industriales

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Sistema inalámbrico de monitorización cardíaco: Integración con dispositivos móviles inteligentes mediante el diseño de una aplicación en la capa Software as a Service (SaaS) en el entorno de Google Cloud.

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Sergio Sáez Sánchez**
Director: Juan Antonio López Riquelme
Codirector: Nieves Pavón Pulido

Cartagena, 10 Octubre de 2015

Universidad Politécnica de Cartagena

AGRADECIMIENTOS

- A Juan Antonio y a Nieves por su ayuda prestada durante la elaboración de este trabajo, incluso en días y horarios no lectivos.
- A mis padres por apoyarme, ayudarme y aconsejarme durante los estos años, y por no dejar nunca de creer en mí.
- A mi hermana Laura por estar siempre para desconectar y desahogarme cuando lo necesito.
- A Vanessa por estar disponible siempre y por guiarme en todos los sentidos.
- A mi club de triatlón, por todo el apoyo y los ánimos para acabar los estudios y por todas las horas que pasamos juntos. Especialmente en estos momentos a ti, Pablo, porque dentro de poco tenemos que estar entrenando juntos (#VamosPablo).

Índice

Índice de ilustraciones

Capítulo 1

Introducción

1.1 Planteamiento inicial del proyecto

El presente proyecto final de grado está enfocado al estudio de las posibilidades de almacenamiento y procesamiento que se pueden encontrar hoy en día en la nube, así como su relación con las aplicaciones móviles para smarthpones, más concretamente en el ámbito de la salud. Para poder cumplir el objetivo mencionado, se realizará el estudio de un caso práctico de una toma de datos de un sensor de frecuencia cardiaca, y se le aplicarán una serie de algoritmos para poder tener toda la información necesaria para el paciente.

Partiendo de la base de otro trabajo final de grado realizado por un compañero, en el cual mediante un microprocesador, un circuito electrónico y un sensor, es capaz de medir las pulsaciones de un paciente, se realizarán una serie de implementaciones en la nube escogida para el caso (Google cloud) y se utilizará la plataforma de Android para el desarrollo de una aplicación móvil.

De esta manera, se iran describiendo a continuación todo lo necesario para llevar a cabo este proyecto, desde todas las posibilidades que abarca el mundo de la computación

en la nube, las ventajas e inconvenientes, las limitaciones, etc. Hasta los procedimientos utilizados para la creación de dichos algoritmos.

El presente proyecto se ha llevado a cabo utilizando el sistema operativo de Windows 7 y tanto la aplicación para dispositivos móviles inteligentes como el tratamiento de los datos y almacenamiento en la nube, se ha utilizado el software Eclipse, en su versión 4.5 conocida como Eclipse Mars.

Por otro lado, la aplicación para dispositivos móviles inteligentes ha sido programada para móviles con sistema operativo Android.

1.2 Fases del proyecto

Para el correcto desarrollo del trabajo final de grado se han establecido las siguientes fases que se exponen a continuación:

1. Estudio del estado del arte en todos los campos que engloba el presente trabajo.
2. Estudio de la plataforma de Google Cloud.
3. Estudio de los algoritmos de detección de anomalías en la señal cardiaca.
4. Diseño de una aplicación en Android para enviar los datos a la nube de Google.
5. Diseño del front-end y el back-end de la aplicación de cloud computing necesaria.
6. Validación del sistema en un entorno real.
7. Redacción del documento final de memoria del trabajo.

1.3 Desarrollo de la memoria

La memoria de este trabajo se divide en cinco capítulos, en los que se describe en detalle y por partes cada sección del trabajo. A continuación se ofrece al lector un pequeño anticipo de lo que se encontrará posteriormente desarrollado en cada uno de los capítulos.

- Capítulo 1. Introducción.
  Este primer capítulo presenta de forma general el contenido del trabajo final de grado, indicando las fases en las que se puede dividir, desde el inicio hasta el final, de la realización de este.

- Capítulo 2. Estado del Arte.
  Este capítulo está dedicado al estudio de las herramientas utilizadas. Es por esto que se podrá observar una introducción al cloud computing así como sus características principales, los tipos de nubes, las capas, funcionalidades etc.
  De la misma manera, se hará una vista rápida al estado de los microprocesadores, móviles inteligentes y, para acabar, a los sensores de salud.

- Capítulo 3. Tecnología utilizada.
  Tal como indica el nombre de este capítulo, en este se describirán todos los elementos que intervienen, unido también a su instalación y puesta a punto.

- Capítulo 4. Estudio de un caso práctico.
  En este capítulo se redactará toda la información necesaria para poner en la práctica todos los conocimientos previamente adquiridos en los capítulos anteriores.
  Se dejará constancia por escrito de todos los pasos a seguir para la implementación del sistema final de este trabajo, con el fin de servir de ayuda y/o guía para futuros trabajo o investigaciones.

- Capítulo 5. Resultados y conclusiones.
  Como su propio nombre indica, en este capítulo se obtendrá un breve resumen de lo que se haya conseguido junto con los resultados obtenidos durante la realización de este trabajo final de grado. Se propondrán mejoras que puedan surgir a lo largo del desarrollo y se dejará una puerta abierta para futuras investigaciones.

Capítulo 2

Estado del Arte

2.1 Introducción

A lo largo de este apartado se va a definir el estado actual de la tecnología en la que se basa este trabajo final de grado.

Se van a estudiar en detalle todos y cada uno de los componentes utilizados y explicará el porqué de los materiales seleccionados para llevar a cabo este trabajo.

Por un lado, se tratará de explicar todo aquello relacionado con el cloud computing, seguido de los Smartphones, los microprocesadores y, para concluir, lo sensores de salud.

2.2 Cloud Computing

2.2.1 Introducción

Antes de comenzar este capítulo, es necesario aclarar el concepto de Cloud Computing.

Traducido al español como computación en la nube, servicios en la nube, informática en la nube, nube de cómputo o nube de conceptos, este término hace referencia a un paradigma que permite ofrecer servicios de computación a través de internet.

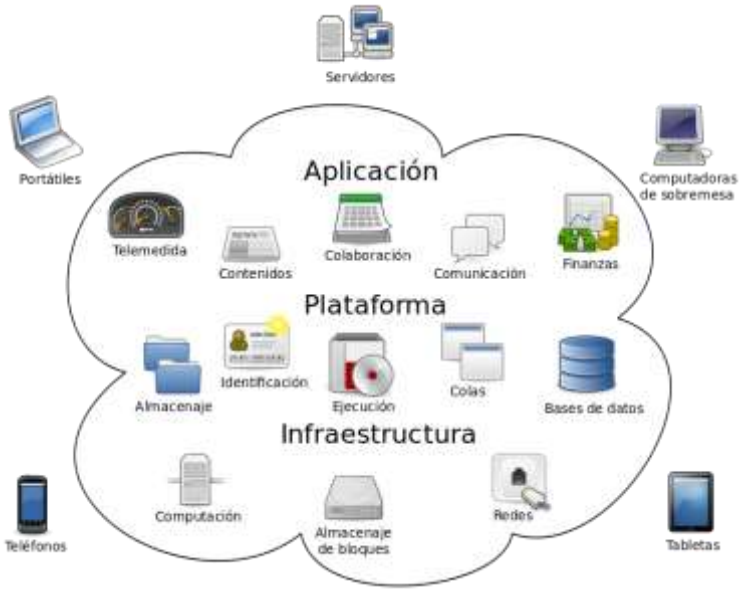

Ilustración 1: Computación en la nube

La principal ventaja de este tipo de computación es que los usuarios pueden acceder a los servicios disponibles en la nube sin apenas poseer conocimientos (o al menos sin ser expertos) en la gestión de recursos que usan.

Los servicios alojados en internet se encargan de atender las peticiones en cualquier momento. Se puede tener acceso a su información o servicio mediante una conexión a internet desde cualquier dispositivo móvil o fijo ubicado en cualquier lugar y sirven a sus usuarios desde varios proveedores de alojamiento repartidos frecuentemente por todo el mundo.

El cambio que ofrece la computación desde la nube es que permite aumentar el número de servicios basados en la red. Esto genera beneficios tanto para los proveedores que pueden ofrecer de una forma más rápida y eficiente un mayor número de servicios, como para los usuarios que consiguen la oportunidad de poder acceder a ellos.

2.2.2 Origen del Cloud Computing

La computación en nube ha recorrido un largo camino desde que fue marcada por primera vez como una perspectiva de futuro por parte de algunos investigadores. La historia inicial de la computación en nube nos lleva a finales del siglo veinte, cuando la

prestación de servicios de computación comenzó. Sin embargo el concepto se remonta a J.C.R. Licklider y John McCarthy.

El concepto básico del cloud computing o computación en nube se le atribuye a John McCarthy, responsable de introducir el término "inteligencia artificial". En 1961, durante un discurso para celebrar el centenario del MIT, fue el primero en sugerir públicamente que la tecnología de tiempo compartido (Time-Sharing) de las computadoras podría conducir a un futuro donde el poder del cómputo e incluso aplicaciones específicas podría venderse como un servicio (tal como el agua o la electricidad). Esta idea de una computadora o utilidad de la información era muy popular en la década de 1960, incluso algunas empresas comenzaron a proporcionar recurso compartidos como oficina de servicios, donde se alquilaba tiempo y servicio de cómputo. El sistema de tiempo compartido proporcionaría un ambiente operacional completo, incluyendo editores de texto y entornos de desarrollo integrado para lenguajes de programación, paquetes de programas informáticos, almacenamiento de archivos, impresión masiva y de almacenamiento offline. A los usuarios se les cobraba un alquiler por el terminal, las horas de tiempo de conexión, tiempo del CPU y kilobytes mensuales de almacenamiento en disco. Sin embargo, esta popularidad se desvaneció a mediados de los setenta cuando quedó claro que el hardware, software y las tecnologías de comunicación simplemente no estaban preparados.

Las empresas de telecomunicaciones hasta la década de los 90s eran quienes ofrecían redes privadas virtuales (VPN) con una calidad de servicio semejante, pero a un costo mucho menor. Al ser capaces de equilibrar el tráfico pudieron hacer uso del ancho de banda total de la red con mayor eficacia. Incluso el símbolo de la nube se utiliza para indicar el punto de demarcación entre lo que es la responsabilidad del proveedor y lo que era la responsabilidad del usuario. Ahora la computación en nube extiende este límite para cubrir servidores, así como la infraestructura de red.

Uno de los pioneros en la computación en nube fue Salesforce.com, que en 1999 introdujo el concepto de entrega de aplicaciones empresariales a través de una sencilla página web. Amazon era el siguiente en el tren, al lanzar Amazon Web Service en 2002. Entonces llegó Google Docs en 2006, que realmente trajo el cloud computing a la vanguardia de la conciencia del público. 2006 también vio la introducción de Elastic Compute Cloud de Amazon (EC2) como un servicio web comercial que permitió a las empresas pequeñas y particulares alquilar equipos en los que pudieran ejecutar sus propias aplicaciones informáticas.

Esto fue seguido por una colaboración de toda la industria en 2007 entre Google, IBM y una serie de universidades de los Estados Unidos. Luego vino Eucalyptus en 2008, como la primera plataforma de código abierto compatible con el API-AWS para el despliegue de clouds privados, seguido por OpenNebula, el primer software de código abierto para la implementación de nubes privadas e híbridas. Microsoft entraría hasta el 2009 con el lanzamiento de Windows Azure. Luego en 2010 proliferaron servicios en distintas capas de servicio: Cliente, Aplicación, Plataforma, Infraestructura y Servidor. En 2011, Apple lanzó su servicio iCloud, un sistema de almacenamiento en la nube, para

documentos, música, videos, fotografías, aplicaciones y calendarios, que prometía cambiar la forma en que usamos la computadora.

2.2.3 Principales características del Cloud Computing

Las principales características que presenta la computación en la nube son:

- Reducción de costes: Con la computación en la nube, abaratamos los costes asociados al alquiler o compra de servidores, el mantenimiento de este, etc.
- Accesibilidad: basta con tener conexión a internet para poder ejecutar las aplicaciones. No está asociado a un solo equipo si no que donde sea que esté el usuario, le basta con tener una conexión a internet.
- Elasticidad y escalabilidad: Las aplicaciones en la nube se adaptan al número de clientes que haya conectados en cada momento de manera automática. El usuario no tiene por qué preocuparse de ello.
- Recuperación: Los proveedores de servicios en la nube ofrecen herramientas para la recuperación de copias de seguridad realizadas de manera automática por el propio sistema.
- Seguridad: Puede mejorar debido a la centralización de datos. Es a menudo tan buena o incluso mejor que otros sistemas tradicionales ya que los proveedores son capaces de dedicar recursos a la solución de problemas de seguridad que muchos clientes no pueden permitirse abordar.
- Mantenimiento: es más sencillo ya que las aplicaciones no necesitan ser instaladas en el ordenador y cada usuario puede acceder desde diferentes lugares.

2.2.4 Ventajas e inconvenientes

La decisión de utilizar esta tecnología, al igual que en todos los ámbitos, conlleva asumir una serie de ventajas e inconvenientes a valorar para ver de verdad si resulta interesante. Entre las ventajas, las que más destacan son:

- Prestación de servicios a nivel mundial: las infraestructuras de cloud computing proporcionan mayor capacidad de adaptación, recuperación completa de pérdidas de datos mediante copias de seguridad realizadas de manera automática cada cierto tiempo, y reducción al mínimo de los tiempos de inactividad.

- Trabajar con infraestructuras de cloud computing permite al cliente prescindir de la instalación de software ya que este se encuentra disponible teniendo acceso a la nube. En muchos casos, esta característica puede reducir costes de inversión al empezar a trabajar.
- Instalación más rápida. Esta característica viene de la mano de la ventaja anterior puesto que al encontrarse disponible en la nube, desde el momento que se paga el servicio se tiene acceso inmediato a este, sin depender de que tenga que venir un técnico ajeno a instalarlo, ahorrando días y costes de trabajo.

- Contribuye al uso eficiente de energía puesto que la energía utilizada por las aplicaciones en la nube no son más que las requeridas por los servidores cuando estos están en funcionamiento.

En cuanto a las desventajas que encontramos al hacer uso de esta tecnología, pueden verse enumeradas en los siguientes puntos:

- La primera y más importante es que la accesibilidad a las aplicaciones está sujeta a la disponibilidad de internet.
- Aunque cada día se consiguen más avances en este ámbito, la seguridad es otro de los puntos a tener en cuenta. La información recorre varios nodos hasta llegar a su destino por lo que cada uno de estos nodos es un foco de inseguridad.

  Así mismo, los datos más importantes de cada empresa se encuentran alojados en la nube, lo que puede dar lugar a ser una vulnerabilidad para la sustracción de información.
- Por último, y como consecuencia de las dos desventajas anteriores, se puede nombrar una última desventaja y no es más que la dependencia que genera esta tecnología a los servidores de los proveedores, al estar todas las aplicaciones y el almacenamiento de datos centralizados.

2.2.5 Capas o modelos de servicio del cloudcomputing

De forma general podemos hacer una clasificación de los modelos de servicio tal como muestra la ilustración 2:

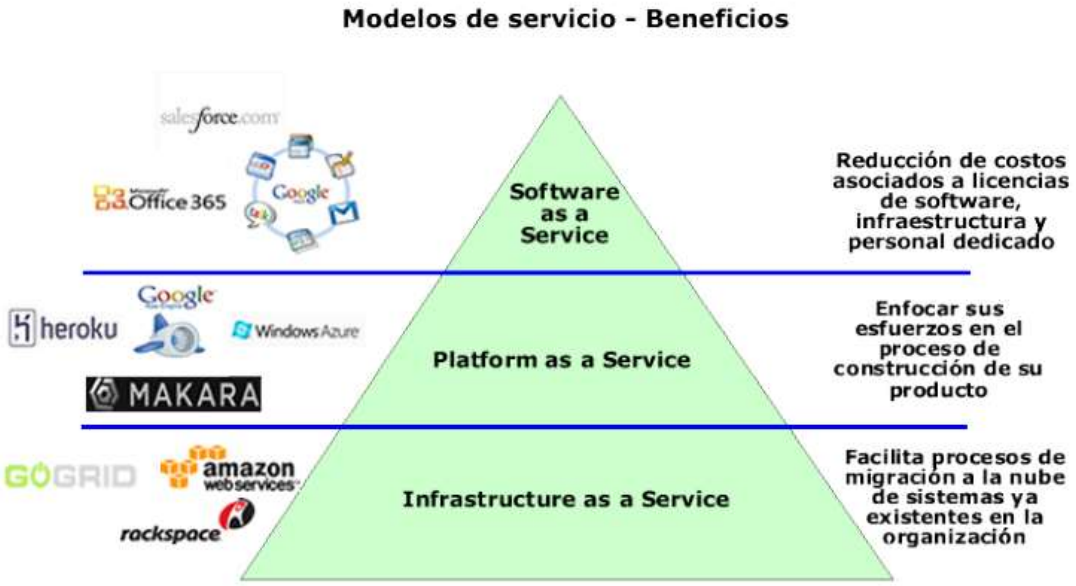

Ilustración 2: Pirámide de modelos de servicio en la nube

No obstante, se procede a definir cada uno de estos tres modelos para que quede claro la función de cada uno de ellos.

Software as a service (software como servicio)

Ilustración 3: Capa SaaS

Describe cualquier servicio cloud en el que los consumidores puedan acceder a aplicaciones de software a través de internet. Esas aplicaciones están alojadas en la nube y pueden utilizarse para una amplia variedad de tareas, tanto para particulares como para organizaciones. Google, Twitter, Facebook y Flickr son ejemplos de SaaS, en los cuales los usuarios pueden acceder a los servicios a través de cualquier dispositivo que pueda conectarse a internet. Los usuarios empresariales pueden utilizar aplicaciones para resolver necesidades muy diversas, desde la contabilidad y la facturación hasta el seguimiento de ventas, planificación, control de rendimiento y comunicaciones (por ejemplo, el correo web y la mensajería instantánea).

El modelo SaaS (Software as a Service) se conoce también a veces como "Software como servicio", y la forma de utilizarlo se parece más a alquilar el software que a comprarlo. Con las aplicaciones tradicionales, el software se compra al principio como un paquete, y una vez adquirido se instala en el ordenador del usuario. La licencia del software puede también establecer limitaciones en cuanto al número de usuarios y/o dispositivos en los cuales puede instalarse. Por el contrario, los usuarios del Software como Servicio se suscriben al software, en lugar de comprarlo, generalmente por períodos mensuales. Las aplicaciones se compran y utilizan a través de internet, y los archivos se guardan en la nube, no en el ordenador del usuario.

Platform as a service (plataforma como servicio)

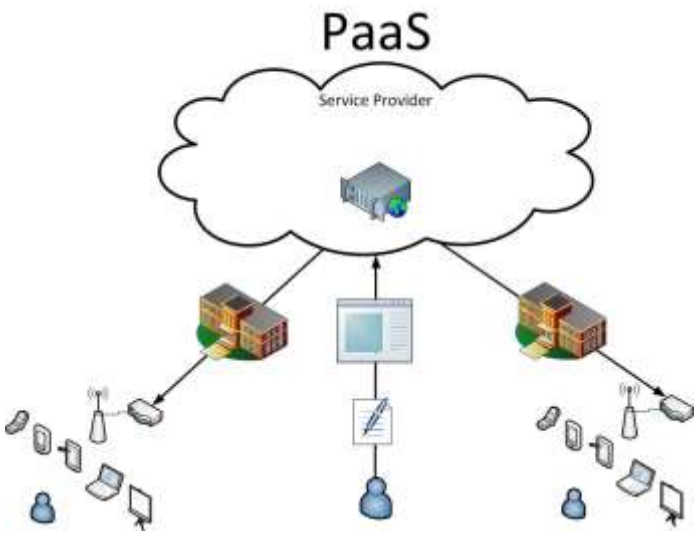

Ilustración 4: Capa PaaS

El concepto de Plataforma como Servicio (PaaS, Platform as a Service) es una categoría de servicios cloud que proporciona una plataforma y un entorno que permiten a los desarrolladores crear aplicaciones y servicios que funcionen a través de internet. Los

servicios PaaS se alojan en la nube, y los usuarios pueden acceder a ellos simplemente a través de su navegador web.

El modelo PaaS permite a los usuarios crear aplicaciones de software utilizando herramientas suministradas por el proveedor. Los servicios PaaS pueden consistir en funcionalidades preconfiguradas a las que los clientes puedan suscribirse, eligiendo las funciones que deseen incluir para resolver sus necesidades y descartando aquellas que no necesiten. Así, los paquetes pueden variar desde un sencillo entorno que se maneje con el ratón y no requiera ningún tipo de conocimiento o instalación especial por el lado del usuario, hasta el suministro de opciones de infraestructura para desarrollo avanzado.

Al igual que en la mayoría de las propuestas de servicios cloud, los servicios PaaS suelen facturarse como una suscripción en la que el cliente acaba pagando al final sólo por lo que realmente utiliza. Además, puede beneficiarse de las economías de escala que aporta el hecho de estar compartiendo una misma infraestructura física subyacente entre muchos usuarios, lo que se traduce en una reducción de costes.

El modelo PaaS aporta ventajas tanto a los desarrolladores de software como a los programadores de webs y a las empresas. Tanto si se trata de crear una aplicación que tengan previsto ofrecer a través de internet como de un software para vender en las tiendas, una solución PaaS proporciona grandes ventajas a un desarrollador de software. Por ejemplo, los desarrolladores para web pueden utilizar entornos PaaS diferentes en cada una de las fases del proceso de creación de sus webs, desde el desarrollo hasta las pruebas y su alojamiento final. Y también las empresas que desarrollan internamente su propio software pueden sacar partido al modelo de Plataforma como Servicio, por ejemplo para crear entornos de pruebas y de desarrollo completamente aislados entre sí.

Infrastructure as a service (infraestructura como servicio)

Ilustración 5: Capa IaaS

Infraestructura como servicio, es el último de los tres modelos de computación en la nube que quedan por estudiar.

La infraestructura como servicio (infrastructure as a service, IaaS) también llamado en algunos casos hardware as a service, HaaS) se encuentra en la capa inferior y es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red. Servidores, sistemas de almacenamiento, conexiones, enrutadores, y otros sistemas se concentran (por ejemplo a través de la tecnología de virtualización) para manejar tipos específicos de cargas de trabajo desde procesamiento en lotes ("batch") hasta aumento de servidor/almacenamiento durante las cargas pico. El ejemplo comercial mejor conocido es Amazon Web Services, cuyos servicios EC2 y S3 ofrecen cómputo y servicios de almacenamiento esenciales (respectivamente).

En el caso de Iaas el recurso de computación proporcionado es específicamente la de hardware virtualizado. Esta definición incluye servicios tales como espacio virtual del servidor, conexiones de red, ancho de banda, direcciones IP, etc. Físicamente, el grupo de recursos de hardware se obtiene de una multitud de servidores y redes normalmente distribuidos a través de numerosos centros de datos, los cuales tienen la responsabilidad de ser mantenidos por los proveedores. El cliente por su parte, tiene acceso a dichos componentes virtualizados con el fin de construir sus propias plataformas.

Al igual que con los otros dos modelos de cloud computing, el modelo de infraestructura como servicio puede ser utilizado por clientes empresariales para crear soluciones tecnológicas de bajo coste y fácilmente escalables, en las cuales la complejidad y los gastos de gestión del hardware corren a cargo del proveedor de la nube. De la misma forma, dichas soluciones son escalables por lo que siempre se pueden contratar más servicios en la nube con el fin de expandir, instalar o integrar diferentes software.

2.2.6 Tipos de nube

Existen diversos tipos de nubes atendiendo a las necesidades de las empresas, al modelo de servicio ofrecido y a como se desplieguen en las mismas.

Dependiendo de dónde se encuentren instaladas las aplicaciones y qué clientes pueden usarlas tendremos nubes públicas, privadas o híbridas, cada una de ellas con sus ventajas e inconvenientes.

Las nubes públicas, los servicios que ofrecen se encuentran en servidores externos al usuario, pudiendo tener acceso a las aplicaciones de forma gratuita o de pago. Se manejan por terceras partes, y los trabajos de muchos clientes diferentes pueden estar mezclados en los servidores, los sistemas de almacenamiento y otras infraestructuras de la

nube. Los usuarios finales no conocen qué trabajos de otros clientes pueden estar corriendo en el mismo servidor, red, discos como los suyos propios. La ventaja más clara de las nubes públicas es la capacidad de procesamiento y almacenamiento sin instalar máquinas locales, por lo que no tiene una inversión inicial o gasto de mantenimiento en este sentido, si no que se paga por el uso. La carga operacional y la seguridad de los datos (backup, accesibilidad, etc.) recae íntegramente sobre el proveedor del hardware y software, debido a ello, el riesgo por la adopción de una nueva tecnología es bastante bajo. El retorno de la inversión se hace rápido y más predecible con este tipo de nubes. A veces puede resultar difícil integrar estos servicios con otros sistemas propios.

Las nubes privadas, las plataformas se encuentran dentro de las instalaciones del usuario de la misma y no suele ofrecer servicios a terceros. Son una buena opción para las compañías que necesitan alta protección de datos y ediciones a nivel de servicio. Como ventaja de este tipo de nubes, al contrario que las públicas, es la localización de los datos dentro de la propia empresa, lo que conlleva a una mayor seguridad de estos, corriendo a cargo del sistema de información que se utilice. Incluso será más fácil integrar estos servicios con otros sistemas propios. Las nubes privadas están en una infraestructura local manejada por un solo cliente que controla qué aplicaciones debe correr y dónde. Son propietarios del servidor, red, y disco y pueden decidir qué usuarios están autorizados a utilizar la infraestructura. Sin embargo, como inconveniente se encuentra la inversión inicial en infraestructura física, sistemas de virtualización, ancho de banda y seguridad, lo que llevará a su vez a pérdida de escalabilidad y desescabilidad de las plataformas, sin olvidar el gasto de mantenimiento que requiere. Esta alta inversión supondrá un retorno más lento de la inversión.

Las nubes híbridas combinan los modelos de nubes públicas y privadas. Esto permite a una empresa mantener el control de sus principales aplicaciones, al tiempo de aprovechar el Cloud Computing en los lugares donde tenga sentido. Usted es propietario de unas partes y comparte otras, aunque de una manera controlada. Las nubes híbridas ofrecen la promesa del escalado aprovisionada externamente, en demanda, pero añaden la complejidad de determinar cómo distribuir las aplicaciones a través de estos ambientes diferentes. Una nube híbrida tiene la ventaja de una inversión inicial más moderada y a la vez contar con SaaS, PaaS o IaaS bajo demanda. En el momento necesario, utilizando las APIs de las distintas plataformas públicas existentes, se tiene la posibilidad de escalar la plataforma todo lo que se quiera sin invertir en infraestructura. Este tipo de nubes está teniendo buena aceptación en las empresas de cara a un futuro próximo, ya que se están desarrollando software de gestión de nubes para poder gestionar la nube privada y a su vez adquirir recursos en los grandes proveedores públicos.

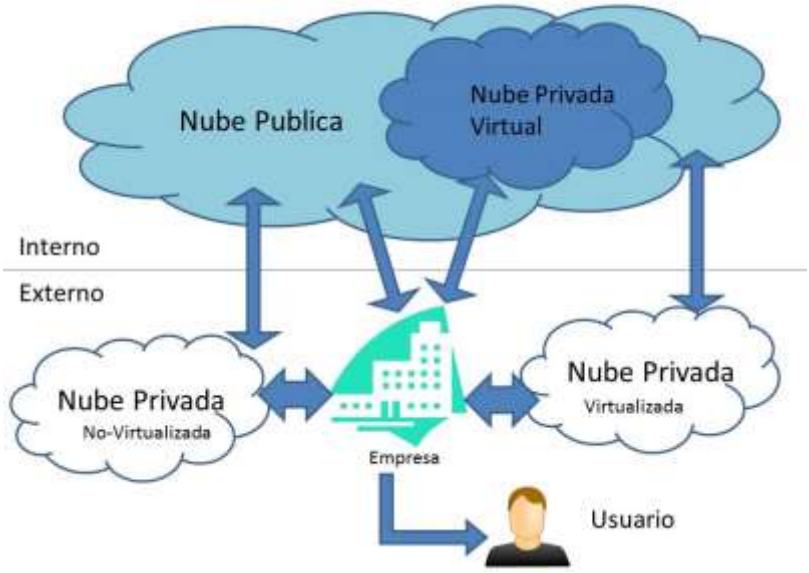

Ilustración 6: Tipos de nube

2.2.7 Plataforma del internet de las cosas

Existen diversas plataformas o ecosistemas en la nube que permiten el almacenamiento, visualización, gestión y tratamiento, entre otras funcionalidades, de la información enviada. Estos ecosistemas permiten crear aplicaciones, tomando como fuente los datos enviados por los dispositivos, sensores y actuadores conectados a la red. Son una parte clave del Cloud Computing, en concreto en el ámbito de las Smart Cities e Internet of Things, dado que ayudarán a procesar la ingente cantidad de datos disponibles para obtener resultados de gran utilidad a la hora de crear proyectos.

Actualmente los servicios que se suelen prestar se enumeran a continuación:

- Almacenamiento. Se ofrece almacenamiento, limitado por número de Gigabytes, de archivos en la nube para tener acceso a ellos desde cualquier sitio.

- Informática. Existe la posibilidad de tener alojada en la nube una computadora, generalmente representada como máquina virtual (VM) o instancia. Es posible escalarla para tener mayor capacidad o menos.

- Bases de datos. Almacenar información en bases de datos ubicadas en un servidor remoto.

- Aplicaciones Web. Se trata del uso de software sin la necesidad de tener instalado el software en nuestro equipo.

- Análisis de datos. Cualesquiera que sean los datos enviados a la plataforma, pueden ser analizados y procesados para obtener informes, gráficas y resultados de utilidad.

- Servicios de Red. Es posible crear una red local virtual entre equipos que realmente están en lugares remotos

Al igual que ocurre con los sistemas empotrados hay disponibles diversas plataformas de desarrollo de aplicaciones en la nube. A continuación se van a describir las más importantes y relevantes en la actualidad.

2.2.8 Amazon Web Service (AWS)

Es una colección de servicios de computación en la nube lanzada en 2006 [1], que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com. Es usado en aplicaciones populares como Dropbox, Foursquare, HootSuite. Es una de las ofertas internacionales más importantes de la computación en la nube.

AWS está situado en 9 Regiones geográficas: EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Norte de California), EE.UU. Oeste (Oregón), AWS GovCloud (EE.UU.), São Paulo (Brasil), Irlanda, Singapur, Tokio y Sydney. Cada región está totalmente contenida dentro de un solo país y todos sus datos y servicios permanecen dentro de la región designada.

Presta servicios de informática con instancias de Linux, cuyo coste se evalúa por horas de uso, almacenamiento casi ilimitado con una tarifa por GB (Amazon Cloud Drive), bases de datos SQL , Redes y CDN, análisis de datos almacenados y servicios de aplicaciones para software ejecutándolo en streaming desde la nube.

Se denomina Amazon EC2 (Amazon elastic compute cloud) a la parte central de la plataforma de cómputo en la nube de la empresa Amazon.com. EC2 permite a los usuarios alquilar PCs virtuales, en los cuales poder correr sus propias aplicaciones

Se trata de una plataforma muy desarrollada, ofrece gran cantidad de servicios adaptadas a diferentes exigencias y es fácil de implementar. En contra, la mayoría de servicios son de pago, aunque algunos sean adquiribles de forma gratuita.

En la Ilustración 7 se muestra la interfaz del portal de AWS.

Ilustración 7: Vista del portal de Amazon Web Services.

2.2.9 Google Cloud Platform

Google Cloud Platform [2] (ver Ilustración 8) es la plataforma de Google de Cloud computing. Sus inicios se remontan a abril de 2008, cuando por primera vez apareció Google App Engine. Finalmente, ha sido lanzada en marzo de 2014 con un gran elenco de servicios a disposición del usuario.

Los servicios que presta esta plataforma, entre otros de menor importancia para este proyecto, son:

- Google Compute Engine, aplicaciones Web de recursos escalables.
- Google App Engine, informática en la nube para creación de VM.
- Google Cloud Storage para almacenamiento en la nube.
- Google Cloud SQL gestor de bases de datos.
- Google BigQuery análisis de datos en Big Data.

Dada la creciente aparición de este tipo de plataformas, las empresas deben ofrecer características distintas a las de las demás. En el caso de Google Cloud Platform, cuenta con una gran red propia de fibra óptica que se extiende por medio mundo, y que va a permitir a los clientes obtener una gran velocidad de transferencia de datos al usar sus servicios. Además, se destaca el hecho de que todo está montado sobre los servidores de

Google, que a su vez están conectados a dicha red y que va a hacer el sistema mucho más rápido y fiable. Google Cloud Platform pone además varias API's a disposición de los usuarios para la creación de las aplicaciones web, de forma que cada cual puede crear sus propios proyectos.

Al igual que ocurre con Amazon Web Services, Google Cloud Platform cuenta con diferentes posibilidades en sus servicios, que en la mayoría de los casos serán de pago.

Ilustración 8: Vista del portal de Google Cloud Platform.

2.2.10 Fi-Ware

La Comisión Europea (CE) y las principales empresas TIC europeas emprendieron en 2011 un programa de Colaboración Público-Privada (PublicPrivatePartnership – PPP), con el objetivo de definir una plataforma que represente una opción abierta para el desarrollo y despliegue global de aplicaciones en la Internet del Futuro. La tarea fue encomendada a Telefónica, la cual creó Fi-Ware [3]. Las especificaciones de las APIs (ApplicationProgramming Interfaces) ofrecidas por los componentes de esta plataforma son abiertas y completamente libres de royalties. En la actualidad se está implementando en grandes ciudades de Europa para la creación de Smart Cities.

Los principales servicios prestados por esta plataforma son: elementos de Cloud Hosting, que proporcionan los recursos básicos de computación, de red y de almacenamiento; elementos de manejo de datos, que proporcionan herramientas para análisis tipo "Big Data"; módulos para integración de aplicaciones, que proporciona

elementos para integrar aplicaciones, permitir su publicación, etc., así como aspectos de negocio, elementos para el manejo de la sensores e Internet de las cosas, que permiten utilizar de forma fácil y estandarizada los recursos de los sensores y actuadores; módulos para el acceso a la red de comunicaciones y control de terminales, y elementos para dotar de seguridad y privacidad a las aplicaciones.

En general, se denominan General Enablers a los diferentes servicios disponibles en su catálogo. Posee un entorno web de Cloud donde es posible enlazar unas aplicaciones o widget con otras, hasta conseguir otras nuevas de mayor complejidad sin tener que crearlas desde cero.

Fi-Ware está en continuo desarrollo, promueve la creación de nuevos proyectos e ideas por parte de desarrolladores y usuarios, y a diferencia de la mayoría de ecosistemas en la nube, es completamente abierto y gratuito.

Ilustración 9: Vista de la web principal de FI-WARE

2.2.11 Xively

Xively [4] (ver Ilustración 10) es una división de LogMeInInc, que ofrece una plataforma de Internet de las Cosas como servicio para empresas y usuarios.

En 2007, fue fundada Pachube, una infraestructura de Internet de las Cosas, en Londres. En 2011 se anunció que habían sido comprados por LogMeIn y se pasó a llamar Cosm. Finalmente, en Mayo de 2013, dado el poco éxito de Cosm se volvió a renombrar y pasó a llamarse Xively, que además pasó a ser pública.

Principalmente cuenta con los siguientes servicios:

- Xively Cloud Services, construido para Internet de las Cosas. Incluye servicios de datos, un servicio de seguridad Trust Engine y aplicaciones gestionadas desde la web. La comunicación con Xively está construida mediante protocolos publish-subscribe llamados MQTT. El API soporta REST, WebSockets y MQTT.

- Xively Business Services, el cual va orientado a los servicios y aplicaciones para empresas.

- XivelyPartner Network: es una asociación con compañías de chipsets como ARM, Atmel y TI como proveedores, así como alianzas con compañías de Internet de las Cosas como OASIS.

Al igual que otras plataformas, cuenta con API's que permiten al usuario crear aplicaciones Web, conectar dispositivos y gestionar los datos enviados y recibidos desde ella. Xively está muy desarrollada y permite realizar todas estas tareas de forma rápida y sencilla. Muestra de ello es la consola Web para la gestión de todos los dispositivos con la que cuenta, donde se muestra información detallada de todos ellos.

Ilustración 10: Vista del portal web de Xively

2.2.12 Microsoft Azure

Microsoft Azure [5] (denominada anteriormente Windows Azure y AzureServicesPlatform) es una plataforma alojada en los Data Centers de Microsoft. Anunciada en 2008 en su versión beta, pasó a ser un producto comercial el 1 de enero del 2010. Windows Azure es una plataforma general que proporciona diferentes servicios para

aplicaciones, desde servicios que alojan aplicaciones en alguno de los centros de procesamiento de datos de Microsoft, para que se ejecute sobre su infraestructura (Cloud Computing), hasta servicios de comunicación segura y federación entre aplicaciones.

El servicio de proceso de Windows Azure ejecuta aplicaciones basadas en Windows Server. Estas aplicaciones se pueden crear mediante .NET Framework en lenguajes como C# y Visual Basic, o implementar sin .NET en C++, Java y otros lenguajes.

Windows Azure utiliza un sistema operativo especializado, llamado de la misma forma, para correr sus "capas" (en inglés "fabriclayer"). Un clúster localizado en los servidores de datos de Microsoft, que se encarga de manejar los recursos almacenados y el procesamiento para proveer los recursos (o una parte de ellos) para las aplicaciones que se ejecutan sobre Windows Azure.

Los servicios prestados por Windows Azure son:

- Windows Azure Compute: es una plataforma para hospedar y administrar aplicaciones en los centros de datos de Microsoft.
- Windows Azure Storage: tiene servicios de básicos como parte de la cuenta de almacenamiento.
- Microsoft SQL Azure: es un servicio de base de datos en la nube basado en las tecnologías de SQL Server.
- Content Delivery Network (CDN): coloca copias de los datos cerca de donde estos se encuentran.
- AzureAppFabric: ofrece diferentes servicios para aplicaciones. Los servicios de autenticación, autorización y mensajería permiten la comunicación segura entre aplicaciones y servicios desplegados tanto en la nube y en local.
- AzureMarket Place es un mercado en línea global compartir, comprar y vender aplicaciones SaaS completas y conjuntos de datos.
- Azure Virtual Network es una serie de funciones de red.

Ilustración 11: Vista del portal de Microsoft Azure.

2.2.13 Sensor Cloud

SensorCloud [6] es una plataforma de MicroStrain que se centra en el almacenamiento, visualización y gestión remota de datos. Es muy potente y permite el procesamiento en la nube de la información proporcionando gran escalabilidad de los datos, rápida visualización y análisis programados por el usuario.

Originariamente se creó para dar soporte a despliegues de sensores inalámbricos de MicroStrain sin embargo, ahora soporta cualquier dispositivo físico, sensor o red de sensores a través de una simple API OpenData.

Como ya se ha mencionado esta plataforma está optimizada para el manejo de datos, y es por ello, que suministra gran cantidad de servicios para el manejo, análisis y control de sensores de todo tipo, pudiendo recibir notificaciones de múltiples formas: SMS, correo electrónico, notificaciones en Smartphone, etc.

Mathengine es un potente motor de cálculo ofrecido por esta plataforma para operar sobre los datos representados gráficamente de forma instantánea, de forma que es posible filtrar, suavizar, aplicar transformaciones (FFT's), todo ello en tiempo real.

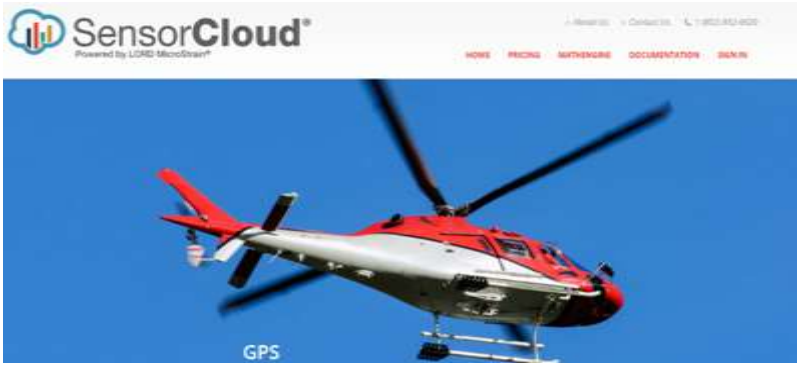

Ilustración 12: Vista del portal web de SensorCloud

2.3 Smartphones

2.3.1 Introducción

Un Smartphone también llamado teléfono inteligente es un dispositivo electrónico que funciona como un teléfono móvil con características similares a las de un ordenador personal.

Ilustración 13**:** Ejemplos de Smartphones en la actualidad

Una vez aclarado el concepto de telefonía inteligente nos remontaremos a la historia, pues según se cuenta, el primer teléfono inteligente lo creo la IBM en el año 1992 y se llamó Simón, sin embargo fue liberado en 1993 y fue comercializado por BeelSounth. En ese entonces el Smartphone te permitía hacer muchas cosas, es decir, recibir y realizar llamadas, tenía calendario, libreta de direcciones, hora mundial, libreta de anotaciones,

enviaba y recibía FAX. Pero ¿Cuál es la diferencia entre un Smartphone y un teléfono móvil?

Existen muchas diferencias, ya que un teléfono inteligente debe poseer las siguientes características:

- Soporta correo electrónico.
- Posee sistema de posicionamiento GPS.
- Permiten la instalación de programas de terceros.
- Utiliza cualquier interfaz para el ingreso de datos, como por ejemplo teclado QWERTY, pantalla táctil.
- Te permiten ingresar a Internet.
- Poseen agenda digital, administración de contactos.
- Permitan leer documentos en distintos formatos, entre ellos los PDF `s y archivos de Microsoft Office.
- Debe contar con algún sistema operativo

Con un teléfono inteligente puedes hacer de todo al mismo tiempo, esto es que puedes recibir llamadas, revisar tu agenda mientras ves unos videos en Media Player, o mientras sincronizas tu dispositivo con otros, y todo esto sin necesidad de interrumpir alguna de las tareas.

Para no ir tan lejos, es lo mismo que se hace en tu ordenador, abres ventanas y todas funcionan al tiempo y no como en un teléfono convencional que si vas a revisar tu agenda debes dejar de escuchar música para hacerlo.

En síntesis podemos decir que un Smartphone es algo como un teléfono móvil, pero mucho más potente. Sin embargo por poseer características similares a las de un computador, hace que estos dispositivos puedan ser vulnerables a virus y ataques al SO, tal como sucede en la actualidad con los equipos portátiles o de escritorio.

2.3.2 Clasificación según su sistema operativo

Como se ha nombrado antes, una de las principales diferencias entre un Smartphone y un teléfono convencional es que los smartphones utilizan sistemas operativos móviles, por lo que en función de estos, podemos realizar una clasificación.

A vista de un usuario general que no es experto en la materia, a la pregunta de cuántos sistemas operativos móviles existen en la realidad, probablemente respondería que hay dos sistemas operativos: Android y IOS. Pero en la realidad existen muchos más sistemas operativos que no son tan conocidos como pueden ser: Windows phone, BlackBerry, Symbian, FireFox, Ubuntu touch, etc.

Dado que hoy por hoy, los grandes líderes del mercado son Android y IOS, se va a obviar la explicación del resto con el fin de poder focalizar la información sobre los más comunes para un ciudadano de a pie.

Sistema operativo Android

Android es un sistema operativo basado en el núcleo Linux. En sus inicios se diseñó principalmente para dispositivos móviles con pantalla táctil (teléfonos inteligentes o tablets) y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y que posteriormente compró.

Ilustración 14: Logotipo de Android

Android fue presentado en 2007 para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y salió a la venta en octubre de 2008.

Como curiosidad, cabe añadir que los dispositivos de Android venden más que las ventas combinadas de Windows Phone e IOS.

El 25 de junio de 2014 en la Conferencia de Desarrolladores Google I/O, Google mostró una evolución de la marca Android, con el fin de unificar tanto el hardware como el software y ampliar mercados. Para ello mostraron nuevos productos como Android TV, Android Auto, Android Wear o una serie de "smartphones" de baja gama bajo el nombre de Android One. Esto sirvió para estabilizar la imagen de la marca de cara a los mercados y al público.

En cuanto a su historia, como se ha comentado anteriormente, fue desarrollado inicialmente por Android Inc., una firma comprada posteriormente por Google. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre. A escala mundial alcanzó una cuota de mercado del 50,9% durante el cuarto trimestre de 2011, más del doble que el segundo sistema operativo (iOS de Apple, Inc.)

Android posee una gran comunidad de desarrolladores creando aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se ha llegado ya al 1.000.000 de aplicaciones disponibles para la tienda de aplicaciones oficial de Android: Google Play, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android como la tienda de aplicaciones Samsung Apps de Samsung, slideme de java y amazonappstore.

Google Play es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. La tienda F-Droid es completamente de código abierto así como sus aplicaciones, una alternativa al software privativo. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un frameworkOpenCore, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizadoWebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de XML, 2,8 millones de líneas de lenguaje C, 2,1 millones de líneas de Java y 1,75 millones de líneas de C++.

Por último y como curiosidad por parte de la compañía desarrolladora, cabe añadir las versiones estables a día de hoy de este sistema operativo. Como se puede observar, cada

vez que ha salido una nueva versión ha sido nombrada en orden alfabético. De esta manera, se tienen las siguientes versiones del sistema operativo Android:

- A: Apple Pie (v1.0)
- B: Banana Bread (v1.1)
- C: Cupcake (v1.5)
- D: Donut (v1.6)
- E: Éclair (v2.0/v2.1)
- F: Froyo (v2.2)
- G: Gingerbread (v2.3)
- H: Honeycomb (v3.0/v3.1/v3.2)
- I: Ice Cream Sandwich (v4.0)
- J: Jelly Bean (v4.1/v4.2/v4.3)
- K: KitKat (v4.4)
- L: Lollipop (v5.0/v5.1)

Sistema operativo iOS

iOS es un sistema operativo móvil de la multinacional Apple Inc. Originalmente desarrollado para el iPhone, después se ha usado en dispositivos como el iPod touch y el iPad.

Ilustración 15: Logotipo iOS

En enero-febrero de 2015, más del 52% de todo el mundo que poseen iPhone, iPod touch o iPad tienen instalado en su dispositivo iOS 8.

Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo o al rotarlo en alguna dirección.

iOS se deriva de OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Tipo Unix. iOS cuenta con cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios" y la capa de "CocoaTouch".

La existencia del iPhone OS se remonta a Enero de 2007 en la MacworldConference& Expo, aunque el sistema no tuvo un nombre oficial hasta que salió la primera versión beta del iPhone SDK un año más tarde, en marzo de 2008. Antes de esto se consideraba simplemente que el iPhone ejecutaba OS X o una versión modificada de NewtonOS. A partir de entonces se llamaría iPhone OS. El lanzamiento oficial del iPhone OS tuvo lugar en junio de 2007.

El interés en el SDK aumentaría notablemente en los siguientes meses debido al aumentante crecimiento de la plataforma iPhone, que se vio incrementado en septiembre de 2007 por el lanzamiento del iPod Touch, un dispositivo con las capacidades multimedia del iPhone pero sin la capacidad de hacer llamadas telefónicas solo por redes.

En enero de 2010 Steve Jobs, anunció el iPad, un dispositivo muy similar al iPod Touch pero con un propósito orientado hacia la industria de contenidos. Este dispositivo, apoyado en una pantalla táctil de mayor dimensión, compartiría sistema operativo con sus dos exitosos hermanos, y vendría acompañado de una aplicación oficial para la compra y lectura de libros electrónicos, iBooks.

Tras más de 185 000 las aplicaciones disponibles para iPhone OS a través de la App Store8, durante la presentación del iPhone 4, en Junio de 2010, Steve Jobs anunció que iPhone OS pasaría a ser llamado oficialmente como iOS.

En de 2013 es presentado iOS 7 en la WWDC 2013 a las 10:00 tiempo de San Francisco como "El mayor cambio de iOS desde el iPhone original", cambia por completo el diseño gráfico del sistema, haciéndolo más plano y con nuevos íconos, trae nuevas características como AirDrop, Filtros de cámara, Fondo dinámico entre muchas otras, ese mismo día se liberó la beta 1 para desarrolladores. En la misma conferencia de dieron a conocer los datos oficiales de iOS hasta la fecha, indicando que han sido vendidos más de 600 millones de iDevices, los usuarios de iOS utilizan un 50 % más sus dispositivos que los de Android, que el mercado web lo domina iOS con un 60 % y en tabletas el iPad tiene el 82 % del tráfico web.

La última versión estable del sistema operativo a día de hoy es la versión iOS 8.4.

2.4 Microcontroladores

2.4.1 Introducción

Un microcontrolador no es más que un circuito integrado programable que es capaz de ejecutar las órdenes que hay almacenadas en su memoria. Se compone de diversos bloques funcionales, como la CPU, memoria RAM y ROM y periféricos de entrada y salida.

Algunos microcontroladores funcionan a velocidades de reloj con frecuencias muy bajas lo que implica un consumo de potencia muy bajo, del orden de los microvatios.

Una característica que poseen todos los microcontroladores, es la capacidad que tienen de mantenerse a la espera de que suceda un evento (como puede ser pulsar un botón). Esto es lo que se conoce como interrupción, y consigue que, bajo una programación del microcontrolador optimizada, este reduzca más aun su consumo de energía hasta el orden de los nanovatios.

Por defecto, un microcontrolador no contiene ningún dato en la memoria ROM cuando se fabrica. Por tanto, para que se pueda controlar algún proceso, es necesario generar en la EEPROM un programa el cual permita ser programado en lenguaje ensamblador. Es importante saber que el programa deberá ser grabado en la memoria codificada en el sistema numérico hexadecimal.

2.4.2 Diferencias entre un microcontrolador y un microprocesador

Los microprocesadores y los microcontroladores son dispositivos electrónicos que tienen la capacidad de llevar a cabo procesos lógicos.

El microprocesador, tiene la característica de que sus unidades están físicamente separadas, esto significa que interactúa con una memoria RAM, una memoria ROM y con dispositivos de entrada y salida por medio de buses de comunicación. El microcontrolador, por su parte, es un solo dispositivo que internamente contiene todo lo necesario para poder llevar a cabo sus acciones. Este contiene su propio CPU, memorias RAM y ROM y dispositivos de entrada y salida. Por lo tanto es superior al microprocesador debido a su reducido tamaño y capacidad de ser implementado en circuitos electrónicos. Sin embargo, los microprocesadores continúan empleándose en equipos como los computadores.

Estructura gráficamente del microcontrolador (derecha) y el microprocesador (izquierda):

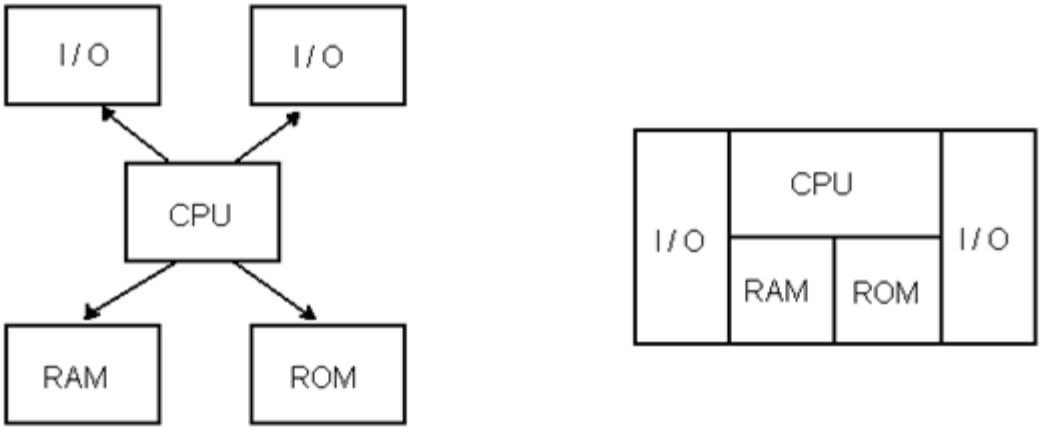

Ilustración 16: Diferencia gráfica entre microcontrolador y microprocesador

2.4.3 Partes de un microcontrolador

A continuación se listan y se detallan las partes de las que consta un microcontrolador:

Registros:

Es un espacio de memoria muy reducido pero necesario para cualquier microprocesador, ya que de aquí se toman los datos para varias operaciones que debe realizar el resto de los circuitos del procesador. Los registros sirven para almacenar los resultados de la ejecución de instrucciones, cargar datos desde la memoria externa o almacenarlos en ella.

La importancia de los registros se ve cuantificada cuando escuchamos que un procesador es de 4, 8, 16, 32 o 64 bits, puesto que con este dato nos estamos refiriendo a procesadores que realizan sus operaciones con registros de datos de ese tamaño, y en consecuencia, esto determina muchas de las potencialidades de estas máquinas.

Mientras mayor sea el número de bits de los registros de datos del procesador, mayores serán sus prestaciones, en cuanto a poder de cómputo y velocidad de ejecución, ya que este parámetro determina la potencia que se puede incorporar al resto de los componentes del sistema.

Unidad aritmética lógica (ALU)

Como los procesadores son circuitos que hacen básicamente operaciones lógicas y matemáticas, se le dedica a este proceso una unidad completa, con cierta independencia. Aquí es donde se realizan las sumas, restas, y operaciones lógicas típicas del álgebra de Boole.

Actualmente este tipo de unidades ha evolucionado mucho y los procesadores más modernos tienen varias ALU, especializadas en la realización de operaciones complejas como las operaciones en coma flotante. De hecho en muchos casos le han cambiado su nombre por el de "coprocesador matemático", aunque este es un término que surgió para dar nombre a un tipo especial de procesador que se conecta directamente al procesador más tradicional.

Su impacto en las prestaciones del procesador es también importante porque, dependiendo de su potencia, tareas más o menos complejas, pueden hacerse en tiempos muy cortos, como por ejemplo, los cálculos en coma flotante.

Buses

Es el medio de comunicación que se utiliza para unir los diferentes componentes del procesador y así poder intercambiar información entre ellos.

De esta manera, existen tres tipos de buses:

- Dirección: Son utilizados para determinar el dispositivo con el cual se quiere trabajar o en el caso de las memorias, determinar el dato que se quiera leer o escribir.
- Datos: Se utiliza para mover los datos entre los dispositivos de hardware.
- Control: Gestiona los procesos de escritura-lectura y también controla la operación de los dispositivos del sistema.

Memorias

- ROM (readonlymemory): es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite solo la lectura de la información y no su escritura, independientemente de la presencia o no de una fuente de energía. Los datos almacenados en la ROM no se pueden modificar, o al menos no de manera rápida o fácil. Sin embargo, las ROM más modernas, como EPROM y Flash EEPROM, se pueden borrar y volver a programar varias veces, aun siendo descritos como "memoria de sólo lectura" (ROM).
    - EPROM: Es un tipo de chip de memoria ROM no volátil inventado por el ingeniero DovFrohman. Está formada por celdas de FAMOS o "transistores de puerta flotante", cada uno de los cuales viene de fábrica sin carga, por lo que son leídos como 1.
    - EEPROM: Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente, a diferencia de la EPROM que ha de borrarse mediante un aparato que emite rayos ultravioleta. Son memorias no volátiles. Las celdas de memoria de una EEPROM están constituidas por un transistor MOS, que tiene una compuerta. Aunque una EEPROM puede ser leída un número ilimitado de veces, sólo puede ser borrada y reprogramada entre 100.000 y un millón de veces.
    - RAM (random Access memory): se utiliza como memoria de trabajo de computadoras para el sistema operativo, los programas y la mayor parte del

software. En la RAM se cargan todas las instrucciones que ejecutan la unidad central de procesamiento (procesador) y otras unidades de cómputo.

Se denominan de acceso aleatorio porque se puede leer o escribir en una posición de memoria con un tiempo de espera igual para cualquier posición, no siendo necesario seguir un orden para acceder a la información de la manera más rápida posible.

Unidad de control

La unidad de control es la encargada de la decodificación y ejecución de las instrucciones, el control de los registros, la ALU, los buses y todo aquello que se quiera meter en el procesador, todo esto mediante la lógica necesaria interna.

Esta unidad es una de las partes fundamentales del microcontrolador para determinar las prestaciones del procesador, ya que su estructura determina parámetros tales como el tipo de conjunto de instrucciones, velocidad en ejecución, tiempo del ciclo de máquina, tipo de buses que puede tener el sistema, manejo de interrupciones, etc.

Las unidades de control son el elemento más complejo de un procesador y normalmente están divididas en unidades más pequeñas trabajando de conjunto. La unidad de control agrupa componentes tales como la unidad de decodificación, unidad de ejecución, controladores de memoria cache, controladores de buses, controlador de interrupciones, pipelines, entre otros elementos, dependiendo siempre del tipo de procesador.

2.4.4 Tipos de arquitecturas de los microcontroladores

Según la arquitectura que tengan los microcontroladores, se pueden diferenciar en aquellos que tengan una arquitectura de Von Neumann o una arquitectura Harvard.

A continuación se detallan cada una de ellas.

Arquitectura Von Neumann

La arquitectura Von Neumann utiliza el mismo dispositivo de almacenamiento tanto para las instrucciones como para los datos, siendo la que se utiliza en un ordenador personal porque permite ahorrar una buena cantidad de líneas de E/S, que son bastante

costosas, sobre todo para aquellos sistemas donde el procesador se monta en algún tipo de zócalo alojado en una placa madre.

En un ordenador personal, cuando se carga un programa en memoria, a éste se le asigna un espacio de direcciones de la memoria que se divide en segmentos, de los cuales típicamente tenderemos los siguientes: código (programa), datos y pila. Es por ello que podemos hablar de la memoria como un todo, aunque existan distintos dispositivos físicos en el sistema (disco duro, memoria RAM, memoria flash, unidad de disco óptico...).

A pesar de que en los sistemas integrados con arquitectura Von Neumann la memoria esté segregada, y existan diferencias con respecto a la definición tradicional de esta arquitectura, los buses para acceder a ambos tipos de memoria son los mismos, del procesador solamente salen el bus de datos, el de direcciones, y el de control.

Las principales limitaciones de esta arquitectura es que por tener un único bus de datos, hace que el CPU tenga que hacer varios accesos a la memoria para buscar instrucciones complejas y también se reduce el tiempo en realizar las operaciones, impidiendo superponer los tiempos de acceso.

Arquitectura Harvard

Es la más utilizada en microcontroladores y sistemas integrados en general. En este caso, además de la memoria, el procesador tiene los buses segregados, de modo que cada tipo de memoria tiene un bus de datos, uno de direcciones y uno de control.

La ventaja fundamental de esta arquitectura es que permite adecuar el tamaño de los buses a las características de cada tipo de memoria. Además, el procesador puede acceder a cada una de ellas de forma simultánea, lo que se traduce en un aumento significativo de la velocidad de procesamiento.

La desventaja está en que consume muchas líneas de E/S del procesador, sin embargo, en los microcontroladores y otros sistemas integrados, donde usualmente la memoria de datos y programas comparten el mismo encapsulado que el procesador, este inconveniente deja de ser un problema serio y es por ello que encontramos la arquitectura Harvard en la mayoría de los microcontroladores.

2.4.5 Los microcontroladores hoy en día

Como se puede deducir de lo explicado anteriormente acerca del microcontrolador, el uso desde su invención hasta hoy en día ha supuesto un avance para tecnología usada

por el día a día. Desde la incorporación de estos en aparatos de uso cotidiano como pueden ser electrodomésticos hasta la incorporación en la industria en cadenas de montaje en la automoción.

Con el paso del tiempo es más fácil encontrar también microcontroladores que vienen con un kit de desarrollo incorporado el cual facilita a un usuario no experto la programación de estos.

Estos kits han provocado un salto tecnológico, abriendo un amplio abanico de posibilidades que han provocado una gran documentación en la nube acerca de proyectos de todo tipo de aplicaciones.

Ejemplos de esto, son las plataformas Arduino, Raspberry pi, ardupilot, funduino, etc.

Todos ellos son "open source" y han conseguido crear una comunidad con millones de usuarios que cada día aportan proyectos e ideas nuevas.

A modo de ejemplo, se pasa a detallar brevemente las características principales de la plataforma Arduino, puesto que junto con Raspberry Pi, es la plataforma con mayor número de usuarios en su comunidad.

Arduino

Arduino [7] es una plataforma de electrónica abierta (open Hardware) para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores.

El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing).

Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software.

¿Qué es Processing? Es un lenguaje de programación de código abierto, para las personas que quieran crear imágenes, animaciones e interacciones. Inicialmente

desarrollado para servir como un software como block de dibujo y para enseñar los fundamentos de la programación de computadora en un contexto visual.

Processing también ha estado involucrado como una herramienta para generar trabajos finales profesionales.

¿Qué es Wiring? Wiring es una plataforma de abierta de prototipos electrónicos compuestos de un entorno de programación, una tarjeta de prototipo, documentación (creada por ingeniero y artistas y una comunidad de expertos), personas con experiencia tanto principiante como nivel intermedio comparten sus ideas, conocimiento y su experiencia colectiva.

Las placas pueden ser hechas a mano o compradas las cuales fueron hechas en una fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta, así pues eres libre de adaptarlos a tus necesidades.

El Hardware de Arduino está basado en un microcontrolador AVR, en particular el ATmega8, ATmega 168, ATmega328 y el ATmega1280. Los programas de Arduino están basados en C/C++ y compilados con el compilador de código abierto avr-gcc y enlazado con la Libc de AVR de código abierto.

El entorno de desarrollo de Arduino contiene un editor de texto para escribir el código, un área de mensaje, una consola de texto, barra de herramientas, etc. Los programas escritos usando Arduino son llamados en inglés "sketches". Cuando cargas un sketch (proyecto), estas usando el bootloader de Arduino, el cual es un pequeño programa que ha sido precargado al microcontrolador de la tarjeta. Esto permite cargar un código sin usar ningún hardware adicional. El bootloader es activado durante unos pocos segundos cuando se reinicia el microcontrolador; después de eso inicia el programa que le hayas cargado.

¿Qué sucede cuando pulsamos el botón "build"? El software de Arduino realiza algunas pequeñas transformaciones para verificar que el código en C o C++ este correcto. Éste programa pasa por el compilador avr-gcc, el cual pasa el código de alto nivel a lenguaje de máquina. Luego el código es combinado con librerías estándar de Arduino que proveen las funciones. Como resultado se obtiene un archivo .hex, el cual contiene los bytes necesarios para programar la memoria del microcontrolador que se encuentra presente en la placa de Arduino y luego enviado por el puerto USB o serial a través del bootloader que se encuentra en el microcontrolador.

Entre las principales aplicaciones que se han realizado con Arduino, dado su carácter de importancia, cabe destacar estos proyectos:

- Xoscillo: Osciloscopio de código abierto.
- Equipo científico para investigaciones.
- Arduinome: Un dispositivo controlador MIDI.
- OBDuino: un económetro que usa una interfaz de diagnóstico a bordo que se halla en los automóviles modernos.
- Humane Reader: dispositivo electrónico de bajo coste con salida de señal de TV que puede manejar una biblioteca de 5000 títulos en una tarjeta microSD.
- The Humane PC: equipo que usa un módulo Arduino para emular un computador personal, con un monitor de televisión y un teclado para computadora.
- Ardupilot: software y hardware de aeronaves no tripuladas.
- ArduinoPhone: un teléfono móvil construido sobre un módulo Arduino.
- Impresoras 3D.

No obstante, dependiendo de la finalidad del proyecto que queramos realizar, la plataforma Arduino ofrece una amplia gama de productos con distintas especificaciones.

Para más información, se puede consultar su página oficial.

Ilustración 17: Logotipo de la marca registrada Arduino

2.5 Sensores de salud

La constante y rápida evolución del internet de las cosas (Internet of things, IoT) ha revolucionado el uso de la tecnología cotidiana para los ciudadanos.

Son muchos los sensores que se han ido diseñando para hacer una vida más cómoda. Estos sensores abarcan todos los campos posibles, desde sensores para proteger nuestros dominios (alarmas, cámaras de vigilancia en tiempo real,…) hasta sensores para controlar nuestros jardines, pasando por sensores para la salud.

Dichos sensores para la salud se encargan de capturar datos, almacenarlos e incluso crear alarmas mediante una aplicación de forma que nos avisen si algún valor de nuestras constantes vitales no está dentro de un rango determinado e incluso alertar a personal sanitario, si es necesario.

Ejemplo de estos sensores pueden ser las famosas pulseras de actividad.

Grandes marcas como Garmin, Polar, Xiaomi, Samsung, etc han sacado al mercado estos "wearables" que monitorizan datos como los pasos que caminamos, las pulsaciones, las kilocalorías o incluso las horas de sueño, y mediante una aplicación para Smartphone, podemos visualizar todos estos datos en nuestro móvil, comparando dichos datos con cualquier usuario, o incluso creando desde la misma aplicación eventos o alarmas para unos determinados valores.

Ilustración 18: Pulsera de actividad Polar Loop

Ilustración 19: Pulsera de actividad Xiaomi Band

De una forma gráfica, para ver cómo funcionan estos wearables, la imagen que aparece a continuación puede dar una idea de las funcionalidades:

Ilustración 200: Funcionamiento de wearables

Pero no todos los sensores de salud se limitan a pulseras de actividad. Es evidente, con el avance tecnológico en esta última década, que existen multitud de sensores destinados a la salud. Muchos de ellos son independientes y tan solo son capaces de medir una constante vital en concreto. Otros, como las pulseras de actividad, incorporan diversos sensores para evaluar diferentes valores.

Entre las principales funciones que realizan dichos sensores, encontramos la medición de los siguientes valores:

- Ritmo cardiaco
- Frecuencia respiratoria
- Temperatura corporal
- Número de pasos dados
- Nivel de estrés
- Fases del sueño
- Nivel de glucosa en sangre
- Postura del cuerpo
- Kilocalorías consumidas

Lo sensores recopilan y transmiten la información para que sea procesada por ordenadores o smartphones. La información se puede transmitir de manera segura, en cualquier parte del mundo, incluso fuera de él. Las instituciones de salud quieren en tiempo real resultados de diagnósticos fiables y precisos proporcionados por dispositivos que se pueden supervisar de forma remota, por si el paciente se encuentra en un hospital, en una clínica o en casa.

Las aplicaciones de sensores en diagnósticos médicos están creciendo a un ritmo asombroso. Por ejemplo, Apple se está lanzando a expandir su personal que trabaja con ordenadores portátiles y dispositivos médicos con sensores mediante la contratación de más científicos y especialistas en el campo sensor médico. Cada fabricante importante de tecnología ve lo que parece evidente: la salud digital es la próxima gran frontera.

Aunque a simple vista lo veamos como un avance más de la tecnología, incluso haya gente que no deposite su confianza en este campo, desde la perspectiva de un ciudadano son muchas ventajas las que podemos obtener con este desarrollo tecnológico.

En primer lugar, se tendrá un mayor y mejor acceso a la información sobre salud y enfermedades. Se contará con una mayor posibilidad de comunicación con otras personas o familias que tengan relación con una enfermedad concreta, dado que se crearán comunidades de usuarios. También se crearán nuevas vías de comunicación con profesionales sanitarios, se tendrá más acceso a nuestros datos de salud y habrá herramientas que ayuden a la toma de decisiones en situaciones particulares.

En definitiva, los beneficios son muy claros:

- Conectividad móvil y comodidad
- Respuesta en tiempo real ante el análisis de algún valor
- Privacidad de los datos
- Creación de un historial clínico para cada paciente

Hoy en día existen diversos kits de desarrollo para microcontroladores como los que se han comentado anteriormente (Arduino, Raspberry Pi, etc) que traen consigo una amplia gama de sensores acondicionados para la conexión a la placa y puesta en marcha.

Además, dichos kits suelen tener integrados una aplicación compatible con los diferentes sistemas operativos de los smartphones, por lo que basta con hacer una pequeña inversión económica para poder hacer uso de esta tecnología que se está comentando en este apartado, sin tener apenas conocimientos en el campo de la electrónica.

Ejemplo de estos kits, puede ser el que ha sido puesto en el mercado hace escasos meses: e-health.

Este kit no deja de ser más que una serie de sensores con acondicionamiento y conexión directa a un microcontrolador y que tiene predefinidas una serie de librerías para usar en función del sensor que queramos utilizar.

Ilustración 21: Kit de desarrollo e-Health

En este caso, como se puede apreciar en la imagen, este kit está diseñado para ser conectado directamente a un Arduino, aunque ofrece la posibilidad de ser conectado a otro microcontrolador diferente.

Contiene sensores de todo tipo como de temperatura corporal, presión arterial, acelerómetro, etc. En la siguiente imagen, se muestra un ejemplo de todo y cuanto este kits te permite controlar:

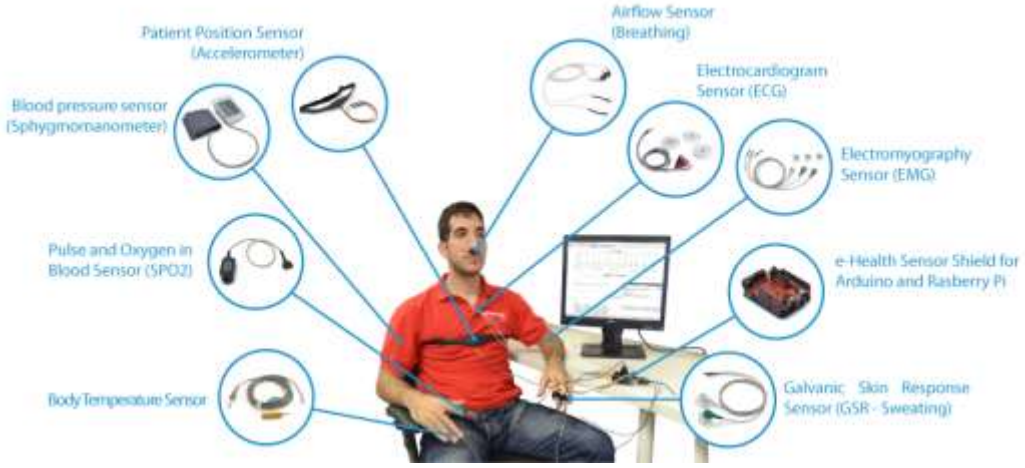

Ilustración 22: Ejemplo de sensores kit e-Health

Para más información acerca de este kit de desarrollo se puede visitar su página web oficial en:

https://www.cooking-hacks.com/ehealth-sensors-complete-kit-biometric-medical-arduino-raspberry-pi/

2.6 Conclusiones

A lo largo de este tercer capítulo se ha podido ver el estado actual de los campos en los que se basa este proyecto final de grado. Como se puede observar, la peculiaridad de realizar este proyecto no es más que dejar abierta la puerta a futuras investigaciones para poder llegar a implementar un sistema tan preciso como los que ya hay en el mercado actual, tal como se ha visto en este capítulo (wearables, pulseras de actividad, etc) con la ventaja de ser un producto lowcost.

El rápido avance de la tecnología, del campo de la electrónica destinada a este ámbito y las nuevas comunidades de programación de software libre, hacen intuir la cercana aparición de dichos gadgets.

Capítulo 3

Tecnología utilizada

3.1 Introducción

A lo largo de esta sección se va a profundizar en las herramientas que han sido seleccionadas para llevar a cabo este proyecto.

En primer lugar se hablará del microcontrolador seleccionado, para posteriormente hablar del sensor que se ha acondicionado para el caso práctico y por último el sistema operativo elegido.

No obstante, todos los detalles que tengan que ver con el montaje o la programación se verán detallados en el siguiente capítulo, en el estudio de un caso práctico.

3.2 Antecedentes

Para la realización del presente trabajo final de grado, se ha partido de la base de otro proyecto realizado durante el mismo periodo, el cual se encarga de realizar la medición del pulso cardiaco y la comunicación con el Smartphone mediante el bluetooh de baja energía.

Es por este motivo por el cual los detalles acerca de la parte relacionada con la implementación del otro trabajo final de grado no serán tan vistos en detalle como los propios de este trabajo. No obstante, sí que se nombrarán y comentarán para que el lector posea la información necesaria para comprender el propósito final.

3.3 Psoc

En primer lugar, tal como se ha mencionado anteriormente en la introducción, el microcontrolador seleccionado ha sido el PSoC [8]. No obstante, tanto la elección como la programación de este no es objeto de este trabajo final de grado, sino que ha sido realizado por un compañero en su trabajo. Por tanto, no se va a comentar en detalle pero sí que se comentan a continuación las principales características.

Este micro-controlador está desarrollado por la empresa cypress. Consta de una CPU con arquitectura ARM que funciona a 48 MHz. El almacenamiento con el que contamos para cargar el programa es de unos 128 KB de memoria flash. Aunque pueda parecer que la potencia y la memoria son insignificantes al lado de los ordenadores actuales, hay que recordar que para el tipo de tarea que tiene que llevar a cabo el micro controlador es más que suficiente. Además, al ser una velocidad de procesamiento tan "limitada", facilita enormemente el bajo consumo del dispositivo, que es lo que estamos buscando.

Las dimensiones del encapsulado son 7 mm de largo, 7 mm de ancho y 0.6 mm de alto.

En cuanto al bluetooth, el chip consta de un transceptor que transmite y recibe datos a 1 Mbps utilizando la banda de los 2.4 GHz, siendo compatible con la especificación 4.1 de bluetooth. La capa de enlace proporciona protección AES de 128 bits, encriptación, y como novedades del bluetooth 4.1, el chip es capaz de entrar en modo advertising en un bajo ciclo de trabajo. El controlador de banda base soporta tanto modo maestro como modo esclavo, así como los protocolos L2CAP, ATT y SM. La API proporciona acceso a GATT, GAP y L2CAP. El PSoC soporta todos los perfiles del bluetooth de baja energía definidos por el SIG.

En cuanto a los componentes físicos con los que cuenta el chip y que se pueden programar, cuenta con cuatro amplificadores operaciones en modo comparador, así como con un sensor de temperatura que se puede desactivar para ahorrar batería. A ello hay que añadirle cuatro contadores, temporizadores y PWM para implementar la lógica necesaria o para controlar motores si fuera necesario. También cuenta con cuatro bloques digitales que permiten la conexión de periféricos para poder generar interrupciones en el chip.

No hay que olvidar los dos bloques dedicados a las comunicaciones serie, pudiendo implementar cada bloque los modos I2C, UART y SPI. Para complementar las comunicaciones, se incluyen 36 GPIO que pueden programarse tanto para tareas analógicas como digitales.

En cuanto a las funciones especiales de este chip, en todos los pines soporta la conexión de una unidad LCD de 7 segmentos, así como de un detector capacitivo.

Este chip puede comprarse junto con un kit de desarrollo por sólo 49€, por lo que es bastante asequible, en la siguiente página web: http://www.cypress.com/?rID=102636

Para una información más detallada del funcionamiento del PSoC, se puede ver en el trabajo final de grado desarrollado por Elías Hernández Gómez bajo el título de "Estudio y diseño de un sistema de monitorización del ritmo cardiaco basado en un PSoC y un dispositivo móvil inteligente".

3.4 Google App Engine (GAE)

3.4.1 Introducción

Google App Engine es uno de los servicios que conforman la familia de Google Cloud Platform. Este servicio es del tipo *Plataforma como Servicio* o *Platform as a Service* (PaaS), permite publicar aplicaciones web en línea sin necesidad de preocuparse por la parte de la infraestructura y con un enfoque 100% en la construcción de la aplicación y en la posibilidad de ejecutarla directamente sobre la infraestructura de Google, es decir, la que Google usa para sus propios productos.

Como cualquier otra Plataforma como Servicio, App Engine facilita construir, mantener y escalar la aplicación en la medida que sea necesario.

Otra de sus características es que las aplicaciones pueden ser escritas en diferentes lenguajes. A día de hoy, GAE permite la programación en Java, Python, PHP y Go.

Cuando usamos Google App Engine (GAE) no hay que preocuparse por la escalabilidad de nuestra aplicación ya que cuenta con un balanceador de carga y escalamiento automático.

Así todas las aplicaciones solamente serán atendidas por las máquinas necesarias para tener un perfecto comportamiento y para que la respuesta de estas sea la más óptima.

GAE es completamente amigable con otros productos de Google Cloud Platform, de manera que tenemos la capacidad de integrarlos entre ellos.

GAE también asegura almacenamiento persistente y veloz, ya sea usando Google Cloud Datastoreo usando Google Cloud SQL, así como la posibilidad de tener tareas asíncronas corriendo en procesos de colas o tareas continuas y regulares corriendo en ciertos intervalos de tiempo.

Para el almacenamiento de datos tendremos dos posibilidades en casi cualquier lenguaje en el que se esté desarrollando: Cloud Datastore y Cloud SQL.

Cloud Datastore es una base de datos NoSQL ideada como la opción de almacenamiento principal a la hora de que corremos aplicaciones en App Engine.

Una de las características de Cloud Datastore es que eventualmente es consistente, lo que significa que la información que almacenemos no será inmediatamente registrada, sino más bien, lo hará eventualmente.

Está basado en transacciones atómicas, que pueden contener más de una operación a la base de datos. Una transacción en Cloud Datastore no puede ser finalizada a menos que todos sus procesos hayan sido concluidos, lo que lo hace muy útil en situaciones en las que se hacen muchas operaciones sobre la misma información al mismo tiempo. Cloud Datastore no está disponible si se programa en el lneguaje PHP.

Por otro lado *Cloud SQL* es una base de datos MySQL que corre en la nube de Google y tiene todas las características de cualquier base MySQL, entre otras.

Google Cloud SQL se encarga de todo el mantenimiento y no hay que preocuparse por servidores de bases de datos y cosas por el estilo. A diferencia de una base MySQL común, Google Cloud SQL ofrece la capacidad de replicar las bases de datos en diferentes puntos geográficos para tener mayor disponibilidad y durabilidad con facilidad.

Entre todos los servicios compatibles y complementarios de Google App Engine se encuentra el Blobstore para el almacenamiento y procesamiento de archivos grandes y pesados como imágenes y videos; Google Cloud Endpoints para generar servicios REST para ser consumidos por todo tipo de clientes, desde móviles hasta aplicaciones web; App EngineMemcache como capa de cache en la que existen: memcache compartido completamente gratuito y servido por App Engine y memcache dedicado que tiene un cobro por GB/hora de cache usado.

También se cuenta con un servicio de logs para monitorear en tiempo real el estado de la aplicación y un servicio de colas para poder procesar grandes tareas de manera asíncrona, fuera de la petición del usuario, entre otros servicios.

Si se decide usar Python como lenguaje de programación en un proyecto de Google App Engine, cabe recalcar que no hay una manera oficial y nativa de correr proyectos de Django. Aunque existen algunas formas de usar Django en Google App Engine por medio de hacks, no es la tecnología recomendada para aprovechar al 100% el poder de Google App Engine. Para esto se cuenta con el framework web desarrollado por Google específicamente para correr sobre Google, webapp2.

Por último, cabe destacar entre los clientes más destacados de Google App Enginea Rovio (creador de AngryBirds), KhanAcademy, BestBuy y Feedly quienes han ahorrado mucho tiempo en la creación y diseño de la infraestructura usando App Engine y sus servicios.

3.4.2 Google Web Toolkit

A la hora de programar una aplicación, surge la posibilidad de trabajar con Google Web ToolKit (GWT) [9]. GWT no es más que un framework de desarrollo en Java de código abierto, que permite al usuario depurar y desarrollar aplicaciones AJAX usando el lenguaje de programación Java y utilizando el entorno de programación que se quiera. Una vez se ha terminado la fase de programación del código, GWT compila y traduce el programa de Java a JavaScript y HTML, de forma que es compatible con cualquier navegador web.

Las ventajas de utilizar este "traductor" GWT para traducir de Java a JavaScript son variadas. En primer lugar, como se ha comentado, se puede utilizar con el IDE que se desee, como por ejemplo para este caso de estudio, se puede utilizar Eclipse.

Se evitan también así los errores de sintaxis en JavaScript ya que son detectados de forma muy sencilla mientras se desarrolla la aplicación y no es necesario percatarse del error mientras se está ejecutando por el usuario final.

También son más sencillos los diseños en Java basados en la programación orientada a objetos, haciendo que el código AJAX sea más comprensible con menos documentación.

Para trabajar con GWT en Eclipse, a la hora de crear un nuevo proyecto basta con asegurarse de dejar marcada la casilla que se muestra en la siguiente ilustración:

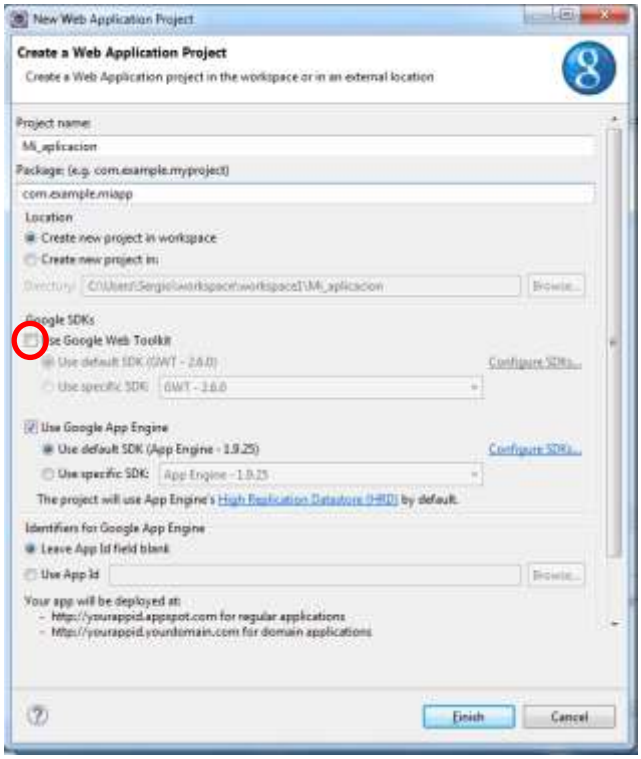

Ilustración 21: Detalle de la selección de GWT

Pero antes de esto, se debe tener instalado GWT en el ordenador donde se vaya a programar. En el caso de estar trabajando con Windows, GWT se puede obtener en la siguiente dirección:

http://www.gwtproject.org/download.html

Haciendo clic en descargar e instalando el ejecutable, se estará en disposición de trabajar con GWT.

3.4.3 Google Cloud EndPoints

Los Google Cloud EndPoints son un conjunto de herramientas que permiten de una forma sencilla (incluso a veces, de forma automáica) generar las APIs para poder comunicar las aplicaciones clientes con aplicaciones BackEnd.

Esto es, con las herramientas de los EndPoints se va a disponer de unos mecanismos sencillos para poder generar, exponer y utilizar una API REST con la que compartiremos datos entre las aplicaciones que hagan la función de clientes y las

aplicaciones que hagan la función de BackEnd. Esta última parte, el BackEnd, se va a gestionar mediante Google App Engine.

Se puede decir que lo que se tienen son una serie de objetos de datos a los que se les va a poder realizar las operaciones conocidas como CRUD, que no son más que las de crear, leer, actualizar y borrar (create, read, update, delete).

Al mismo tiempo, con los EndPoints se facilita la tarea de autenticación mediante OAuth 2.0 en la API REST del proyecto que se esté programando.

Aunque EnPoints sí que es una herramienta gratuita, sí que habrá que pagar por el resto de productos que ofrece la plataforma Google Cloud. No obstante, para estos productos (como puede ser el Cloud DataStore para el almacenamiento de persistencia) existe una versión de prueba gratuita con ciertas limitaciones pero que hacen que se prueba probar previamente dicha tecnología sin necesidad de pagar.

3.4.4 DataStore de Google

Google DataStore es una parte fundamental de Google App Engine que está destinada únicamente al almacenamiento de datos, por lo que todos aquellos datos que queramos almacenar y posteriormente recuperar, estarán alojados aquí.

Cabe destacar que bajo el DataStore no hay una base de datos relacional, y por tanto, sirve para almacenar datos de manera consecuente, por lo que no funciona como las bases de datos de tipo MySQL u Oracle. Aun con todo, el concepto de almacenar y recuperar los datos es el mismo, pero la tecnología y la manera de trabajar no lo son. Esto implica que DataStore presentará una serie de ventajas e inconvenientes respecto las otras tecnologías.

La diferencia de las tecnologías radica en que Google posee su propia tecnología basada en GFS y en BigTable. La primera de ellas, GFS, es un sistema de archivos propio, mientras que el segundo es un sistema de almacenamiento distribuido diseñado para almacenar cantidades de datos de orden muy superiores en un gran número de ordenadores. Ambas tecnologías han sido diseñadas específicamente para poder funcionar de manera distribuida en DataCenter's con cientos y miles de ordenadores en forma de grid.

El DataStore no es más que una capa que se sitúa sobre BigTable. Por tanto, todas las operaciones que se realicen sobre este sistema de almacenamiento tendrán que ser invocadas con el DataStore.

Para operar con el DataStore, utilizando el lenguaje de programación Java, hay tres formas disponibles de operar:

- JPA (Java Persistence API)
- JDO (Java Data Objects)
- El API de nivel inferior

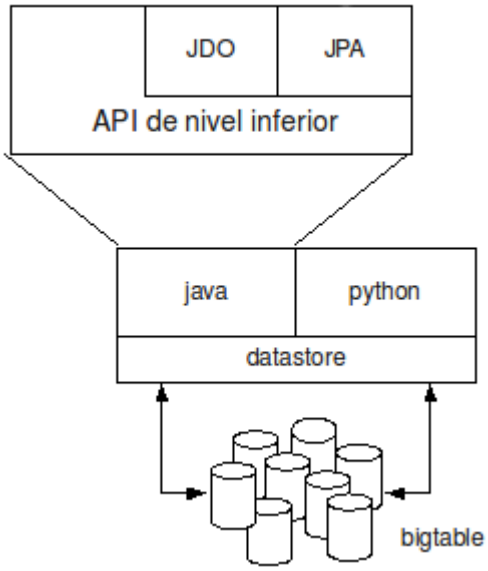

Ilustración 22: Estructura del DataStore

JPA es el 'API de persistencia para Java' o 'interfaz de programación de aplicaciones de persistencia para Java'. Este interfaz define un estándar de almacenamiento de datos en una base de datos relaciona que se encuentra adaptado para poder almacenar datos en el DataStore. Debido a esta adaptación, el soporte de GAE para JPA no admite algunas funciones de JPA. A grandes rasgos, JPA sirve para definir un marco de trabajo con un nivel de abstracción alto que, mediante la anotación de objetos Java, permita que la recuperación de objetos a través de consultas, que el almacenamiento de las mismas y que la interacción con una base de datos a través de transacciones sea bastante sencilla.

JDO son los 'objetos de datos de java' y también consiste en una especificación de almacenamiento de objetos en Java.

Tiene muchas similitudes y algunas diferencias con JPA. La idea de definir un marco de trabajo a un alto nivel de abstracción se mantiene, pero JDO no se centra en almacenar datos en bases de datos relacionales. JDO encaja, por tanto, mejor que JPA con el DataStore ya que el éste no es relacional. Sin embargo también se encuentra limitado y no se puede utilizar todas las funciones de JDO en nuestra aplicación si esta tiene que funcionar en Google App Engine.

Por último, el API de nivel inferior es el 'interfaz de programación de aplicaciones de nivel inferior'. Se dice que es de nivel inferior ya que es la capa de almacenamiento de datos que se encuentra más cerca del DataStore.

Este API opera con datos y con operaciones relativamente primitivos. Al programar con este API hay que ser explícito ya que es necesario indicar no sólo qué quieres almacenar o recuperar sino cómo quieres hacerlo.

Este API, al igual que el propio DataStore, opera con entidades.

La ventaja de trabajar contra esta API es que permite tener un mayor control a costa de tener que comprender mejor cómo funciona el DataStore.

3.5 Sensor óptico

Al igual que ha ocurrido con el microcontrolador, para este proyecto, se ha utilizado un sensor óptico para captar las pulsaciones. No obstante, la selección, acondicionamiento y puesta a punto del sensor ha sido parte del trabajo final de grado "Estudio y diseño de un sistema de monitorización del ritmo cardiaco basado en un PSoC y un dispositivo móvil inteligente" realizado por Elías Hernández Gómez, por lo que en dicho trabajo se puede encontrar toda la información referente a esto.

No obstante, a continuación se detallan unos conceptos simples acerca del funcionamiento de estos sensores.

La aparición de este tipo de sensores se remonta unos tres años atrás, momento en el que empezaron a aparecer instalados en relojes deportivos, pulseras de actividad y otros tipos de wearables, encabezado por la multinacional Philips, la cual lanzó al mercado los primeros productos en este sector.

Hasta día de hoy, se han probado en variedad de situaciones e instalaciones para intentar obtener una respuesta lo suficientemente fiable.

El primer sensor óptico para medir la frecuencia cardiaca instalada en un dispositivo entre los mencionados anteriormente, apareció en el mercado a finales del año 2012 y su funcionamiento se basaba en un solo led que era capaz de emitir un haz de luz y captar con un receptor las pulsaciones, detectando el tiempo entre estas en milisegundos.

La fiabilidad de este tipo de sensores se vio superada por el siguiente modelo en salir a la luz que se basaba en el mismo comportamiento pero con dos leds, en vez de uno.

Esta vez, la precisión mejoró y no presentaba tantos problemas como su antepasado al medir la frecuencia cardiaca. Aun con todo, no es del todo fiable en pacientes con un color de piel más oscuro o en pacientes con mucho bello en la zona donde se situaba el sensor.

Por el contrario, este tipo de sensores empezó a ser apto para la práctica deportiva, puesto que el anterior modelo no era fiable en este ámbito.

No estando del todo satisfechos con los resultados obtenidos con estas tecnologías, el pasado año 2014, salió a la luz un nuevo sistema que lleva integrado tres dispositivos leds y cambió la ubicación de la toma de datos.

Si bien los otros sistemas colocaban los dispositivos en la muñeca, este nuevo sistema se colocaba en el antebrazo.

No obstante, la incomodidad que presentaba este sistema frente al poco incremento de precisión que implicaba, ha conseguido que no se termine de implantar en el mercado este tipo de sensores ópticos.

Por último, en el presente año, la multinacional Apple ha sacado al mercado su propia gama de relojes inteligentes con su modelo personalizable y disponible en tres versiones (clásica, deportiva y…….) en el cual viene integrado un sistema para la medición de la frecuencia cardiaca mediante cuatro leds, como se puede apreciar en la imagen:

Ilustración 23: Apple watch

A día de hoy, no se tiene constancia de la fiabilidad de este dispositivo en cuanto a lo que se refiere a la toma de pulsaciones en movimiento.

Todos estos funcionamientos no son más que un intento de llevar la tecnología empleada en la medicina al usuario no experto del que hemos comentado anteriormente en esta memoria.

En este caso de estudio, se va a realizar un pequeño circuito electrónico aprovechando la placa PSoC y su circuitería interna para realizar una pinza que consiga medir la frecuencia cardiaca, como si de un pulsioxímetro se tratara.

El funcionamiento de un pulsioxímetro es el mismo que el descrito anteriormente para el caso de los sensores ópticos incorporados en los wearables.

El dispositivo emite luz con dos longitudes de onda de 660 nm (roja) y 940 nm (infrarroja) que son características respectivamente de la oxihemoglobina y la hemoglobina reducida. La mayor parte de la luz es absorbida por el tejido conectivo, piel, hueso y sangre venosa en una cantidad constante, produciéndose un pequeño incremento de esta absorción en la sangre arterial con cada latido, lo que significa que es necesaria la presencia de pulso arterial para que el aparato reconozca alguna señal.

Mediante la comparación de la luz que absorbe durante la onda pulsátil con respecto a la absorción basal, se calcula el porcentaje de oxihemoglobina. Sólo se mide la absorción neta durante una onda de pulso, lo que minimiza la influencia de tejidos, venas y capilares en el resultado.

El pulsioxímetro mide la saturación de oxígeno en los tejidos, tiene un transductor con dos piezas, un emisor de luz y un fotodetector, generalmente en forma de pinza y que

se suele colocar en el dedo, después se espera recibir la información de: saturación de oxígeno, frecuencia cardíaca y curva de pulso.

Por tanto, para conseguir simular este comportamiento, el circuito electrónico a realizar será como el siguiente:

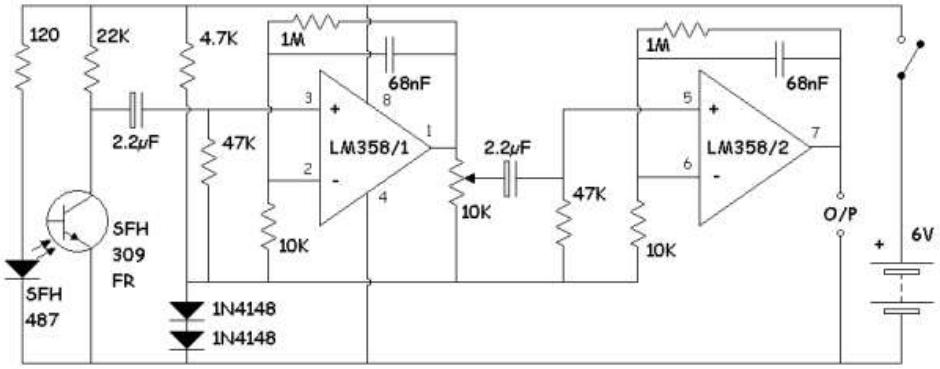

Ilustración 24: Circuitería para emular un pulsómetro

Con este circuito, obtendremos el acondicionamiento necesario para la conversión de milivoltios, que será la salida de nuestro circuito, a pulsaciones por minuto para poder monitorizarlas y crear alarmas, que es el objetivo principal de este proyecto.

El shield que se encarga del circuito de acondicionamiento ha quedado de la siguiente forma:

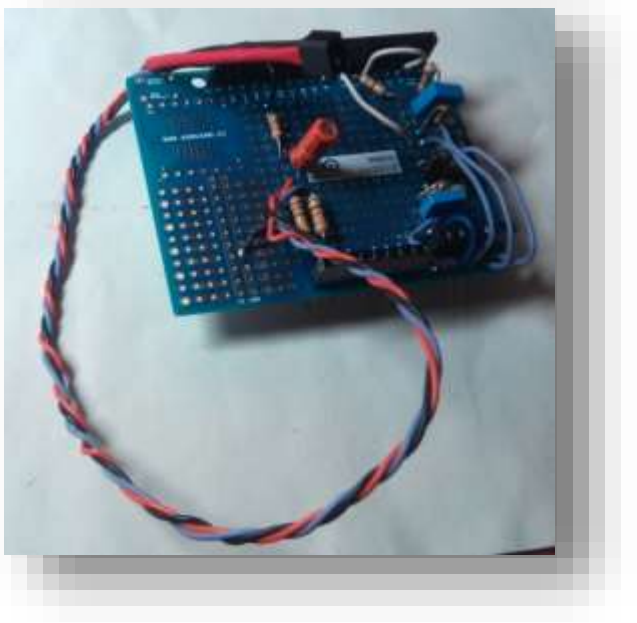

Ilustración 25: Shield de acondicionamiento del circuito

3.6 Android

Para el estudio de un caso práctico para este proyecto final de grado se ha utilizado el sistema operativo Android.

La elección de este sistema operativo, aparte de ser el más vendido en la industria del Smartphone, viene dada por la posesión de varios smartphones en casa con dicho sistema operativo instalado en ellos.

La versión concreta para la que se ha estado trabajando, es la versión 5.0, conocida por los propios desarrolladores de Android como "Lollipop", según se vio en la lista de versiones de este sistema operativo en el capítulo del "Estado del arte".

3.6.1 Eclipse

Tanto para la programación de la aplicación en Android como para la programación de la parte de Google App Engine, se ha utilizado el software "Eclipse".

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java DevelopmentToolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la CommonPublicLicense, pero después fue re-licenciado bajo la Eclipse PublicLicense. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

La interfaz que presenta el software Eclipse mientras se está trabajando con un proyecto es tal como la que se puede ver en la ilustración 4.

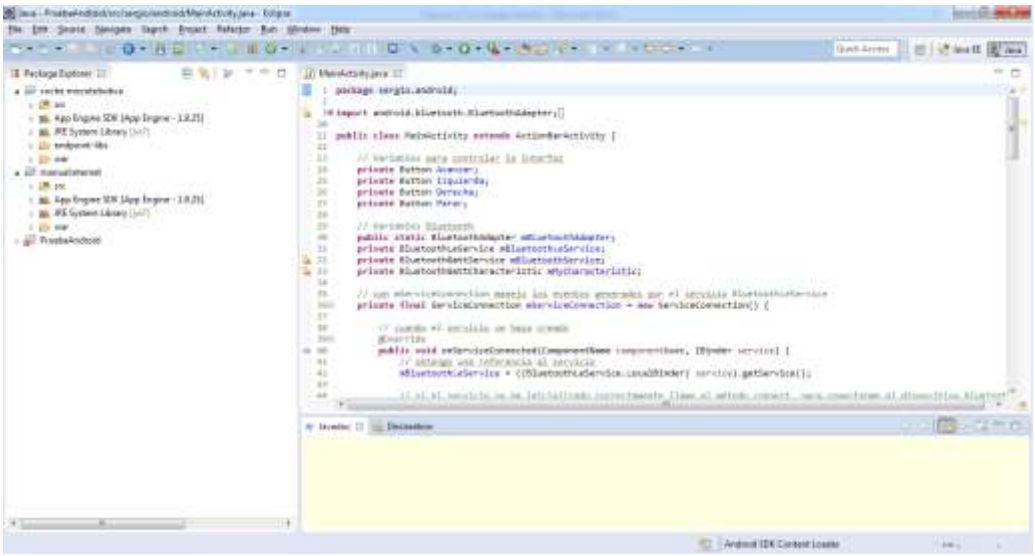

Ilustración 26: Interfaz gráfica del software Eclipse

Pero antes de seguir avanzando, es conveniente describir el proceso de instalación del software ya que hay que reunir una serie de requisitos y seguir un determinado orden, puesto que son varios paquetes a instalar para que todo funcione correctamente.

En primer lugar, visitando a la página oficial del software Eclipse, se encuentra un botón disponible dedicado a la descarga de las diferentes versiones del software. En este caso, se está trabajando con la versión Eclipse Mars, por lo que se seleccionará sobre esta para que comience la descarga en segundo plano de un ejecutable.

Acto seguido, se realizaran los pasos necesarios que vaya pidiendo el asistente de instalación hasta completar esta de manera satisfactoria.

Por otro lado, es imprescindible descargar la versión 6 del JDK (java development kit), siendo conscientes de que existen versiones posteriores a la 6. Dicha versión se puede encontrar indagando en su páginal oficial: http://www.oracle.com/index.html

Por último, para concluir la instalación del entorno de programación, también es necesario descargar las herramientas para Android conocidas como los SDK. Esto último es un paquete que incluye las herramientas necesarias para el desarrollo básico de aplicaciones en el entorno de Android.

Su descarga, se puede gestionar a través de la siguiente dirección: https://developer.android.com/sdk/index.html

El orden de instalación de las herramientas necesarias comentadas hasta ahora, si tienen que ver a la hora del correcto funcionamiento. Es necesario instalar la versión de desarrollo de java 6 en primer lugar. A continuación, se está en disposición de instalar el SDK manager de Android, puesto que este consulta si se ha instalado o no la versión de Java. Ya por último, se puede instalar el propio software de Eclipse Mars con total seguridad.

Una vez instalado correctamente el programa, la interfaz que presenta de cara al usuario al ejecutarlo, es tal como la que se muestra en la siguiente ilustración.

Ilustración 27: Vista principal al ejecutar Eclipse

A la hora de crear un nuevo proyecto, basta con pulsar en el menú superior sobre File -> new -> y seleccionar el tipo de proyecto que queremos generar. Al hacer estos pasos se despliega una variedad de tipos de proyectos más comunes, sin embargo, si el proyecto buscado no se encuentra entre esa lista, al final de esta se encuentra la opción de "Others" y al pulsar sobre ella se abre un "wizard" para la selección de todo tipo de proyectos que soporta el software Eclipse.

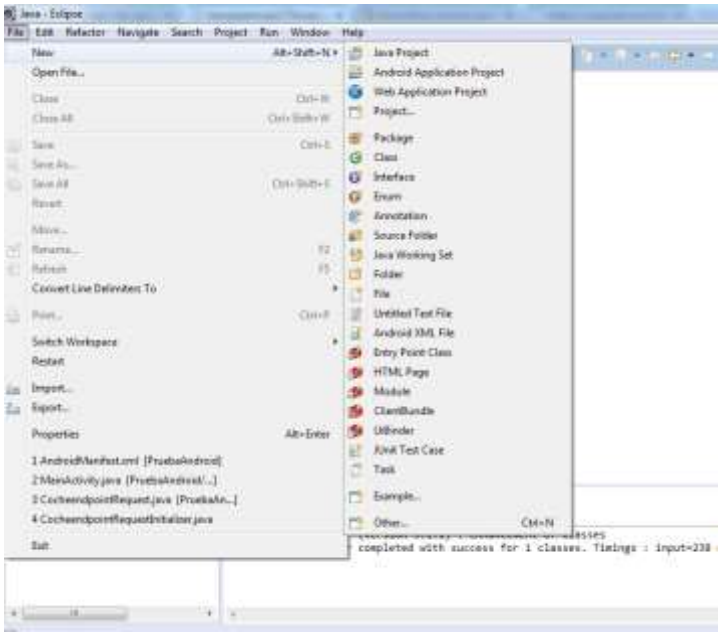

Ilustración 28: Creación de un nuevo proyecto

No obstante, para la realización de un proyecto del tipo "web application" para ejecutar en Google App Engine como en nuestro caso, antes de nada debemos instalar una serie de plugins para poder trabajar sobre el proyecto.

Para ello, en el menú superior, haciendo clic sobre "help"->"Instal new software.." se abre una ventana con el mismo aspecto de la mostrada en la siguiente ilustración.

Para poder instalar todos los plugins necesarios, se escribe en la barra la dirección https://dl.google.com/eclipse/plugin/4.4 y a continuación se hace clic sobre el botón "add".

Se desplegaran, de manera automática, todas las actualizaciones disponibles acerca de los plugins necesarios.

Basta con seleccionarlos todos mediante el botón "SelectAll"->"Next" y aceptar los términos y condiciones para comenzar con la instalación.

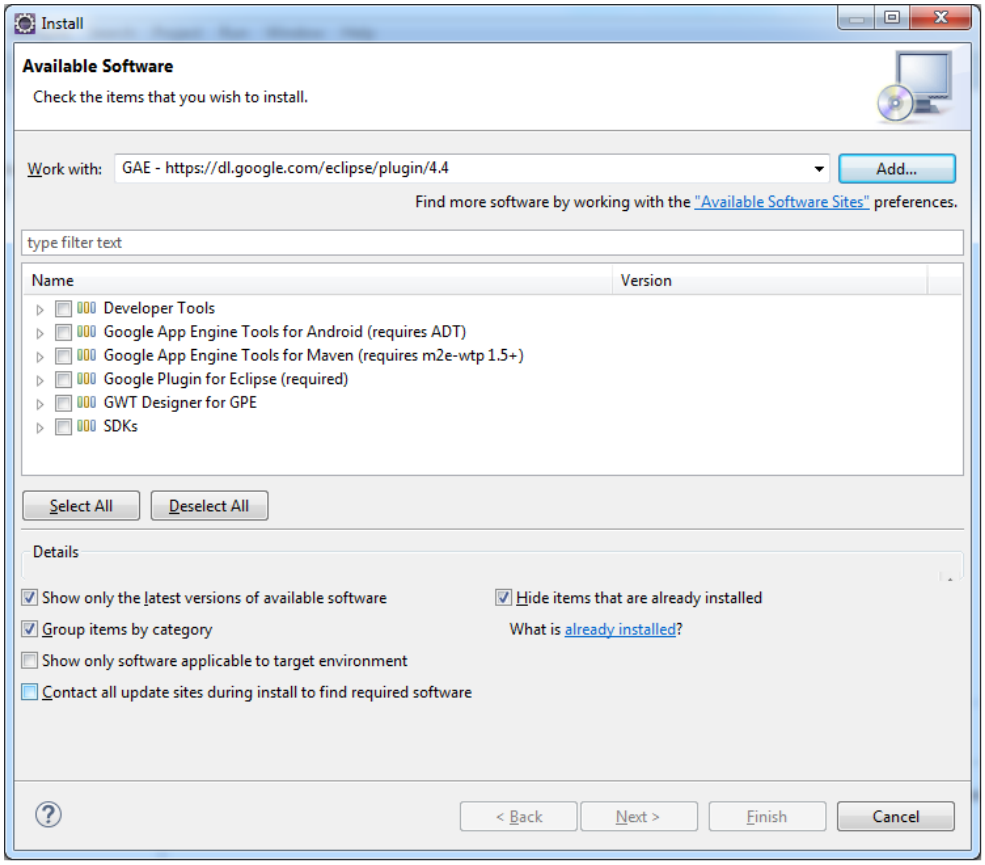

Ilustración 29: Instalación plugins Eclipse

Una vez instalados, para verificar que todo está instalado correctamente para comenzar a programar sobre la plataforma de Google, se tiene que verificar que sobre uno de los menús superiores aparece el siguiente icono, mediante el cual se tiene un acceso directo a la creación de proyectos relacionados con Google.

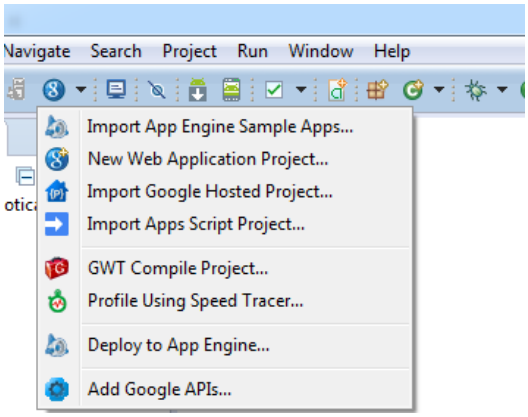

Ilustración 30: Detalle botón Google

Capítulo 4

Estudio de un caso práctico

4.1 Introducción

Una vez redactado los capítulos anteriores en los cuales se detallan los pros y las contras, comparativas y especificaciones de todos y cada uno de los elementos que han intervenido este proyecto, es hora de dar paso a la escritura de este último capítulo.

La finalidad de este capítulo no es más que la elaboración de un documento que sirva de utilidad al lector para poder realizar un proyecto de la misma envergadura de la cual consta este proyecto final de grado.

Para ello, se explicará con el máximo número de detalle los pasos que se han seguido para la elaboración del caso práctico.

4.2 Primeros pasos con Eclipse

4.2.1 Creación deuna web application project

Para la creación de un aplicación web, tal como se adelantó en las últimas páginas del capítulo anterior, si la instalación tanto del software como de los plugins han sido

correctamente ejecutadas, aparecerá un icono como el que se mostró también en el capítulo anterior que representa las opciones de proyectos que se pueden escoger y que incluyen a Google.

Para nuestro caso particular, haremos clic sobre "New WebApplication Project…" tal como se muestra en la siguiente ilustración.

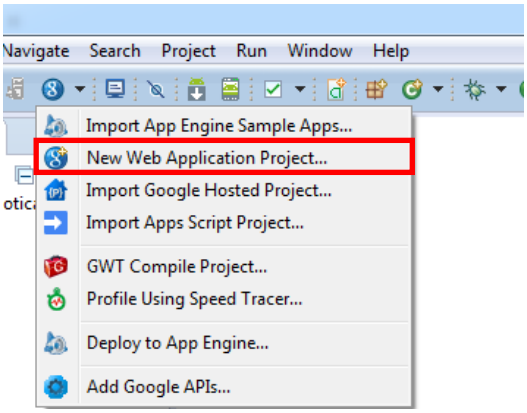

Ilustración31: Detalle de New Webb Application Project

Al pulsar sobre esta opción, se abrirá una nueva ventana con un asistente de ayuda para la creación del proyecto. Este asistente permitirá la selección de diferentes opciones así como la elección del nombre del proyecto, etc. Se puede ver en la siguiente ilustración una visión de la interfaz que presenta este asistente.

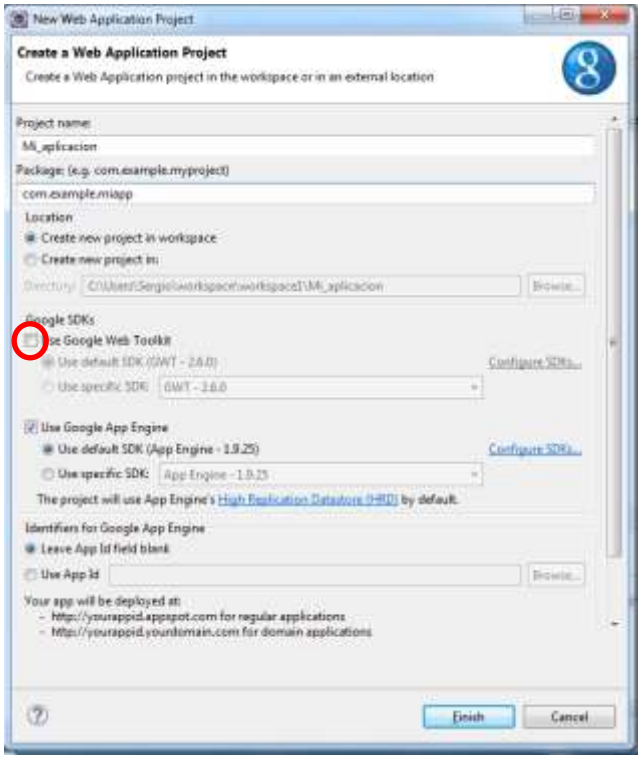

Ilustración 32: Asistente creación de un proyecto

El primer campo a rellenar será el nombre del proyecto que vamos a realizar y de la misma forma, a continuación se debe seleccionar el dominio en el que se va a crear, que por defecto será de la forma com.example.nombre_de_mi_app. A continuación basta con dejar las opciones marcadas tal como aparecen por defecto, con la única excepción de cerciorar que la opción "Use Google Tool Kit" esté desmarcada, y hacer clic en Finish para cerrar el asistente y crear el proyecto.

Tras realizar los pasos anteriores, en el margen izquierdo del panel de la vista principal de Eclipse se creará el directorio con las carpetas que se generan de forma automática cada vez que se cree un proyecto.

Dicha estructura, consta de dos carpetas, src y war, junto con una serie de librerías. A su vez, dichas carpetas se pueden desplegar en diferentes archivos.

Por una parte, en la carpeta src se alojarán los archivos .java que sean necesarios para la realización del proyecto.

Y por otro lado, en la carpeta war, se almacenarán librerías, archivos con extensión .html y archivos con extensión .xml, por lo que se puede ver perfectamente que esta segunda carpeta se utiliza para la creación de una interfaz web y todo lo que está relacionado con esta.

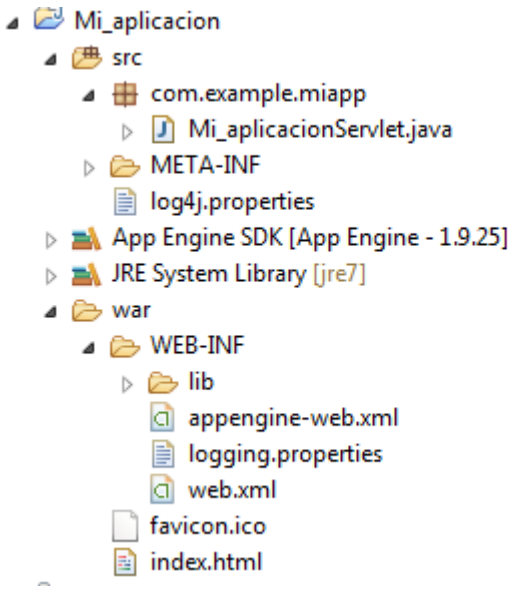

Ilustración 33: Detalle de la jerarquía de un proyecto

Una vez se tenga desarrollada la aplicación web con todo el código desarrollado incorporado en su carpeta correspondiente, para ejecutarlo en los servidores de google se hará clic con el botón derecho del ratón sobre la carpeta del proyecto y se seleccionará la opción de Run as y se escogerá WebbApplication.

A continuación, se podrá ver una secuencia de ejecución del código en la consola de información que está situada en la parte inferior de la ventana principal, indicando en todo momento el estado de la compilación del código programado.

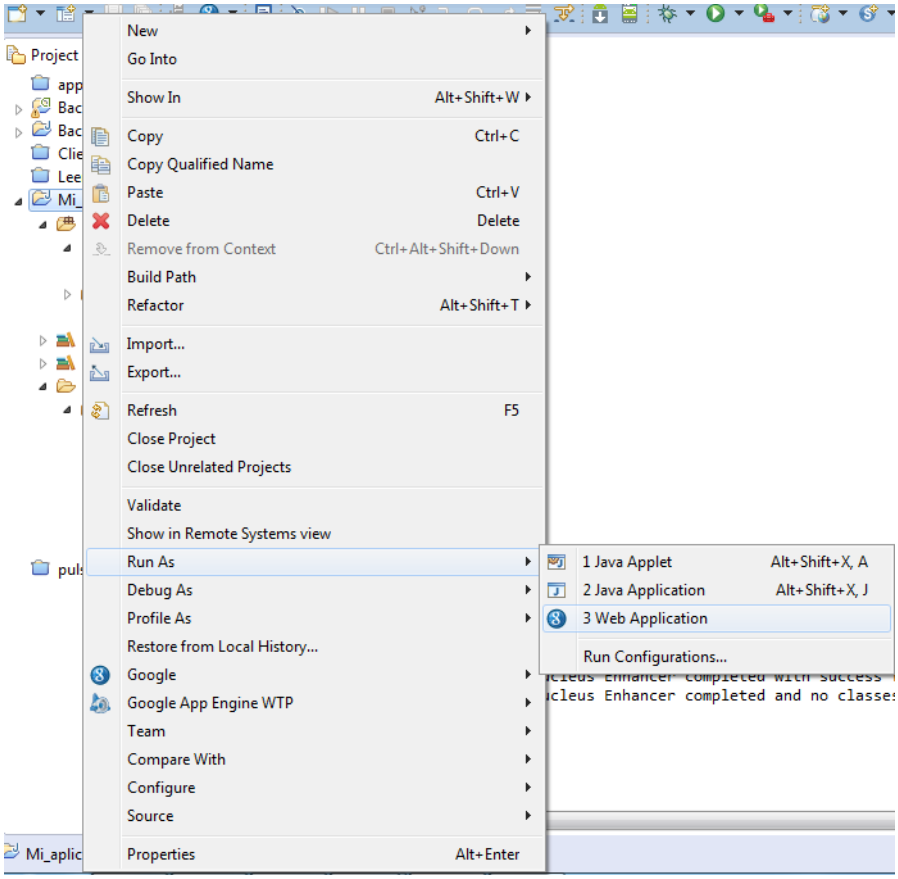

Ilustración 34: Ejecución del programa en los servidores de google

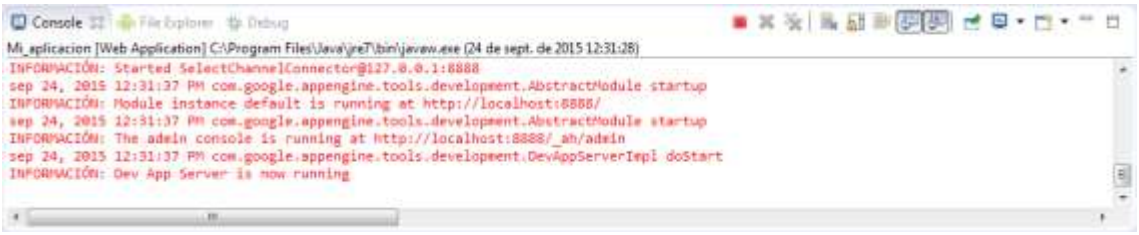

Ilustración 35: Información del compilador en la consola

Como se puede observar en la ilustración anterior, una vez finalizada la correcta compilación del programa, aparecerá el mensaje: "INFORMACION: Dev App Server isnowrunning" indicando de esta manera que el programa está siendo ejecutado en este mismo instante.

Al mismo tiempo, unas líneas más arriba, se puede observar en qué puerto local se está ejecutando el programa, puesto que aún no hemos desarrollado el código para ejecutarlo en la nube, sino que se está haciendo en local.

Si introducimos la dirección en la barra de direcciones de nuestro navegador (http://localhost:8888/ en nuestro caso), se puede ver lo siguiente:

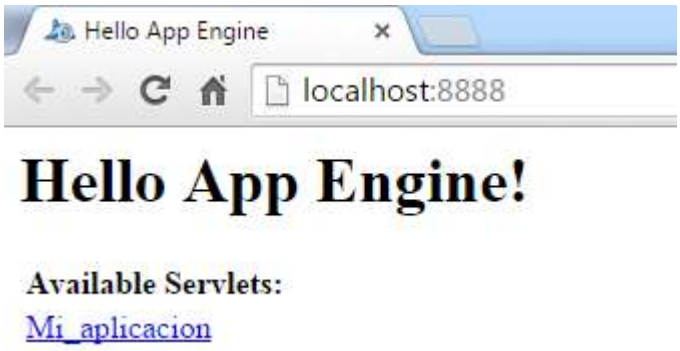

Ilustración 36: Detalle ejecución local del código

Si llegados a este punto pulsamos sobre "Mi_aplicacion" que es el proyecto que se ha creado y que se está ejecutando, veremos como se muestra en el navegador el famoso mensaje de programación básica "Hola mundo".

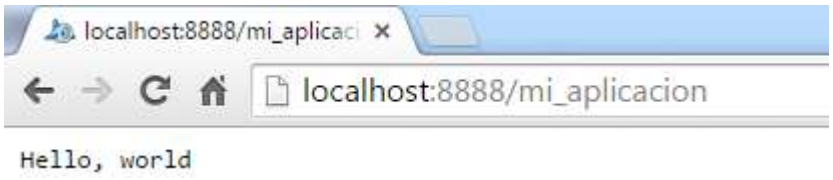

Ilustración 37: Mensaje Hola mundo

4.2.2 Creación de una aplicación Android

De la misma forma que en el apartado anterior se ha explicado la creación de un proyecto de tipo Web App, en este segmento se procede a describir los pasos a seguir para la creación de un proyecto Android, con todos los puntos a tener en cuenta.

En primer lugar, y de forma similar al apartado anterior, crearemos el proyecto haciendo clic en File -> New ->Others y del asistente para crear el nuevo proyecto, se desplegará la carpeta de Android y se seleccionará AndroidApplication Project, tal como se muestra en la siguiente ilustración:

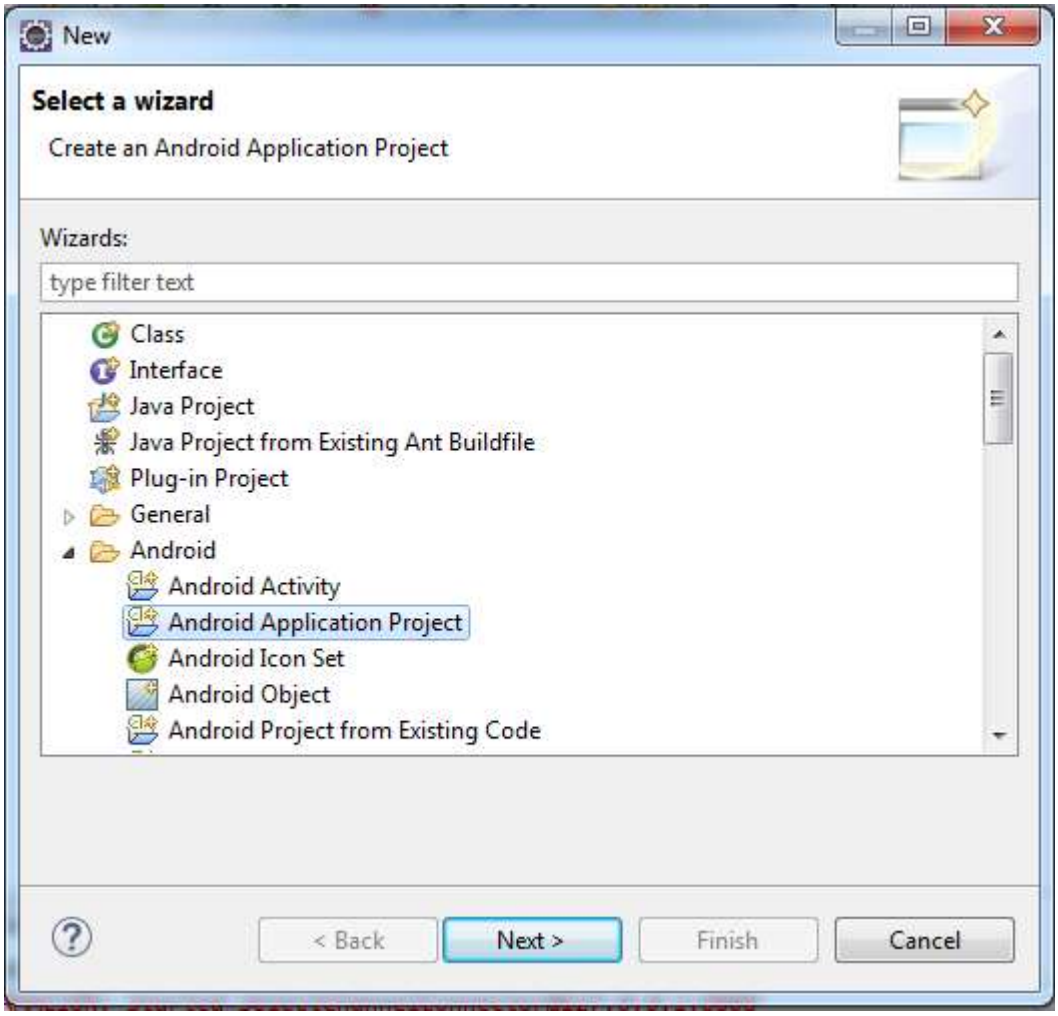

Ilustración 38: Creación de un proyecto Android

De igual manera que en el caso anterior, a partir de la selección del tipo de proyecto que queremos realizar, se abrirá un asistente para la creación del mismo en el cual podremos indicar desde el nombre del proyecto, el icono de la aplicación y diversos parámetros más.

NOTA: Esta vez, al no ser ninguno de estos puntos de especial interés se procede a describir lo que sucede una vez creado el proyecto.

Una vez creado el proyecto, se generarán de forma automática una serie de directorios sobre los que se ubican los diferentes archivos que componen una aplicación Android. Concretamente, en la carpeta "src" y "res" encontraremos los principales archivos necesarios para la creación de una aplicación, junto con el archivo "AndroidManifest.xml" situado suelto, dentro de la jerarquía del proyecto.

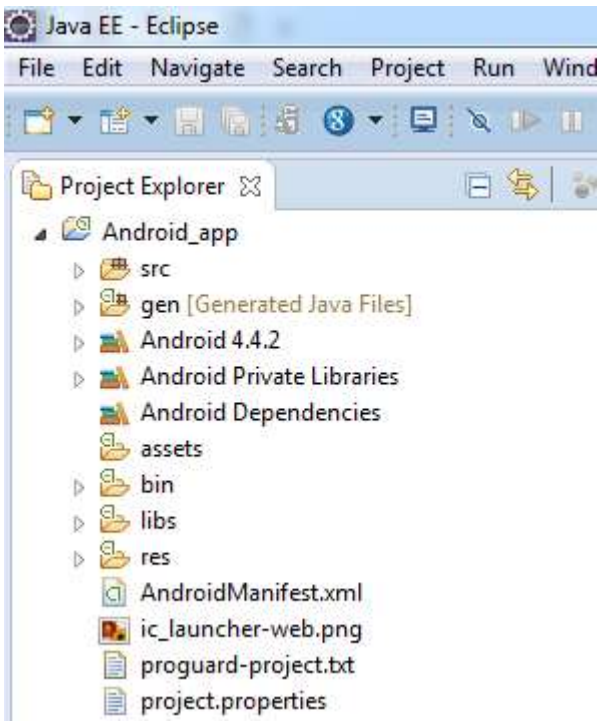

Ilustración 39: Jerarquía de un proyecto Android

De la misma forma que pasaba en el caso anterior, en un proyecto Android, dentro de la carpeta "src" se ubicarán los archivos con extensión java. Dentro de la carpeta "res" se encuentran todos aquellos archivos que están relacionados con la apariencia o interfaz de la aplicación, entre otros.

A la hora de compilar y ejecutar el programa, se plantean dos opciones. En primer lugar, la más sencilla de todas es haciendo clic sobre el botón que está presente en el menú superior que tiene aspecto de un botón "play" en color verde.

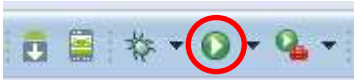

Ilustración 40: Detalle botón para compilar yejecutar

Este botón puede llevar a equivocaciones puesto que si se tiene más de un proyecto abierto al mismo tiempo, tenemos que asegurarnos que estamos compilando el programa deseado. Para ello, si se estuviera en ese caso, se pulsaría sobre la flecha desplegable y se seleccionaría el proyecto actual.

Por otro lado, la segunda opción que se plantea, es la misma que la que se ha descrito en el apartado anterior. Para ello basta con hacer clic derecho sobre la carpeta que representa nuestra aplicación Android y seleccionar "Run As" y a continuación hacer clic sobre "AndroidApplication", tal como se ve en la siguiente ilustración.

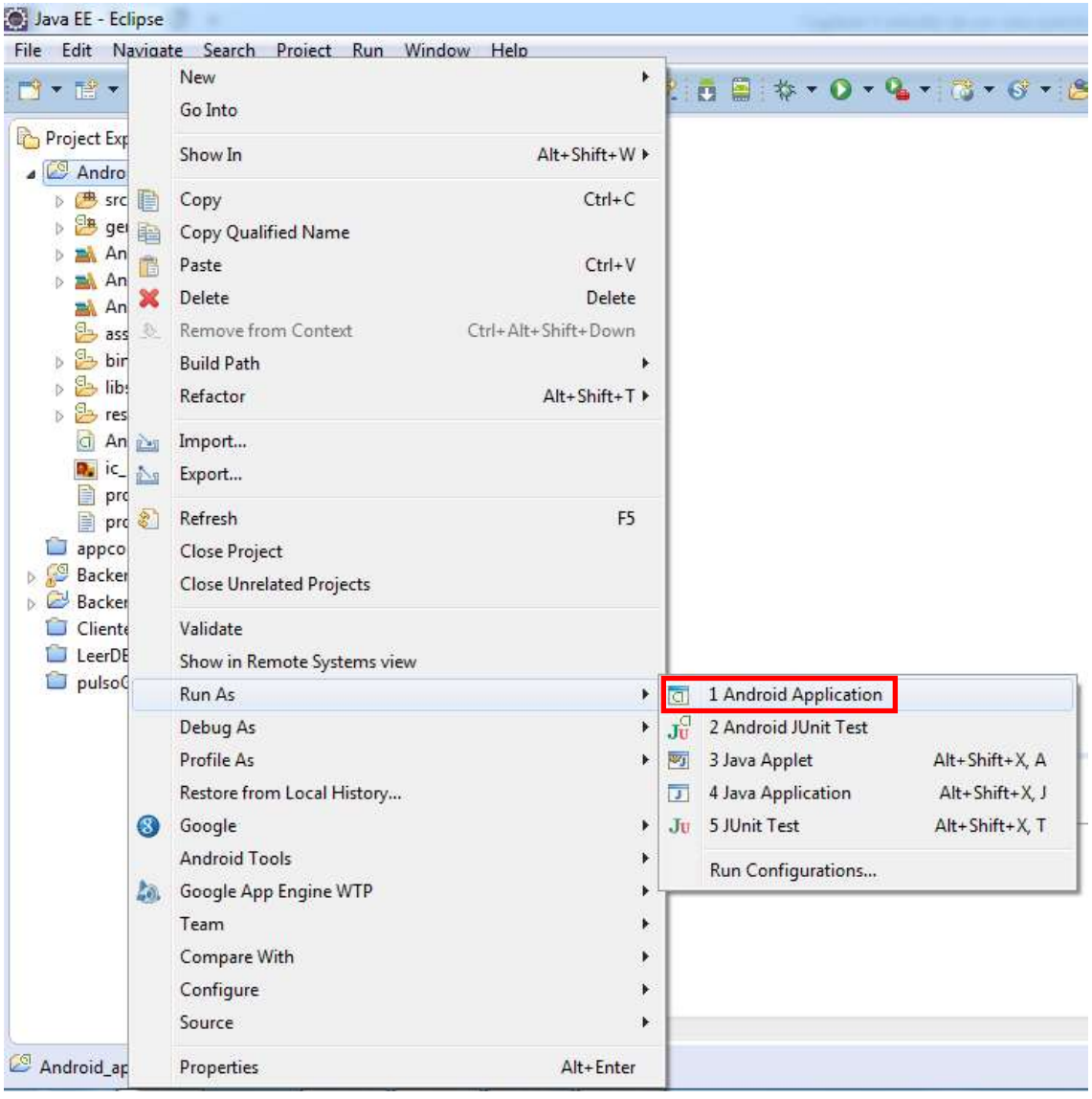

Ilustración 41: Detalle para compilar y ejecutar el proyecto

Una vez compilada la aplicación, se plantean de nuevo dos formas de comprobar el correcto funcionamiento de esta.

Por un lado, la opción más rápida es la de buscar en el directorio donde se almacena el proyecto en nuestro ordenador, y enviar al dispositivo móvil en cuestión que queramos probar la aplicación el archivo que se ha generado del tipo "mi_aplicación.apk". Basta con moverlo a la memoria del Smartphone e instalarlo.

Por otro lado, la opción que proporciona el software eclipse, no es más que simular el caso anterior en un dispositivo Android virtual generado por el propio software. Para ello, en primer lugar se creará un dispositivo pulsando sobre el icono del menú superior, tal como se muestra en la siguiente ilustración.

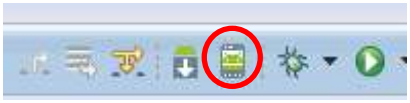

Ilustración 42: Detalle icono dispositivo móvil virtual

A continuación se desplegará una nueva ventana con un asistente en el cual se listan los dispositivos disponibles, y en el menú lateral derecho, ofrece la posibilidad de generar nuevas máquinas virtuales.

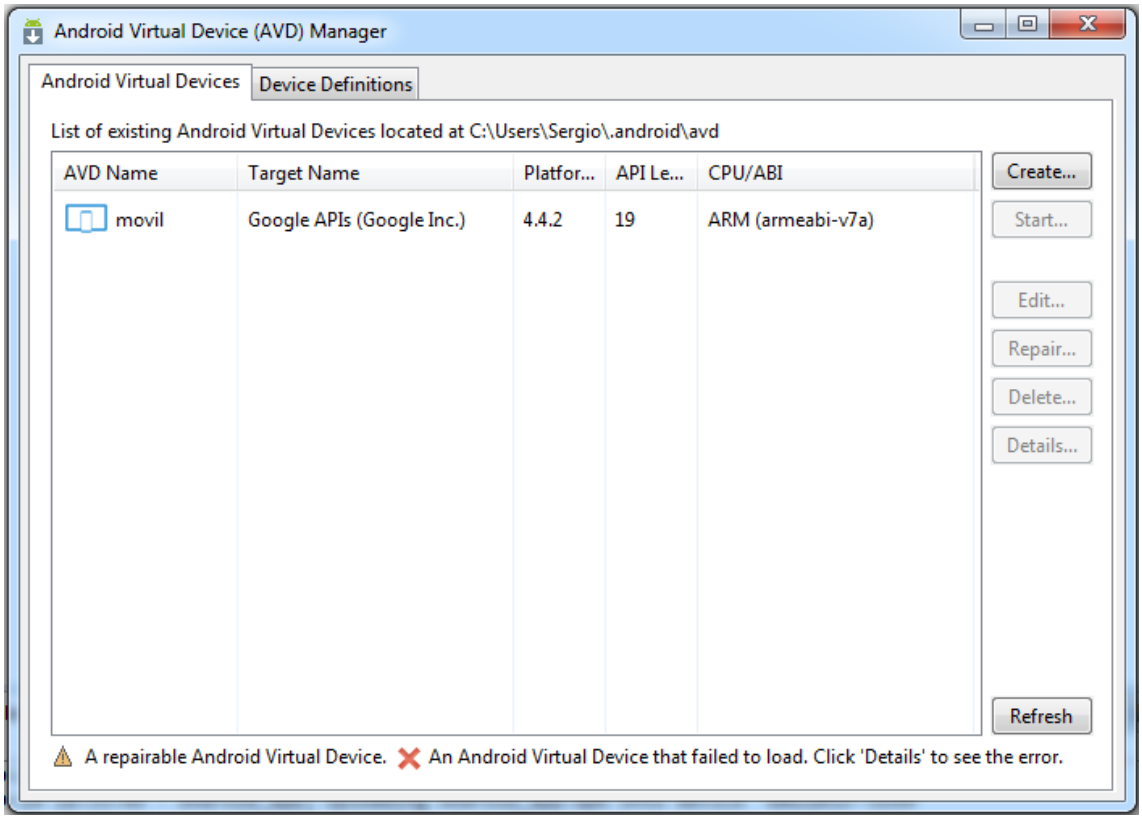

Ilustración 43: Asistente para la creación de dispositivos virtuales

Una vez generado el dispositivo virtual, haciendo clic sobre el botón "Start" situado en el lateral derecho, se iniciará el simulador. Es conveniente saber que este proceso puede tardar unos minutos.

Una vez inicializado, se mostrará en pantalla una nueva ventana con una interfaz que simulará la de un Smartphone, en la cual, tras desbloquear el dispositivo, ejecutará la aplicación que se haya creado, simulando su comportamiento real en un dispositivo móvil.

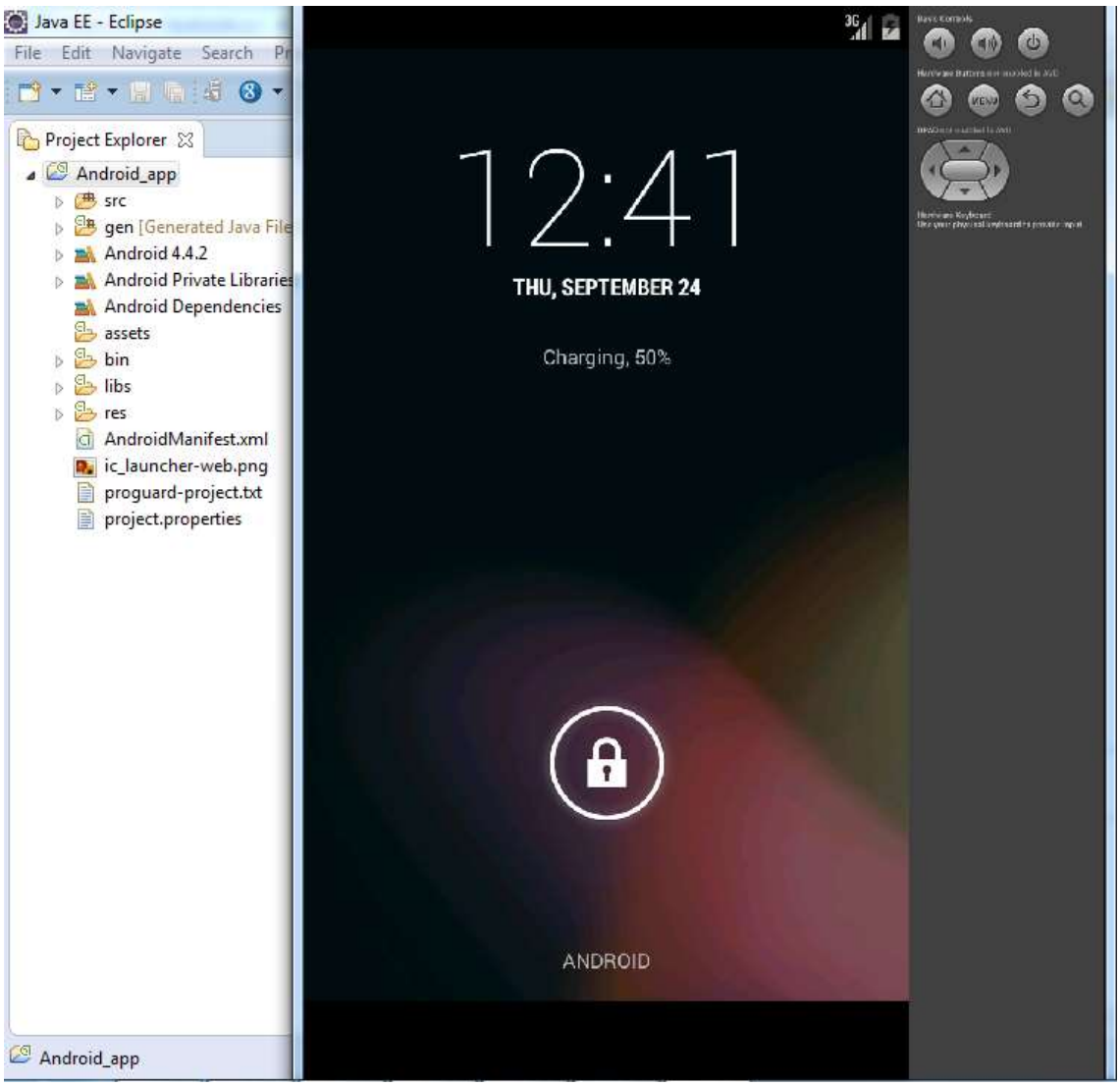

Ilustración 44: Vista general del dispositivo Android simulado

4.3 Creación de un Backend con googleEndPoints

A partir de este punto, y conforme vaya avanzando este apartado, se centrará la información en el trabajo realizado para poder llevar a cabo este proyecto final de carrera, en el cual se han utilizado diversas herramientas para la creación de un Backend y un Frontend para una aplicación de Android.

En primer lugar, se tratará de explicar los procedimientos necesarios para la creación del BackEnd alojado en los servidores de Google.

Para ello, el software Eclipse proporciona muchas facilidades. Muestra de ello es que haciendo clic con el botón derecho sobre la carpeta del proyecto con el que queremos trabajar, nos aparecen una serie de opciones como se ha visto anteriormente en alguna ocasión. Dejando el cursor sobre "Google" se despliegan otra serie de opciones entre las que se incluye "Generate App EngineBackend", tal como se puede ver ilustrado a continuación.

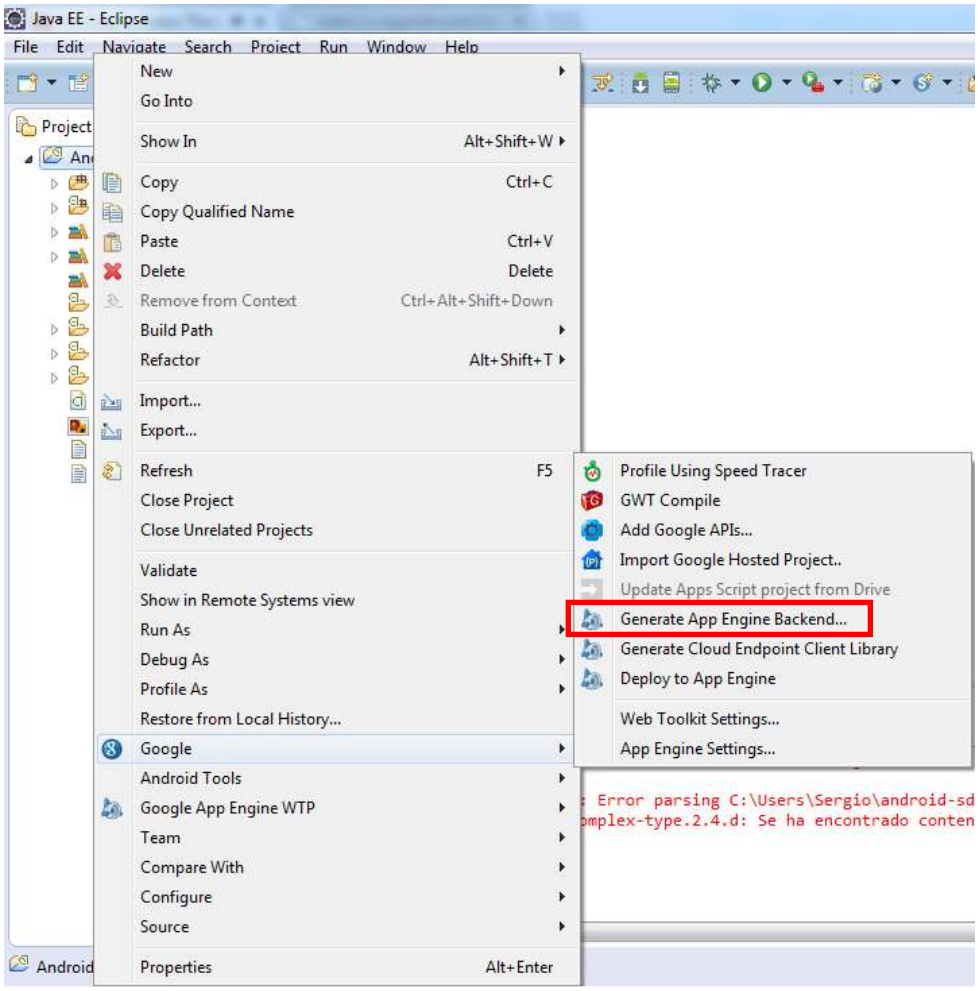

Ilustración 45: Generación de un Backend

.

Acto seguido se abrirá un asistente para finalizar el proceso de creación de un Backend. En nuestro caso, como se está ejecutando todo en local, basta con hacer clic en Create.

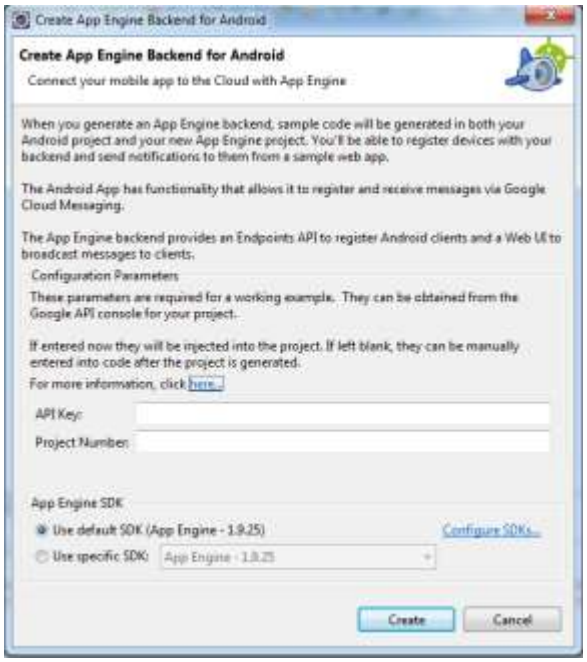

Ilustración 46: Asistente de creación del Backend

Una vez creado, en el explorador de archivos se podrá observar cómo se ha creado un nuevo directorio referente al Backend, como es la carpeta Android_app-App Engine.

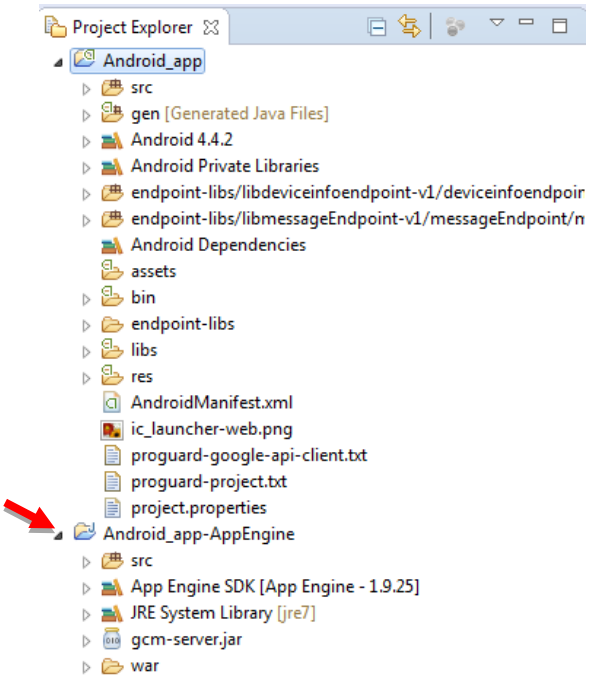

Ilustración 47: Directorios creados

Analizando los directorios que se tienen, se puede ver con claridad que estamos de nuevo ante los casos estudiados en los apartados 5.2.1 y 5.2.2, solo que esta vez se ha conseguido unificar los dos directorios en un solo proyecto. De esta forma, se puede comenzar a comentar el caso práctico de este proyecto final de grado.

Para el caso que se está tratando, se van a almacenar los datos correspondientes al id, al pulso cardiaco del paciente y fecha y hora a la que se realizó la toma de datos, mediante un tipo de dato '*timestamp*'.

Por esto, se crea una nueva clase en el proyecto del backend, a la que se le llamará ClasePulso, y presentará el siguiente aspecto.

```
@PersistenceCapable

public class ClasePulso {

    @PrimaryKey
    @Persistent(valueStrategy=IdGeneratorStrategy.IDENTITY)
    private Key key;

    @Persistent
    private int id;
    @Persistent
    private long fecha;
    @Persistent
    private int pulso;

        public long getfecha() { return fecha; }
        public void setfecha(long Fecha) { fecha = Fecha; }

        public int getpulso() { return pulso; }
        public void setpulso(int Pulso) { pulso = Pulso; }

        public int getid() { return id; }
        public void setid(int Id) { id = Id; }

        public Key getKey() {
            return key;
        }

        public void setKey(Key key) {
        this.key = key;
        }

        }
```

Como se puede observar, es tan sencillo como crear una clase con los parámetros que se desean almacenar en los servidores de google, haciéndolos persistentes mediante la anotación @Persistent y realizando a continuación los gets y los sets de estas variables. De la misma forma, la clase pública ClasePulso, también se hace persistente mediante la anotación @PersistanceCapable.

Un detalle a tener en cuenta es la creación de una clave de tipo 'Key' dado que el DataStore de Google trabaja con claves que genera aleatoriamente. Para evitar este problema, se genera una clave de tipo Key y más adelante se explicará el procedimiento para que utilice dicha clave para almacenar datos.

Una vez creada la clase ClasePulso, haciendo clic derecho sobre esta en la jerarquía de carpetas que aparece en el menú del lateral izquierdo de la pantalla principal, dejando el ratón sobre 'Google', aparece la posibilidad de pulsar sobre 'Generate Cloud EndpointClass'. Haciendo uso de esta herramienta, se puede ver como de manera automática se generan dos nuevos archivos java en el mismo directorio donde se ubica la ClasePulso; por un lado se genera la clase PMF.java y por otro lado se genera la clase ClasePulsoEndpoint.java.

Estas dos clases son las encargadas de generar los servicios para poder gestionar los datos desde la aplicación a los servidores de google. Es decir, de manera automática se han generado las clases necesarias para realizar los servicios de insert, update, list, get y remove, o lo que es lo mismo, se ha generado el código para insertar, listar, actualizar, borrar u obtener datos en el DataStore de google.

No obstante, dentro de la clase ClasePulsoEndpoint la cual contiene los sercicios, hay que realizar unas modificaciones.

Como se ha comentado anteriormente, es conveniente generar una clave de tipo key, y mediante el método KeyFactory se puede generar una clave a partir de una variable de nuestro programa. Esto solucionará los posibles errores al invocar a la función 'Get' dado que el método insert se realizará con normalidad.

La razón, es porque al realizar un 'Insert' mediante la invocación de su función, los datos se subirán correctamente al DataStore pero se almacenarán con una clave generada aleatoriamente y la cual se desconoce. Si acto seguido se quiere invocar a la función 'Get' no bastará con pasarle el id del cual queremos conseguir, ya que el DataStore necesitará la clave. Es por esto por lo que se modifican estos servicios para que se genere la clave a partir del id y, por tanto, conocer en cualquier momento su clave asociada para tener acceso a estos datos.

Por tanto, las modificaciones se han de realizar en la clase ClasePulsoEndpoint, concretamente dentro de los servicios de 'insert' y 'get', para que quede de la siguiente forma:

```
@ApiMethod(name = "getClasePulso")
public ClasePulso getClasePulso(@Named("id") Long id) {
    PersistenceManager mgr = getPersistenceManager();
    ClasePulso clasepulso = null;
    try {
        Key miclave= new KeyFactory.Builder(ClasePulso.class.getSimpleName(), id).getKey();
        clasepulso = mgr.getObjectById(ClasePulso.class, miclave);
    } finally {
        mgr.close();
    }
    return clasepulso;
}
```

Y de la misma forma, el servicio de insertar nuevos datos quedaría:

```
@ApiMethod(name = "insertClasePulso")
public ClasePulso insertClasePulso(ClasePulso clasepulso) {
    PersistenceManager mgr = getPersistenceManager();
    clasepulso.setKey(KeyFactory.createKey(ClasePulso.class.getSimpleName(),clasepulso.getid()));
    try {
        if (containsClasePulso(clasepulso)) {
            throw new EntityExistsException("Object already exists");
        }
        mgr.makePersistent(clasepulso);
    } finally {
        mgr.close();
    }
    return clasepulso;
}
```

Llegados a este punto, es conveniente hacer un punto y aparte en la secuencia de creación del programa para explicar el procedimiento de ejecución de este en la nube, y no en servidores locales como hasta ahora se ha visto.

Para que el código programado se ejecute en servidores de Google y tengamos acceso a un dominio que proporcione Google, en primer lugar habrá que crearse un nuevo proyecto en la consola de desarrollo de Google. Para ello, se introduce la siguiente dirección en el navegador que se esté usando:

https://console.developers.google.com/

En caso de no estar iniciada la sesión con una cuenta de Google, este será el primer paso a realizar para poder acceder a la consola de desarrollo. Por esta razón, al introducir la dirección en el navegador se llegará a la siguiente ventana:

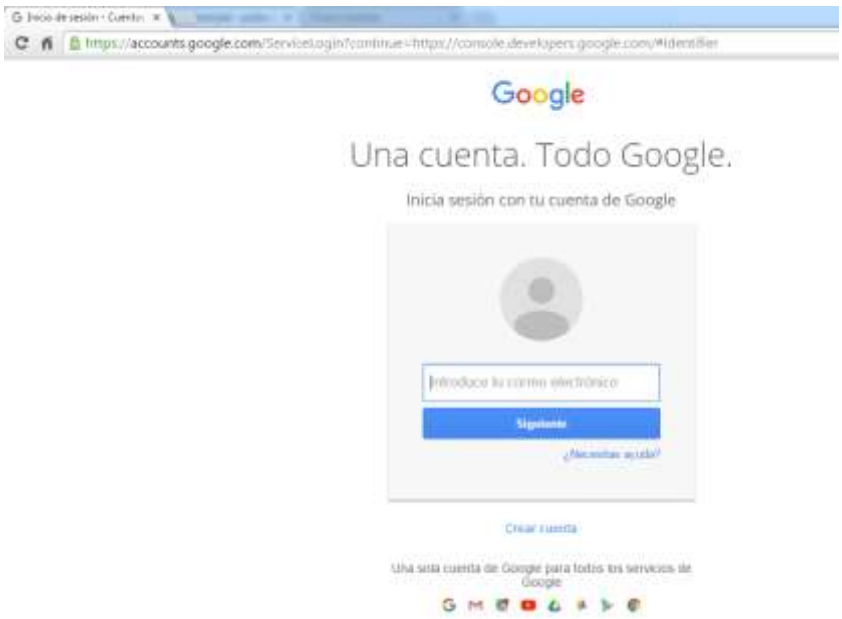

Ilustración 48: Inicio de sesión en la cuenta de Google

Tras iniciar sesión, ya se está en disposición de crear un nuevo proyecto. Para ello es tan sencillo como hacer clic sobre 'Selecciona un proyecto' y elegir la opción 'Crear proyecto…'.

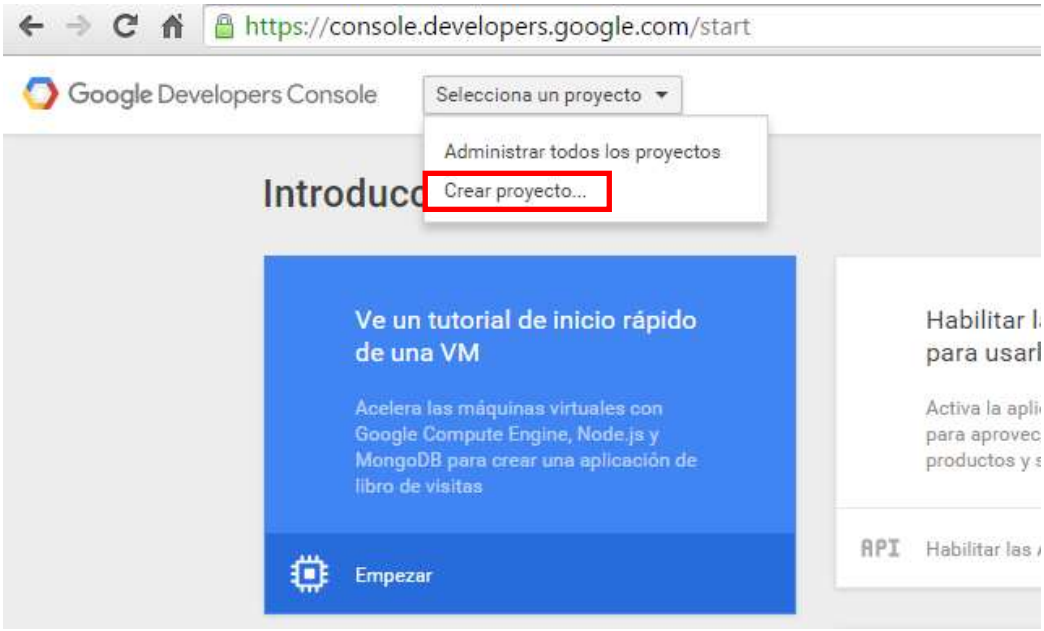

Ilustración 49: Creación de un nuevo proyecto

Acto seguido se abrirá el asistente para la creación del proyecto en el cual permitirá al usuario la elección del nombre del proyecto, el ID, y una serie de opciones.

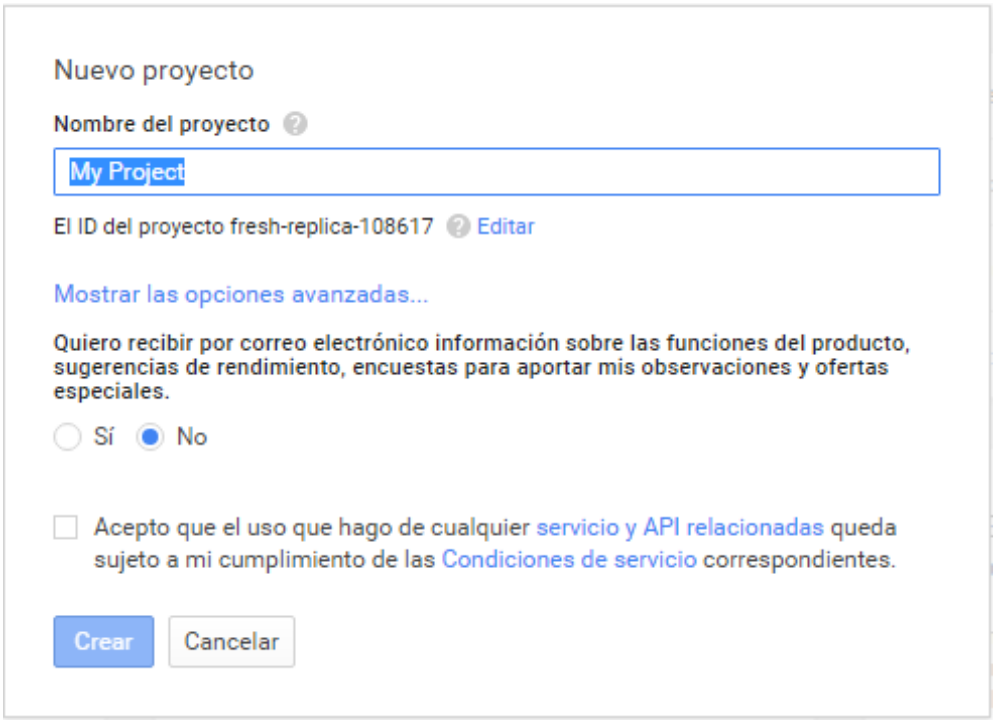

Ilustración 50: Asistente para la creación del proyecto

Ahora bien, para que el proyecto, al cual se le está realizando el caso de estudio, se ejecute bajo el dominio asociado al proyecto creado en la consola de Google, habrá que realizar una serie de cambios en el código fuente del programa.

Por un lado el ID que se ha generado de manera aleatoria al introducir el nombre del proyecto, o bien el que hayamos elegido nosotros para nuestro proyecto (véase que aunque se genere aleatoriamente permite la posibilidad de crearlo a nuestro gusto) hay que introducirlo dentro de la carpeta correspondiente a la web app del proyecto en Eclipse. Concretamente, dentro de la carpeta 'war' desplegamos la carpeta 'WEB-INF' y se abre el archivo 'appengine-web.xml' y en la tercera línea, se debe introducir el ID de la aplicación creada en la consola de Google.

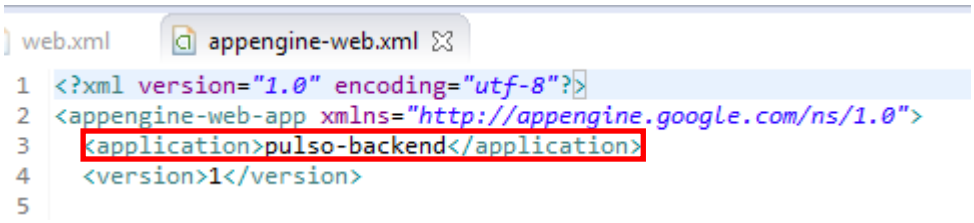

Ilustración 51: Inserción del ID del proyecto

Para finalizar la ejecución en los servidores de Google de la aplicación, habrá también que modificar dos aspectos más. Esta vez se modifican dentro del proyecto de Android, concretamente dentro de la carpeta 'src' dentro, a su vez, del paquete de nuestra aplicación, y más específicamente en la clase 'CloudEndpointsUtils'.

Dentro de esta clase, se pude observar en primer lugar una línea que hace referencia a la ejecución en local de la aplicación y que por tanto, al estar por defecto en 'true' tendremos que cambiarla a 'false' quedando de la siguiente forma:

```
protected static final boolean LOCAL_ANDROID_RUN = false;
```

Y por último, se deberá indicar en qué dominio se aloja nuestra aplicación. Dicho dato no es más que el nombre de nuestro proyecto expresado de la siguiente manera: nombre_aplicación.appspot.com, por lo que dicha línea a modificar, para el caso de estudio que se está redactando, quedaría así:

```
protectedstaticfinal String LOCAL_APP_ENGINE_SERVER_URL_FOR_ANDROID =
"www.pulso-backend.appspot.com";
```

Ahora bien. Una vez creadas las clases EndPoints y modificadas para que generen la clave a partir del ID se está en disposición de programar la aplicación de Android, la cual se encargará de llamar al servicio Insert para realizar la subida de datos al DataStore de Google, o a la función listar para obtener los datos desde el cliente, etc.

Se presenta un amplio abanico de posibilidades que permitirán realizar diversas operaciones de interés para el programador y posteriormente para el usuario final de la aplicación.

4.3.1 Creación de la clase 'Insertar datos en la nube'

En primer lugar, se debe aclarar la forma en la que ambas aplicaciones (tanto la aplicación que se está desarrollando en este proyecto, como la aplicación realizada en el otro proyecto comentado y que sirve de enlace y comunicación entre el PSoC y el Smartphone) se comunican. Este procedimiento se realiza mediante un contentprovider y un content resolver.

Este procedimiento es la única forma de intercambiar datos entre dos aplicaciones Android instaladas en un mismo dispositivo. La implementación del contentprovider se realiza en la aplicación que se encarga de la comunicación con el microcontrolador, y la implementación del content resolver se implementará en este caso de estudio para poder acceder a dichos datos.

Para invocar al content resolver, en primer lugar se deben crear las variables que definan las columnas de la tabla a la que se va a acceder así como la dirección de la Uri que contiene la base de datos en el otro proyecto. Por tanto, nuestro proyecto tiene que contar con las siguientes líneas de código en cualquiera de las funciones que se quieran recuperar o trabajar con datos de la base de datos creada por otra aplicación.

```
publicstaticfinal String COL_CAMPO1 = "created_at";
publicstaticfinal String COL_CAMPO2 = "dato";
privatestaticfinal String uri = "content://elias.pulsaciones/pulsaciones";
publicstaticfinal Uri CONTENT_URI = Uri.parse(uri);
```

Dichas líneas de código deben de verificar con la implementación del contentprovider en el código de la aplicación encargada de la comunicación del microcontrolador con el Smartphone.

A continuación, se procede a explicar el fragmento de código que se encarga de realizar la conexión con el servidor:

```
@Override
protected Void doInBackground(Void... params) {
	try {

	String[] projection = new String[] { Ejemplo._ID,
	Ejemplo.COL_CAMPO1, Ejemplo.COL_CAMPO2 };

	Uri ejemploUri = CONTENT_URI;

ClasepulsoendpointBuilderendpointBuilder = newClasepulsoendpoint.Builder(
                                AndroidHttp.newCompatibleTransport(),
newJacksonFactory(), newHttpRequestInitializer() {

		publicvoid initialize(HttpRequesthttpRequest) {

	}
});

catch (Throwablee) {
		msError = e.toString();
		e.printStackTrace();

		}
```

De esta forma, se realiza la creación del Builder, y para la conexión con el content resolver y su posterior procesamiento de datos, basta con crear un cursor para ir recorriendo la base de datos e ir realizando las operaciones oportunas. En este caso, se ha implementado de la siguiente forma.

```
ContentResolvercr = contexto.getContentResolver();
Cursor cur = cr.query(ejemploUri, projection, null, null, null);
	if (cur.moveToFirst()) {
		JDateloDate = newJDate();
			do {
				intelcampo0 =
Integer.valueOf(String.valueOf(loDate.getMes()) +
String.valueOf(loDate.getDia()) + String.valueOf(loDate.getHora()) +
String.valueOf(loDate.getMinuto())+ String.valueOf(loDate.getSegundo()))
+ cur.getInt(cur.getColumnIndex(Ejemplo._ID));

mselcampo2 = cur.getString(cur.getColumnIndex(Ejemplo.COL_CAMPO1));
loDate = newJDate(mselcampo2);
intelcampo2 = cur.getInt(cur.getColumnIndex(Ejemplo.COL_CAMPO2));
mselcampo2 = String.valueOf(elcampo2);
ClasePulsodatouno = newClasePulso();
datouno.setId(elcampo0);
datouno.setFecha(newDateTime(loDate.getDate()));
datouno.setPulso(elcampo2);
endpoint.insertClasePulso(datouno).execute();
}
while (cur.moveToNext());
	}
```

De esta forma en el campo 0 se está insertando en el DataStore, mediante el servicio 'Insert', definido y explicado anteriormente, el valor del ID. De la misma manera se está realizando lo propio para el campo 2, que en este caso inserta la fecha de la forma año, mes día, hora, minuto y segundo.

Como aclaración, para simplificar el manejo con los datos de la fecha, se ha incluido una clase que aporta una serie de funciones como comparación, cambio de tipo de dato, cambios de formato de mostrar el valor de cara al usuario, etc. Dicha función se puede ver adjunta en los anexos de este proyecto, bajo el nombre de JDate.

Por último, para concluir esta función, se ha implementado el siguiente código que se procede a explicar a continuación:

```
@Override
publicvoidonPostExecute(Void result) {
	try {

//Activo el botonmientrasejecutalafuncion, paraqueestédesactivoantes y no
acumulartoques

		moBoton.setEnabled(true);

		if(msError.equals("")){

		//Lanzounmensajepara saber que he acabadolatarea
			Toast.makeText(contexto, "Procesoterminadocorrectamente",
Toast.LENGTH_LONG).show();

		}else{

		Toast.makeText(contexto, msError, Toast.LENGTH_LONG).show();
			}
		} catch (Throwablee) {

			e.printStackTrace();
		}
	}
```

Dentro del onPostExecute simplemente se activa el botón para poder ejecutar la función y se lanza un mensaje en la aplicación para que muestre al usuario una notificación flotante con un mensaje que avisará de cuándo el proceso de subir datos al DataStore se ha terminado correctamente o si ha habido algún error.

4.3.2 Creación de la clase 'Listar datos de la nube'

Para la creación de esta clase, la implementación del código es muy similar a la implementación de la clase anterior. Hasta la creación del Builder, es tal y como se ha definido y explicado para el caso anterior.

A partir de este punto, sí que hay que realizar diversos cambios para poder implementar correctamente el servicio que ejecuta la acción de listar los datos en la nube para poder trasegar con ellos y, como en este caso, graficarlos en la pantalla de nuestro Smartphone.

El primer paso a realizar, antes de ejecutar la función onPostExecute, es declarar las dos variables, una local y otra modular, que permitirán el manejo de los datos obtenidos del DataStore de Google. Para ello, definimos las siguientes variables:

```
ListClasePulsoloDatos = endpoint.listClasePulso();
moDatos = loDatos.execute().getItems();
```

A continuación, se almacenan los datos en una variable de tipo ArrayList, se comparan y se ordenan por fecha, para finalmente ir convirtiéndolos mediante un cast a tipo entero, para de esta forma poder graficarlos.

La función para graficar se ha añadido mediante una librería, puesto que el propio Software de Eclipse no incluye en sí las herramientas necesarias para añadir un campo de tipo gráfico.

Con todo lo comentado, el código queda de la siguiente forma:

```
@Override
publicvoidonPostExecute(Void result) {
	try {
		moBoton.setEnabled(true);

		//Almacenolosdatosenunarraylist
		ArrayList<Number>loNumero = newArrayList<Number>();
		Arrays.sort(moDatos.toArray(),newjava.util.Comparator<Object>()
{

	//Comparolosdatosparaordenarlosporfecha
		publicint compare(Object o1, Object o2) {
		return (int) (((ClasePulso)o2).getFecha().getValue() -
((ClasePulso)o1).getFecha().getValue());
				}
			});
```

```
//Recorrolosdatos
                    for(ClasePulsoloDato : moDatos){
//Conviertolosdatos a tipointegerporproblemasde casting conlaclasedegraficar
                          loNumero.add(new Integer(loDato.getPulso()));
                    }
```

Una vez hecho esto, basta con graficar los datos recuperados y mostrar una serie de mensajes de cara al usuario.

En primer lugar, la gráfica se genera de la siguiente manera:

```
      // Añadimosunaseriededatos:

      XYSeriesseries1 = newSimpleXYSeries(loNumero,
      SimpleXYSeries.ArrayFormat.Y_VALS_ONLY, "Series1");

// Modificamosloscoloresdelaprimeraserie

LineAndPointFormatterseries1Format = newLineAndPointFormatter(
                    Color.rgb(0, 200, 0), color.rgb(0, 100, 0),
      Color.rgb(150, 190, 150),
      newPointLabelFormatter(Color.WHITE));

      mySimpleXYPlot.addSeries(series1, series1Format);

      //Limitamos el numerodeetiquetasporgráfica

            mySimpleXYPlot.setTicksPerRangeLabel(10);
```

De esta forma estaría solucionada la gráfica, por lo que solamente faltaría implementar los mensajes flotantes que aparecerán en la aplicación de cara al usuario, para saber en todo momento si se está ejecutando de forma correcta o si hay cualquier error.

```
if(msError.equals("")){
      Toast.makeText(contexto, "Proceso terminado correctamente, Número
datos(" + String.valueOf(moDatos.size()) + ")" , Toast.LENGTH_LONG).show();
            }
            else{
                  Toast.makeText(contexto, msError + ", Númerodatos(" +
String.valueOf(moDatos.size()) + ")", Toast.LENGTH_LONG).show();
                  }
            } catch (Throwablee) {
                  Toast.makeText(contexto, e.toString(),
Toast.LENGTH_LONG).show();
                  e.printStackTrace();
                  Toast.makeText(contexto, msError + ", Númerodatos(" +
String.valueOf(moDatos.size()) + ")", Toast.LENGTH_LONG).show();
            }
      }
```

De esta forma, se puede saber en todo momento si el proceso se ha realizado de la manera deseada o si, por el contrario, ha habido cualquier error.

4.3.3 Creación de la clase 'Buscar anomalías en los datos'

La última clase que se ha implementado ha sido la de buscar, entre los valores delimitados por dos fechas introducidas por el usuario de la aplicación, anomalías en el pulso registrado; si bien pueden ser bradicardias o taquicardias.

Para la creación de esta clase, se han utilizado dos variables: 'mlBradi' y 'mlTaqui' de forma que se ha creado un algoritmo con la siguiente lógica:

En primer lugar, se obtienen las fechas límite mediante una caja de texto que se muestra por pantalla al usuario. Este campo ha sido configurado como modo de entrada de tipo 'Date' para facilitar la introducción de la fecha al usuario de la aplicación.

A continuación, mediante varios bucles 'if' se comprueba si dentro de dicho rango de fechas ha habido algún caso de pulsaciones inferiores a 60 (en cuyo caso se tomaría como bradicardia) o superiores a 120 (taquicardia). De esta forma, si se encontrar algún valor por debajo o por encima de dichos rangos, se incrementaría la variable 'mlBradi' o 'mlTaqui', respectivamente.

Una vez recorridos todos los datos para el rango establecido, se hace un recuento de las variables que se han ido incrementando, de manera que si se han recibido más de cinco datos en cualquiera de estas, saltará un mensaje en la pantalla y avisará al usuario si ha habido bradicardia o taquicardia. Además, en dicho mensaje se mostrará el número de datos que ha contabilizado como anomalías, que no es más que el valor de las variables 'mlBradi' o 'mlTaqui', o en algunos casos, el valor de ambas, puesto que dentro de un mismo rango se puede dar el caso de que haya habido ambas anomalías.

Por tanto, y según lo descrito hasta ahora, la función queda implementada a través del siguiente código, realizando la búsqueda de anomalías perfectamente.

```
        ListClasePulsoloDatos = endpoint.listClasePulso();
        moDatos = loDatos.execute().getItems();
        Arrays.sort(moDatos.toArray(),newjava.util.Comparator<Object>() {

        publicint compare(Object o1, Object o2) {

return (int) (((ClasePulso)o2).getFecha().getValue() -
((ClasePulso)o1).getFecha().getValue());

  }
        });

        mlBradi = 0;
        mlTaqui= 0;
        for(ClasePulsoloDato : moDatos){
              booleanlbcon = true;
```

```
	if(loDato.getFecha().getValue()<moDate1.getDate().getTime()){

			lbcon=false;
				}

	if(lbcon&&loDato.getFecha().getValue()>moDate2.getDate().getTime()){

			lbcon=false;
				}

	if(lbcon&&loDato.getPulso() < 60){

			mlBradi++;
				}

	if(lbcon&&loDato.getPulso() > 120){

			mlTaqui++;
				}
			}

		} catch (Throwablee) {
			msError = e.toString();
			e.printStackTrace();
		}
		returnnull;
	}

@Override

publicvoidonPostExecute(Void result) {
	try {
		moBoton.setEnabled(true);

		if(mlBradi>5){
			msArritmia="Bradicardia";

	}
		if(mlTaqui>5){
			msArritmia+=" Taquicardia";
			}

		if(msArritmia.equals("")){
			msArritmia="No tienes anomalías";
			}

	AlertDialog.BuilderdlgAlert  =new
	AlertDialog.Builder(contexto);
	dlgAlert.setMessage(msArritmia);
	dlgAlert.setTitle("Tieneanomalías?");
	dlgAlert.setPositiveButton("OK", null);
	dlgAlert.setCancelable(true);
	dlgAlert.create().show();
```

4.3.4 ActivityMain y Layaout de la aplicación Android

A pesar de incluir en el anexo la programación completa del trabajo, se adjunta en este apartado el código correspondiente al ActivityMain y al Layaout de la aplicación. Al estar el código comentado, carece de sentido explicar qué hace cada variable, puesto que tanto los nombres, como las funciones, están hechos con la intención de ser transparentes para el lector de este trabajo.

Empezando en el ActivityMain, las librerías que hay que exportar son las siguientes:

```
importcom.androidplot.xy.XYPlot;
importandroid.app.Activity;
importandroid.os.Bundle;
importandroid.view.Menu;
importandroid.view.MenuItem;
importandroid.view.View;
importandroid.widget.Button;
importandroid.widget.EditText;
importandroid.widget.TextView;
importandroid.widget.Toast;
```

Por otro lado, nada más iniciar la clase MainActivity, se declaran los siguientes botones:

```
private Button moBoton;
privateXYPlotmySimpleXYPlot;
private Button moBoton2;
private Button mobuttonArritmia;
privateEditTextmoeditTextFecha1;
privateEditTextmoeditTextFecha2;
```

A continuación, dentro del método onCreate, se ejecuta el siguiente código:

```
//Botónparaactualizarla BD y parabuscaranomalías
moBoton = (Button)this.findViewById(R.id.button1);
moBoton2 = (Button)this.findViewById(R.id.button2);
mobuttonArritmia = (Button)this.findViewById(R.id.buttonArritmia);

//Botónparaejecutarlagráfica
mySimpleXYPlot = (XYPlot) findViewById(R.id.mySimpleXYPlot);

//Camposdestinados a lasfechas
moeditTextFecha1 = (EditText)this.findViewById(R.id.editTextFecha1);
moeditTextFecha2 = (EditText)this.findViewById(R.id.editTextFecha2);
moeditTextFecha1.setText( newJDate().getDia()  + "/" + newJDate().getMes()  +
"/" + newJDate().getAno() );
moeditTextFecha2.setText( newJDate().getDia()  + "/" + newJDate().getMes()  +
"/" + newJDate().getAno() + " " + newJDate().getHora() + ":" +
newJDate().getMinuto() + ":" + newJDate().getSegundo());
```

Y por último, las funciones que darían acceso mediante el clic de los botones a sus correspondientes clases, se ejecutan de la siguiente manera:

```
publicvoidaccederResolver(View v){

//Botónactivopara no estartocandolapantalla y llamandocte a latareaasincrona
moBoton.setEnabled(false);

//Accesoalconten provider

TareaAsincronaResolvertarea = newTareaAsincronaResolver(this,
(TextView)findViewById(R.id.tView1), (TextView)findViewById(R.id.tView2),
moBoton);
tarea.execute();
      }
```

De igual manera, para la gráfica:

```
publicvoidaccederGrafico(View v){

//Botónactivopara no estartocandolapantalla y llamandocte a latareaasincrona
moBoton2.setEnabled(false);

//Accesoalconten provider

TareaAsincronaListartarea = newTareaAsincronaListar(this, mySimpleXYPlot ,
moBoton2);
tarea.execute();
      }
```

Y por último, para la detección de anomalías:

```
publicvoidarritmia(View v){

//Botónactivopara no estartocandolapantalla y llamandocte a latareaasincrona
mobuttonArritmia.setEnabled(false);

//Accesoalcontenprovider

TareaAsincronaAnomaliastarea;
      try {
tarea = newTareaAsincronaAnomalias(this,
moeditTextFecha1.getText().toString(), moeditTextFecha2.getText().toString(),
mobuttonArritmia);
      tarea.execute();
            } catch (Exception e) {
            Toast.makeText(this, e.toString(), Toast.LENGTH_LONG).show();
                  }

      }
```

Y por otro lado, para que la interfaz que presente al usuario sea la adecuada, se ha implementado el siguiente Layout en xml:

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context="com.example.contentresolver.MainActivity">

<TextView
android:id="@+id/tView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/hello_world"/>

<TextView
android:id="@+id/tView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/tView2"
android:layout_below="@+id/tView2"
android:text="Large Text"
android:textAppearance="?android:attr/textAppearanceLarge"/>

<Button
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:onClick="accederResolver"
android:text="ACTUALIZAR DB"/>

<TextView
android:id="@+id/tView3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Detectararritmias"/>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">

<EditText
android:id="@+id/editTextFecha1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="datetime">

<requestFocus/>
</EditText>
```

```
<EditText
android:id="@+id/editTextFecha2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="datetime">

<requestFocus/>
</EditText>


</LinearLayout>

<Button
android:id="@+id/buttonArritmia"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:onClick="arritmia"
android:text="Buscaranomalías"/>

<Button
android:id="@+id/button2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:onClick="accederGrafico"
android:text="Gráfico XY"/>

<com.androidplot.xy.XYPlot
android:id="@+id/mySimpleXYPlot"
android:layout_width="272dp"
android:layout_height="100px"
android:layout_marginLeft="10px"
android:layout_marginRight="10px"
android:layout_marginTop="10px"
android:layout_weight="0.62"
title="Pulso"/>


      </LinearLayout>
```

4.3.5 Android Manifest

Para este archivo, solo hay que tener en cuenta el permiso que hay que dar a la aplicación para el acceso al contentprovider de la otra aplicación, puesto que los permisos de conexión a internet para la conexión con el DataStore se realizan de forma automática al generar el BackEnd. Por tanto, hay que añadir tan solo los siguientes permisos:

```
<uses-permissionandroid:name="elias.pulsaciones.READ_DATABASE"/>
uses-permissionandroid:name="elias.pulsaciones.WRITE_DATABASE"/>
```

4.4 Conclusiones

Durante este capítulo, se ha podido ver, desde los pasos necesarios para la instalación de todas las herramientas necesarias, hasta la programación que conlleva el diseño y la implementación de una aplicación de estas características y ha quedado reflejado las principales opciones con las que cuenta el entorno de programación utilizado junto con la plataforma Android.

Como se ha podido observar, con conocimientos de programación en Java para la parte de Google platform, y de Android para la parte de la interfaz e interconexión con el BackEnd, son muchas las posibilidades que ofrece este entorno de desarrollo para aplicaciones Android.

Capítulo 5

Conclusiones

5.1 Conclusiones

Tal como se ha podido ver a lo largo del desarrollo de este trabajo final de grado, las herramientas que presenta Google Cloud Platform son muy amplias. Abarca muchas temáticas y ha conseguido que, tanto juntando una aplicación en Android, como haciendo tan solo una aplicación web, se haya convertido en una herramienta muy polivalente y versátil.

Para el caso de estudio que se ha realizado, si se consiguiera optimizar el error en la medición de pulsaciones, sería una buena herramienta para tener controladas las frecuencias cardiacas de los posibles pacientes y podría servir de utilidad a deportistas o incluso para personas con alguna patología cardiaca.

5.2 Futuros trabajos

Con Google Cloud se ha visto que prácticamente se puede implementar cualquiera de las aplicaciones que se deseen, gracias a la multitud de servicios que ofrece.

Es por esta razón por la que el campo de investigación hacia nuevos trabajos es muy amplio y se podría mejorar este trabajo incluyendo multitud de funciones diferentes a las ya creadas, como bien podría ser la generación de alertas ante un ritmo cardiaco determinado, bien sean por correo, o mor mensajería móvil.

También cabe la posibilidad de añadir funciones para fitness de manera que mediante la creación de un algoritmo se puedan calcular, de forma aproximada, el gasto calórico del paciente.

No obstante, hay que tener en cuenta que se ha estado limitando las dimensiones de trabajo a la versión gratuita, por lo que si en vez de utilizar DataStore como método de almacenaje de datos, utilizamos cualquier otro servicio de los que Google ofrece u otra plataforma de las que se han podido ver a lo largo de este trabajo, se abre otro abanico de posibilidades en el desarrollo de aplicaciones.

Referencias

[1] Amazon Web Services: https://aws.amazon.com/es/

[2] Google Cloud Platform: https://cloud.google.com/

[3] Fiware: https://www.fiware.org/

[4] Xively: https://xively.com/

[5] Microsoft Azure: https://azure.microsoft.com/es-es/

[6] Sensor Cloud: http://www.sensorcloud.com/

[7] arduino: https://www.arduino.cc/

[8] PSoc: http://www.cypress.com/products/programmable-system-chip-psoc

[9] GWT: http://www.gwtproject.org/

Bibliografía básica

- Google App Engine Java and GWT Application Development (Daniel Guermeur & Amy Unruh)
- El gran libro de programación avanzada con Android (José Enrique Amaro Soriano)
- Desarrollo de Aplicaciones con Java (Henry terrero & José Paredes)