que recibe una señal.

Listado A.3: Obtención de datos de los sensores y envío a la nube para actualización de la base de datos

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <IRremoteESP8266.h>

//Creamos todas las conexiones que vamos a utilizar del NodeMCU
int RECV_PIN = D2; //an IR detector/demodulatord is connected to GPIO pin D2
int ledBoxes = D8; //a boxes led is connected to GPIO pin D8
int ledRooms = D7; //a rooms led  is connected to GPIO pin D7
int ledTriaje = D1; //a triaje led  is connected to GPIO pin D6
int ledEntry = D5; //a entrada led  is connected to GPIO pin D5
int ledInfo = D4; //a info led  is connected to GPIO pin D4

//Creamos las variables para el env\'io de peticiones por WiFi
WiFiClient espClient;
//Y para el demodulador de infrarrojos
IRrecv irrecv(RECV_PIN);
decode_results results;

long lastMsg = 0;
char msg[50];
int value = 0;
// Por \'ultimo, para crear el mensaje que vamos a enviar
String header;
HTTPClient http;

//La funci\'on para conectar el WiFi como antes
void setup_wifi() {
   delay(100);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
      delay(500);
    }
  Serial.println(WiFi.localIP());
}


void setup() {
  Serial.begin(115200);
  setup_wifi();
  pinMode(ledBoxes, OUTPUT);
  pinMode(ledRooms, OUTPUT);
  pinMode(ledTriaje, OUTPUT);
  pinMode(ledEntry, OUTPUT);
  pinMode(ledInfo, OUTPUT);
  digitalWrite(ledBoxes, HIGH);
  digitalWrite(ledRooms, HIGH);
  digitalWrite(ledTriaje, HIGH);
  digitalWrite(ledEntry, HIGH);
  digitalWrite(ledInfo, HIGH);
  //Arrancamos el demodulador una vez hemos seteado todas las salidas
  irrecv.enableIRIn();
}

void loop() {
  long now = millis();
  if (now - lastMsg > 2000) {
     lastMsg = now;
     //Cada 2 segundos decodificamos lo que tengamos almacenado en el demodulador IR
    if (irrecv.decode(&results))
    {
      irrecv.resume(); // Recibimos el valor emitido por el mando
       //Actuamos en consecuencia, creando la petici\'on y enviandola
```

A.2.2 Google Functions

La programación que podemos encontrar durante esta subsección permanece ejecutándose en los servicios cloud de Google, por lo que forman parte del modelo IaaS, al ser programadas con Visual Studio Code y simplemente subidas a un repositorio de Google Functions en la nube.

También podemos ver el modo de programación que requieren las Google Functions en Javascript, a través de NodeJS. Podemos ver como la función requiere exportar un método de entrada, y nosotros aprovechamos esa entrada, para llamar a otras funciones del código según entren por un método HTTPS u otro, es decir, gestionamos una función distinta por GET, PUT, DELETE y POST, ya que no necesitamos más.

Listado A.4: Manejador de eventos y punto de entrada de la GFunction Squawker

```
/**
 * Responds to a request with new identifier, based on method. Forbids other request.
 *
 * @example
 * gcloud functions call squawker
 *
 * @param {Object} req Cloud Function request context.
 * @param {Object} res Cloud Function response context.
 */
exports.squawker = (req, res) => {
  switch (req.method) {
    case 'GET':
      handleGET(req, res); //Return the value for the identificator
      break;
    case 'POST':
      handlePOST(req, res); //Return a new ticket and set the identificator on redis
      break;
    case 'PUT':
      handlePUT(req, res); //Resets Tickets to 0
      break;
    case 'DELETE':
        handleDELETE(req, res); //Deletes from Redis a identificator when is used in
            frontdesk
        break;
    default:
      res.status(405).send({error: 'Something blew up!'});
      break;
  }
```

Aquí vemos uno de los métodos, el POST, que utilizamos para retornar un nuevo ticket al usuario para que lo introduzca en el frontdesk de urgencias, también se gestiona que el usuario no tenga una urgencia ya en marcha y que no se exceda del número de ticket 999.

Listado A.5: Manejador de evento POST y lógica de trabajo de la función

```
function handlePOST(req, res) {
    //Utilizamos el contenido de la request para establecer nuestra variable
    let identification = req.body.identification;
    let redisClient    = redis.createClient({port: 10426, host: 'redis
        -*****.***.**-*****-*-*.ec2.cloud.redislabs.com', password: '
        **************************'});
    //Nos conectamos a Redis Cloud
    redisClient.on('connect', function() {
            //Obtenemos el valor si existiera para no devolver otro, si no existiera se
                crea uno nuevo
          redisClient.get('ticket', function (error, result) {
            var parsedResult = parseInt(result, 10);
            //Comprobamos que no excede el m\'aximo de tickets
              if (parsedResult >= 999) {
                redisClient.set('ticket', '000');
                let newTicket = 000;
                //Creamos el valor en redis
                redisClient.set(identification, newTicket, redis.print);
                //Cerramos la conexi\'on con Redis Cloud
                redisClient.quit();
                //Lo devolvemos al cliente
                res.status(200).send('Identificator: ' + newTicket);
              }else if(error) {
                throw error;
              }else{
                let newTicket = parsedResult+1;
                redisClient.set('ticket', newTicket);
                redisClient.set(identification, newTicket, redis.print);
                redisClient.quit();
                res.status(200).send('Identificator: ' + newTicket);
              }
            });
            //Manejamos errores si ocurrieran
          redisClient.on('error', function(err) {
              redisClient.quit();
              console.log('RC -> Squawk 7600, '+ err);
          });
    });
  }
```

En esta otra Google Function correspondiente al gestor de eventos del hospital, EmergDialer, gestionamos las conexiones con MongoDB Atlas y además utilizamos el servicio de notificaciones push que se implantó en Android, para que lsa actualizaciones lleguen al mismo de modo inmediato, a la vez que se guardan en MongoDB. En este caso, la function trabaja de cliente de publicación sobre el topic del broker de notificaciones push.

Listado A.6: Manejador de evento POST y lógica de trabajo de la función EmergDialer

```
function handlePUT(req, res) {
    //Obtenemos los datos de la request y configuramos el canal de notificaciones push
    let identification = req.headers.identification;
    let update = null;
    var channels_client = new Pusher({
      appId: '8****2',
      key: 'c************1',
      secret: '8******************d',
      cluster: 'eu',
      encrypted: true
    });

    //Generamos una funci\'on especial que permite reutilizar conexiones de MongoDB
    co(function*() {
      if (client == null) {
        //Esta conexi\'on se puede reutilizar bajando dr\'asticamente los tiempos de
            ejecuci\'on
        client = yield mongodb.MongoClient.connect(uri);
      }

      let filterQuery = { dni: identification };
      //Actualizamos los valores en MongoDB seg\'un lo que hayamos recibido, pudiendo
          actualizar m\'as de 1, por eso concatenamos IFs
      if(req.headers.location != null){
        update = { $set: { "location": req.headers.location }};
        channels_client.trigger('patientData', 'newLocation', {
          "location": req.headers.location
        });
      }
       if(req.headers.actualState != null){
        update = { $set: { "actualState": req.headers.actualState }};
        //Generamos un mensaje-evento para notificaciones push, y lo enviamos
        channels_client.trigger('patientData', 'newState', {
          "actualState": req.headers.actualState
        });
      }
       if(req.headers.actualDiagnosis != null){
        update = { $set: { "actualDiagnosis": req.headers.actualDiagnosis }};
      }
       if(req.headers.dietToday != null){
        update = { $set: { "dietToday": req.headers.dietToday }};
      }
       if(req.headers.dietTomorrow != null){
        update = { $set: { "dietTomorrow": req.headers.dietTomorrow }};
      }
       if(req.headers.room != null){
        update = { $set: { "room": req.headers.room }};
        channels_client.trigger('patientData', 'newRoom', {
          "room": req.headers.room
        });
      }
      //Realizamos la actualizaci\'on con la query
      const docs = yield client.db('test').collection('emergencies').updateOne(
          filterQuery, update);
      res.send(200);

    }).catch(error => {
      res.send('Error: ' + error.toString());
    })
  };
```

A.2.3 Android código y XMLs

En el apartado de Android, podemos ilustrar algunas partes del código realizado en Andorid Studio, un código que se ejecutará como aplicativo en el dispositivo móvil Android del que dispongamos, como por ejemplo la gestión de peticiones HTTP a través de Volley, la libería recomendada para Android en estos casos, también se puede mostrar como se gestiona de forma independiente el hilo de «User Interface».

Listado A.7: Peticiones asíncronas con Volley en Android

```
//Creamos una cola de peticiones de Volley
requestQueue = Volley.newRequestQueue(this);
//Indicamos la URL a la que vamos a enviar los datos
this.url = "http://********************.cloudfunctions.net/emergDialer";
//Obtenemos el intent para obtener acceso a los elementos gr\'aficos
Intent intent = getIntent();
if(intent.getStringExtra("viewRole").equals("admin")) {
    StringRequest stringRequest = new StringRequest(Request.Method.GET, url, new
        Response.Listener<String>() {
        @Override
        //Cuando recibimos una resupuesta, tratamos de obtener todos los datos
            del object que se nos responde
        public void onResponse(String response) {
            try {
                String newTimestamp, newDoctor, newRoom, newUrgency, newState,
                    newLocation;
                JSONArray jsonarray = new JSONArray(response);
                JSONObject patientDataObj = jsonarray.getJSONObject(0);
                if (!patientDataObj.getString("timestamp").equals("null")) {
                    newTimestamp = patientDataObj.getString("timestamp");
                } else {
                    newTimestamp = "Desconocido";
                }
                    ...

                }
                //Llamada a una funci\'on que se encargar\'a de actualizar el
                    hilo de interfaz de usuario
                setInitialData(newTimestamp, newDoctor, newRoom, newUrgency,
                    newState, newLocation);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {

        }
    }) {
        @Override
        //Seteamos el header del que queremos pedir informaci\'on
        public Map getHeaders() throws AuthFailureError {
            HashMap headers = new HashMap();
            headers.put("identification", userID);
            return headers;
        }
    };
    //A\~nadimos la petici\'on a la cola y esta la gestionar\'a en segundo plano
    requestQueue.add(stringRequest);
```

A continuación mostramos el otro lado de las notificaciones push, realizadas a través de un servicio cloud facilitan mucho la configuración y conexión gracias a que tienen una librería propia para ello.

Listado A.8: Suscripción como cliente y gestión del sistema de notificaciones push

```
//Pusher Subscribe options
            PusherOptions options = new PusherOptions();
            options.setCluster("eu");
            Pusher pusher = new Pusher("c**************1", options);
            //Nos suscribimos a uno de los canales
            Channel channel = pusher.subscribe("patientData");
            //Y a continuaci\'on emparejamos con los distintos eventos
            channel.bind("newLocation", new SubscriptionEventListener() {
                @Override
                //Cada vez que sucede un evento de un tipo al que estamos emparejado
                public void onEvent(String channelName, String eventName, final String
                    data) {
                    try {
                        //Ejecutamos una acci\'on y un cambio en la interfaz, obteniendo
                            los datos de la notificaci\'on
                        JSONObject wrapperObject = new JSONObject(data);
                        changeLocation(wrapperObject.getString("location"));
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            });

            channel.bind("newState", new SubscriptionEventListener() {
                @Override
                public void onEvent(String channelName, String eventName, final String
                    data) {
                    ...
                }
            });

            ...
```

En este otro fragmento se aprecia como se puede modificar el hilo de interfaz de usuario, ya que al trabajar con fragments, no podemos acceder a ellos desde la actividad principal, que es desde la que se lanzan las funciones por el usuario, que actualizan los campos del fragment, por eso utilizamos este hilo y no el propio fragment.

Listado A.9: Actualización del hilo de interfaz de usuario desde la actividad principal

```
    public void changeLocation(final String newLocation){
        //Nos movemos al hilo de interfaz de usuario
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                //Corremos la siguiente funci\'on en este hilo.
                fragment_hospitalization.setLocation(newLocation);
            }
        });

    }
```

Por último, una muestra del manifest de Android, siendo uno de los XML de configuración que

aparecen en un aplicativo android, junto con los layouts.

Listado A.10: Android Manifestl

```
<!--> Cabecera regular del manifiesto de Android <-->
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.medicalreport.medicsolutions.medicalreport">
    <!-->Permisos que vamos a utilizar y requerimos del tel\'efono <-->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <!-->Configuraci\'on general de la aplicaci\'on en cuanto a estilos , nombres y
    actividad principal<-->
    <application
        android:allowBackup="true"
        android:icon="@mipmap/icon"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/icon"
        tools:replace="android:icon,android:theme"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivityLogin" />
        <activity android:name=".WaitingActivity" />
        <activity
            android:name=".MainNavigation"
            android:label="@string/title_activity_main_navigation" />
        <activity
            android:name=".MainActivity"
            android:theme="@style/SplashTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".EmergencyData"
            android:label="@string/title_activity_emergency_data"></activity>
    </application>
</manifest>
```

A.2.4 Unity código y configuraciones

Unity requiere para usar la libreria de Vuforia de varias configuraciones, como los nombres de la base de datos, los secretos de conexión que crea la libreria y desarrollar algunos scripts para soportar los botones de un smartphone android, incluimos un ejemplo de ello. Remarcar, que este código genera una librería o aplicativo, según nos convenga, que puede ejecutarse en cualquier plataforma debido al motor multicompilación del que dispone Unity. En nuestro caso, generaremos código Android que se ejecutará sobre el dispositivo móvil, aprovechando su hardware y sencilla portabilidad.

Listado A.11: Soporte para el botón de atrás de cualquier smartphone

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SupportBackScript : MonoBehaviour {
 void FixedUpdate(){
  if (Application.platform == RuntimePlatform.Android)
  {
   if (Input.GetKey(KeyCode.Escape))
   {
    Application.Quit();
   }
  }
 }
}
```

A.3 Algoritmos

NodeRed se basa en la conexión de dispositivos y hardware de forma indiferente, para ello se crean algoritmos que se traducen en la consecución y cableado de cajas, que representan los diferentes dispositivos y conexiones.

Data: Salida del sensor de movimiento
Result: Cambio del estado de los semáforos o permanencia de los mismos
initialization;
if *sensorOutput >0* **then**
    activate usb camere and take photo;
    analize photo through LPR algorithm;
    if *license plate exists in database* **then**
        change IoT device outputs to give priority in car entrance by closing the others;
        wait for car to enter the roundabout;
        reset traffic ligths to carefull mode;
    else
        do nothing;
else
    do nothing;

Algoritmo A.1: Cómo controlar el tráfico

Otro algoritmo que podemos mostrar pertenece al sistema de localización de pacientes en el hospital

Data: Entrada de datos de la antena RFID
Result: Actualización de la localización del paciente
initialization;
while *patient gots rfid tag* **do**
    walk arround hospital;
    if *antenna detects tag* **then**
        take tag identifier;
        send tag identifier and antenna identifier to google function;
        google function decodes antenna identifier and tag identifier;
        if *antenna identifier is not the same as the last patient location* **then**
            take actual state of patient;
            if *if actual state is other intervention or pre-op* **then**
                change actual state to post-op or intervention using state machine (pre-op
                 ->surgery room location = intervention state : intervention ->other than surgery
                 room = post-op state;
            else
                do nothing;
            change location of patient to the new one;
        else
            do nothing;
    else
        do nothing;

Algoritmo A.2: Cómo detectar la posición del paciente

Apéndice B

Pliego de Condiciones

Las herramientas necesarias para la elaboración del proyecto han sido:

- PC con conexión a Internet y capacidad para modelado y renderizado de modelos 3D, necesitando para esto, como mínimo 8GB de RAM, procesador i5, 4 núcleos a 2.50Ghz de alto rendimiento (no bajo consumo) o AMD equivalente y 250 GB de HDD o SSD.
- Raspberry Pi 3B
- Terminal inteligente Android con API mínima 26
- Sensores varios y componentes IoT como Weemos D1
- Sistema operativo Raspbian para Raspberry Pi [22] y Windows 10 para PC [23]
- Entorno de desarrollo Visual Studio code [24]
- Entorno de desarrollo Unity[25]
- Entorno de desarrollo NodeRed [26]
- Entorno de desarrollo Android Studio [27]
- Entorno de desarrollo Visual Studio 2019 [28]
- Entorno de modelaje y renderizado MagicaVoxel[29]
- Entorno de modelaje y renderizado Blender[30]
- Procesador de textos LaTeX[31]
- Paquete de desarrollo NodeJS y NPM [32]
- Control de versiones, compilación y despliegues mediante Microsoft Azure DevOps[33]
- Gestor de tareas gradle [34]
- Plataforma de servicios y servicios en la nube, Google Cloud Platflorm [35]

Apéndice C

Presupuesto

Se muestra, seguidamente, una estimación de costes a modo de presupuesto del coste que supondría el desarrollo de este proyecto de modo profesional por una empresa que pueda proporcionar todos los servicios, desde el rol de integrador hasta desarrollador de soluciones, cubriendo un ala del hospital junto con el servicio de urgencias y de forma productiva y no de desarrollo como es el proyecto y con 2 años de soporte, incluidas suscripciones de Cloud de máxima disponibilidad, a renovar tras 2 años según mercado.

C.1 Coste Directo

Siguiendo normalmente una regla de precios conocido como «Keystone», pudiendo variar la ganancia del 50 % en algunos costes por algo superior o inferior, obtenemos los siguientes:

Tabla C.1: Costes directos del proyecto

	Producto / Recurso	Unidades	Coste Empresa (Euros)	Coste Cliente (Euros)
Materiales (IVA incluido)				
	Raspberry Pi 3B	1	34,00	62,00
	NodeMCU	4	28,00	70,00
	Antena RFID	10	65,00	119,00
	Pulsera Tag RFID	5.000	250,00	359,00
	Cámara Cloud	10	600,00	1099,00
	Cableado	1	400,00	800,00
Cloud Infraestructure Bianual (IVA incluido)				
	Functions	3+	PayAsYouGo	600,00
	IoT Hub	4	PayAsYouGo	200,00
	MongoDB Atlas AWS M10	1	2.000,00	3.750,00
	Redis Cloud Multi AZ	1	552,00	950,00
Humanos (IVA incluido)				
	Senior DevOps Engineer Hours	60	2.520,00	6.300,00
	Senior Develop Engineer Hours	120	1.620,00	4.000,00
Total	Electric Integrator Hours	8	72,00	180,00
	DevOps Engineer Hours Mantenimiento	30	750,00	1.875,00
	Develop Engineer Hours Mantenimiento	40	600,00	1.500,00
				21.864,00

C.2 Coste Indirecto

Aparece un coste indirecto en el proyecto debido a la necesidad de gestionar aspectos internos al hospital, como la creación de un formulario de registro con una correcta «letra pequeña» sobre ley de protección de datos, la gestión de la obra pública para la colocación de las cámaras, etcétera. Se propone un precio estimado para la empresa y un 25 % de sobrecoste para el cliente que suele ser la media de ganacias aplicadas a este tipo de coste.

Tabla C.2: Costes indirectos del proyecto

	Coste Empresa (Euros)	Coste Cliente (Euros)
Servicios externos varios	2.500,00	3.125,00
Total		3.125,00

C.3 Otros Costes

Enumeramos otros gastos que podrían añadirse a un presupuesto porque incurren en gastos hacia la empresa que deben ser tenidos en cuenta para que el cliente los cubra, pero que no podemos calcular en este momento por la dificultad de acceso y cálculo sobre estos datos.

C.3.1 Coste estructural

Coste derivado del alquiler de las oficinas, consumo de electricidad y otros del que cliente se le hace cargo proporcionalmente a las horas de trabajo.

C.3.2 Coste Amortizaciones

La devaluación de las herramientas de trabajo suponen una perdida de su valor, ya sea por su uso prolongado, desgaste u otros motivos. Este gasto se puede añadir a los costes de un proyecto, de forma proporcional.

C.4 Coste Total

El coste total sobre los importes que si se han podido calcular ascienden a ***24.989,00 euros***

Apéndice D

Manual de Ejecución

D.1 Introducción

Con este último apéndice se pretende otorgar al usuario una guía de despliegue del proyecto y ejecución del mismo. El flujo de uso del trabajo ya se explicó durante el capítulo de desarrollo mediante el ejemplo de Iris y sus familiares, por lo que aquí no abordaremos tanto su uso sino las necesidades y despliegues a completar antes del mismo

D.2 Requisitos de despliegue

Para el despliegue de los diferentes componentes del proyecto, primero se requiere la configuración de ciertas herramientas.

- Azure DevOps, siendo esta una herramienta que auna todas las necesidades DevOps básicas, desde SCM (Source Control Management o Version Control), CI/CD a webhooks y planes de testeo más complejos. Desde aquí realizaremos tanto el control de versiones como los depliegues a Google Cloud o a los diferentes elementos IoT.
- Google Cloud, nuestra suscripción en Google Cloud Platform para poder usar los servicios del mismo, como las fuinctions.
- Google Cloud SDK, para permitir el trabajo programático sobre GCP a través de scripts o command line.
- Azure DevOps agents, pequeñas piezas software de Microsoft, que se instalan en las máquinas que deseemos y las convierte en agentes de compilación y/o despliegue.
- Artefactos, siendo estos los resultados de las compilaciones o despliegues que realicemos.
- Terraform, conocimientos sobre esta plataforma que nos provee del llamado «Infraestructure as a code», es decir, creamos despliegues mediante programación.

Una vez disponemos de todo lo anterior, podemos proceder a la creación de «pipelines» para las compilaciones y despliegues de nuestro código a través de la plataforma Azure DevOps.

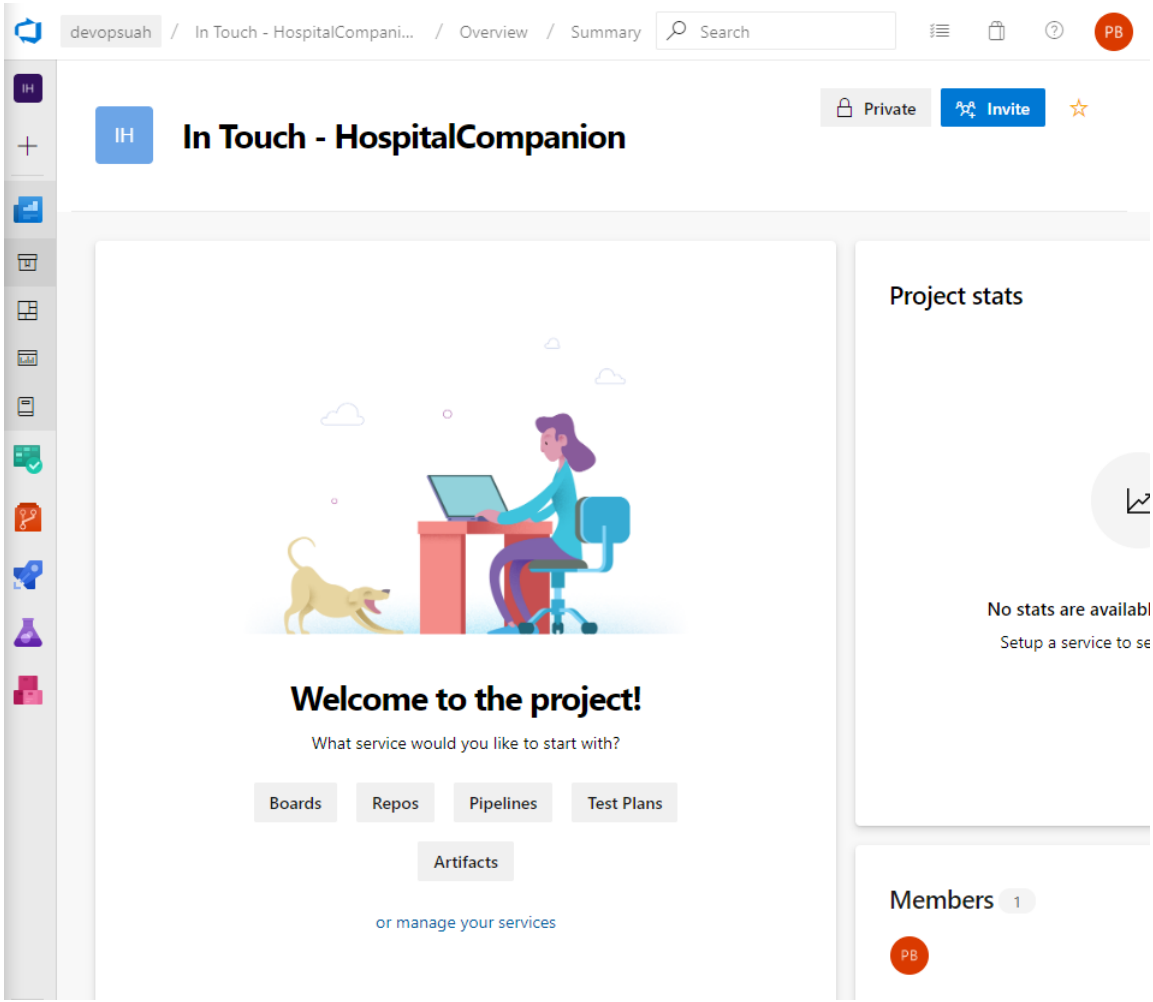


Figura D.1: Navegación principal de Azure DevOps

D.3 Azure DevOps

D.3.1 SCM

Dentro de la plataforma Azure DevOps existen unos servidores de almacenamiento que nos sirven con la capacidad de tener un control de versiones basado en Git. Para crearnos los diferentes repositorios donde almacenaremos el código a liberar, utilizaremos la interfaz de la que nos provee.

1. Acceder a la pestaña de Files, dentro del menú Repositories
2. En el menú superior, pulsar sobre el desplegable y a continuación sobre el botón «New Repository»
3. Completamos los datos que nos solicita, introduciendo el nombre que deseemos sobre nuestro proyecto, en el ejemplo usaré HC_EmergDialer_CloudFunction_NodeJS, utilizando la convención de nombres que deseemos.
4. Creamos las ramas correspondientes siguiendo las normas y estándar de GitFlow WorkFlow.
5. Cuando queramos liberar alguna versión, realizaremos un pull request de las diferentes features a develop, y de ahi extraeremos una rama release, que será la usada para deplegar. Para actualizar master, realizaremos el pull request de la rama release y un cherry-pick a develop si la rama hubiera sufrido algún cambio desde su creación.

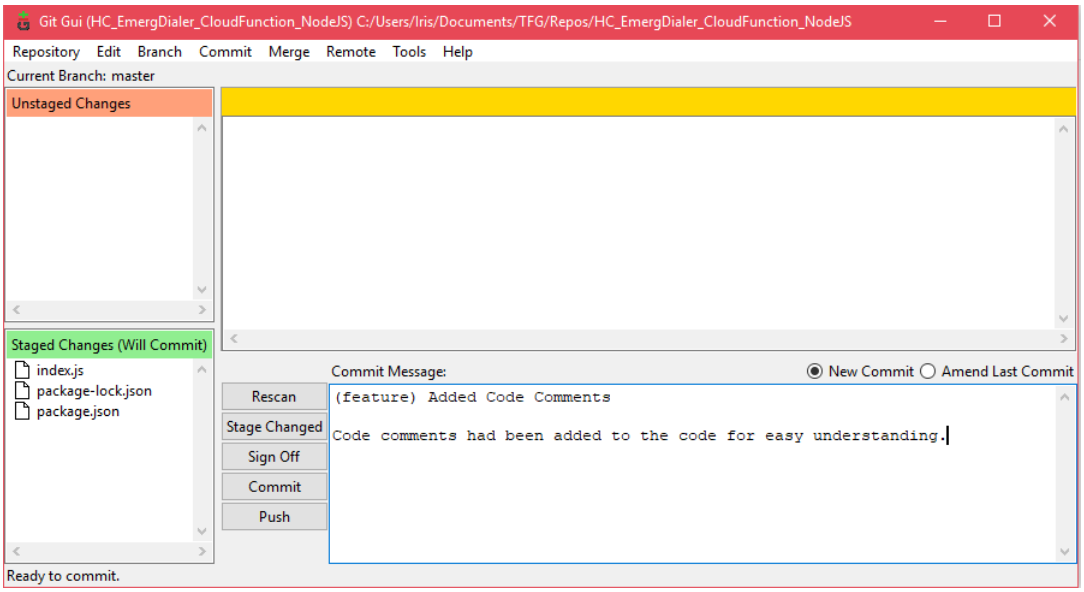


Figura D.2: Git commit y push a rama feature de Azure DevOps

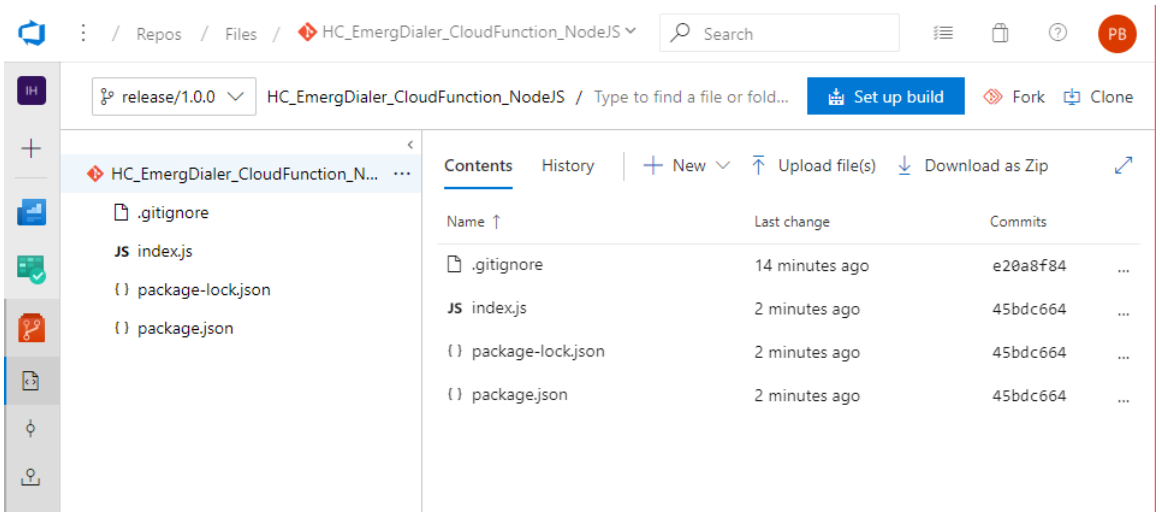


Figura D.3: Repositorios de Azure DevOps

Una vez disponemos de nuestro código en la nube, puede ser necesario compilarlo como el caso de Java o no, como sería Javascript.

D.3.2 Compilaciones

Para la creación de compilaciones desde Azure DevOps, seguiremos los siguientes pasos, una vez completados los del punto anterior. El siguiente ejemplo de creación de una pipeline de compilación corresponde a Java mediante Gradle.

1. Acceder a la pestaña de Pipelines, dentro del menú con el mismo nombre.
2. En el menú superior, pulsar sobre sobre el botón «New Pipeline» y seguidamente Azure Repos Git, junto con el repositorio y rama que queramos.
3. Empezamos con la opción «Empty Job» para que no nos introduzca tareas que no queremos usar automáticamente.

4. Seleccionamos la tarea de gradle y la configuramos para que ejecute la tarea que queramos, en nuestro caso la tarea de compilación la hemos llamado build.

5. Copiamos los artefactos generados mediante la tarea «Copy Files» y los publicamos en nuestro Azure DevOps para poder descargarlos si lo necesitamos.

6. Guardamos la pipeline y ejecutamos cuando necesitemos compilar, esta compilación se puede configurar a través de agentes para que se realice en máquinas propias o utilizar las que Microsoft pone a nuestra disposición desde Azure.

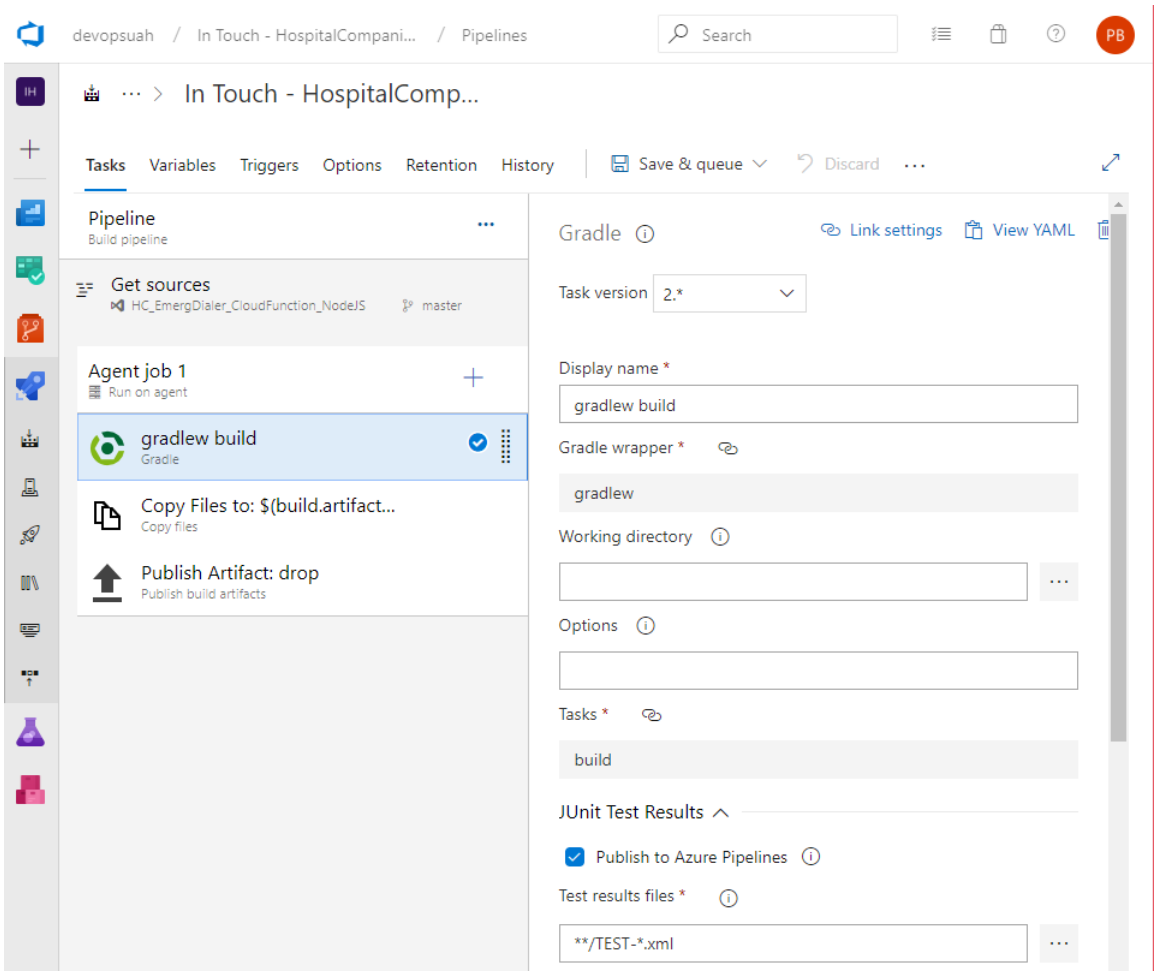


Figura D.4: Pipelines de Azure DevOps

D.3.3 Despliegues

Para realizar un despliegue necesitamos, o bien el compilado que se genera de una build del punto anterior, o el código que no necesita compilado de un repositorio de la primera sección. Una vez poseamos uno de los dos, podemos seguir los siguientes pasos para que el despliegue quede configurado con un botón, generando un entorno con Continuous Integration y Continuous Deployment.

1. Acceder a la pestaña de Releases, dentro del menú Pipelines.

2. Acceder a la pestaña de Releases, dentro del menú Pipelines.

3. En el menú superior, pulsar sobre sobre el botón «New Pipeline» y seguidamente la opción «Empty Job».

4. En el diagrama, añadimos artefactos, estos artefactos pueden ser o los resultados de la build anterior o código disponible desde nuestros repositorios, entre otros disponibles que no nos interesan ahora mismo. Completamos la configuración correspondiente. En nuestro ejemplo usamos el código subido a los repositorios.

5. Completamos el diagrama de artefactos y pasamos al de enviroments, para este ejemplo utilizaremos solo Producción, nombrado por nosotros, pero deberiamos establecer un entorno distinto para desarrollo y calidad. Completamos con tareas según necesitemos. Existen 2 lugares de despliegue, en cloud que es el que vamos a usar en este ejemplo a través de los agentes de Microsoft u on-premises, donde deberemos instalar el agente de DevOps que escribimos en los requisitos y seleccionarlo en la pipeline.

6. Para nuestro despliegue de la function, añadimos las tareas de Terraform, la primera que instale la última versión disponible de Terraform y las demás que configuran, validan y ejecuta el script que le indiquemos para realizar la release. Pero para esto necesitamos linkear nuestra nube de Google con Terraform, y para ello se crea un nuevo «service connection» desde Azure, obteniendo los datos de Google. Mostramos una imagen abajo de los campos de esta conexión sin completar debido a que no se deben compartir estos datos.

7. Guardamos la pipeline y ejecutamos cuando necesitemos deplegar, esta se realizará automáticamente sin necesidad de entrar en las másquinas, reconfigurar opciones o pulsar ningún botón adicional.

D.3.4 Otros

Igual que hemos realizado con estos ejemplos, también podriamos realizar despliegues y compilaciones para el resto de componentes, únicamente nos preocuparemos de buscar y utilizar las tareas adecuadas de entre las que nos ofrece Azure DevOps.

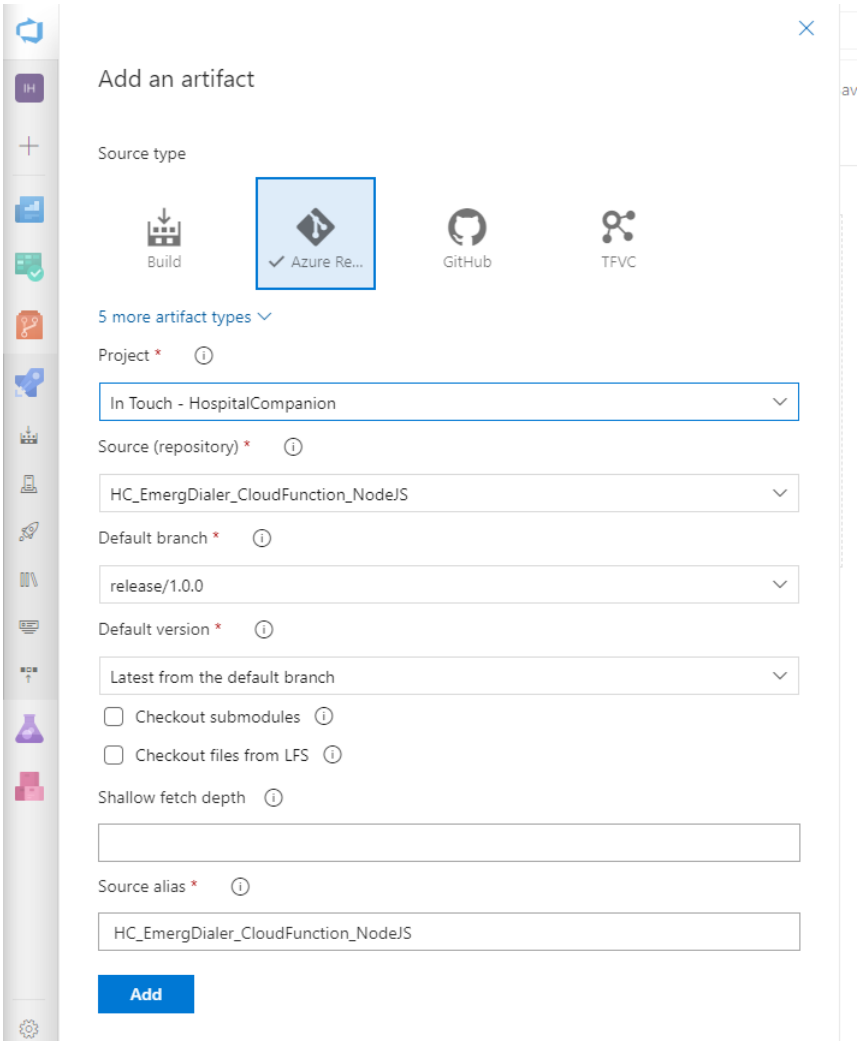


Figura D.5: Añadiendo artefactos a pipelines de despliegue

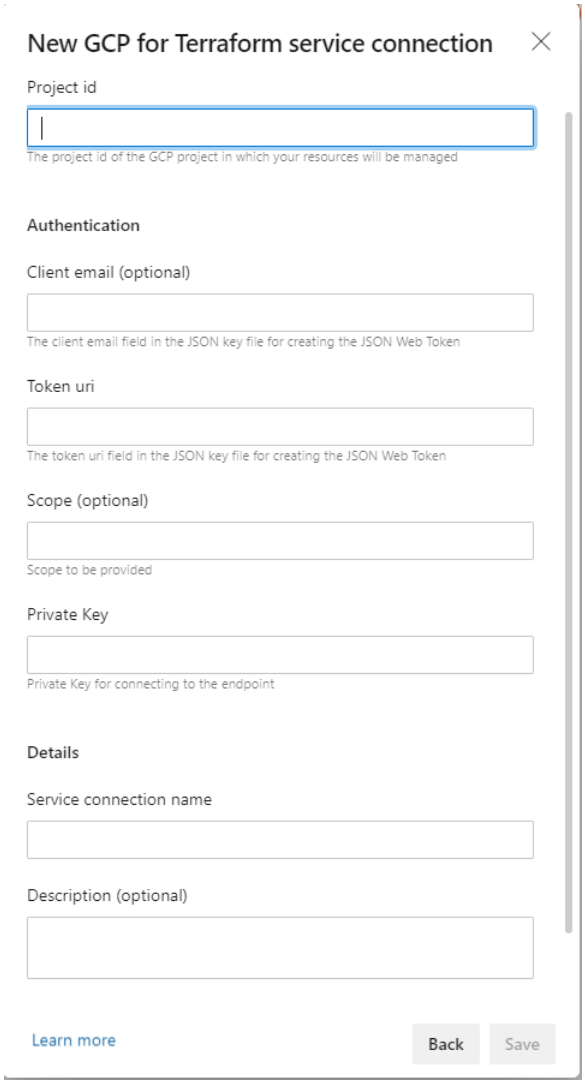


Figura D.6: Service connection para Terraform hacia GCP

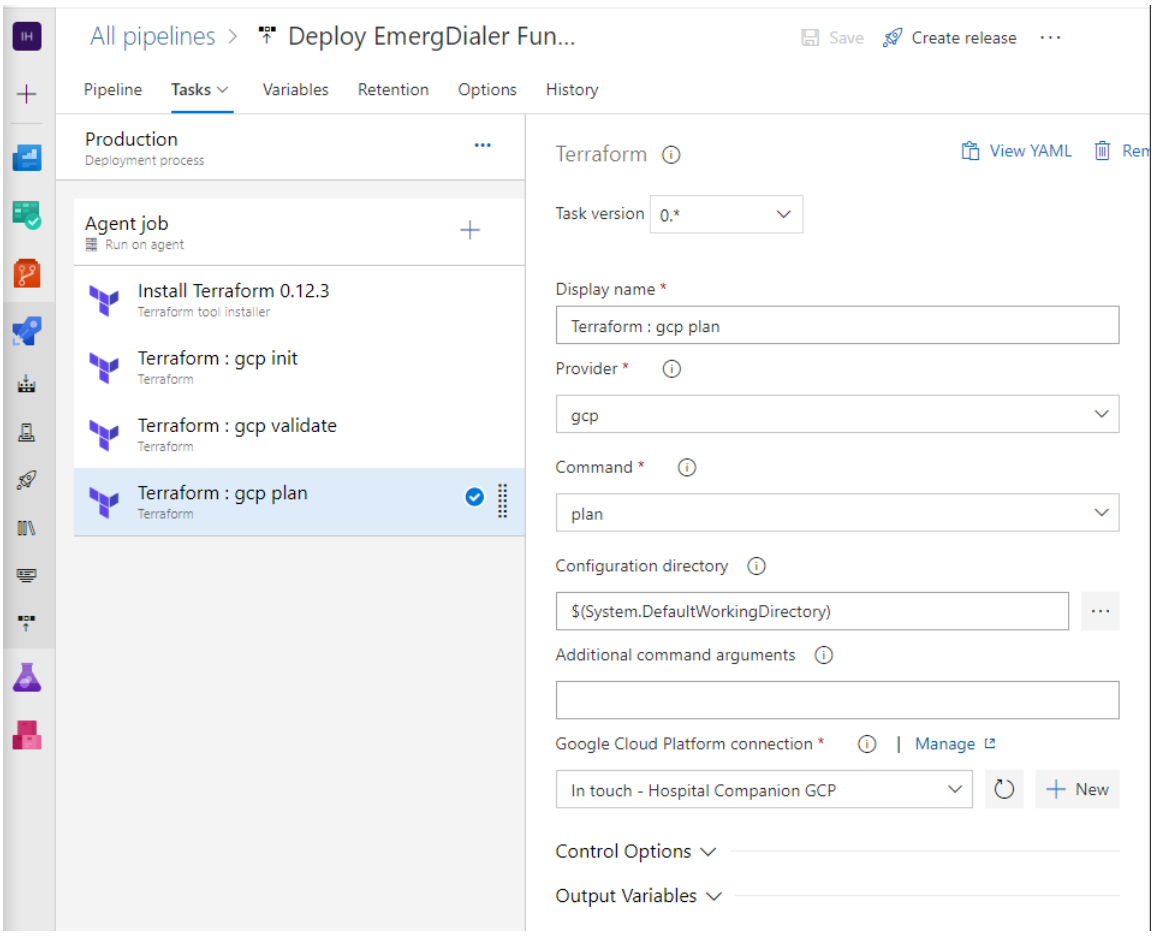


Figura D.7: Deslpiegue de Google Function mediante Azure DevOps

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá