Tesis doctoral en Ingeniería Informática

OPENVEND: HACIA UN ECOSISTEMA ABIERTO PARA EL VENDING EN LA ERA DE INTERNET DE LAS COSAS

ANTONIO SOLANO TARROC

Ingeniero Superior de Telecomunicaciones

Director: Dra. NATIVIDAD DURO CARRALERO

Co-director: Dra. RAQUEL DORMIDO CANTO

Departamento Informática y Automática
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia

Madrid, 2017

Tesis doctoral en Ingeniería Informática

OPENVEND: HACIA UN ECOSISTEMA ABIERTO PARA EL VENDING EN LA ERA DE INTERNET DE LAS COSAS

ANTONIO SOLANO TARROC

Ingeniero Superior de Telecomunicaciones

Director: Dra. NATIVIDAD DURO CARRALERO

Co-director: Dra. RAQUEL DORMIDO CANTO

Departamento Informática y Automática
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia

Madrid, 2017

DEDICATORIA

A mi hija Martina

AGRADECIMIENTOS

Agradezco a las profesoras Natividad Duro Carralero y *Raquel Dormido Canto por brindarme la oportunidad de emprender este proyecto con el respaldo de la UNED, así como a los 17 alumnos que han contribuido con sus proyectos fin de carrera a construir OpenVend: José Ignacio Mantero, Iván Pérez, Francisco J. González, Patxi Ballesteros, Ricardo Cárdenas, Ángel Cifuentes, Juan Miguel Sánchez, Víctor J. González, José Daniel Fernández, Rafael Alonso, Luis Miguel Cabezas, Pedro Martínez, Francisco Javier González, Javier Mira, Manuel García, Francisco Paños y David Díaz. Agradecer también a Joan Gispert por compartir sus conocimientos sobre la industria del vending.*

Todo esto nunca hubiera sido posible sin el apoyo incondicional de mi familia, y sin el amor de Merche y Martina. Este es también vuestro premio.

RESUMEN

Esta tesis describe cómo aplicar los paradigmas de Internet de las Cosas (IoT) a los puntos de venta desatendidos y en concreto a las máquinas de vending.

De cara a afrontar con garantías la inminente cuarta revolución industrial en el sector del vending, esta tesis propone acelerar su transformación digital. Para ello, cada máquina de vending es modelada en Internet como una tienda *online*, lo que permite a los usuarios comprar y pagar desde su móvil. A partir de este simple escenario se analizan y proponen diversas tecnologías web para construir una solución de uso sencillo y bajo coste.

Teniendo en cuenta que el sector del vending está muy fragmentado y opera con márgenes comerciales muy bajos, parte de la tesis se centra en la búsqueda de eficiencias operacionales que justifiquen las inversiones necesarias para conectar las máquinas de vending a Internet. En esta línea, se propone una solución de bajo coste basada en dispositivos de *hardware* abiertos y en un modelo de *software* como servicio (SaaS). La solución tiene que ser muy fácil de gestionar con el objetivo de minimizar los costes de operación y mantenimiento. Para lograr estos objetivos, se ha diseñado una arquitectura en nube privada con capacidades de *Big Data.* Dicha arquitectura permite automatizar la provisión de los dispositivos IoT y registrar todos los flujos de información para su posterior procesado, y poder así generar analíticas avanzadas.

Por otro lado, se propone un sistema de pago móvil alternativo a los actuales basados en aplicaciones nativas (Apps). No es necesario descargarse ninguna aplicación porque se usan tecnologías web (webapps). El sistema propuesto se focaliza en la usabilidad, ya que se busca la simplicidad e inmediatez de los pagos con monedas. Es un sistema universal, es decir, el usuario no tiene que estar vinculado a ninguna institución financiera en concreto, operador de telefonía móvil, o fabricante de dispositivos para poder pagar desde su teléfono móvil. El diseño de la solución debe asegurar que el consumidor,

cuando realiza la compra, se encuentra delante de la máquina para recoger el producto, ya que éste se ha comprado en una tienda virtual en Internet y hay que detectar la presencia del usuario para poder dispensarlo.

Bajo estas premisas, nace OpenVend, como una solución integral de *software* y *hardware* abierto que mejora la eficiencia operativa de las empresas de vending y que permite a los consumidores comprar los productos de las máquinas de vending desde sus teléfonos móviles.

Este trabajo también introduce un mecanismo de auto-configuración o *plug-and-play* que permite crear la representación digital de un sistema complejo o servicio en Internet de forma automática. Una vez creada la representación digital del servicio o sistema, el usuario puede interactuar con el sistema físico en cuestión desde su representación digital, y todo ello, con plenas garantías de confidencialidad y privacidad de los datos del usuario.

Los resultados de esta investigación pueden, por extensión, aplicarse a diferentes sectores y escenarios donde sea necesario realizar el pago de bienes y servicios de forma desatendida, o simplemente para eliminar colas. Por ejemplo: el pago de un parking, el alquiler de una bicicleta, el pago en restaurantes y supermercados, etc.

SUMMARY

OPENVEND: TOWARDS AN OPEN ECOSYSTEM FOR VENDING MACHINES IN THE ERA OF INTERNET OF THINGS

This work describes how to apply Internet of the Things (IoT) paradigms to unattended point of sales such us vending machines.

In order to face the 4th industrial revolution and contribute to mitigate climate change, this thesis proposes to accelerate the digital transformation of vending sector. To this end, each vending machine is modeled on the Internet as an online shop, allowing end-users to buy and pay from their mobile. Bear in mind this simple scenario, pervasive web and mobile technologies are analyzed to build up a low cost and easy-to-use solution.

Taken into account that vending sector is very fragmented and operates at very low margins, this thesis also looks for operational efficiencies which may justify the investments needed to connect all vending machines to Internet. In this sense, a low-cost solution is proposed based on open hardware devices and a software as a service (SaaS) model. The solution has to be very easy to operate to minimize maintenance cost and must be ready for Big Data. To achieve these objectives, a private cloud architecture is designed to automatize the provisioning and management of devices.

On the other hand, an alternative mobile payment system is proposed that seeks the simplicity and immediacy of payments with coins. The system must be universal, i.e., the end-user does not have to be affiliated to any financial institution, mobile operator or device manufacturer in order to pay with his mobile phone in any of the vending machines connected to the Internet. The design of the solution must ensure that the consumer is in front of the machine when the purchased product is dispensed.

Under these premises, OpenVend has been implemented. OpenVend is a comprehensive open source & hardware solution that improves the operational efficiency of vending operators and enables e-commerce transactions from consumers' mobile phones by modeling the machine as an online shop.

The outcomes of this research can be extended and applied to different sectors and scenarios where it is necessary to pay for goods and services in an unattended manner, or simply to eliminate queues. For example: paying a parking lot, renting a bicycle, paying in restaurants and supermarkets, etc. In any of these scenarios, the digital modeling of a complex system or service (digital twin) can be instantiated in the cloud automatically through the plug-and-play mechanism presented on this work. Once the digital twin is instantiated, the end-user can interact with the physical system through his digital twin.

OpenVend has been built with security by design and therefore guarantees the privacy of end-user data.

LISTA DE PALABRAS CLAVE

Internet of Things, Vending, OpenVend, Cloud Computing, Big Data, Mobile Computing, Pervasive Computing, HTML5, Webapps, Web of Things, Cashless Payments, Mobile Payments, Near Field Communications, Digital Twin.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

DESCRIPCIÓN DE LA TESIS

A continuación, se exponen los capítulos que componen esta tesis doctoral.

En el capítulo 1 de **Introducción**, se presenta el concepto de Internet de las Cosas y se comenta como este nuevo paradigma va a trasformar nuestra sociedad y por qué es necesario replantearse la manera de hacer las cosas. En este sentido, esta tesis se centra en la transformación de puntos de venta desatendidos, comúnmente conocidos como máquinas de vending.

En el capítulo 2, **Contexto**, se proporciona una retrospectiva de cómo se llegó a concluir que el sector del vending presentaba los suficientes retos para que su estudio mereciese ser objeto de esta tesis. Afrontar tales retos desde una perspectiva de negocio, ha requerido proponer soluciones viables y realizar prototipos mínimamente funcionales. La realización de dichos prototipos se ha instrumentado a base de varios ciclos de innovación con proyectos de fin de Carrera/Grado de estudiantes de la Escuela Técnico Superior de Ingeniería Informática de la UNED.

En el capítulo 3 se exponen las **Hipótesis y objetivos** a alcanzar, indicando en qué publicación, o publicaciones, se reflejan dichos objetivos, al tratarse de una tesis por compendio de publicaciones.

Se continúa con el capítulo 4, **Marco teórico,** en el que se enmarca el tema principal de la tesis y se introducen las tecnologías de base utilizadas, con su correspondiente remisión a las publicaciones.

En el capítulo 5, **Publicaciones**, se transcriben los tres artículos académicos que conforman esta tesis doctoral. Como los artículos están en inglés, se preceden los mismos con un breve resumen en castellano para facilitar su lectura.

A continuación, se incluye el capítulo 6 de **Conclusiones y aportaciones,** indicando de qué publicación, o publicaciones, se desprenden.

El capítulo 7 incluye **Otras aportaciones científicas** derivadas directamente de la tesis doctoral. En este capítulo se presenta una adaptación de la metodología *Scrum/Agile* a un entorno académico a distancia como es la UNED y aplicado a la dirección de proyectos de fin de carrera o grado. Se exponen los resultados obtenidos en cada ciclo de innovación, aplicando dicha metodología, y se resaltan las contribuciones más relevantes de cada estudiante en el ámbito de esta tesis.

En capítulo 8 se incluye un informe con el **Factor de impacto** y el cuartil del *Journal Citation Reports* en el que se encuentran las publicaciones presentadas.

En el capítulo 9, **Bibliografía,** se recogen todas las referencias mencionadas en la tesis, excepto las ya incluidas en los propios artículos.

El capítulo 10 incluye un **Glosario y acrónimos.**

Por último, se incluyen dos anexos sobre la metodología *Scrum/Agile* particularizada a la enseñanza a distancia. El **ANEXO A** transcribe los proyectos de fin de carrera ofertados en el curso 2015-2016 y el **ANEXO B** es un ejemplo de formulario para apuntarse al programa, como muestra del compromiso explícito de los estudiantes con cada ciclo de innovación.

1 INTRODUCCIÓN

Internet de las Cosas (IoT: "*Internet of Things*"), es un nuevo paradigma que conlleva un cambio sustancial en los usos y costumbres de nuestra sociedad, y que plantea una cuarta revolución industrial al transformar las cadenas de producción de bienes y servicios. Una definición simplista de Internet de las Cosas es conectar a Internet lo no conectado, es decir, conectar todo aquello que sea susceptible de mejorar nuestra calidad de vida. Se conforma así la web física, donde los objetos cotidianos pasan a tener su representación virtual en Internet, y gracias a nuestros dispositivos móviles, accedemos a un mundo de información que vamos descubriendo a medida que paseamos, y todo ello desde la palma de nuestra mano. Por tanto, la navegación por la web se convierte en algo físico, interactuamos con el medio y nos beneficiamos de lo mejor de ambos mundos, el físico y el virtual. Como contraprestación, los procesos productivos se harán más eficientes, no sólo por alcanzar nuevos umbrales de automatización gracias a la ubicuidad de las redes de sensores, sino más bien, por adaptarse dinámicamente a la demanda de bienes y servicios de una sociedad, que está en constante evolución por la rápida adopción de nuevas Tecnologías de la Información y Comunicación (TIC). Poder acceder en tiempo real a información relevante de los usos y costumbres de nuestra sociedad, permitirá acortar los ciclos de diseño de nuevos productos y servicios, y alcanzar así, niveles de personalización hasta hace bien poco inimaginables.

Sin embargo, estos cambios no serán abruptos. De hecho, ya se están produciendo porque IoT es una consecuencia y evolución de otros paradigmas en tecnologías de la información ya consolidados, como la computación en la nube y el almacenamiento y procesado masivo de datos ("*Cloud Computing*" y "*Big Data*" respectivamente). El rápido desarrollo de estos nuevos paradigmas

debe su origen a iniciativas colaborativas de código abierto que han transformado el sector de las TIC. Así mismo, IoT viene impulsado por una nueva comunidad de *hardware* libre que rompe el tradicional secreto industrial en el diseño de circuitos electrónicos y sistemas embarcados, democratizando el acceso a dicha tecnología.

Por último, el precio de los componentes electrónicos, su consumo de energía y su miniaturización, no para de descender. En particular, esta bajada de precios viene empujada por una penetración cada vez mayor de la electrónica de consumo y en especial de los dispositivos móviles inteligentes conocidos como "*smartphones*", que incluyen gran variedad de sensores y módulos de comunicaciones de radiofrecuencia, tanto de largo como de corto alcance.

Por tanto, a medida que nuestros usos y costumbres cambian por la incorporación de nuevas tecnologías, nos encontramos con puntos de inflexión donde es posible innovar simplemente haciendo una reingeniería de procesos, que tenga en cuenta el estado del arte de la tecnología. En este sentido, esta tesis se centra en la transformación de puntos de venta desatendidos, comúnmente conocidos como máquinas de vending. Para llevar a buen término la digitalización del sector del vending, se propone una solución integral extremo a extremo denominada OpenVend, porque hace uso exclusivo de *software* libre y *hardware* abierto. En definitiva, se busca que los usuarios se beneficien de unos mejores servicios, sin darse cuenta de que por detrás, todo está conectado.

La visión de OpenVend es que en la era de "Internet de las Cosas" los puntos de venta desatendidos y automatizados como las máquinas de vending, evolucionan para convertirse en "puntos de presencia" donde los consumidores se beneficien de una experiencia superior de compra gracias a la aplicación de los nuevos paradigmas de *Internet of Things, Cloud Computing* y *Mobile Computing.*

Para materializar dicha visión, la misión de OpenVend es proporcionar a las empresas que operan máquinas de vending, una solución tecnológica de bajo coste que les permita conectar "todas" sus máquinas a Internet. El fin último e

incentivo para los operadores de vending es mejorar su competitividad, lo que se traslada en una mejora del servicio a los usuarios finales. Se trata por tanto de conectar incluso la máquina más sencilla de vending a la red y transformarla en una máquina interactiva e inteligente, gracias al poder de la computación en la nube, y al uso de *smartphones* como principal interfaz para interactuar con la máquina de vending.

Esta visión se basa en las siguientes tendencias de mercado que en mayor o menor medida ya son una realidad:

- la adopción masiva en los teléfonos móviles de la tecnología NFC (*Near Field Communication*) permitiendo vincular máquinas y usuarios de manera sencilla facilitando los pagos y transacciones en el punto de venta.

- la comoditización y ubicuidad de las tecnologías de acceso a la Internet del futuro a costes residuales.

- el abaratamiento de los módulos IoT por su adopción masiva en la electrónica de consumo.

- la adopción de nuevos hábitos de consumo haciendo uso del comercio electrónico en entornos de movilidad.

- la regulación sobre información nutricional, trazabilidad de los alimentos y el control de su consumo en lugares como colegios y hospitales.

- la proyección de contenidos y cartelería digital en el sector del "*retail*", como nuevo canal de marketing directo y fidelización (*digital signage*).

Las capacidades que se pretende poner a disposición de las empresas del sector del vending son:

i) un Software de Gestión Integral como Servicio (SaaS), basado en iniciativas de código abierto y tecnologías IaaS (*Infrastructure as a Service*) de Cloud Computing.

ii) unos dispositivos IoT basados en plataformas modulares de *hardware* abierto, que conecten las máquinas de vending a Internet al menor coste.

iii) una implantación sencilla y amigable de la tecnología "*Near Field Communication*" (NFC) para vincular máquinas y usuarios, impulsando el pago por móvil gracias a la integración ubicua de dicha tecnología en los puntos de venta de vending.

iv) un nuevo modelo de negocio basado en la cooperación entre los distintos actores de la cadena de valor, incorporando a nuevos actores como son los operadores de telecomunicaciones y las entidades financieras.

v) la creación de algoritmos de ayuda a la toma de decisiones de negocio convirtiendo el problema de Big Data en una ventaja competitiva: programas de promoción y fidelización de usuarios basados en patrones de consumo, optimización de rutas, inventarios y *product mix*, etc.

vi) la generación de ahorro en consumos energéticos derivados de la aplicación de inteligencia ambiental en las máquinas de vending, tales como la optimización de flotas y rutas, la reducción de inventarios y su mantenimiento, etc.

Tal y como se muestra en la Fig. 1-1, OpenVend se materializa en dos líneas de producto que se complementan para ofrecer una solución integrada extremo a extremo:

- OpenVend-SaaS: Un *software* de gestión en la nube ofrecido como servicio, que modela la máquina de vending como un punto de venta de comercio electrónico. Es decir, además del control del inventario y la contabilidad, introduce una interfaz móvil para los consumidores estilo "*shopping cart*", y permite el pago con las principales pasarelas web de pagos como: Paypal, 2Checkout, Authorize.Net, WorldPay… y en concreto con Redsys: pasarela de pagos de los principales bancos

españoles que soporta las tarjetas de crédito y débito, emitidas por VISA y MasterCard.

- OpenVend-Uino: Un módulo IoT que se instala en las máquinas de vending para dialogar con OpenVend-SaaS y que es compatible con el ecosistema Arduino [1] de *hardware* abierto.

Fig. 1-1 Componentes de la solución OpenVend

En resumen, OpenVend es una solución integral de *software* y *hardware* abierto, que mejora la eficiencia operativa de las empresas de vending y permite comprar y realizar transacciones de comercio electrónico desde los móviles de los consumidores al modelar la máquina como una tienda virtual.

2 CONTEXTO

Esta tesis fue inicialmente propuesta al Departamento de Informática y Automática de la E.T.S.I. Informática de la U.N.E.D. a finales de 2011 tras una experiencia profesional en Silicon Valley en 2011 como empleado de Nokia Siemens Networks (NSN). La experiencia consistió en una estancia de tres meses, de abril a junio, dentro de un programa de aceleración de ideas llamado startup@NSN con metodología *Scrum/Agile* [2]. El programa startup@NSN se llevó a cabo en dos ciclos de innovación. El primer ciclo de innovación se realizó en 2010 sobre tecnologías *Cloud Computing*. En el segundo ciclo de innovación, participamos 40 empleados de la multinacional y se crearon 6 equipos multidisciplinares. El objetivo de dicho ciclo fue el desarrollo de prototipos mínimamente funcional sobre soluciones M2M (*Machine to Machine*). Dichas soluciones debían construirse sobre una plataforma *Cloud Computing*, resultante del ciclo anterior. De las 6 propuestas presentadas, una por cada equipo, sólo 2 de ellas tuvieron continuidad. En concreto una propuesta sobre coches conectados y otra para conectar máquinas de vending. La solución para conectar máquinas de vending fue en la que estuve involucrado, y obtuvo tres patentes americanas [3] [4] [5] de las que soy co-autor. La solución se dio a conocer con la marca VendMe [6]. VendMe fue por primera vez presentada y demostrada en el *Mobile World Congress* (MWC) de 2012 en el stand de Nokia, y días más tarde en el CEBIT (*Global Event for Digital Business*) de 2012. En ambos eventos, VendMe se presentó en colaboración con la operadora británica de telecomunicaciones Everything Everywhere. En junio de 2012, VendMe fue galardonada con un premio a la innovación en el *Global Telecom Business Innovation Awards 2012* [7]. El resultado de los dos ciclos de innovación mencionados dentro del programa startup@NSN culminó a finales de 2012 con una spin-off de Nokia Siemens Networks (NSN), constituyéndose así la empresa

Cumulocity GmbH, con sede social en Düsseldorf, Alemania y que con el tiempo se ha convertido en un referente como plataforma para Internet de las Cosas [8].

Como carta de presentación a las directoras para plantear el tema de la tesis, presenté un artículo publicado en la revista BIT del COIT (Colegio Oficial de Ingenieros de Telecomunicaciones), en concreto en el nº 187 de diciembre de 2011, que incluía un monográfico sobre Internet de las Cosas [9]. El artículo estaba escrito en clave Operador de Telecomunicaciones, como bien se intuye de su título: “Smart Vending Machines: el papel activo de las Telcos en Vending”. Al fin y al cabo, este artículo estaba firmado como empleado de Nokia Siemens Networks, que como empresa proveedora de infraestructuras de telecomunicaciones se debe a sus clientes. Si bien el análisis del mercado del vending y las motivaciones del sector y los usuarios para conectar máquinas de vending prevalecen y se mantienen como hipótesis de partida en esta disertación, desde un principio, se planteó una tesis radicalmente divergente sobre el papel y posicionamiento de los operadores de telecomunicaciones y de las instituciones financieras que respaldan los pagos por móvil.

Tal y como se irá presentando en los siguientes capítulos, se plantea que para que podamos asistir a un mundo con máquinas de vending conectadas a Internet, tenemos que desvincular a las empresas de telecomunicaciones y a los bancos para poder ofrecer una solución universal. Es decir, que tanto los usuarios del vending como las empresas que gestionan las máquinas de vending se puedan beneficiar de los nuevos paradigmas que plantea IoT y mejorar los servicios asociados al vending, sin que por ello, tengan necesidad de estar vinculados explícitamente con una entidad financiera y/o con un operador de telecomunicaciones en concreto.

La hoja de ruta y las hipótesis de partida para llegar a una solución integral para el vending comenzaron con la oferta de una serie de proyectos de fin de carrera que despertaron el interés de varios estudiantes. La experiencia fue tan exitosa, que durante cuatro cursos consecutivos se repitió el programa, y finalmente, por el programa pasaron un total de 17 estudiantes. A esta experiencia docente se dedica el capítulo 7.

Por último, quedaba pendiente poder acceder a un laboratorio con la infraestructura necesaria para poder implementar una nube privada con capacidades de *Big Data*. Con este fin, se produjo un acercamiento al sector privado y en concreto a Huawei Technologies España SL, empresa en la que trabajo desde mediados del 2012. Tales acercamientos concluyeron con la firma de la cátedra universidad-empresa en *Cloud Computing* y *Big Data* que se firmó a finales de 2015 [10].

3 HIPÓTESIS Y OBJETIVOS

Gran parte del trabajo desarrollado en esta tesis se centra en codificar prototipos y descartar tecnologías por el método de prueba error. El objetivo fundamental del proyecto ha sido construir una solución integral de bajo coste, orientada principalmente a las PYMES (Pequeñas y Medianas Empresas) que operan máquinas de vending. En general dichas empresas, al no tener la escala suficiente, no pueden hacer frente a las inversiones necesarias para implantar las soluciones de telemetría y pagos que existen actualmente en el mercado.

Como se verá en el capítulo 7, se ha seguido una metodología basada en *Scrum/Agile* para el desarrollo de una solución integral de bajo coste. En consecuencia, se ha sido muy pragmático a la hora de seleccionar las tecnologías que nos permitiesen conseguir nuestros objetivos en cada momento, partiendo siempre de la premisa de que estas tenían que seguir una filosofía de código abierto. Podríamos decir que esta tesis encaja con el término "*pervasive computing*", es decir, el uso de tecnologías ubicuas que están al alcance de cualquiera. En esta línea, las hipótesis de partida y principales motivaciones para la realización de este proyecto han sido:

- La adopción masiva de dispositivos móviles por parte de los usuarios.
- La ubicuidad de las tecnologías de acceso a Internet.
- El abaratamiento de módulos IoT.
- La adopción de nuevos hábitos de consumo en entornos de movilidad.

La tesis se ha focalizado en el vending porque desde un punto de vista académico nos proporciona un amplio abanico de casos prácticos de interés. Una máquina de vending es un punto de venta desatendido que nos permite modelar casos típicos del sector del "*retail*", por ejemplo: mejorar la experiencia

de compra de los usuarios con pagos por móvil, implantar programas de fidelización, expandir el vending hacia la cartelería digital ("*digital signage*"), etc. Además, una máquina de vending plantea problemas de ruptura de stock, o gestión de productos perecederos, que incluso pueden requerir monitorizar la cadena de frio. Una empresa de vending debe, por tanto, optimizar el inventario y la gestión de sus almacenes, y por ende, necesita gestionar las flotas de sus furgonetas de reparto optimizando las rutas de reposición. También es importante la pronta detección de fallos para ganar eficiencias en el mantenimiento del parque de máquinas. Focalizar esta tesis en el vending, nos ha permitido explorar un sinfín de casos de uso y crear una arquitectura de referencia que es fácilmente replicable a otros sectores y casos de uso como: hostelería, retail, logística, peajes, parking, ticketing, transporte público, taxis, etc.

Conectando las máquinas de vending a Internet, los puntos de venta desatendidos se pueden reconvertir en "puntos de presencia", mejorando la experiencia de compra de los usuarios, y obteniendo más información de su uso para las empresas propietarias de estas máquinas. Por tanto, dos son los objetivos principales de esta tesis:

a) Mejorar la eficiencia de las operaciones de los operadores de vending.
b) Proporcionar una mejor experiencia de usuario.

Ambos objetivos son los principales incentivos que se desprenden del barómetro de 2016 sobre IoT que Vodafone viene publicando anualmente [11].

Por tanto, el primer objetivo es: **"proporcionar a las PYMES que operan máquinas de vending soluciones de bajo coste para conectar sus máquinas a Internet con el fin último de mejorar su competitividad"**. Para cubrir este objetivo se ha trabajado en el análisis de consumo, la gestión de inventarios, la planificación dinámica de rutas y la gestión de incidencias entre otros procesos. Para ello se ha construido una plataforma en una nube privada con capacidades de *Big Data*, con el fin de poder capturar y analizar potencialmente todos los datos generados en dichos procesos.

También ha sido necesario diseñar módulos IoT basados en electrónica abierta para conectar las máquinas. Al conectarlas, las máquinas de vending se han transformado en máquinas interactivas, e inteligentes, usando la computación en la nube. Como consecuencia, se ha mejorado la experiencia de usuario en la compra, ofreciendo servicios adicionales como métodos alternativos de pago con el móvil, recarga de saldo desde pasarelas de pago o desde la propia máquina, e incluso, proporcionar mayor información de los productos con tecnologías web de realidad aumentada, programas de fidelización, etc.

Tras conectar las máquinas, el segundo objetivo de esta tesis es la mejora de la experiencia de compra: **"se debe conseguir que el proceso de compra se realice en tres pasos para poder competir en usabilidad con la compra por monedas"**. Desde la perspectiva del consumidor, la compra en una máquina de vending tradicional requiere, en el mejor de los casos, de tres interacciones:

1. Introducción del importe exacto en la máquina.
2. Selección del producto pulsando un botón en la máquina.
3. Recogida del producto.

La innovación propuesta pasa por mejorar la experiencia de compra del usuario minimizando las interacciones físicas con las máquinas de vending. Tal y como se muestra en la Fig. 3-1, se trata de utilizar el potencial de la tecnología NFC para trasladar al móvil la selección y pago del producto con un solo clic o "*gesture*". Con esta solución se asegura la entrega del producto en la máquina de vending con tres únicas interacciones:

1. Vinculación entre el móvil y la máquina.
2. Acceso a una oferta personalizada en la pantalla del móvil y métodos de comercio electrónico para realizar el pago.
3. Recogida del producto.

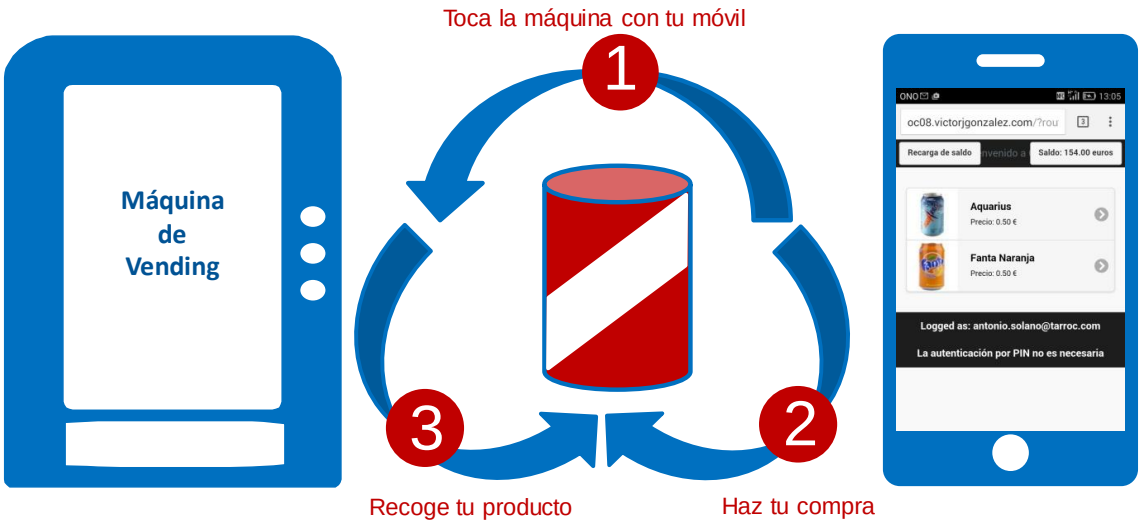


Fig. 3-1 Proceso de compra alternativo en 3 pasos

Concluimos que, si hasta ahora el principal interés en conectar las máquinas de vending viene de los operadores de vending, ya que gracias a la telemetría generan eficiencias operacionales, en un inmediato futuro, van a ser los usuarios los que demanden nuevas formas de comercio electrónico y nuevas experiencias de compra que requerirán que las máquinas estén conectadas.

Éste es justamente uno de los retos que se describe en el artículo, "*Smart vending machines in the era of Internet of things*" [12], cuya transcripción se encuentra en el capítulo 5.2.

El vending plantea un reto adicional al tratarse de un sector muy fragmentado, con multitud de pequeñas empresas que no disponen ni de las competencias técnicas, ni de la financiación necesaria para acometer la implantación de costosos sistemas de información. Desde la perspectiva de negocio existe un tercer objetivo y es que los costes de la solución no sean una barrera para su adopción, y en especial para las pequeñas empresas que operan con muy poco margen.

Por tanto, un tercer objetivo implícito y el más importante para alcanzar la adopción de la solución es: **"minimizar los costes asociados a la solución propuesta"**. Para conseguir reducir los costes al máximo la premisa de partida

es que todo el *software* y el *hardware* utilizado en esta tesis tiene que ser libre, para evitar tener que pagar licencias a terceros. Tampoco se debe depender de proveedores de servicios o de infraestructura más allá de lo mínimamente necesario, por lo que se opta por crear una nube privada. Pero dicha decisión de construir una nube privada viene condicionada por otra decisión respecto a los costes. De no ser así, cualquier proveedor de infraestructura como servicio para IoT [13] justificaría que es más económico alquilar la infraestructura que comprar los servidores y operarlos por uno mismo. En principio esa aseveración es cierta porque mantener una nube privada es complejo y costoso, pero también es cierto que a medida que los negocios escalan, la dependencia con el proveedor de servicios de infraestructura puede llegar a ser vinculante, afectando en gran parte a los futuros costes de operación. En cualquier caso, la decisión de implantar una nube privada viene de la necesidad de acceder a las APIs (*Application Programming Interface*) de bajo nivel para poder automatizar al máximo la gestión del parque de máquinas de vending, de forma que los mecanismos de provisión, configuración y mantenimiento de las máquinas sean lo más sencillos posibles. En definitiva, como hipótesis de partida se debe asumir que los costes operacionales de configurar los sistemas y mantenerlos manualmente, simplemente, no están al alcance de la mayoría de los operadores de vending, ni tampoco tienen la capacitación técnica para hacerlo por sus medios. En consecuencia, el sistema tiene que ser de bajo coste, no solo en lo que respecta al *hardware* y *software,* sino también en los servicios y recursos necesarios para su operación.

En los siguientes apartados, se detallan estos tres objetivos y se añade un nuevo objetivo relativo a la **“seguridad de la solución propuesta”**. Por último, también se incluye como un objetivo explícito de esta tesis, el **“obtener un prototipo de la solución extremo-extremo que sea mínimamente funcional y que permita validar el resto de hipótesis y objetivos presentados”**.

3.1 Micropagos con móvil, sencillo y universal

Este objetivo busca mejorar la experiencia de usuario en el momento de la compra y plantea un modelo alternativo de pagos por móvil en puntos de venta desatendidos que sea universal, es decir, que no esté vinculado a ninguna entidad financiera, tampoco a un operador de telecomunicaciones en concreto y menos aún, a ningún fabricante de dispositivos o sistemas operativos para móviles. Para ello se parte como premisa de diseño que no se requiera descargar ninguna aplicación en el móvil y se haga uso exclusivamente de tecnologías web, estando cada máquina de vending representada por una dirección web o URL (*Uniform Resource Locator*). Dicha dirección web nos conduce a una tienda *online* donde se muestran los productos disponibles en la máquina de vending, y como cualquier tienda *online*, proporciona métodos de pago electrónicos que nos permiten realizar micropagos con bajas comisiones. Una vez confirmado el pago, el sistema se comunica con la máquina de vending para entregar el producto seleccionado.

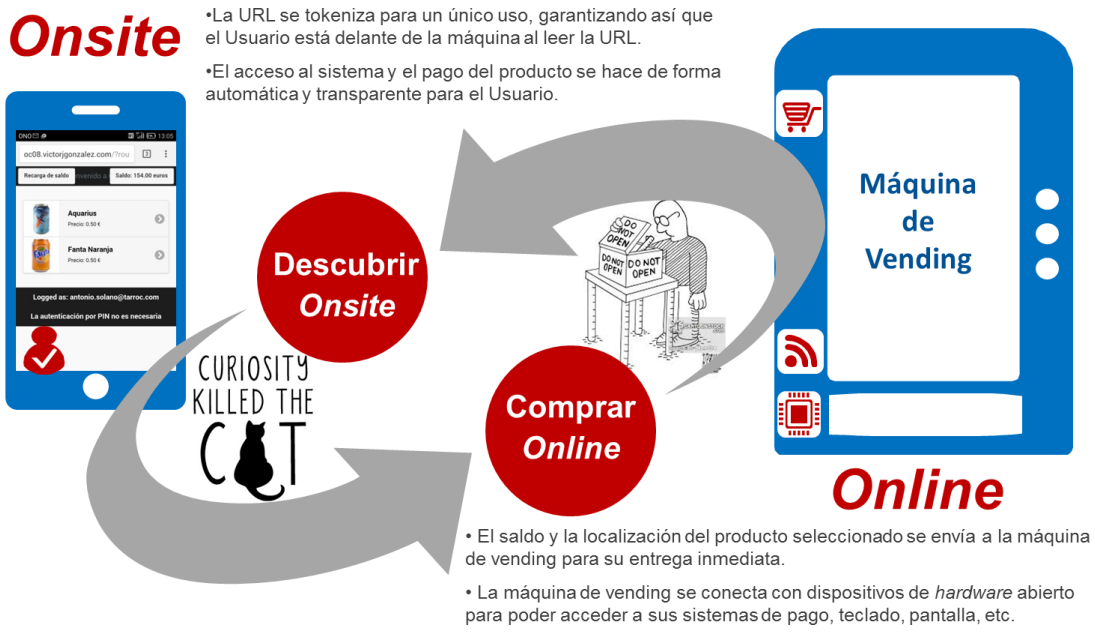


Fig. 3-2 Proceso de compra

La Fig. 3-2 muestra el proceso de compra que requiere dos interacciones del usuario con el móvil para completarlo: primero debe descubrir la URL de la

máquina, por ejemplo, haciendo uso de la tecnología NFC (*Near Field Communications*) y segundo debe seleccionar el producto. El tercer paso implícito sería la entrega del producto.

El artículo "*Smart vending machines in the era of Internet of things*" [12], cuya transcripción se encuentra en el capítulo 5.2, plantea estas premisas de partida, analiza el rol de los distintos agentes que participan en el modelo y propone una solución integral extremo a extremo.

El artículo "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [14], ver capítulo 5.4, explica distintas escenarios que se proponen para publicitar y descubrir las URLs como los mostrados en la Fig. 3-3. Dicho artículo también describe el mecanismo de login transparente multidominio referenciado en la Fig. 3-4.

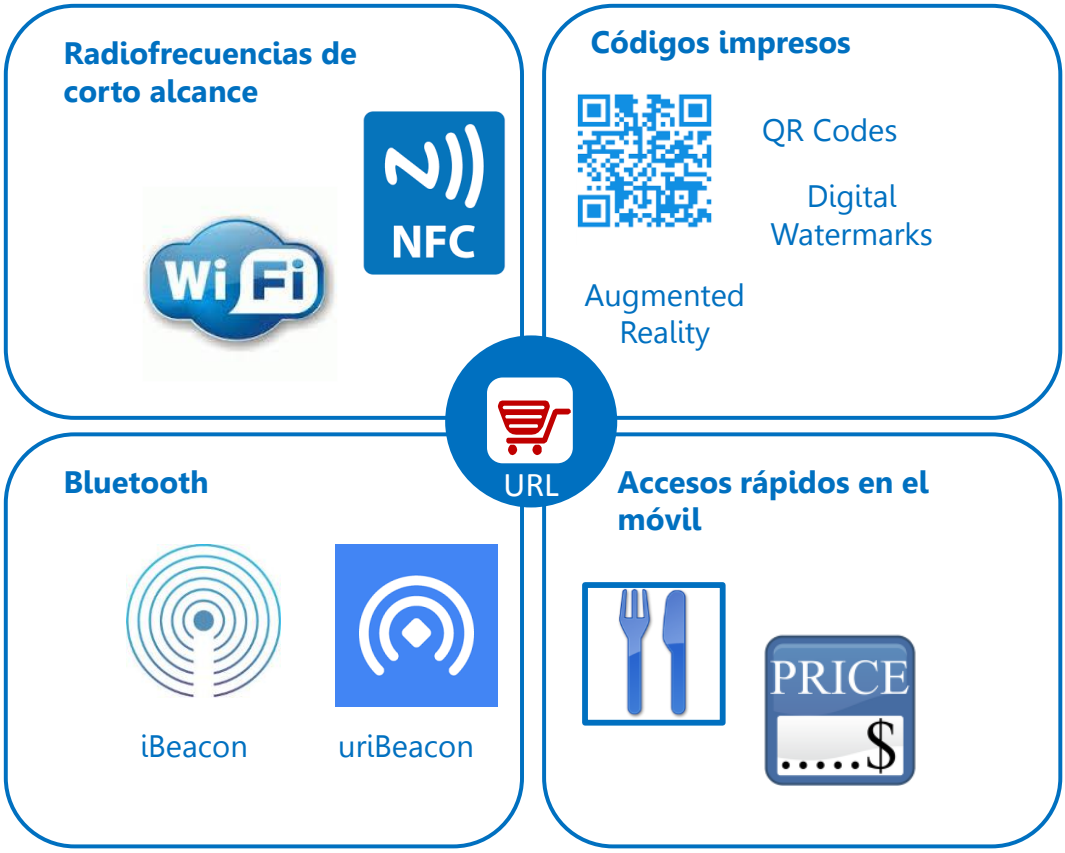


Fig. 3-3 Escenarios para publicar la dirección web de las máquinas de vending

El hecho de no tener que instalar una aplicación específica para cada operador de vending y que el sistema proponga un sistema descentralizado para gestionar las credenciales, permite expandir el diseño a multitud de sectores como los presentados en la Fig. 3-4. Una vez el usuario se ha dado de alta en el sistema, este podrá acceder a cualquier objeto que esté conectado al sistema y transformarlo así en un punto de venta.

Fig. 3-4 Sistema de login transparente multidominio

Los mecanismos descritos hasta ahora justifican la usabilidad para acceder al sistema, pero hay que recordar que es necesario proporcionar un sistema que soporte los micropagos típicos del vending, y por extensión, cualquier tipo de pago. Esto se consigue al compartir un único TPV (Terminal Punto de Venta) virtual entre todas las máquinas de vending.

Como en cualquier tienda *online*, para realizar el pago y completar el pedido del producto el usuario es redirigido a un TPV virtual, tal y como se representa en la Fig. 3-5.

Fig. 3-5 Pago online en máquinas de vending

Por extensión, esta solución de pagos podría utilizarse por ejemplo en un restaurante como se muestra en la Fig. 3-6, donde en vez de ordenar a la máquina de vending que entregase el producto, el pedido podría ir directo a cocina.

Fig. 3-6 Extensión de OpenVend a la hostelería

Esta idea se presentó como el iServilletero al concurso Intel Edison [15] en 2014 y obtuvo el primer premio [16]. En este caso se sustituyó la máquina de vending por una impresora termal que imprimía el ticket del pedido en cocina.

No obstante, pensar que un usuario va a introducir los datos de su tarjeta mientras está realizando una compra en una máquina de vending, contradice nuestro objetivo de usabilidad, ya que eternizaría el proceso de compra. Como alternativas se proponen tres opciones:

1. Integrar Redsys web services, para lo que deben salvaguardarse los datos de la tarjeta en la plataforma siguiendo una regulación muy estricta. Si bien esta solución es técnicamente viable, el problema es que los bancos no facilitan el acceso a web services si el volumen de ventas no es muy elevado, ni tampoco si la empresa que lo solicita no tiene una buena reputación. Realizar esta integración no supone ninguna innovación por lo que su prototipado se queda fuera del alcance de este proyecto. La Fig 3-7 muestra un esquema de cómo quedaría el diagrama de esta solución.

Fig. 3-7 Integración con Redsys web services

2. Implantar un sistema de prepago, que permita a los usuarios disponer de un saldo *online*. El sistema se integra con pasarelas de pago como Redsys o Paypal para realizar la recarga de saldo tal y como se muestra en la Fig. 3-8.

Fig. 3-8 Recarga de saldo en prepago por pasarela de pago

3. Recarga por monedero, disponer de un sistema de prepago plantea la posibilidad de hacer la recarga introduciendo efectivo en la propia máquina. En este escenario se elimina totalmente a la entidad financiera ya que el dinero es recaudado directamente por el operador de vending tal y como muestra la Fig. 3-9.

Para cuantificar la usabilidad en el artículo "*Smart vending machines in the era of Internet of things*" [12] (sección 5.2.6), se han incorporado métricas sobre la duración media de las transacciones. El sistema es capaz de entregar el producto en unos cinco segundos desde el momento en que el usuario ha realizado el pedido desde su móvil, lo cual representa un resultado aceptable.

Fig. 3-9 Recarga de saldo con efectivo en la máquina

3.2 Bajo coste y eficiencia operacional

Sin excepción, todo el *software* empleado es de código libre. Los tres artículos [12] [14] [17] transcritos en el capítulo 5 detallan en mayor o menor medida los componentes de *software* empleados. La introducción a los paquetes de código abierto seleccionados se detalla en el capítulo 4.

El artículo "*Smart vending machines in the era of Internet of things*" [12], plantea que el objetivo subyacente de la solución es reducir todos los costes o el TCO (*Total Cost of Ownership*). Ello se consigue al plantear un modelo de pagos alternativo que no requiere costosos lectores de tarjetas de crédito, con tecnologías propietarias y que suelen estar asociados a altas comisiones en las transacciones electrónicas. Como alternativa se plantea invertir el modelo de pagos y que no sea la máquina, sino el móvil del usuario el que inicia y termina las transacciones por medio de pasarelas de pago convencionales de comercio electrónico (TPV virtual). De esta manera, como se muestra en la Fig. 3-10 se reducen las comisiones sustancialmente permitiendo hacer frente a los micropagos típicos del vending.

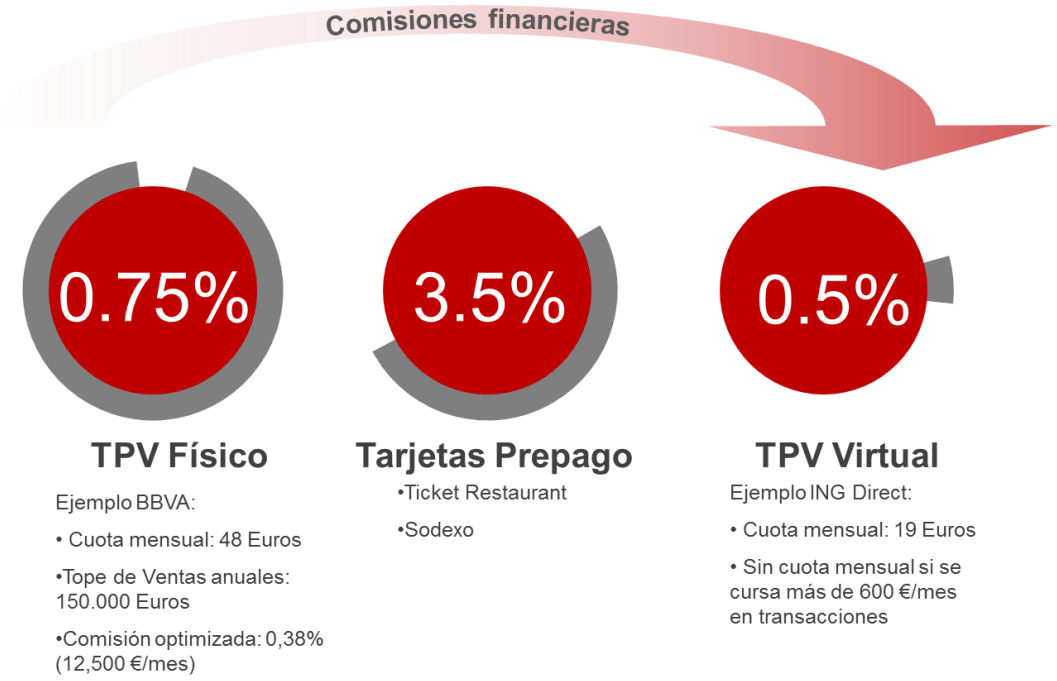


Fig. 3-10 Comisiones en sistemas de pago electrónicos

El artículo "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [14] (capítulo 5.4) muestra varias fotos de cómo se han ensamblado módulos IoT y diseñado circuitos electrónicos "*open hardware*" de bajo coste para poder implementar este modelo alternativo de pagos con dispositivos IoT (comprobar Fig. 5-17 y Fig. 5-25).

El artículo *"A self-provisioning mechanism in OpenStack for IoT devices"* [17] (capítulo 5.3) hace hincapié en que dichos dispositivos IoT sean "*plug-and-play*", es decir, auto configurables. Una vez alimentados y conectados a Internet, los módulos y circuitos que componen los dispositivos IoT, éstos permiten interactuar a los usuarios finales con el "objeto" que conectan, que en nuestro caso es un punto de venta desatendido, y así poder realizar la compra. Para ello, es necesario sincronizar los productos que contiene la máquina con los ofrecidos en su tienda virtual, y eso se hace gracias a los módulos de telemetría que se han diseñado con *open hardware*.

En definitiva, el mecanismo de *plug-and-play* tiene como principal objetivo reducir los costes operacionales y simplificar el proceso de despliegue de la solución, tal y como se describe en la Fig. 3-11.

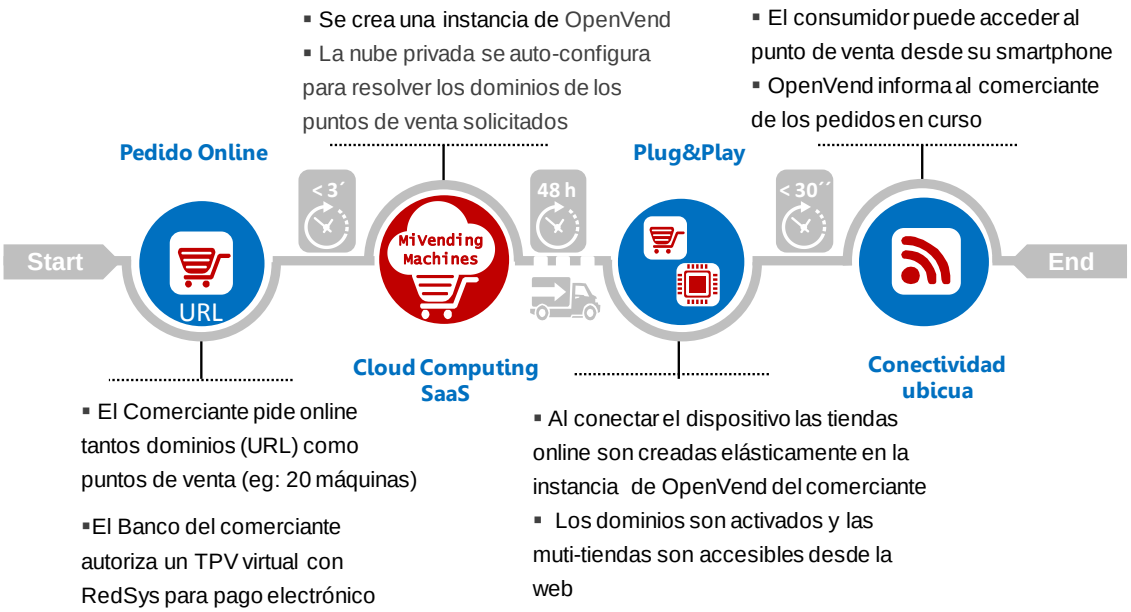


Fig. 3-11 Optimización de las operaciones y despliegue de nuevos dispositivos

3.3 Escalabilidad y flexibilidad con Computación en la Nube

Crear una arquitectura de referencia en la nube que permita escalar y hacer crecer el sistema sin limitar el número de puntos de venta desatendidos a dar servicio, y hacerlo de una manera auto configurable es el principal objetivo del artículo "*A self-provisioning mechanism in OpenStack for IoT devices"* [17], ver capítulo 5.3.

Como se muestra en la Fig. 3-12, el sistema debe poderse replicar de forma que cada operador de vending posea su propia instancia de OpenVend donde conectar todo su parque de máquinas, y por extensión, dar servicio a otros sectores.

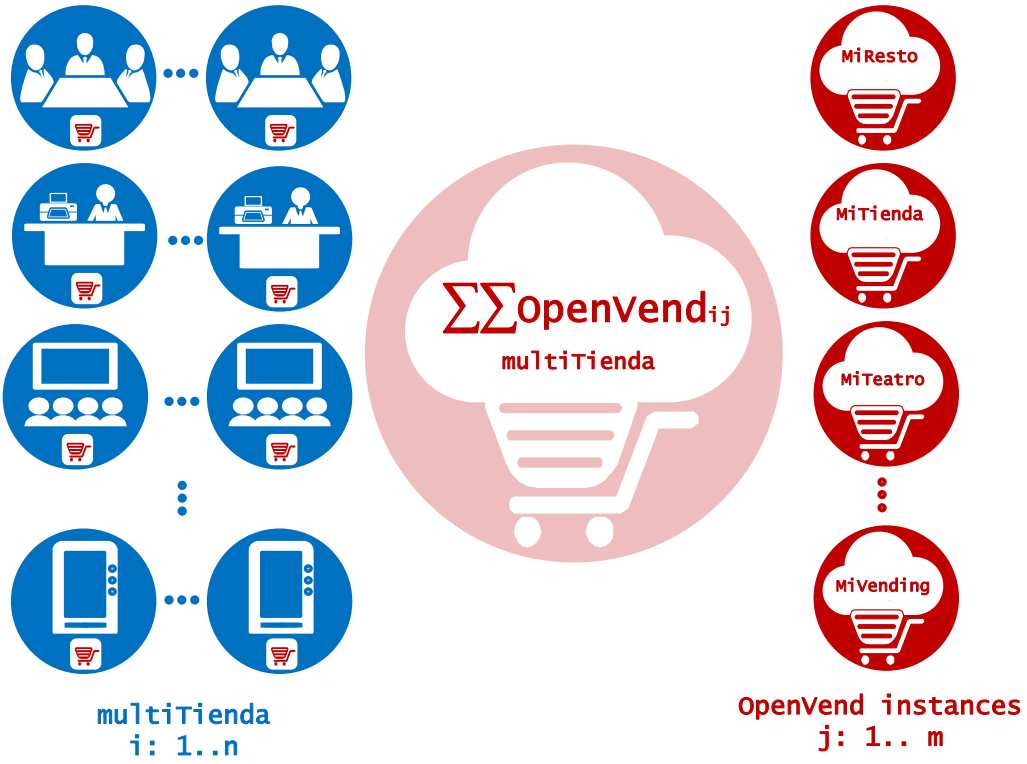


Fig. 3-12 Escalabilidad de la solución OpenVend

La solución debe ofrecerse como un servicio (SaaS), de forma que las empresas finales que comercializan los servicios, no requieran de inversiones en servidores, ni de personal con cualificación técnica en informática para operar y mantener el sistema. La Fig. 3-13 plantea modelos de negocio flexibles para distintos sectores.

Otro requerimiento de partida es diseñar una arquitectura con capacidades futuras de *Big Data*, con el fin de ofrecer analíticas avanzadas y predicciones de la demanda. Al construir la solución sobre una nube privada, las interfaces en cualquier nivel del IaaS, PaaS o SaaS deben ser accesibles, lo que permitirá interceptar todos los flujos y transacciones que pasan por el sistema para su posterior procesado en búsqueda de analíticas avanzadas.

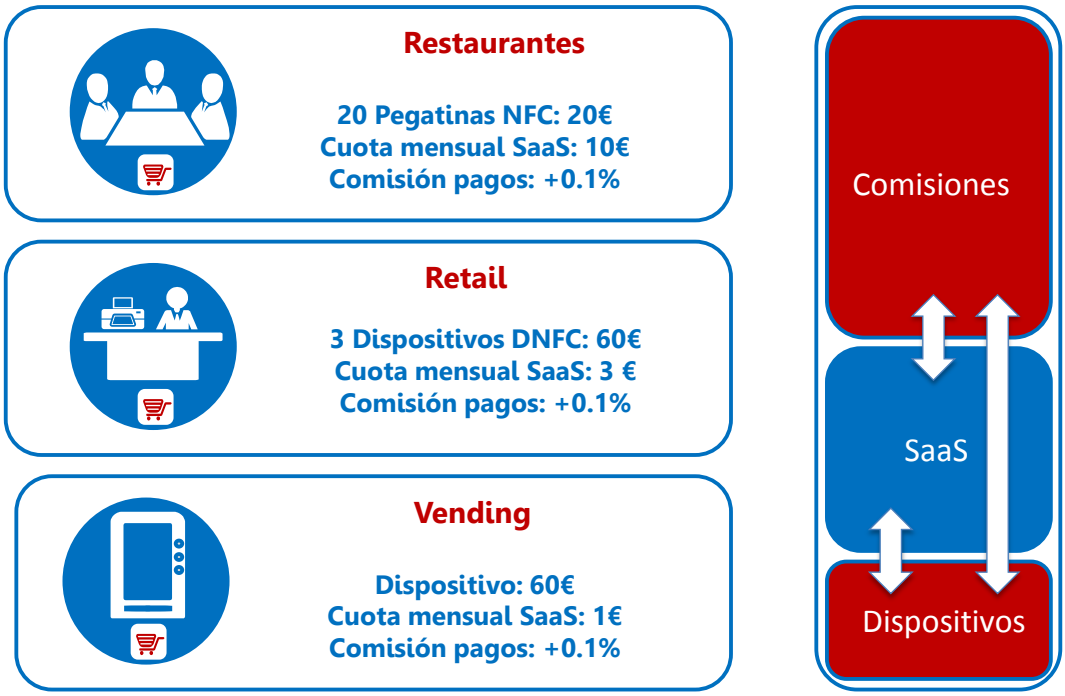


Fig. 3-13 Modelos de negocio flexibles

3.4 Seguridad

La seguridad es por diseño, es decir, la confidencialidad y privacidad de las credenciales y transacciones están garantizadas por el uso de mecanismos de encriptación de las comunicaciones y las credenciales, y por la tokenización de las transacciones y las credenciales. El artículo "*Smart vending machines in the era of Internet of things*" [12] (capítulo 5.2) incluye una sección dedicada a la seguridad donde se presentan los mecanismos implantados en cada nivel de la solución.

Por ejemplo, uno de los requerimientos de la solución es poder garantizar que el usuario se encuentra frente a la máquina de vending en el momento en que realiza la selección y pago del producto. Este requerimiento podría ser resuelto utilizando los servicios de localización de los operadores de telefonía móvil, pero supondría un sobrecoste y la dependencia de éstos, lo que entraría en conflicto con el objetivo de ofrecer el sistema universal mencionado anteriormente. El foco del artículo "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [14] (capítulo 5.4)

justamente intenta dilucidar esta problemática. Para garantizar la presencia del usuario se han desarrollado dos mecanismos diferentes pero que se complementan:

- Tokenización de las credenciales y las transacciones, tal y como se recoge en la Fig. 3-14.
- Solicitud de un PIN de confirmación como se muestra en la Fig. 3-15.

La tokenización de las credenciales forma parte del mecanismo de login transparente multidominio presentado en la Fig. 3-4.

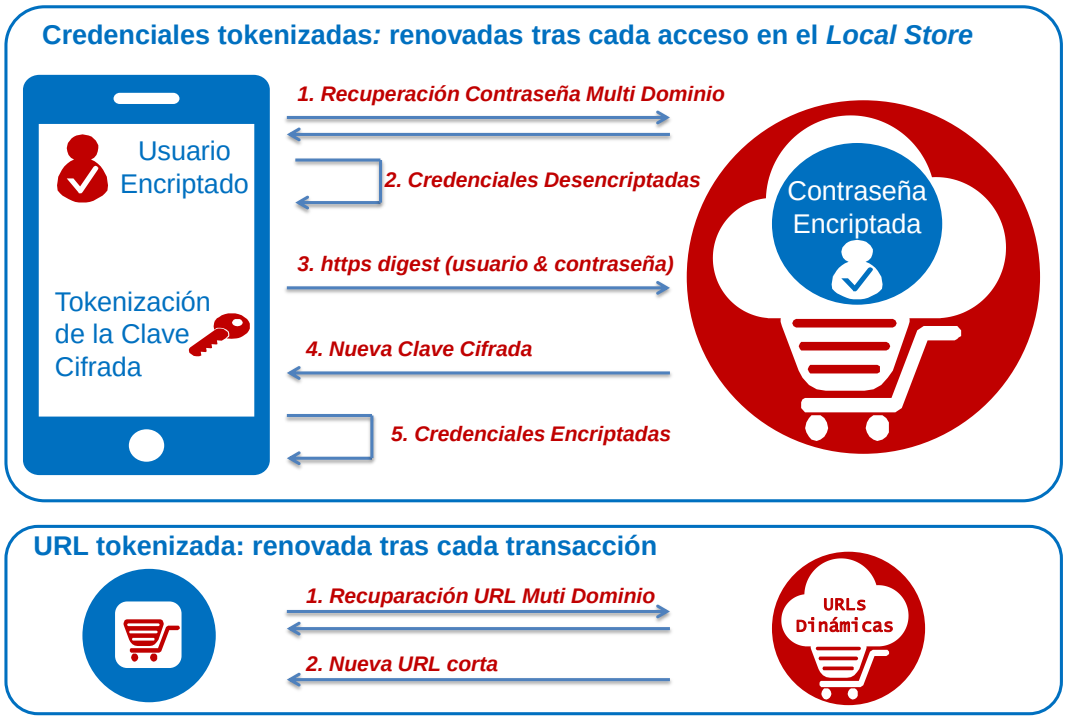


Fig. 3-14 Tokenización de las credenciales y las transacciones

El mecanismo de solicitud de PIN es complementario al de la tokenización de las transacciones, o URLs. Su utilización está contemplada para el caso de que haya un error en la detección de la presencia mediante el token asociado a la URL. Por ejemplo, porque haya habido una falta de sincronización entre la máquina de vending y el servidor haciendo que el token almacenado en la máquina no sea el mismo que piensa el servidor que tiene la máquina asignada.

En ese caso, no es deseable que por un error de comunicaciones se pueda perder una venta, ya que el usuario realmente sí que está frente a la máquina. Para evitar este escenario, se permite al usuario proseguir con el proceso de compra de la forma habitual, pero en el momento de realizar el pago, se le muestra un código de cuatro dígitos en la pantalla de la máquina de vending. Simultáneamente se le pide introducir el código mostrado en su teléfono. La coincidencia de ambos códigos, el mostrado en la pantalla de la máquina y el introducido por el usuario, garantiza que el usuario se encuentra realmente frente a la máquina.

Fig. 3-15 Autenticación con PIN en dos niveles

El PIN también se puede utilizar como un mecanismo de seguridad de dos niveles, como el usado por los bancos cuando nos envían un SMS al realizar una operación por la web. Esta configuración es deseable cuando se accede al sistema escaneando un código QR (*Quick Response code)*, o también conocido como código bidimensional. Si dichos códigos están impresos, la URL es fija y la tokenización de la URL no es viable.

3.5 Prototipo mínimamente funcional

Un objetivo implícito de esta tesis es validar todas las premisas de partida y los mecanismos propuestos con prototipos mínimamente funcionales. A nivel de dispositivo IoT, se han publicado pruebas empíricas que demuestran que se ha conseguido compilar en un simple controlador de 8 bits, mono tarea y con una capacidad máxima de 256 KBytes, toda la lógica expuesta en esta disertación. En los artículos "*Smart vending machines in the era of Internet of things*" [12] y "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [14] (capítulos 5.2 y 5.4 respectivamente) se presentan los resultados determinando las mejoras que el/los mecanismos propuestos puedan ofrecer frente a sistemas ya existentes. Se trata de pruebas de carga de 300 ciclos de venta consecutivos, donde se observa que el código del dispositivo es robusto. La memoria flotante del dispositivo se mantiene estable, evitando así desbordamientos que reiniciarían el sistema. También se demuestra que los tiempos de respuesta para completar un ciclo de compra son aceptables. En ambos artículos también se invita al lector a visitar una demo *online* [18] para darse de alta en el sistema, y experimentar por sí mismos cómo se accede a varias máquinas sin necesidad de introducir las credenciales. La demo usa QRs para codificar URL estáticas como las mostradas en la Fig. 3-16, por lo que se requiere una autenticación de dos niveles por PIN. La demo también incluye un QR dinámico que se genera cada vez que se accede, o refresca la página de la demo *online* [18]. Si se lee dicho QR desde el móvil con una aplicación estándar para leer QRs, se puede experimentar como el proceso de compra valida el token de la URL y no es necesario pasar por el paso del PIN.

En el anexo A del artículo "*A self-provisioning mechanism in OpenStack for IoT devices"* [17] (ver sección 5.3.7) se muestra el código fuente a cargar en el dispositivo IoT. Dicho código desencadena la creación de una máquina virtual en la nube privada, pero solamente la primera vez que el dispositivo es alimentado y conectado a la red.

Fig. 3-16 Demo con URLs no dinámicas

4 MARCO TEÓRICO

4.1 Introducción

Para superar los retos que plantea el vending, esta tesis propone una reingeniería de los procesos de compra y operaciones asociados al vending aplicando nuevos paradigmas como: *Internet of Things*, *Cloud Computing* y *Mobile Computing*.

La Fig. 4-1 presenta dichos paradigmas como los pilares fundamentales de la solución propuesta: OpenVend.

Fig. 4-1 Pilares tecnológicos de OpenVend

En síntesis, estos tres pilares tecnológicos han permitido diseñar:

- Una solución intuitiva diseñada para pagos móviles.
- Una infraestructura, dispositivos y servicios de operación de bajo coste.
- Una solución segura y robusta que garantice la privacidad y confidencialidad de los datos.

De hecho, en la última década, existe una correlación entre el interés que despiertan tecnologías como *Cloud Computing*, *Big Data,* e IoT, tal y como muestra la Fig. 4-2.

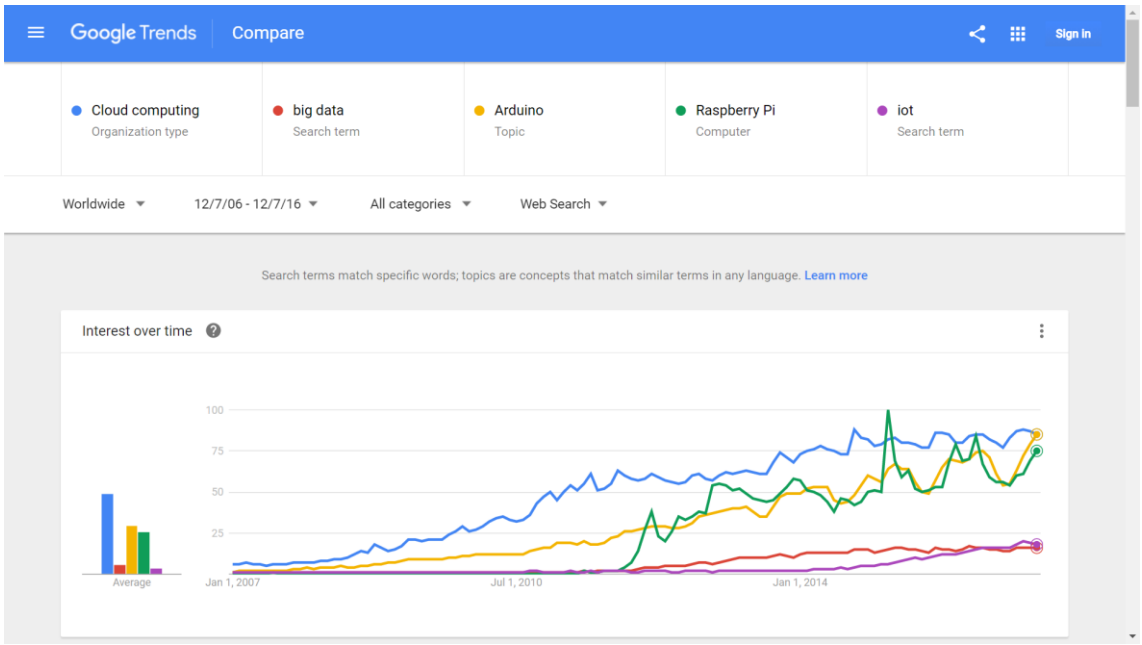


Fig. 4-2 Tendencias de búsqueda en Google

Es interesante ver como IoT supera ya a *Big Data* según Google Trends [19]. Más sorprendente aún, es constatar lo activas que son las comunidades de *open hardware* de Arduino [1] y Raspberry-Pi [20]. Arduino inició sus andaduras en 2008 mientras que Raspberry-Pi lo hizo en 2012, sin embargo, Raspberry-Pi alcanzó rápidamente a Arduino. A la vista de los gráficos de Google Trends, se hace patente que las búsquedas relacionadas con Arduino y Raspberry-Pi están muy correlacionadas, por lo que se podría concluir que prácticamente existe una

única comunidad de código abierto interesada tanto en microcontroladores como en miniordenadores para sistemas embarcados.

Se puede decir que IoT no puede entenderse sin la “revolución silenciosa” que el *hardware* abierto y la comunidad de *makers* está provocando en la industria. Por hacer una analogía, Arduino es para el *hardware* lo que a principio de los noventa *Linux* fue para el *software* [21] . Si hace algo más de dos décadas *Linux* no hubiese irrumpido como sistema operativo de código abierto, hoy no podríamos hablar de ecosistemas de código abierto como Openstack [22] o Hadoop [23], que han democratizado las tecnologías *Cloud* y *Big Data*.

Volviendo a los retos de esta tesis, en el artículo “*Smart vending machines in the era of Internet of things*” [12], se plantea que la principal barrera para conectar las máquinas de vending de forma masiva a Internet son los costes. En consecuencia, tal y como se ha mencionado anteriormente, se ha optado por reducir los costes apostando por tecnologías de código abierto y estándares ampliamente aceptados por la industria, como los mostrados en la Fig. 4-3.

Fig. 4-3 Estándares y paquetes de código abierto en OpenVend

Este capítulo se va a estructurar en torno a los tres paradigmas identificados en la Fig. 4-1: *Internet of Things*, *Cloud Computing* y *Mobile Computing*, y dentro de cada uno de ellos, se presentarán los paquetes de código abierto que han sido utilizados para construir OpenVend. No obstante, se remite al lector a la lectura de las publicaciones del capítulo 5, donde se detallan los bloques funcionales de la arquitectura y los flujos de información que han permitido alcanzar los objetivos propuestos.

4.2 Internet of Things

Internet de las cosas conecta el mundo físico con el mundo virtual de Internet, con el fin último de mejorar nuestra calidad de vida [24]. Como una máquina de vending no deja de ser más que una pequeña tienda desatendida, por analogía, se ha seleccionado un paquete de código abierto como OpenCart [25], para crear tiendas *online* que representen a máquinas de vending. OpenCart es un paquete para realizar comercio electrónico con muchas funcionalidades incluidas de base, como la integración con pasarelas de pago, o la fidelización de clientes. Como consecuencia, con esta integración entre el mundo físico y el mundo virtual, se ha conseguido dotar de más inteligencia a las máquinas de vending, simplemente al agregar las funcionalidades de las tiendas *online* a las propias máquinas. Como resultado, se ha mejorado la experiencia de compra del usuario.

En la Fig. 5-2 del artículo "*Smart vending machines in the era of Internet of things*" [11] (capítulo 5.2) a las instancias de OpenCart se las denomina "Ghost Vending Machines". Para conectar y poder interactuar con las máquinas de vending, se han ensamblado distintas tarjetas electrónicas de *hardware* abierto compatibles con el ecosistema de Arduino, tal y como se muestra en la Fig. 5-4 del mencionado artículo. Arduino se comunica directamente con el servidor de OpenCart y el usuario, a su vez, se comunica también con el servidor de OpenCart. Como resultado, las órdenes recibidas en OpenCart son trasmitidas a la máquina de vending a través de Arduino.

En la Fig. 5-18 del artículo "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [12] (capítulo 5.4) se muestra el flujo de las comunicaciones y se detallan chipsets de comunicaciones susceptibles de ser utilizados para conectar las máquinas. En la Fig. 5-17 del mismo artículo, se muestra una implementación práctica compuesta por diferentes módulos de *hardware* abierto compatibles con Arduino, y más adelante, en la Fig. 5-25 se muestra una placa diseñada para tal fin y 100% compatible con Arduino.

Pero IoT no se reduce a diseñar dispositivos para conectar objetos o desplegar redes de sensores. IoT tiene también mucho que ver con la gestión de los datos y con *Big Data*. No obstante, existe una tendencia a crear plataformas que almacenan y exponen los datos capturados por dispositivos IoT, para sobre dichos datos, crear lógica de negocio que no siempre aporta valor, más allá de crear simples reglas de tipo "*IF-THEN-ELSE*". Dichas plataformas implementan sus propias APIs y las hacen públicas, pero en la práctica añaden un nivel más de integración y complejidad a la hora de construir una solución integral extremo a extremo. En contraposición con esta tendencia, se ha dedicado una publicación entera para plantear una plataforma que permita modelar complejos sistemas, maquinaria o ecosistemas, de forma que al conectar a Internet un dispositivo IoT, el propio dispositivo instancie todo un paquete de *software* que incluya un sinfín de funcionalidades, y acto seguido sea accesible por los usuarios. Este concepto de *plug-and-play* se recoge el artículo "*A self-provisioning mechanism in OpenStack for IoT devices"* [17], ver capítulo 5.3.

4.3 Cloud Computing

Como se ha introducido en la sección 3.2, la necesidad de ahorrar costes nos ha llevado a la búsqueda de eficiencias operacionales y a plantear la creación de una nube privada para automatizar procesos en entornos IoT [26] [27].Para crear una infraestructura como servicio se ha elegido OpenStack [22]. Sobre dicha infraestructura se han creado servidores virtuales bajo demanda para instalar las instancias de OpenCart que representan en el mundo virtual a las máquinas de vending. Como se ha comentado, Opencart es una solución de tienda *online* para realizar comercio electrónico y para su funcionamiento necesita un servidor web como Apache [28], una base de datos relacional mySQL [29] y soporte para PHP (*Hypertext Preprocessor),* que no es más que un lenguaje de programación interpretado que se ejecuta en el servidor y que nos permite generar páginas web dinámicas. Obviamente si OpenCart es código abierto, todo el código de base sobre el que está montado también es código abierto.

Sobre la infraestructura de la nube privada, también se ha creado una plataforma siguiendo un modelo de interfaces Web basadas en REST [30] (*REpresentational State Transfer*), comúnmente referidas como *RESTful Web services.* Para tal fin, dicha plataforma incorpora un paquete de código libre llamado SLIM [31]. SLIM es el módulo de la plataforma responsable de enrutar todas las comunicaciones entre las máquinas de vending y sus respectivos servidores de OpenCart.

Si bien en todas las publicaciones [12] [14] [17] de esta tesis por compendio, se han incluido diagramas de bloques de la arquitectura Cloud desarrollada, el artículo "*A self-provisioning mechanism in OpenStack for IoT devices"* [17] (capítulo 5.3), es el artículo que mejor detalla cómo se ha diseñado la plataforma. Por otro lado, para abordar problemas de *Big Data* necesitamos una arquitectura en la nube que nos permita reutilizar los recursos de la nube cuando están ociosos para procesar los datos. El artículo "*A self-provisioning mechanism in OpenStack for IoT devices"* [17], (capítulo 5.3) termina con una referencia a uno de los proyectos de *Big Data* que se han desarrollado en el marco de este trabajo y que será comentado en el capítulo 7, como el resto de proyectos de fin de carrera asociados a esta tesis. Si bien el apartado anterior terminaba con una crítica a una tendencia en crear plataformas de IoT que capturan y exponen los datos de los sensores de los dispositivos IoT, eso no quiere decir que en esta disertación no se considere el *Big Data* como algo intrínseco a IoT. El diseño de OpenVend sí incorpora capacidades de *Big Data*. La diferencia con otras plataformas, es que el foco está en interceptar y logar todos los flujos entre los diferentes módulos, y no tanto en almacenar el dato en un intento de crear modelos y agentes para cada vertical que da servicio la plataforma. La filosofía adoptada es que el dato fluye entre el dispositivo y la capa final del aplicativo, y la plataforma incluye todo un ecosistema de Hadoop [32] para capturar y procesar los datos en crudo, con el fin de obtener y mostrar cuadro de mandos y analíticas avanzadas [33] usando "R" como principal lenguaje de programación.

4.4 Mobile computing

Para interactuar con las máquinas de vending se ha optado por hacerlo a través de la web creando páginas HTML5 [34] (*Hypertext Markup Language version 5*) que se adaptan a los tamaños de las pantallas de los dispositivos móviles (*responsive web design*). Utilizar webapps basadas en HTML5 y no aplicaciones nativas, o híbridas [35], ha sido una decisión de diseño basada en el menor coste de desarrollo y mantenimiento frente a las aplicaciones nativas. También se ha tenido en cuenta el acceso universal de las webapps, ya que funcionan sobre cualquier navegador web, y son independientes del sistema operativo del dispositivo móvil, es decir, no se tienen que descargar de ningún portal de aplicaciones como el "App Store" de Apple o Google "Play". Como contrapartida, las webapps no acceden a todos los recursos del móvil [36] como pueden ser sus sensores, o sus módulos de comunicaciones (WiFi, Bluetooth, NFC …), y los tiempos de respuesta de las interfaces dependen de la calidad de las comunicaciones. Aun así, para los propósitos marcados de bajo coste, universalidad y sencillez, HTML5 ha sido nuestro aliado en nuestro desarrollo.

Así, a cada máquina de vending se le ha asociado una URL (*Uniform Resource Locator*) apuntando a un servidor de OpenCart y se han empleado mecanismos introducidos en HTML5 como *LocalStorage* [37] y CORS [38] (*Cross Origin Resource Sharing*) para implementar un mecanismo innovador de acceso (*login*) multidominio y transparente para los usuarios. Una vez que el usuario se da de alta con un email, este puede entrar directamente en cualquier tienda *online* asociada a cada máquina de vending, sin necesidad de volver a introducir sus credenciales. Este mecanismo busca que el usuario perciba la webapp, como si fuese una aplicación nativa, o híbrida, con la ventaja que no es necesario descargarse ningún *software* adicional. El inconveniente es que las URLs son públicas, por tanto, cualquiera puede acceder y existe un riesgo de que un usuario curse una orden sin estar delante de la máquina para recoger el producto. Para superar este inconveniente, ha sido necesario desarrollar un mecanismo que garantice que estamos delante de la máquina para poder entregar el producto con garantías de que es entregado al usuario que ha cursado el pedido. La solución propuesta pasa por utilizar tecnologías de

comunicaciones de corto alcance de los móviles como WiFi, Bluetooth, y en concreto NFC para anunciar las URLs de las máquinas de vending. En este punto es cuando los dispositivos IoT incorporados a la máquina de vending dialogan con nuestro dispositivo móvil y anuncian la URL. Para asegurarnos que la URL es utilizada solamente cuando se produce dicho diálogo y no son reutilizadas, a las URLs se les añada un token de un solo uso. Estos mecanismos de tokenización de acceso, tanto a las URLs como a las credenciales de los usuarios, son la principal contribución del artículo "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [14], capítulo 5.4. Dichos mecanismos ofrecen una experiencia de compra inusitada e innovadora desde un teléfono móvil.

5 PUBLICACIONES

En este capítulo se trascriben los tres artículos [12] [17] [14] que conforman esta tesis por compendio de publicaciones.

5.1 Resúmenes en castellano

Como los artículos han sido escritos en inglés, se incluye a modo de introducción, un resumen de los mismos en castellano.

5.1.1 Máquinas de Vending inteligentes en la era de Internet de las cosas

El primer artículo [12] presenta una solución integral para el vending aplicando los nuevos paradigmas de Internet de las cosas. Se introduce un método alternativo de pagos con móvil para máquinas de vending. La idea básicamente consiste en representar virtualmente la máquina de vending en Internet, y ser capaz de realizar pedidos desde un móvil sin necesidad de interactuar físicamente con la máquina, más allá de recoger el producto. Se garantiza que el usuario está delante de la máquina en el momento de entregar el producto. La solución se construye con paquetes de *software* y *hardware* abierto, y toma como hipótesis de partida que la conectividad a Internet es ubicua. También restringe el diseño al uso de tecnologías pervasivas, es decir, tecnologías de uso diario y al alcance de todo el mundo. Reducir el coste de los dispositivos, de su mantenimiento y de su operación, es el objetivo del artículo para que la solución sea adoptada por los operadores de vending. No obstante, se pretende mejorar la experiencia de compra del usuario para que sea la

demanda de servicios avanzados lo que impulse la creación de “la Internet de las máquinas de vending”.

El artículo plantea cómo realizar micropagos sin necesidad de instalar lectores de tarjetas en la máquina de vending, ya que su coste representa una barrera de entrada en un sector muy fragmentado y que opera con muy poco margen comercial. En la sección 2 del artículo, se presenta el escenario y se hace hincapié en la usabilidad para que la solución propuesta sea muy intuitiva y no sea rechazada por los consumidores. La sección 3 muestra el rol de los distintos actores para justificar que la solución es universal. Si bien es necesario un operador de telecomunicaciones que provea la conectividad y una entidad financiera que proporcione un TPV virtual, el operador de vending es libre de elegir a sus proveedores y negociar los precios. La seguridad es muy importante ya que el sistema cursa transacciones electrónicas. Por ello, en la sección 4, se introducen los mecanismos de seguridad que se han implementado en todos y cada uno de los niveles de la solución, desde la capa física a la capa de la aplicación. Para reducir los costes de operación, se plantea comercializar la solución como un servicio, y de nuevo, para no depender de un proveedor de infraestructura, en la sección 5 se describe una arquitectura de nube privada construida con OpenStack. En la sección 6, siguiendo la filosofía de código abierto, se presenta el diseño de los dispositivos electrónicos IoT compatibles con Arduino, y se muestra como resultado el tiempo medio para realizar una compra desde el móvil. Por último, en las conclusiones se contrastan las hipótesis de partida con los resultados obtenidos del prototipo y se plantea como siguiente paso, hacer una prueba comercial para su validación final con usuarios reales.

5.1.2 Un mecanismo de auto-provisión en Openstack para dispositivos IoT

El segundo artículo [17] profundiza en el diseño y la operación de la arquitectura *Cloud* presentada en el primer artículo [12].

El objetivo de este trabajo es introducir un mecanismo de "*plug-and-play*" que, al conectar un dispositivo de Internet de Cosas (IoT), instancie una aplicación de *Software as a Service* (SaaS) en una nube privada, creada con OpenStack. La aplicación SaaS es el avatar digital de un objeto físico conectado a Internet. Como prueba de concepto, se muestra como conectar una máquina de vending a Internet con un dispositivo compatible con el ecosistema Arduino de *hardware* abierto. Una vez completado el mecanismo de auto configuración y provisión, es posible comprar un producto en la máquina de vending desde un móvil.

Este artículo está organizado de la siguiente forma: la sección 1 es una introducción y contextualización de los objetivos ya mencionados. La sección 2 presenta brevemente los modelos de servicios en distintos niveles que ofrece *Cloud Computing*: infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y *software* como servicio (SaaS). La arquitectura y los componentes de OpenStack, el sistema operativo Cloud de código abierto, se describen en la sección 3. Cómo se construye la nube privada y el escenario de creación de máquinas virtuales por el propuesto mecanismo de "*plug-and-play*" se explican en la sección 4. En la sección 5 se detallan los pasos y los algoritmos desarrollados para poder crear automáticamente la representación de una máquina de vending y estar en disposición de acceder a su URL para realizar compras desde nuestro dispositivo móvil. Finalmente, se presentan las conclusiones en la sección 6 y se deja la puerta abierta para ampliar la arquitectura de referencia planteada con capacidades para *Big Data*.

5.1.3 URL de un solo uso: un mecanismo de seguridad por proximidad

El tercer artículo [14] puede también considerarse una extensión del primero [12]. En este caso se detallan los mecanismos de seguridad para salvaguardar las credenciales de los usuarios y asegurar su presencia en el momento de realizar una compra en un punto de venta desatendido.

El objetivo de este trabajo es determinar la proximidad física de los objetos conectados a Internet, cuando se interactúa con ellos desde la web a través de un móvil. Los vínculos entre los objetos conectados y los móviles se crean temporalmente mediante URL dinámicas, que pueden descubrirse fácilmente con las tecnologías inalámbricas de corto alcance disponibles en la mayoría de los teléfonos móviles: WiFi, Bluetooth y NFC principalmente. Además, se presenta un mecanismo de login transparente multidominio, que permite que los usuarios interactúen con los objetos conectados que le rodean desde sus móviles. Los mecanismos propuestos están basados en tecnologías web, por tanto, podemos referirnos a ellos dentro del ámbito de la Web de las Cosas (WoT).

Una vez introducidos estos conceptos en la sección 1, el artículo se organiza de la siguiente manera. La sección 2 presenta la motivación del trabajo presentado, que no es otra que la de proporcionar un método alternativo de pagos. Además, incluye una foto del prototipo ensamblado con *hardware* abierto compatible con Arduino. En la sección 3 se presentan varios casos de uso basados en tecnologías de radiofrecuencia de corto alcance disponible en los móviles y, para cada tecnología, se sugieren dispositivos electrónicos para publicar las URL dinámicas. La sección 4 describe el ciclo de vida de los tokens utilizados para generar las URL dinámicas. En la sección 5, se presenta la arquitectura propuesta basada en *Cloud Computing* y paquetes de código abierto. La sección 6 se dedica a analizar en profundidad las cuestiones de seguridad. La sección 7 explica todos los mecanismos presentados con un escenario que detalla el flujo de llamadas entre todos los bloques funcionales de la arquitectura previamente presentada. La sección 8 muestra una placa hecha a medida y compatible con Arduino, y proporciona algunos parámetros de rendimiento como los presentados en el primer artículo [12]. Finalmente, la sección 9 resume las principales contribuciones desarrolladas en este trabajo y expone como caso de éxito “WeChat Payments”. “WeChat” es la réplica China a “Whatsapp” y hace uso de QRs para anunciar los puntos de venta, que una vez leídos desde la propia aplicación, facilitan la realización de transacciones electrónicas.

5.2 Smart vending machines in the era of Internet of Things

Abstract: The aim of this paper is to propose a real-world deployment in building an Internet of Things (IoT) system for vending machines. We also introduce a new approach for mobile proximity payment for unattended point of sales. The basic idea is to have a digital representation of a vending machine on Internet and be able to order products from a smartphone in a fully contactless way, i.e. without interacting with the vending machine. Our approach guarantees that when the transaction occurs and the products are dispensed the consumer is physically close to the vending machine. Open innovation, ubiquitous connectivity and pervasive technologies are key aspects taken into consideration to build up a cost affordable solution. The ultimate goal is to minimize the Total Cost of Ownership (TCO) for vending operators while enhancing the consumer purchasing experience, driving up the demand for mass adoption of the “Internet of vending machines”.

Index Terms: mobile communication systems, payment schemes, pervasive computing, ubiquitous computing, authentication, web-based services.

5.2.1 Introduction

Connecting vending machines is nothing new. In fact, it is the typical use case for machine to machine (M2M) communications that has been around for the last decades. Traditionally, telemetry and online cashless payments have been the main justifications for connecting vending machines. Telemetry makes more efficient the daily operations of refilling the machine while cashless payments increases consumer convenience and boost sales. To overcome the major interoperability hurdles, the Direct EXchange Uniform Communications Standard (DEX/UCS) [1] was published in 1995 for telemetry, and few years later, the Mutlidrop Bus Standard (MDB) [2] facilitated cashless payments.

Despite all the benefits telemetry offers to vending operators, its penetration rate remains relatively low in developed economies like Europe and USA in contrast with what is observed in the Japanese market [3].

Vending is a very fragmented and competitive market with thousands of small and medium enterprises (SMEs) per country and just a few dozen of multinational companies. Indeed, margins in vending are so low that any investment in advanced technologies bringing operational cost savings requires scale to justify the ROI (Return on Investment). Therefore, at current technology costs, the majority of small vending operators are naturally hesitant to invest in new technologies.

This paper aims at presenting a cost affordable communications solution based on Open Innovation and available free Web services and technologies. In short, vending machines will not only be connected to Internet to gain operational efficiencies [4,5]. Vending machines will also have their own digital representation on the web for human interaction, enlarging our social network of Internet of Things [6] into what we define as “Internet of vending machines”.

The ultimate goal of our approach is to reduce the Total Cost of Ownership (TCO) in operating the vending machine business and, as a consequence, attract a larger number of vending operators that could apply this technology while, at the same time, enhance the consumer purchasing experience. All the above would in turn accelerate the adoption of “Internet of vending machines”, driving the costs even lower.

To these ends, the paper is organized as follows. Section 2 presents the end-user scenario. In Section 3, an alternative model for proximity payments is presented. Section 4 is devoted to analyze the security issues. The proposed architecture based on cloud computing and Open Source software is presented in Section 5. Section 6 summarizes the specifications of our Open Hardware prototype. Finally, Section 7 summarizes the major messages developed in this paper.

5.2.2 User Centric Scenario, Technology Comes Second

The aim of this section is to define a new user-friendly scenario for mobile proximity payments. Ease of use, especially in the retail sector, is key for mass adoption of pervasive technologies.

5.2.2.1 Cash payments, the reference model

Making a purchase in a vending machine requires just three interactions: 1) insert the money, 2) push the button to select the desired product and 3) pick it up as shown in Fig. 5-1a. A basic assumption on this paper is that any technology requiring more user interactions than these simple three steps will be hardly adopted in vending.

5.2.2.2 Card Cashless payments, a financial cost problem model

Cashless payments based on smart cards also allow customers to make a purchase with three simple steps as despite in Fig. 5-1a. First we present the card to the reader of the vending machine and once the credit is granted we select the product and the machine will deliver it.

For close payments based on prepaid smartcards, issuing and maintaining prepaid cards represent an additional operational cost for the vending company. From the consumer perspective, it requires additional steps: first to acquire one of those cards and, second to keep a positive credit on it. This therefore reduces the number of potential customers adopting this system.

For open payments based on credit cards, agreements with financial institutions are required. Transaction fees, including communications fees are still too high for the low micropayments volume of vending sector.

If we add to this the non-negligible cost of the card readers, the total incurred cost associated to cashless payment is a barrier for the vending sector.

As a consequence, vending machines accepting credit cards are only found in frequented locations like shopping centers and airports, where high sales per machine may compensate the incurred costs.

5.2.2.3 Mobile payments, an unresolved problem with many actors

Mobile payments have also been around the vending business for many years. Short Message Service (SMS) based transactional payments [7] did not reach the mass market. Rather than the implicit cost of the connectivity, the failure of such systems was the unfriendly purchasing experience: too complex, far beyond of the mentioned three simple steps consumers are used to.

Recent trends in Mobile Commerce leverage Near Field Communication (NFC) technology [8]. Current approaches are designed as a natural evolution of contactless smartcards. They use cryptographic protocols and rely on a Secure Element managed by a mobile wallet in the smartphone.

As shown in Fig. 5-1a, it is technically possible to provide vending machines with advanced NFC mobile payment systems enabling the use of “mobile wallets”.

How it works:

1. Customer taps or holds their NFC-enabled phone in close proximity to the contactless reader in the vending machine.
2. Customer selects the desired product in the vending machine using the keypad.
3. Customer collects the purchased item.

NFC tap-to-pay technology requires smartphones equipped with NFC chipsets and vending machines equipped with NFC readers that access the information within the smartphones. Nevertheless, there is no wide consensus in the industry on the implementation approaches in NFC and different mobile vendors may apply different techniques. Also, the mobile wallet application requires the settlement of specific agreements between the vendor or telco

carriers and financial services companies. All the above complexity burdens the creation of innovative and low cost applications for the vending machine business. Indeed, the number of players and corresponding transaction overheads, plus the complexity of the implementation make hardly profitable the business case for vending operators.

5.2.2.4 The challenge, lowering the cost barrier

In the previous section we recognized the potential of NFC technology for mobile payments because payments become as easy and convenient as using current credit card systems. Financial institutions are paving the way renewing the Point of Sales (PoS) readers and issuing NFC credit cards. As a result, consumers are starting to be aware of NFC technologies.

Our approach leverages the NFC technology mirroring the concept of Smart Posters using NFC. The NFC tag is written with an Uniform Resource Locator (URL) pointing to the digital representation of the vending machine on the web.

With a NFC-ready smartphone, the user taps onto the NFC tag located at the front end of the vending machine and the communication between the smartphone and the vending machine is ensured through a Web-based HTML5 application.

In just one user interaction, the webapp identifies the user and the vending machine bi-univocally. Once this “discovery” phase is over, the application seamlessly enables e-commerce transactions that allow the user to order, from the smartphone, any product available at the vending machine.

From a consumer perspective, the purchasing process remains extremely simple and it can be still accomplished in three steps as it is shown in Fig. 5-1b.

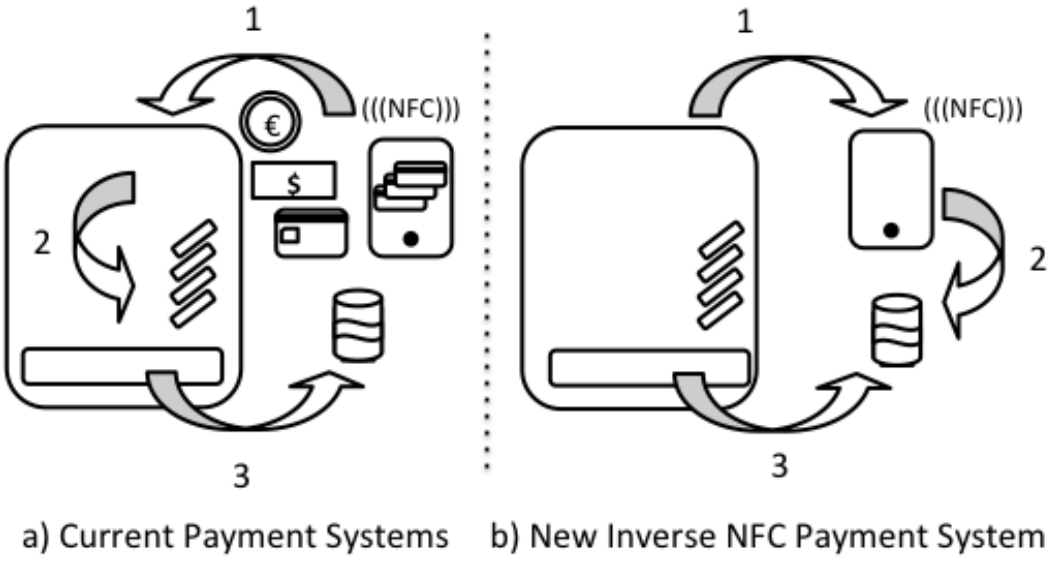


Fig. 5-1 Consumer´s interactions with a vending machine using current payment mechanisms versus a new inverse NFC mobile payment mechanism

How it works:

1. Customer taps the vending machine with the NFC-enabled smartphone.
2. Customer orders a product through the webapp tapping on the desired product icon on his-her smartphone.
3. Customer collects the purchased item.

From a cost perspective, using a NFC tag (Fig. 5-1b), which costs a few euros cents, is much cheaper than a proprietary credit card reader (Fig. 5-1a), which costs several hundreds of euros**.** On the other hand, NFC and HTML5 are both becoming pervasive technologies on smartphones, therefore in our inverse NFC mobile payment mechanism (Fig. 5-1b), the smartphone becomes the reader, resulting in hundreds of euros of savings per vending machine.

Alternatively, quick response (QR) codes [9] or Augmented Reality (AR) markers can be used to announce the URL of vending machine, but requires a native application on the phone to read them.

Other new emerging technologies like BLE (Bluetooth Low Power) may be adopted applying similar concepts. In such a case, the URL of the vending machine may be broadcasted via iBeacons [10] or Physical Web Beacons [11].

5.2.3 Simplifying Mobile Proximity Payments

The ultimate goal of our research is to enable mobile micropayments with an open and universal solution, independent from mobile manufactures, financial institutions, services providers or telecommunications operators. Fig. 5-2 shows the underlying high level interfaces (INT) between the key players and components of our solution.

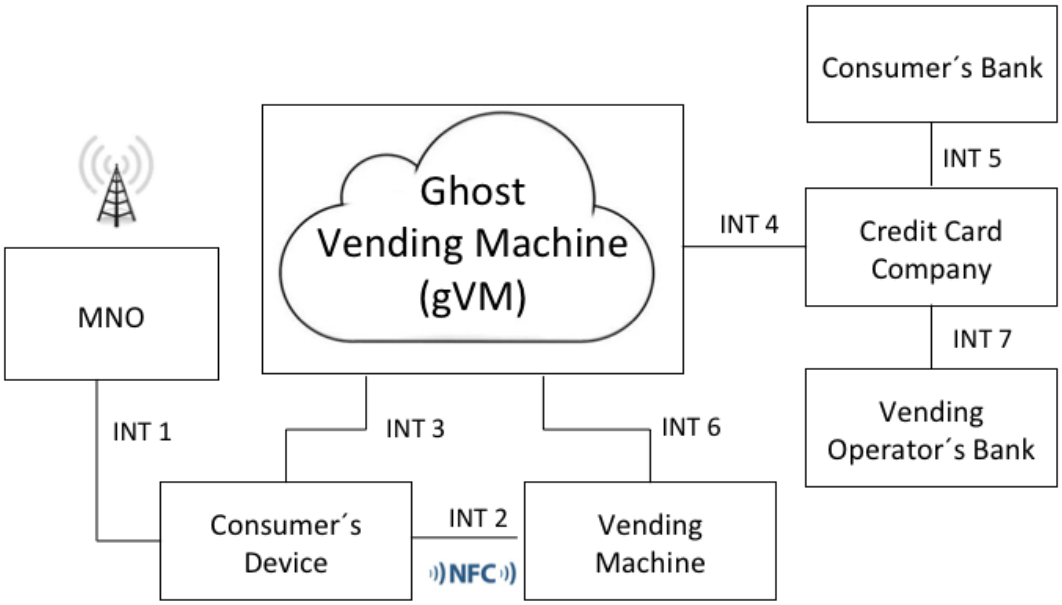


Fig. 5-2 Interfaces and players for inverse NFC mobile payments in vending

A short description of the interfaces 1 to 7 presented in Fig. 5-2 follows:

- INT-1: The Mobile Network Operator (MNO) provides mobile broadband connectivity to the consumer´s device.
- INT-2: Each physical vending machine has a URL. NFC technology at the consumer device is used to ease the access to the URL that is itself linked to an e-commerce shopping cart. The virtual representation of the vending machine is called here ghost vending machine (gVM). In this way, by tapping the NFC tag of the vending machine with a smartphone, the consumer may order products from the vending machine through a webapp.

- INT-3: HTML5 is the technology selected to interact with the gVM. Its benefits are out of the scope of this paper but it is worth to mention that the possibility of using local storage (client-side) in HTML5 Web applications is a powerful feature that puts webapps into parity with native applications in terms of start-up time and responsiveness. Moreover, and this is very important for our purpose, the usage of HTML5 webapps do not need the installation of additional software on the smartphone.
- INT-4: The e-commerce transactions are done through a virtual Point of Sales (PoS) as in any online shop.
- INT-5: Financial transactions will flow through the secured interfaces provided by financial institutions.
- INT-6: The physical vending machine is connected to the Internet through an IoT Gateway installed inside it. This gateway is built in an electronic board designed and assembled with low cost Arduino [12] compatible modules. Once the payment is successfully processed, the machine receives the order to deliver the product. The logical interfaces are based on RESTful technologies commonly used in IoT deployments.
- INT-7: vending operator´s Bank is in charge to reconcile the different consumer payments.

When using credit card enabled NFC readers for vending machines, Fig. 5-1a, service providers may charge up to 5% for each transaction. While in our inverse model, Fig. 5-1b, e-commerce fees are around 0.5% because vending operators can negotiate directly, with financial institutions, the commission fees and configure their own e-commerce account without intermediates.

To summarize, our approach brings the scale and ubiquity of e-commerce web technologies, resulting in lower transaction fees from up to 5% to lower than 0.5%.

5.2.4 Security by Design

Security concerns when using cloud computing services [13]. Designing a secured proximity mobile payment system for un-attended point of sales, such as vending machines, requires the analysis of security from different angles as described below.

5.2.4.1 Physical Security: tampering un-attended point of sales

Vandalism is an issue in all the countries and therefore the system should be designed with ruggedized hardware. Our design will expose only the NFC Tag on the front of the vending machine. The NFC Tag can be easily protected with a plastic cover.

5.2.4.2 Proximity payments: collocate consumers and vending machines

NFC peer-to-peer (P2P) technology is commonly used in mobile proximity payments in such a way that the user and the PoS must be close (collocated) to exchange information. To lower the costs, our design is based on a simple NFC Tag or QR code that stores the URL of the vending machine in a similar way to that followed in the NFC smart posters. Our enhanced approach is based on a one-time URL that is “consumed” after each transaction. Reading this URL, stored in a dynamic NFC tag or P2P NFC module, becomes a proof of mobile proximity as it is uniquely associated with one machine and can only be read through a NFC smartphone.

At this point, it is worth to mention a successful payment mechanism based on tokenized QRs that WeChat [14] has recently introduced in China.

Conceptually this system behaves similar to our design: it generates one-time QRs on consumer WeChat App. Then merchants use their smartphones to scan the QR in order to validate transactions without need of credit card readers.

If mobile payments based on tokenized QRs become pervasive, our solution fits well on modern vending machines with displays, where tokenized

QRs may be shown. Reading these QRs will point consumers to our one-time URLs.

5.2.4.3 Unauthorized users: encryption of user credentials on smartphones

The number of interactions between the user and the terminal must be minimal. Unfortunately, this requirement collides with the security requirements.

In order to achieve that goal, the local storage of the HTML5 web browser is used to store the user credentials to allow silent login to many vending machines. A demo is available at [15].

The drawback of this solution is that anyone who has access to the smartphone can get these credentials unless they are encrypted. The solution then includes the following procedures:

1. The user name and password are stored in the server ciphered using the Advanced Encryption Standard-256 (AES-256) algorithm. This information is not kept in the smartphone while the server only stores encrypted data.
2. The encryption key is stored in the local storage of the smartphone of the user, along with a one-use-token generated by the server.
3. Every time the user is going to perform a purchase, both the encryption key and the token are sent to the server. The token is checked and if it is correct, the key is used to decrypt the user credentials to log into the server. The encryption key is not kept in the server after its use.
4. A new token is generated and sent to the user´s smartphone.

The advantage of this mechanism is that, if the smartphone is compromised, the attacker can only get access to the key to decipher the credentials but not to the credentials themselves (which are stored encrypted in the server). The one-use-token guarantees that if an attacker gets the key, she/he can only access the system once.

5.2.4.4 *Unauthorized transactions: a cross-platform two factor authentication security mechanism*

Optionally or when the dynamic URL of the vending machine cannot be read, a personal identification number (PIN) is generated randomly for each transaction and showed in the display of the vending machine. The user has to enter this PIN into the webapp to validate the transaction. A demo is available at [15]. This mechanism allows verifying the collocation of the user with the machine. It also permits the usage of other ¨discovery¨ technologies that use fixed URL like printed QR codes or Augmented Reality (AR) markers, URLs in browser history, webapp shortcuts, etc.

5.2.4.5 *Secure Web communications: adding Secure Socket Layer (SSL)*

Our webapps use HTTPS to secure all information exchanged online.

5.2.4.6 *Secure IoT communications: implementing Lightweight Machine-to-Machine (LWM2M) security model*

The IoT module is based on Arduino Open Hardware. When several vending machines are located in proximity to each other it is possible to create ad-hoc mesh networks to reduce the number of IoT modules acting also as IoT gateways. Hence, several machines can be connected between them and to the one that is equipped with the IoT gateway.

To secure the IoT module communications, our design is guided by LWM2M [16] recommendations and uses Pre-Shared Keys (PSKs). PSK is preferred to other mechanisms also proposed by LWM2M because of Arduino's limited computing resources.

This design decision implies that the same PSKs and PSK IDs need to be generated, and installed on the IoT module and on the backend.

LWM2M proposes two modes:

- TLS_PSK_WITH_AES_128_CCM_8
- TLS_PSK_WITH_AES_128_CBC_SHA256

Both options provide all the requirements of a cryptographic system: confidentiality, authentication and data integrity. We have selected the second mode considering availability of Arduino libraries.

When we encrypt a message the result is a random sequence of bits. This result has to be encoded into printable characters to be sent in the message. There are two options for this: hexadecimal or American Standard Code for Information Interchange (ASCII) characters. With hexadecimal encoding each byte of information is encoded in two bytes while with ASCII, using the Base64 encoding, 3 bytes are encoded into 4 bytes. Due to the resource-constrained memory of Arduino we have chosen the Base64 encoding (lower overhead).

5.2.5 Cloud Based Architecture

Cloud Computing mechanisms allow the solution to seamless scale from managing few machines to several thousands of them. As a consequence, the size of the vending operator, measured in number of vending machines to be connected, should not be at all a barrier to use this solution.

The Fig. 5-3 describes the building blocks of the architecture based on Open Source software and its main interfaces (INT).

Fig. 5-3 also shows the three distinct service categories common in Cloud Computing architectures: Infrastructure as a Service (IaaS) [17], Platform as a Service (PaaS) and Software as a Service (SaaS).

In the following subsections we describe the three categories in detail.

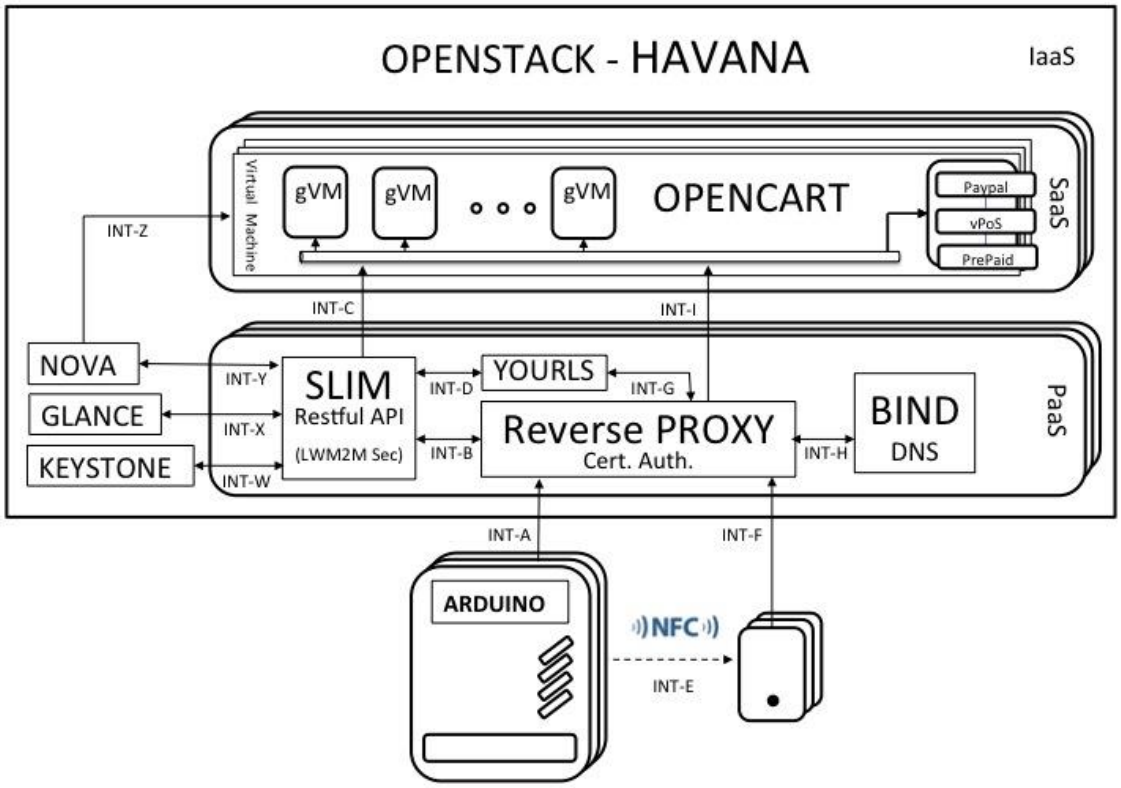


Fig. 5-3 Building blocks of the architecture using Open Innovation

5.2.5.1 IaaS: Building our private cloud for the infrastructure layer.

Our design uses OpenStack Havana on Ubuntu 12.04 TLS. OpenStack is a free and open-source software that includes several key components such as Compute, Identity Service, Networking, Image Service, Block Storage, Object Storage, Telemetry, Orchestration, and Database.

For simplicity sake, we have only shown in Fig. 5-3 the main components and interfaces that configure a plug-and-play with zero-configuration solution. When we connect “the first” IoT module allocated to a new vending operator (INT-A), the platform layer (INT-B) will interact with the infrastructure layer (INT-W, INT-X, INT-Y) to deploy an “image” of OpenCart for this vending operator at the application layer (INT-Z).

5.2.5.2 PaaS: A multi-tenant architecture where multiple vending operators share resources

The design of our PaaS aims at facilitating the deployment of gVMs. The main Open Source PaaS building blocks are:

- BIND (Berkeley Internet Name Domain): we use it for our internal network resolution to address each gVM (INT-H).
- YOURLS (Your Own URL Shortener): it is a URL shortening service to write short URLs on NFC tags (INT-D/INT-B/INT-A).
- SLIM: it is a RESTful framework we use to communicate with our IoT modules (INT-B/INT-A).
- Reverse Proxy based on Apache web server: it is the entry point to the gVM in the application layer. It provides HTTPS for the webapps (INT-F/INT-G/INT-H/INT-I) and routes to SLIM the RESTful queries to IoT modules (INT-A/INT-B/INT-C).

The first time we connect an IoT module (INT-A) the platform will create a new multi-shop in the right instance of OpenCart (INT-C), represented in the diagram as a gVM.

5.2.5.3 SaaS: The application layer where ghost vending machines reside

We need to provide a Solution that allows every vending operator to benefit from Telemetry and Cashless systems at very low cost. A SaaS model combined with Pay-as-You-Grow (PAYG) mechanisms minimize upfront investments and keep under control the operating expenses.

Our approach leverages OpenCart, a free e-commerce software which supports multi-store management. In our implementation, we deploy an OpenCart “image” for each vending operator. Each store inside OpenCart represents a gVM. OpenCart has built-in order management and payment

gateways and therefore, for our purpose, we have just added our webapp in HTML5 as an OpenCart module extension.

To buy a product in our connected vending machine, the process starts when the consumer taps the NFC tag with his smartphone (INT-E). The webapp is launched and opens a secured https session towards the reverse proxy (INT-F). The reverse proxy translates the shortened URL into the original URL (INT-G) and resolves the URL via the internal Domain Name Server (INT-H). Finally, the reverse proxy routes the traffic to the right gVM (INT-I). At this moment the consumer makes the selection of the product from the touch screen of his smartphone, billing is performed and product is delivered. To deliver the selected product, the vending machine gets instructions from the gVM through interfaces INT-A/INT-B/INT-C.

In summary, building up our own private cloud, allows us to implement self-provisioning mechanisms to ease the daily operations and contribute to reach our target: minimize the Total Cost of Ownership (TCO) for vending operators.

5.2.6 Open Hardware Prototype and Preliminary Benchmarks

To build our Open Hardware prototype we have selected Arduino compatible electronic components, interacting with the Vending Machine Controller (VMC) as despite in Fig. 5-4.

But, why not choose Raspberry-Pi [18]? Cost saving was our mantra and obviously Arduino is cheaper than Raspberry-Pi. Arduino costs around $10-20 depending on the version, while the price of Raspberry is around $25-40. However, cost was not the only reason to choose Arduino. To retrofit any vending machine, we need a plug-and-play device that can be turned ON and OFF at any point of time. Raspberry-Pi runs on an Operating System (OS) so it must be properly shut down before turning OFF the power, otherwise OS may get corrupt and Pi can be damaged. As a consequence, after a power outage, the risk to have the vending machine out of order is too high. This is also one of the main reasons why we did not choose Raspberry-Pi.

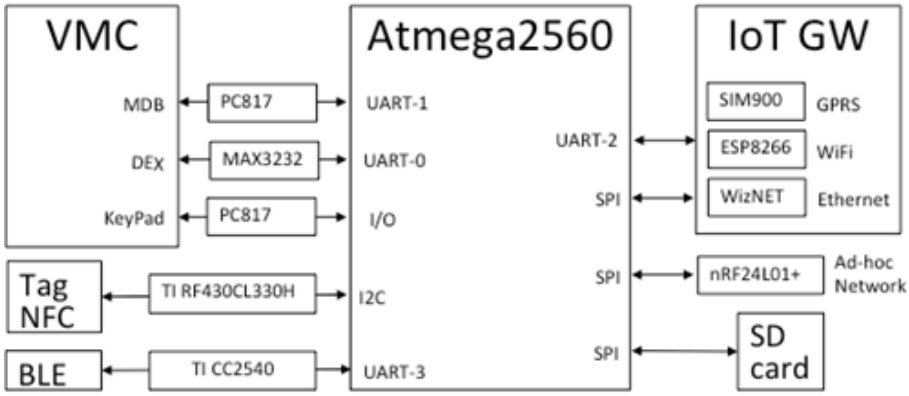


Fig. 5-4 Open Hardware design based on Arduino Mega

On the other hand, Arduino is well suited for repeated type of works like keep alive the slave-master synchronization mechanism, which requires to send every 200ms acknowledge messages to the vending machine controller (VMC).

As disadvantage, Arduino is a simple microcontroller and is not capable of doing multiple tasks at a time, like a computer as Raspberry-Pi does. As shown in Fig. 5-4 our design is quite demanding for Arduino. Other than behave as a slave cashless device, it has to manage communication flows to enable payments and telemetry. To overcome such limitations, we programed a pseudo multitask mechanisms consisting of:

a) Interrupts every 200ms, to keep alive the master-slave synchronization with the VMC.
b) Finite-State Machine (FSM) computation models, to handle the call flows with the back-end servers and to interact with other peripheral devices such as NFC modules, hacked vending machine keypad, etc.

To improve the response time using FSM handlers, it is important not to lose cycles of microcontroller, therefore we avoid the use of the directive “delay()”. Unfortunately, this is a directive commonly used in Arduino libraries, which force us to discard them and code at very low level. After optimizing and control the memory of our code it results in a firmware below the 60 Kbytes, and a stable dynamic memory allocation of 6331 bytes. In Table 5-1 are summarized the

performances results after running a stress test of three hundred consecutive orders.

Table 5-1 Performance test result

Test	Count	Minimum (ms)	Maximum (ms)	Average (ms)
Cycles	300	7918	13,457	9049
GET	295 *	2235	5189	2972
VEND	295 *	1499	1501	1499
POST	295 *	4140	6994	4565
FREE Memory Allocation	300	6331	6331	6331

* Difference due to 5 timeouts from server.

Fig. 5-5 represents an average response time as a call flow diagram. The average time to complete an order is based on mentioned stress test (Table 5-1).Two main calls are required to complete an order (GET and POST). First, once the product has been selected on the smartphone, the vending machine performs a GET RESTful call. This call retrieves the product selection, including the price and the slot on which the product is located in the vending machine. Gathering such info takes 2972 milliseconds. At this moment the vending machine performs the VEND in 1499 milliseconds and the product is delivered. That is, the product is delivered in 4472 milliseconds providing a good purchasing experience.

To complete the ordering cycle a second POST RESTful call is performed to update the system and inform the user that the order has successfully been delivered. The POST requests requires an additional 4565 milliseconds, which overlaps with the time used by the consumer to collect the order from the vending machine. As outcome the overall cycle is completed in 9049 milliseconds and the system is ready to serve new orders. The performance of the stress test shows the robustness and viability of our open hardware design.

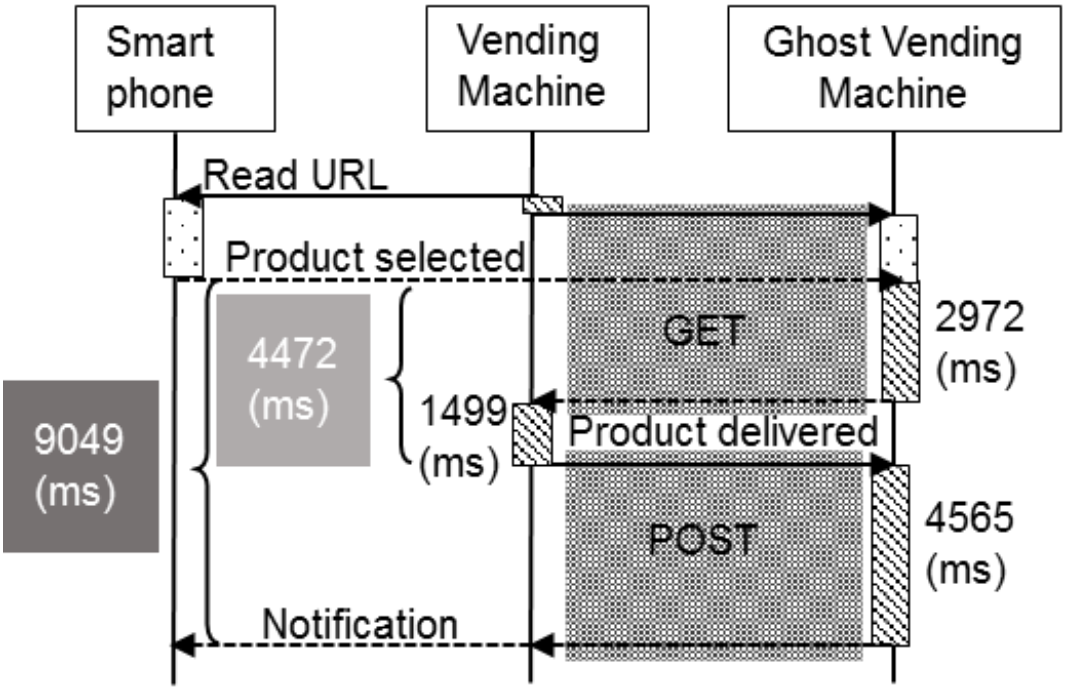


Fig. 5-5 Average response time in a call flow diagram

It is worth to note that with good cellular coverage, using a 3G MiFi router instead of a standard WiFi did not impact the consumer purchase experience. In such scenarios, the overall cycle to complete an order increased just a few hundred milliseconds.

5.2.7 Conclusions

The emerging Internet of Things (IoT) covers a wide range of industries, devices and applications [19, 20]. This paper presents a real deployment of an IoT system for vending machines to enable mobile micropayments with an open and universal solution, independent from hardware manufacturers, financial institutions or telco operators. Dealing with micropayments is a challenging long tail problem. To solve it, our solution uses new paradigms like cloud computing, IoT and web technologies.

The project covers all the phases needed to fully integrate the vending machines in the IoT in a scalable manner, leveraging cloud and open source technologies and targeting the lowest cost-of-ownership for vending companies.

The new approach presented in this paper allows to “log” into a vending machine using a smartphone, and purchase one of the available products using a web application in the smartphone. The webapp identifies the user and the vending machine and allows the consumer to order products. The same phone is used to make the product selection and the payment over the cloud using an alternative and secured proximity payment model.

We have also presented the cloud-based architecture, which supports the solution and allows a low cost scalable solution.

Our functional prototype complies with three basic design principles: it is a cost affordable solution, it is user friendly [21] and it is easy to operate.

A significant take away of our work is the realization that new innovative services and payment architectures can be deployed in a scalable and cost effective way in a very demanding and margin-constrained business like the vending one. Our approach can also be extended into other industries that would also benefit from easy to implement mobile e-commerce systems, e.g. public transportation, fast food ...

Following our prototyping phase, the next step in this project would be to demonstrate the benefits of our approach in a real case scenario.

5.2.8 References

[1] Data Transfer Standard EVA DTS 6.1.1, Dec 2010, European Vending Association aisbl, 44 rue van Eyck – 1000 Brussels, <http://www.vending-europe.eu>

[2] Multi-Drop Bus / Internal Communication Protocol Version 4.2, February, 2011, National Automatic Merchandising Association, 20 N. Wacker Drive, Suite 3500 Chicago, Illinois 60606-3120 USA, <http://www.vending.org>

[3] T. Yokouchi. Today and tomorrow of vending machine and its services in Japan. Proc. IEEE, Service Systems and Service Management (ICSSSM), 2010 7th International Conference on, 1–5, 2010, doi: 10.1109/ICSSSM.2010.5530240.

[4] Y. Park, S. Yoon. A comparison study of stock-out policies in vending machine systems. Proc. IEEE, Engineering and Industries (ICEI), 2011 International Conference on, 1-4,2011

[5] T. C. Poon, K. L. Choy, C. K. Cheng, S. I. Lao. A realtime replenishment system for vending machine industry. Industrial Informatics (INDIN), 2010 8th IEEE International Conference on, pp: 209–213, July 010 doi: 10.1109/INDIN.2010.5549432

[6] L. Atzori, A. Iera, G. Morabito. The Internet of Things: A survey. Computer Networks, 54, (2010), 2787–2805

[7] Z. Wen, Z. X. Long. Design and Implementation of Automatic vending machine Based on the Short Massage Pay-ment. Proc. IEEE, Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on, 1-4, 2010, doi: 10.1109/WICOM.2010.5600192.

[8] M. Jovanovic, M. Organero. Analysis of the Latest Trends in Mobile Commerce using the NFC Technology. Journal of Selected Areas in Telecommunications (JSAT), 2011, ISSN: 1925-2676

[9] Muradzikwa, Gresham, et al. "Designing of Android Mobile Based System Using QR Code." (2014). [10] <https://developer.apple.com/ibeacon/>

[10] <https://google.github.io/physical-web/>

[11] <https://arduino.cc/>

[12] S. Subashini, V. Kavitha. A survey on security issues in service delivery models of cloud computing (Review). Journal of Networks and Computer Applications, 1-11, 34, 1, January 2011

[13] Huang, Wei, and Jialian Tang. "Explore the Development of WeChat Payment from User Behavior." 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). Vol. 3. IEEE, 2015.

[14] <http://openvend.es/demo>

[15] Lightweight Machine to Machine Technical Specification Candidate Version 1.0 – 10 Dec 2013, <http://openmobilealliance.org/>

[16] S. M. Sunilkumar, K.S. Gopal. Resource management for infrastructures as a service (IaaS) in cloud computing: A survey. Journal of Networks and Computer Applications, 424-440, May 2014

[17] <https://raspberrypi.org/>

[18] D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamta. Internet of things: Vision, applications and research challenges. Ad Hoc Networks, 10, (2012), 1497–1516.

[19] P. Patel, D. Cassou. Enabling high-level application development for the Internet of things. The Journal of Systems and Software, 103, 62-84, 2015

[20] BHATTI, Tariq. Exploring factors influencing the adoption of mobile commerce. The Journal of Internet Banking and Commerce, 2015, vol. 2007.

5.3 A self-provisioning mechanism in OpenStack for IoT devices

Abstract: The aim of this paper is to introduce a plug-and-play mechanism for an Internet of Things (IoT) device to instantiate a Software as a Service (SaaS) application in a private cloud, built up with OpenStack. The SaaS application is the digital avatar of a physical object connected to Internet. As a proof of concept, a Vending Machine is retrofitted and connected to Internet with an Arduino Open Hardware device. Once the self-configuration mechanism is completed, it is possible to order a product from a mobile communication device.

Keywords: Internet of Things; cloud computing; openstack; arduino

5.3.1 Introduction

The Internet has evolved and grown beyond our expectations. It is expanding much more rapidly than it has done in the last decade. Internet of Things (IoT) is its new revolution, being one of the most relevant trends in the software industry. IoT is the fusion of the digital and physical world. In a world of IoT, billions of things or devices of all types and sizes are interconnected and uniquely identified. Devices are becoming instrumented, intelligent and interconnected. In this sense, maker and hobbies communities are hacking daily objects and connecting them to the Internet, discovering new ways to interact with them. For instance, how to retrofit a lamp and switching the lights from our smartphone is a simple case published in many blogs of such communities. Thanks to tiny embedded and cheap microcontrollers and sensors, it is not difficult to build up your own home automation solution. A relay and a WiFi or Bluetooth communication module below 5 USD, plus some lines of code borrowed from Open Hardware community would be enough to this end.

In this way, today our daily objects are becoming “smart”. This smarter and connected world has the potential to completely change our way of life. Examples of IoT solutions can be cars that talk each other about traffic congestion or

medicine containers that remind the time to take your pills. In fact, clothing, factories will eventually be “smart” as well. The possibilities are endless. By moving the logic devices uses to be embedded in electronics to the cloud it is possible leveraging cloud computing paradigms [1]. To this end, it is only required to connect our daily objects with low cost communication modules to Internet and to integrate the machinery of such objects with some sensors and actuators enabling the discovery of new ways to interact with them.

In just a couple of years a boom has occurred in the cloud based platforms to enable the IoT [2,3]. Early in 2009 Pachube [4] sets the foundations for such platforms and today there are hundreds of them enabling to collect data from our network of sensors and providing north-bound interfaces for data manipulation [5,6]. All of them claim to have plug-and-play mechanism to connect sensors and simple devices and they usually provide in-build simple scenarios such the mentioned example to control our retrofitted lamp. However, due to the intrinsic complexity of our physical world, in order to create digital version of complex objects or devices, which may be composed of many sensors and actuators [7], it is necessary to deploy bespoke logic at the application layer. In this context, to provide end to end self-configuration mechanisms is not an easy task.

The main challenge of this paper is to develop a simple plug-and-play mechanism to automate the deployment of digital version of complex objects in Internet, the so called in this paper digital avatars. These avatars are deployed following a model of Software as a Service (SaaS) in a cloud platform. In other words, the SaaS at the application cloud layer is the digital avatar of a physical object connected to Internet. To this end, a private cloud infrastructure with minimum hardware requirements using OpenStack [8] is deployed. OpenStack allows the creation of a very cost effective, flexible and elastic Information Technology (IT) infrastructure, taking full control of the resources and configuration required at the platform and the application layers. The key point of our work is to deploy a cloud-based plug-and-play mechanism for IoT devices in a simple way, with no need of performing ad-hoc and complex configuration actions by the cloud system administrator.

This plug-and-play mechanism and the cloud developed can be used by small, middle sized and large scale organizations with high efficiency and security. Multiple projects for multiple clients can be created in a cost efficiency way using this infrastructure.

As a proof of concept in this paper a vending machine to make it smarter is retrofitted. The evolution of the traditional architecture of buying in a vending machine by a cloud-based architecture is proposed. The core processes of the buying are offered through a SaaS business model. In this way, vending machines are connected and integrated in a cloud environment. It reinforces the concept of IoT by making objects smarter thanks to ubiquitous connectivity and new cloud computing paradigms [9]. This approach is achieved by moving business logic from real vending machines to the cloud. Usually, vending machines are owned and managed by vending operators. Therefore, vending machines are grouped and configured in a cloud multi-tenancy architecture where tenants are associated to vending operators and each tenant serves several vending machines. The open software used in the SaaS layer is OpenCart [10], a multistore shopping cart. This platform makes possible to offer service to many vending machines using a single domain. In this way there is no dependence on external Domain Name Server (DNS), apart from the public DNS where the domain is registered.

This paper is organized as follows. Section 2 briefly presents the cloud computing services and model. In Section 3, OpenStack architecture and components are described. The built of our private cloud and the plug-and-play automation are explained in Section 4. In Section 5 the model developed is explained and applied to retrofit a vending machine. Finally, some conclusions are presented in Section 6.

5.3.2 Cloud Computing at a Glance

Cloud computing is a modern computing paradigm that provides IT infrastructures. It involves deploying groups of remote servers and software network that allow the users to access different information from anywhere. The

cloud computing removes the need for user to be in the same physical location as the hardware that stores data. The cloud provider can both own and house the hardware and software necessary to run home or business applications [11].

Cloud computing can be classified into three main categories attending to the service model it offers (see Fig. 5-6):

- Infrastructure-as-a-Service (IaaS) is the most basic cloud service model. It provides virtual machines (VMs), load balancers, raw block storage, firewalls and networking services. Service provider owns the equipment and is responsible for housing, running and maintaining it.
- Platform-as-a-Service (PaaS) provides a computing platform including application program interfaces (APIs), operating system, development environments, programing languages execution environment and web servers. Users can access and use these tools to create applications on the service provider's platform over the Internet.
- Software-as-a-Service (SaaS) offers users the hardware infrastructure, the software product and interrelates with the users through a portal. Cloud providers install and operate the application software in the cloud, authorizing an application to clients.

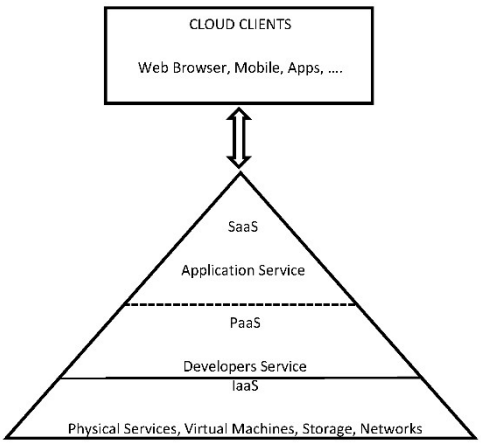


Fig. 5-6 Cloud Service Models

Cloud computing allows three deployment models: public, private or hybrid. If the services are provided over the Internet then it is public cloud, also called external cloud. When services are provided within an organization through

intranet then it is a private or internal cloud. Hybrid cloud is an internal/external cloud, which allows a public cloud to interact with the clients but keep their data secured within a private cloud.

5.3.3 OpenStack Overview

There are different free and open-source software solutions for setting up a private cloud [12]. Due to the simple, elastic, consistent and massively scalable services OpenStack offers, the proposed system is implemented using this software.

5.3.3.1 OpenStack Basic Architecture

OpenStack is able to control large pools of compute, storage and networking resources making use of a modular architecture, which uses different components to work together as a service. The three main components are the following:

1. OpenStack Identity Service. It provides identity, token, catalog of available services and policy. It tracks all OpenStack services installed.
2. OpenStack Compute Service. It is the cloud group controller. It provides a tool to deploy cloud including things like managing block storage, networking, computing resources, scheduling, authorization and hypervisors.
3. OpenStack Image Service. It is a mirror storage, query and retrieval system of virtual machines.

Fig. 5-7 shows the architecture of the cloud operating system [8]. The OpenStack Storage Service shown in the figure is a highly scalable object storage system although it is not an essential component in the operated mode. It is worth to note that OpenStack allows the management of all the resources through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

Concrete implementation of each component in our development is shown in the next section.

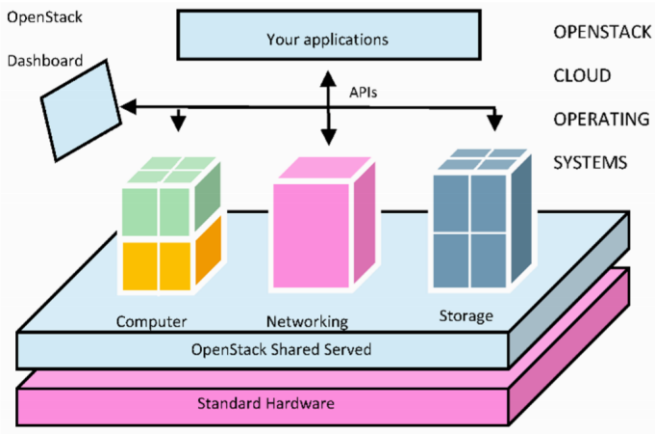


Fig. 5-7 OpenStack Cloud Computing Operating System

5.3.3.2 *Components of OpenStack*

OpenStack includes several key components such as Compute, Identity, Networking, Image, Block Storage, Object Storage, Telemetry, Orchestration, and Database. Fig. 5-7 shows the OpenStack system architecture. A brief description of the different components and what they provide is given below.

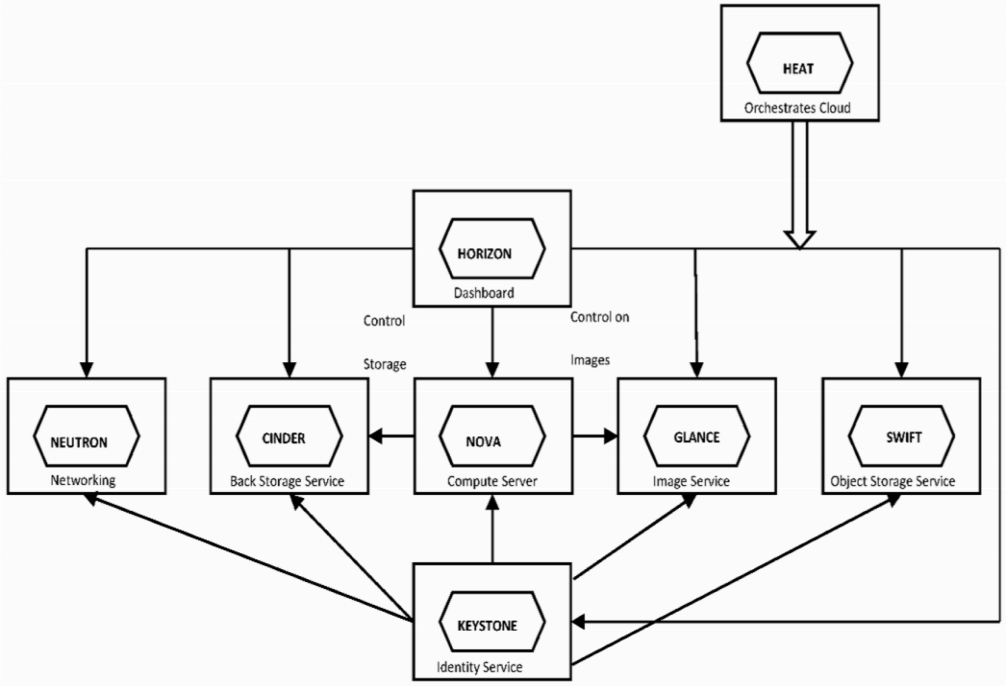


Fig. 5-8 OpenStack system architecture

- *KeyStone* provides a unified authentication and high level authorization service for all the components in the OpenStack family. It supports token based authentication.
- *Nova* is the computing controller for the OpenStack cloud. It is used to manage various compute resources, networking, authorization, and scalability needs of the OpenStack cloud.
- *Cinder* is a block storage component, which provides persistent block-level storage devices for use with OpenStack compute instances.
- *Glance* allows spinning up virtual machines quickly when users request them. Glance helps accomplish this by creating templates for virtual machines. It can copy or snapshot a virtual machine image and allow that to be recreated. Glance can also be used to back up existing images to save them and it integrates with Cinder to store the images.
- *Swift* is an object storage system for objects and files. Swift plays an important role in scalability.
- *Horizon* implements the dashboard. It allows the user to access cloud services platform by a web front-end interface. Things like manage instances and images, create keypairs or attach volumes to instances can be accomplished using it.
- *Neutron* is related with the networking. It enables tenants to create advanced virtual network topologies, improving performance and security.
- *Heat* implements an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files.

5.3.4 Proposed General Architecture

5.3.4.1 The Challenge, Modeling Complex Digital Avatars

Similar to the Physical Web Google approach [13], each connected vending machine is identified by means of a Uniform Resource Locator (URL). The URL points to a Web application (Webapp), which is in fact the digital avatar of the vending machine. By accessing this URL from a smartphone, consumers will be able to interact with the vending machine and order products online.

To create a digital avatar of a vending machine two facts have been considered: (i) a vending machine is an un-attendant point of sales; and (ii) nowadays, a point of sale on the Internet is an online shop. Therefore, the vending machine is modelled by means of open-source e-commerce software. However, to have a working online shop several steps are required. First, it is necessary to mirror the vending machine settings and product's information such a price, stock and then, to keep the vending machine and its digital avatar synchronized. Moreover, to reach the online shop, the URL has to be announced to consumers and obviously it has to be provided online payment mechanisms.

The challenge is how to streamline all these configuration steps with a simple plug-and-play mechanism. The proposed approach consists in building up an own private cloud to take full control of the deployment of virtual machines, which contains all the software and logic to become digital avatars of complex objects.

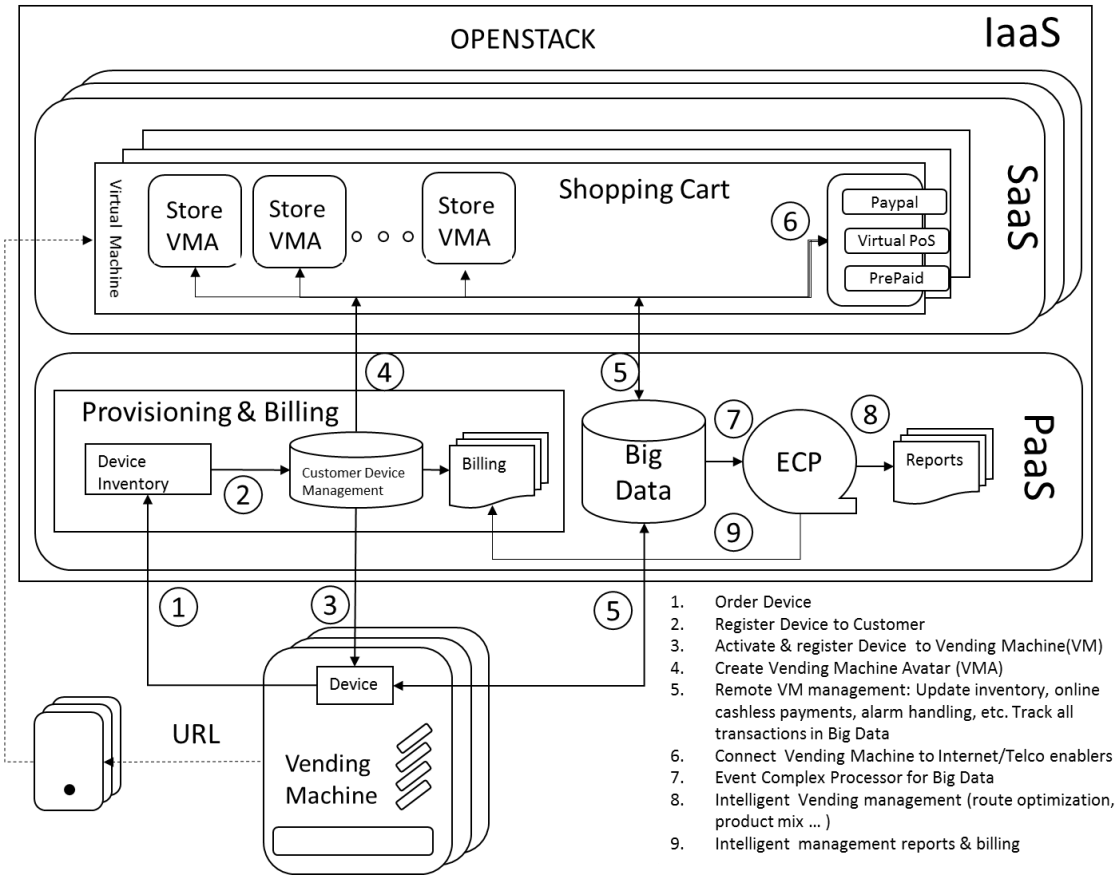


Fig. 5-9 High level end to end system design applied to vending machines

The Fig. 5-9 shows the proposed high level system design applied to vending machines. It also describes the provisioning, management and billing flows to have a fully functional end to end solution. This paper focuses mainly on step 4: the instantiation of the vending machine avatar in the Cloud.

5.3.4.2 The Target Scenario

A vending operator buys a device to retrofit a vending machine. When the device is plugged into the vending machine, it initiates a self-configuration process consisting in:

1. Launching an instance of an online shop.
2. Reading the Telemetry of the vending machine to configure the online shop.
3. Publishing the URL of the online shop.
4. Buy online and dispense products onsite.

As reference, Fig. 5-10 shows one our bespoke Arduino Mega open hardware designs. This board is powered by the Mutlidrop Bus Standard (MDB) interface of the vending machine and it is able to communicate with our Cloud Solution via WiFi. The first time this board is powered on, it will initiate the plug-and-play mechanism describe on this paper. As outcome, we have published a demo [14] to show how to access to different vending machines and order products.

Fig. 5-10 Arduino Mega compatible prototype.

The self-provisioning process should last not more than 3–5 min.

5.3.4.3 System Architecture

Fig. 5-11 shows a general block representation of the main components and interfaces that implement the global architecture of the system. In the Section 5.3.5 the mechanism of the plug-and-play with zero-configuration solution proposed is detailed.

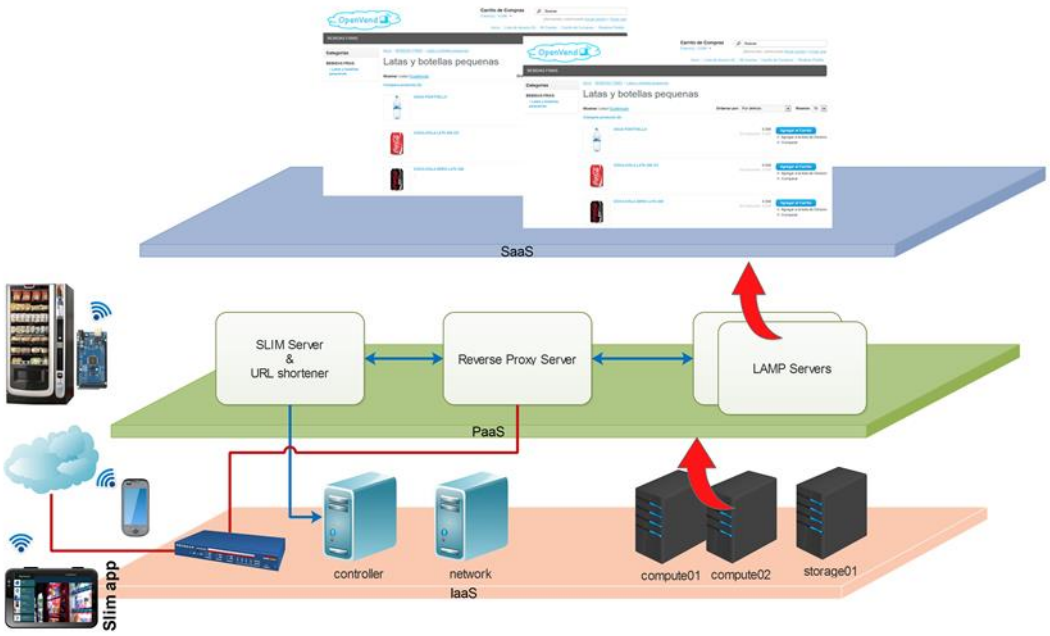


Fig. 5-11 Building blocks of the proposed architecture

The digital avatar of the vending machine resides in the SaaS layer. OpenCart is the free open software used to implement the e-commerce shopping cart. An OpenCart image is deployed for each vending operator. OpenCart is multistore, therefore each store inside OpenCart represents a vending machine.

The PaaS layer aims to facilitate the deployment of complex digital avatars such as our digital version of a vending machine in the Internet. The vending machine is connected to the Internet through an IoT module installed inside it. This module is built in an electronic board designed and assembled with low cost Arduino [15] compatible modules. The logical interfaces between the IoT module and the PaaS are based on REpresentational State Transfer (REST) technologies, commonly used in IoT deployments. Indeed, a RESTful API is an

application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. The first time an IoT module is connected, the platform spins up a digital avatar of the vending machine in OpenCart's multistore.

In the IaaS layer, among the different OpenStack components described in Section 5.3.3, NOVA is used to store and retrieve virtual disks ("images") and associated metadata in GLANCE. The format chosen in GLANCE to store the actual virtual disk files in the Object Store is QEMU Copy-On-Write file (QCOW2), a flexible format, which allows images to grow on demand. Kernel-based Virtual Machine (KVM) is used for virtualization. KEYSTONE is the entry Service to the infrastructure, where all RESTful API queries from PaaS layer are received.

5.3.5 Implementing the Self-Provisioning Mechanism

OpenCart's multistore mode, allows user to add more stores to the current installation by creating a subdomain structure for the stores, e.g., "http://store1.domain.com", "http://store2.domain.com" ... "http://storeN.domain.com". However, this approach does not suit our needs. A subdomain, essentially, is an actual DNS entry. Therefore, creating a subdomain is not necessarily so immediately obvious if our own DNS is not deployed. In addition, at times, even deploying our own DNS, the addition of a subdomain may not be immediately available due to potential DNS or Server-side propagation issues. In addition, from a SEO standpoint, it is difficult to increase rank in search engines and get traffic for N subdomains because Google treats them as different websites, regardless if they have one shared parent host.

To overcome the subdomain management issues it has been created a subfolder model for our OpenCart's multistore. In this way, subfolders to address the digital avatars of our vending machines, e.g., "http://domain.com/store1", "http://domain.com/store2" ... "http://.domain.com/storeN" are used. OpenCart documentation does not provide a full description of multistore, which may lead people to believe that subdomains are the only possible solution, but as it is shown below, it is possible. The following steps detail how to make a subfolder model working on OpenCart's multistore:

1. Make a new Folder inside your OpenCart structure. Let's call it "operatora001" because it will be the vending machine number "001" owned by "operator A".
2. Go to the new folder titled "operatora001" and create an ".htaccess" file. Then copy all the strings from the original .htaccess file to it.
3. Add the following to the ".htaccess" file:

```
RewriteCond %{REQUEST_URI} !^/operatora001/.*$
```

4. Create a new file inside the "*operatora001*" folder and name it "index.php." The structure of this file is the following:

```
<?php include('../index.php');
```

5. From the OpenCart admin panel go to Settings and create new Store. Add full URL path to the 'Store URL' of the sub-store like this: http://domain.com/operatora001/.

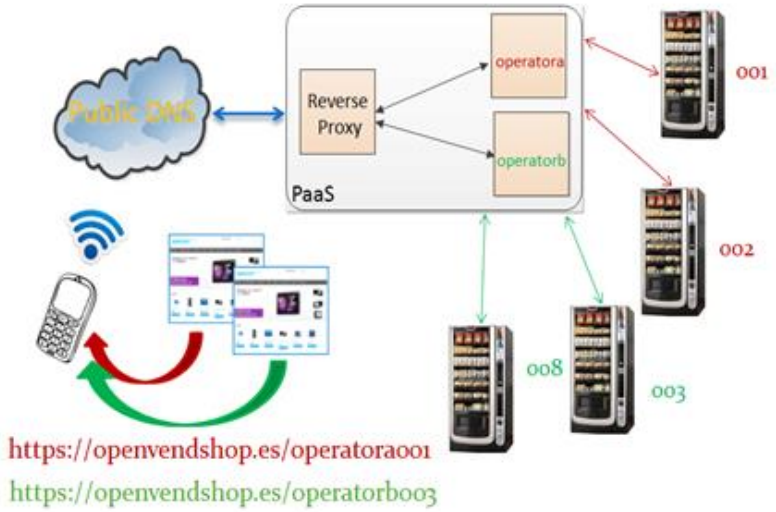


Fig. 5-12 Access through domains

At this point it is really important to note that different vending machines are accessed through only one registered domain, e.g., "openvendshop.es", (see Fig. 5-12). This mechanism allows not having to register domains for each

vending operator, or for each vending machine. It makes the plug-and-play process independent of 3rd party DNS and it contributes to cost affordable solutions, as domain providers use to limit or charge for subdomains.

The automatic handling of the configuration files in the plug-and-play mechanism is detailed in Sections 5.3.5.3 and 5.3.5.4. This is a complex task because of the mapping from subdomains to subfolders is performed in a reverse proxy which acts as the main entry point to the platform.

5.3.5.1 *Underlying Reference Infrastructure*

Fig. 5-13 shows a typical OpenStack deployment without High Availability (HA) used as reference for our project. The proposed design uses OpenStack Havana on Ubuntu 12.04 TLS. For this purpose, the deployment consists of:

- one *controller node*, where services for the environment run.
- one *network node*, responsible for the virtual networking.
- two *compute nodes*, servers where Virtual Machines (VMs) are created.
- one *storage node* to store cinder volumes and images.
- one *util node* used to provide system administration functions, for monitoring and for maintenance purposes.

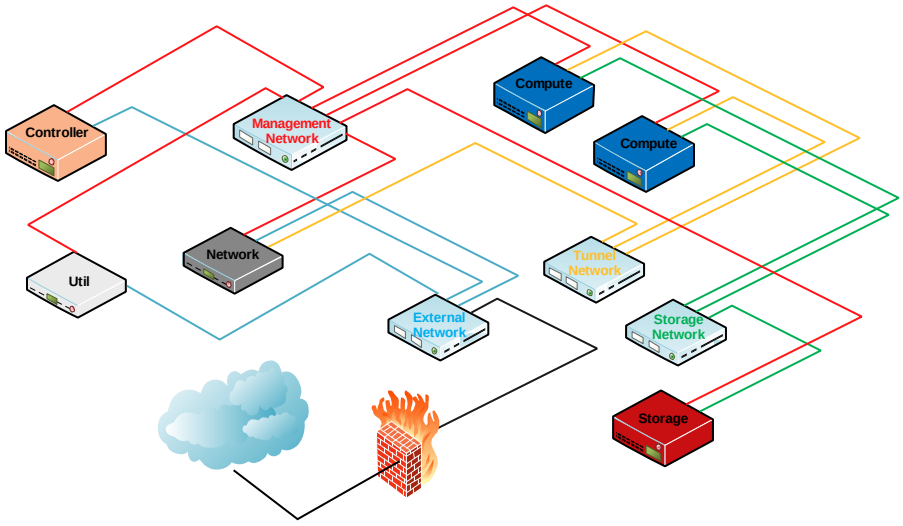


Fig. 5-13 Typical OpenStack deployment

Regarding networking, four different networks are created and connected through switches. The usage of the networks is as follows:

- *external network*: it is a public network used for Internet access for all the nodes. Allows both inbound and outbound connections for VM's.
- *management network*: used for communication between the controller and the compute nodes. It supports the internal communication between OpenStack components.
- *tunnel*: used for VM data communications.
- *storage*: used for communication between the storage nodes (cinder) and the compute nodes.

5.3.5.2 Automation of OpenCart Instantiation

Two of the main PaaS building blocks in Fig. 5-11 are SLIM [16] and reverse proxy. SLIM is a PHP micro framework, which allows a quick deployment of RESTful APIs to communicate with the IoT modules based on Arduino open Hardware. Reverse proxy is based on Apache web server and is the entry point to the platform. It provides HTTPS for the webapps and RESTful APIs to IoT modules. Therefore, each time a new IoT module is plugged, SLIM initiates the plug-and-play process.

RESTful API queries are sent from the SLIM block to interact with the underlying IaaS controller node as Fig. 5-14 shows.

The main interactions result in:

1. Retrieving a Universally Unique IDentifier (UUID) token for subsequent secure interactions.
2. Retrieving an OpenCart image reference to be launched.
3. Retrieving "flavors" (number of virtual CPU's, RAM, Disk capacity, Ephimeral Disk capacity)", floating IPs and access keys used by new Virtual Machines instantiated in the PaaS.

4. Launch the new Virtual Machine (as a simple example of how to instantiate a virtual machine in OpenStack from an Arduino open hardware device, refer to source code provided in Appendix A).
5. VM is ready to receive telemetry data from vending machines to configure their digital avatars.

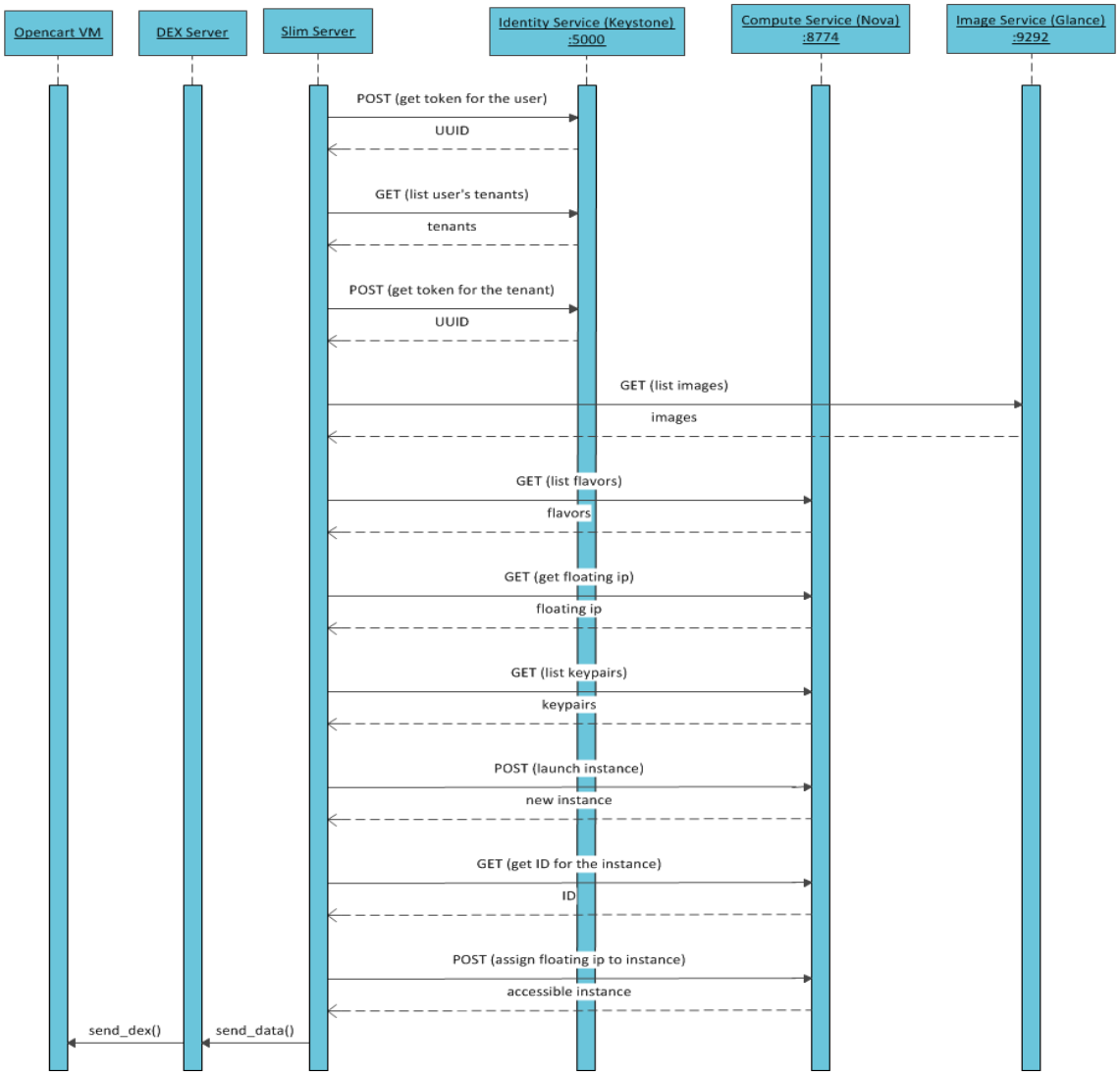


Fig. 5-14 Sequence flow between SLIM and OpenStack's components to instantiate OpenCart

5.3.5.3 *Self Configuration of OpenCart and Reverse PROXY*

A self-configuration process is done by using shell scripts. The aim of the shell scripts is to automate the generation of VirtualHosts on the Apache servers and make some initial OpenCart configurations.

These scripts are pre-programmed into the OpenCart and reverse proxy images and once a new instantiation is required from the SLIM server, their execution start. At this time, base64 encoded user data for the scripts are injected.

Table 5-2 shows the OpenStack RESTful API specification to launch a virtual machine.

Table 5-2 OpenStack RESTful API specification to launch a virtual machine

Method: POST	**URL:** http://iaasopenstack.dyndns.org:8774/v2/{tenant_id}/servers
Header Name	**Value**
Content-Type	application/json
X-Auth-Token	<UUID>
Body	

```
{
"server" :
   {
    "name" : "name",
    "imageRef" : "image_id",
    "flavorRef" : "flavor_id",
    "key_name" : "keyname",
    "security_groups":[
      {
      "name": "default"
      },
      {
      "name": "webserver"
      }
    ],
   "user_data": "script encodedb64"
   }
}
```

The decrypted script injected in the value of the “user_data” field inside the body of the POST query looks as follows:

```
#!/bin/bash
echo -e "O\ndomain\nnombreOperador\nip_flotante_tenant\n  \
nombreTienda\n...\nnombreTiendaN\n" > /etc/scripts/domainstores
source /etc/scripts/frontscript
```

These data are mainly, the domain, the vending operator's name and the number of vending machines in OpenCart's multistore instance.

Fig. 5-15 describes the scripts' logical flow (see details in [17]). It can be noted that initially scripts are executed in the virtual machine where OpenCart is roll out, during instantiation time. From there, the execution flows to the reverse proxy. Once the cycle is finished, vending machines can be accessed from the Internet. Shell Scripts Flow

This section presents an overview of the main interactions among scripts during the instantiation process.

The process starts with an OpenStack API request from SLIM block (Fig. 5-14). Then the user data information is injected by an encoded base64 script (Table 5-2) that it is necessary in the rest of the process. Following this request, an automatic sequence of calls is triggered:

1. Initial script execution. It allows starting the logic to insert directives of Apache into the VirtualHosts. It also sends data to reverse proxy and execute scripts remotely.
2. VirtualHost creation on server where OpenCart resides. This script creates the VirtualHost for the domain if it does not exist.
3. Store data insertion into the VirtualHost. This script first includes injected data from OpenStack and then adds substitute directive into the VirtualHost. It also makes the necessary changes in the VirtualHost when a new default store is launched (see example in Appendix B).
4. Changes in OpenCart config.php file are also carried out. These scripts use some templates to fill the data. These are denoted in Fig. 5-15 as Vhtemplate and Storestemplate.

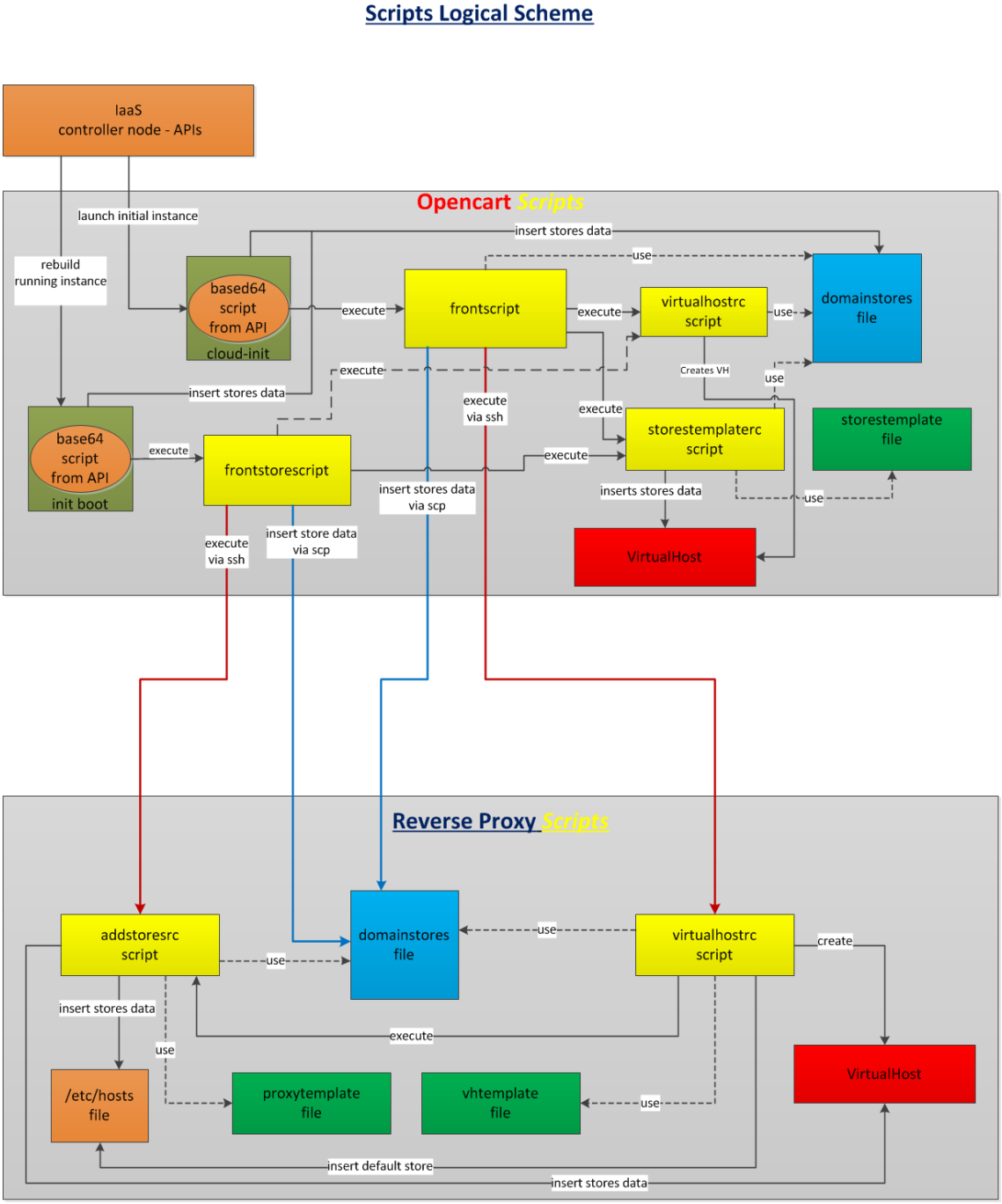


Fig. 5-15 Scripts logical flow

5. Once the above process is completed, another similar process triggers in the reverse proxy, to configure an access from the Internet to OpenCart stores. The different scripts in this case accomplish the following actions:

   - VirtualHost creation for a particular domain to be used. This includes the creation of the default store into the VirtualHost, the addition of proxy

directives for default store into VirtualHost, the addition of the domain and the IP address for the default store.

- Insert stores into the VirtualHost to allow the access through Apache directives (see example in Appendix C).

These scripts also use some templates to fill the data (Vhtemplate and Proxytemplate).

5.3.5.4 *Certification Authority (CA)*

As regards security, it is used SSL certificates signed by our own CA created inside the Reverse proxy, by means of a root certificate, avoiding external CAs. HTTPS between reverse proxy and Internet is provided (see Fig. 5-16). For simplicity's sake, the certificates are only retained in the reverse proxy, not affecting the security of the communications in the PaaS as they are performed through secured networks provided by OpenStack.

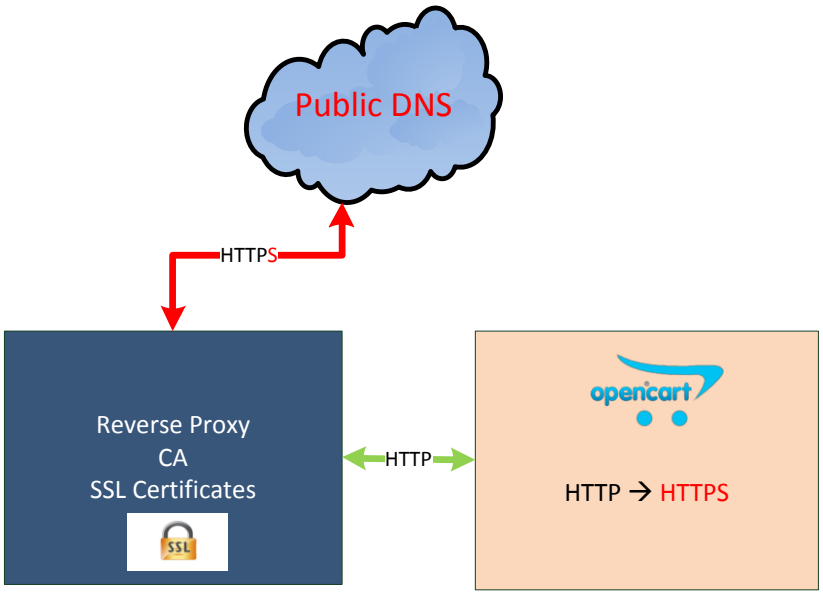


Fig. 5-16 Https access

For the creation of the certificates OpenSSL tools are used. Each domain has its own certificate.

5.3.6 Conclusions

The great potential of the Internet of Things (IoT) is widely known. To unlock its full potential in order to develop IoT solutions it is necessary to bring together connected devices and cloud computing.

Something like a universal plug-and-play to simplify programming and enable devices to be smarter is demanded from many forums. In this paper, a plug-and-play mechanism for IoT is presented and it is applied to retrofit a vending machine. Open software like OpenStack, OpenCart, Arduino . . . has been used in the implementation in order to get an affordable solution in terms of cost issues.

Following our prototyping phase, the next step in this project would be to demonstrate the benefits of our approach in a real case scenario and integrate big data analytics such as the algorithms described at [18] to benefit of having a single entry point towards our cloud-based platform through a unique domain. A first approach of this work can be found in [19].

5.3.7 Appendix A

Below it is the source code for an Arduino Mega board using as communication module a General Packet Radio Service (GPRS) shield connected on serial port number 2 of Arduino Mega board. In this example, once the device is powered on, it instantiates a virtual machine in a private cloud based on OpenStack Havana (Ubuntu 12.04). As prerequisite it is needed to generate a Unique Universal Identifier (UUID) to access to OpenStack RESTful interfaces.

```
/** Copyright 2014 Dpto. Informatica y Automatica, ETSI Informatica, UNED
Juan del Rosal 16, 28040 Madrid, Spain.
*
* Licensed under the EUPL, Version 1.1 or – as soon they will be
approved by the European Commission – subsequent versions of the
EUPL (the "Licence");* you may not use this work except in
compliance with the Licence.
* You may obtain a copy of the Licence at:
```

```
*
* http://www.osor.eu/eupl/european-union-public-licence-eupl-v.1.1
*
* Unless required by applicable law or agreed to in writing,
software distributed under the Licence is distributed on an "AS
IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
* See the Licence for the specific language governing permissions
and limitations under the Licence.
*
* Author: Mr. Antonio Solano Tarroc
* Directors: Dr. Natividad Duro, Dr. Raquel Dormido
* Date: 19 may 2014
* Published: http://blog.tarroc.com/?p=40
*/

#define UUID “45a884a2718f98e7addac7da2677a2fd” // replace your api key here

void setup()
{
  Serial.begin(9600);
  while (Serial.available())
  Serial.read();

  //GPRS
  Serial.println(F(“Initializing  GPRS …”));
  Serial2.begin(9600);  //GPRS Shield baud rate
  TcpSettings();
  delay(2000);
  if (CreateVirtualMachine)) Serial.println(F(” \n Virtual Machine created”));
}

void loop()
{
}
void ShowSerialData(){
 while(Serial2.available()){
   Serial.write(Serial2.read());
 }
}
void TcpSettings()
{
  Serial2.println(F(“AT+CREG?”)); //Check the registration status
  delay(300);
  ShowSerialData();

  Serial2.println(F(“AT+CGACT?”)); //Check attach status
```

```
  delay(300);
  ShowSerialData();

  Serial2.println(F("AT+CGATT=1")); // Attach from GPRS serivce
  delay(300);
  ShowSerialData();

  Serial2.println(F("WAIT=7")); // Wait for Attach
  delay(300);
  ShowSerialData();
  Serial2.println(F("AT+CSTT=\"movistar.es\"")); // Start task and set APN
  delay(300);
  ShowSerialData();

  Serial2.println(F("AT+CIICR")); // Bring up wireless connection with GPRS
  delay(300);
  ShowSerialData();
  Serial2.println(F("WAIT=6")); // Wait for Attach
  delay(300);
  ShowSerialData();
  Serial2.println(F("AT+CIFSR")); // Get local IP address
  delay(300);
  ShowSerialData();
  Serial2.println(F("AT+CIPSPRT=1")); // Set its prompt echo '>'and shows "send
ok" when sending is successful
  delay(300);
  ShowSerialData();
 Serial2.println(F("AT+CIPQSEND=0")); // Set normal data transmitting mode
  delay(300);
  ShowSerialData();
}
bool CreateVirtualMachine()
{
  Serial2.println(F("AT+CIPSTART=\"TCP\",\"myiaasopenstack.org\",\"8774\""));
//Start up TCP connection
  delay(2000);
  ShowSerialData();

  Serial2.println(F("AT+CIPSEND")); // Begin send data to platfrom through TCP
connection
  delay(300);
  ShowSerialData();
  // Request Syntax for TCP Socket Connection Type
   Serial2.println(F("POST
http://myiaasopenstack.org:8774/v2/483ff61f336e4ad9b4ba0b7c0bcd5a54/servers
HTTP/1.1"));
   Serial2.println(F("Content-Type: application/json"));
```

```
   Serial2.print(F("X-Auth-Token: "));
   Serial2.println(UUID);
   Serial2.println(F("Host: myiaasopenstack.org"));
   Serial2.println(F("Content-Length: 228"));
   Serial2.println(F(""));
   Serial2.print(F("{\"server\":{\"name\":\"arduino-test-api\","));
   Serial2.print(F("\"imageRef\":\"49dca96d-1b20-488c-add1-4d4aad801902\","));
   Serial2.print(F("\"flavorRef\":\"5ac7af64-f132-4c15-84e3-1080e8f25e0b\","));
   Serial2.print(F("\"key_name\":\"accesskeyTarroc\","));

Serial2.print(F("\"security_groups\":[{\"name\":\"webserver\"},{\"name\":\"defa
ult\"}]}}"));
   Serial2.println(F(""));
   Serial2.println(F(""));
   Serial2.print(0x1A, 0); //Arduino 1.0

   // End of Request Syntax
  // Serial2.println((char)26); // After sending AT+CIPSEND ,tap CTRL+Z to send
data

   delay(300);
   ShowSerialData();
   Serial2.println(F("AT+CIPCLOSE")); // Close TCP connection
  delay(300);
   ShowSerialData();
   return 1;
}
```

5.3.8 Appendix B

As outcome of the plug-and-play mechanism, VirtuaHost are created in the folder /etc/apache2/sites-available/ of the virtual machine where OpenCart resides. An example is showed below.

```
<VirtualHost *:80>

        ServerAdmin webmaster@localhost
        ServerName openvendshop.net
        DocumentRoot "/var/www/public_html/opencart/store"
        ErrorLog /var/log/apache2/error.log
   CustomLog /var/log/apache2/access.log combined

        <Directory "/var/www/public_html/opencart/store">
```

```
        Options -Indexes
        Order allow,deny
                Allow from All
                DirectoryIndex index.html index.php
                AllowOverride All
        </Directory>

        ProxyRequests Off

        SetOutputFilter SUBSTITUTE

        Substitute
"s|http://default.operadora|http://openvendshop.net/operadora|i"

        Substitute
"s|http://operadora01.operadora|http://openvendshop.net/operadora01|i"

        Substitute
"s|http://operadora02.operadora|http://openvendshop.net/operadora02|i"

</VirtualHost>
```

5.3.9 Appendix C

As outcome of the plug-and-play mechanism, VirtuaHost are created in the folder /etc/apache2/sites-available/ of the virtual machine where reverse proxy resides. An example is showed below.

```
<VirtualHost *:80>

                ServerAdmin webmaster@localhost
                ServerName openvendshop.net
                ErrorLog /var/log/apache2/error.log
                CustomLog /var/log/apache2/access.log combined

                <proxy>
                Order deny,allow
                Allow from all
                </proxy>

                ProxyRequests Off

                ProxyPass /operadora http://default.operadora
```

```
                ProxyPassReverse /operadora http://default.operadora

        ProxyPass /operadora01 http://operadora01.operadora
        ProxyPassReverse /operadora01 http://operadora01.operadora

                ProxyPass /operadora02 http://operadora02.operadora
        ProxyPassReverse /operadora02 http://operadora02.operadora

</VirtualHost>
```

5.3.10 References

[1] Kovatsch, M.; Mayer, S.; Ostermaier, B. Moving application logic from the firmware to the cloud: Towards the thin server architecture for the Internet of Things. In Proceedings of the 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Palermo, Italy, 4–6 July 2012; IEEE: New York, NY, USA, 2012.

[2] Parwekar, P. From Internet of Things towards cloud of things. In Proceedings of the 2011 2nd International Conference on Computer and Communication Technology (ICCCT), Allahabad, India, 15–17 September 2011; IEEE: New York, NY, USA, 2011.

[3] Zhou, J.; Leppänen, T.; Harjula, E.; Ylianttila, M.; Ojala, T.; Yu, C.; Jin, H. Cloudthings: A common architecture for integrating the Internet of things with cloud computing. In Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Whistler, BC, Canada, 27–29 June 2013; IEEE: New York, NY, USA, 2013.

[4] Shute, T. Pachube, Patching the Planet: Interview with Usman Haque. Available online: http://www.ugotrade.com/2009/01/28/pachube-patching-the-planet-interview-with-usman-haque/ (accessed on 15 August 2016).

[5] Mineraud, J.; Mazhelis, O.; Su, X.; Tarkoma, S. A gap analysis of Internet-of-Things platforms. *Comput. Commun.* **2016**, *89*, 5–16. [CrossRef]

[6] Perera, C.; Liu, C.H.; Jayawardena, S. The emerging Internet of things marketplace from an industrial perspective: A survey. *IEEE Trans. Emerg. Top. Comput.* **2015**, *3*, 585–598. [CrossRef]

[7] Yuan, Y.; Jia, K.-B. A Semi-supervised Approach for Water Quality Detection based on IoT Network. *J. Inf. Hiding Multimed. Signal Process.* **2016**, *7*, 858–866.

[8] OpenStack Documentation. Available online: http://docs.openstack.org (accessed on 24 June 2016).

[9] Wu, T.-Y.; Pan, J.-S.; Lin, C.-F. Improving Accessing Efficiency of Cloud Storage Based on De-duplication and Feedback Scheme. *IEEE Syst. J.* **2014**, *8*, 208–218. [CrossRef]

[10] OpenCart Documentation. Available online: http://docs.opencart.com/ (accessed on 24 June 2016).

[11] Huth, A.; Cebula, J. *The Basics of Cloud Computing*; Produced for US-CERT2011; Carnegie Mellon University: Pittsburgh, PA, USA.

[12] Yadav, S. Comparative study on open source software for cloud computing platform: Eucalyptus, openstack and opennebula. *Int. J. Eng. Sci.* **2013**, *3*, 51–54.

[13] The Physical Web—Google. Available online: https://google.github.io/physical-web/ (accessed on 24 June 2016).

[14] OpenVend SaaS Demo. Available online: http://openvend.es/demo/ (accessed on 24 June 2016).

[15] Arduino Documentation. Available online: https://arduino.cc (accessed on 24 June 2016).

[16] SLIM Documentation. Available online: http://slimframework.com (accessed on 24 June 2016).

[17] Sánchez, J.M. *Construyendo PaaS con Ubuntu y OpenStack para Internet de las Cosas*; UNED Final Project Report; UNED: Madrid, Spain, 2014.

[18] Wei, H.-W.; Wu, T.-Y.; Lee, W.-T.; Hsu, C.-W. Shareability and Locality Aware Scheduling Algorithm in Hadoop for Mobile Cloud Computing. *J. Inf. Hiding Multimed. Signal Process.* **2015**, *6*, 1215–1230.

[19] Martínez, P. *Big DataPara un Cloud Paas en Internet de las Cosas*; UNED Final Project Report; UNED: Madrid, Spain, 2015.

5.4 One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices

Abstract: The aim of this paper is to determine the physical proximity of connected things when they are accessed from a smartphone. Links between connected things and mobile communication devices are temporarily created by means of dynamic URLs (uniform resource locators) which may be easily discovered with pervasive short-range radio frequency technologies available on smartphones. In addition, a multi cross domain silent logging mechanism to allow people to interact with their surrounding connected things from their mobile communication devices is presented. The proposed mechanisms are based in web standards technologies, evolving our social network of Internet of Things towards the so-called Web of Things.

Keywords: Internet of Things; Web of Things; cloud computing; mobile communication systems; pervasive computing; authentication; web-based services

5.4.1 Introduction

Internet of Things (IoT) emerges as a paradigm to connect things to the Internet providing a bridge between the physical and digital worlds [1]. To this end, the uniform resource locator (URL) arises as an easy and pervasive way of using web technologies to point us towards the virtual representation of things in the Internet, called in this paper digital avatars.

Short-range radiofrequency technologies like near field communications (NFC), Bluetooth low power (BLE) or WiFi, are becoming pervasive technologies on our smartphones [2,3]. Such radiofrequency technologies allow us to discover URLs on the move and automatically initiate Internet connections by providing a physical web browsing experience. Some well known examples of how to facilitate access to specific web pages by using short range radio frequency technologies are:

- smart posters with passive NFC tags [4],
- iBeacons [5] technologies in brick and mortar stores,
- welcome web pages on free WiFi hostpots.

But how do we guarantee that we are accessing the URL of a virtual representation of a thing only when we are in proximity? Short-range radio frequency technologies facilitate the discovering of URLs, but do not guarantee we are located close to the associated thing. For example, the URLs can be retrieved from the browser's history on a mobile.

In this paper, a mechanism to tokenize URLs which are broadcast through open hardware devices [6] by using short-range radio frequency technologies is proposed. To this end, an enhanced approach based on one-time URLs that are "consumed" after usage is presented. In this approach, things are represented by dynamic URLs, which either will change once they are accessed or they will be renewed after a predefined timeout. As a result, reading this URL from a mobile phone will become a proof of close physical proximity. The tokenized URL is uniquely associated with the thing, which can be easily discovered by a smartphone enabled with short-range radio frequency technologies. If a user reopens the URL from the browser's history, and the proximity cannot be guaranteed by other means, the connection will be rejected.

The dynamic URLs are sent from the server to the connected things. These dynamic URLs are generated in a server on the cloud and are usually broadcasted by resource-constrained devices, such as UriBeacons (http://www.uribeacon.org), Dynamic NFC tags or peer to peer (P2P) NFC modules, which become electronic components of our connected things. It is necessary to implement a security mechanism to avoid any hacker from spoofing the URL and using it, or perform man-in-the-middle (MiM) attacks by changing the original URLs and redirecting us to fishing websites that are infected with malware. In this paper, an encryption mechanism to protect key information while using very limited and constrained electronic devices that do not allow us to apply standard HTTPS mechanisms to secure all information exchanged online is also proposed [7,8].

On the other hand, to fully interact with our nearby social things [9], it is not enough to discover the URLs of the things with our smartphone. At some point in time, it is possible that we may have to allow the thing to know who we are. Authentication is the process of verifying that “someone is who he claims to be”, and this is usually carried out via credentials: username and password. Following the authentication, it is the authorization to grant the thing access to end-user profile and/or vice versa. In order to guarantee a good end-user experience, the proposed design uses the minimal number of interactions to register and establish a dialogue between the user’s smartphone and the thing. A silent logging mechanism based in Cross Origin Resource Sharing (CORS) (http://www.w3.org/TR/cors/) and LocalStorage (http://www.w3.org/TR/webstorage) techniques, both available in HTML5 (http://www.w3.org/TR/html5/), is also introduced in this paper. The first time the user access the system, this mechanism will encrypt and store the user credentials on the browser’s memory. Next time the user initiates a browsing session towards any of the URLs published from our solution, silent logging process will happen in background and the links between our mobile smartphone and our nearby connected things will be established automatically, allowing us interact with the things through the Internet. The extensive usage of web technologies in our work, is our main contribution to so-called Web of Things (WoT).

All these introduced mechanisms have been implemented to provide a mobile payment method for unattended point of sale (PoS). Prior to deliver the goods or services, end-user proximity to PoS has to be verified and this is in fact the main challenge to be accomplished in our work.

To these ends, the paper is organized as follows: Section 2 introduces the motivation of presented work and a picture of our open hardware setup is included. Section 3 presents the end-user scenarios based on pervasive mobile short-range radio frequency technologies and suggests cost-effective devices to publish dynamic URLs for each respective technology. Section 4 describes the life cycle of tokens used to generate dynamic URLs. In Section 5, the proposed architecture based on cloud computing and open source software is presented.

Section 6 is devoted to analyze in deep the security issues. Section 7 aims to clarify all mechanisms presented with a call flow scenario. Section 8 provides some performance benchmarks of our open hardware prototype. Finally, Section 9 summarizes the principal contributions developed in this paper.

5.4.2 One-Time URL Motivation

The ultimate goal of our research is to provide an alternative mobile payment for unattended point of sales such as vending machines but it is applicable to toll gates, parking lots, etc. Similar to the Physical Web Google approach [10], each connected vending machine is identified by means of a URL. The URL points to a Web application (webapp), which is in fact the digital avatar of the vending machine. By accessing this URL from a smartphone, consumers will be able to select, from the touchscreen of their smartphones, any of the products of the vending machine and pay for it. Immediately the ordered product will be dispensed by the vending machine. Fig. 5-17 shows an Arduino open hardware prototype to achieve the described mobile payment scenario.

All modules highlighted in Fig. 5-17 are on-the-self products except the MDB/DEX module (multi-drop bus/digital exchange module) and the Keypad Hacking module. These two printed board circuits (PBCs) have been designed for interacting with vending machines in a fully contactless way, i.e., without physically touching the vending machine. This contactless innovation requires implementing a mechanism to guarantee that the consumer is physically close to the vending machine when products are ordered and dispensed. Collocate the end user and the vending machine, at the time the transaction occurs, is the challenge addressed in this paper. This challenge is solved if the only way to access the digital avatar of the vending machine is through the dynamic URL, and if this URL can only be consumed from the smartphone of an authorized end user.

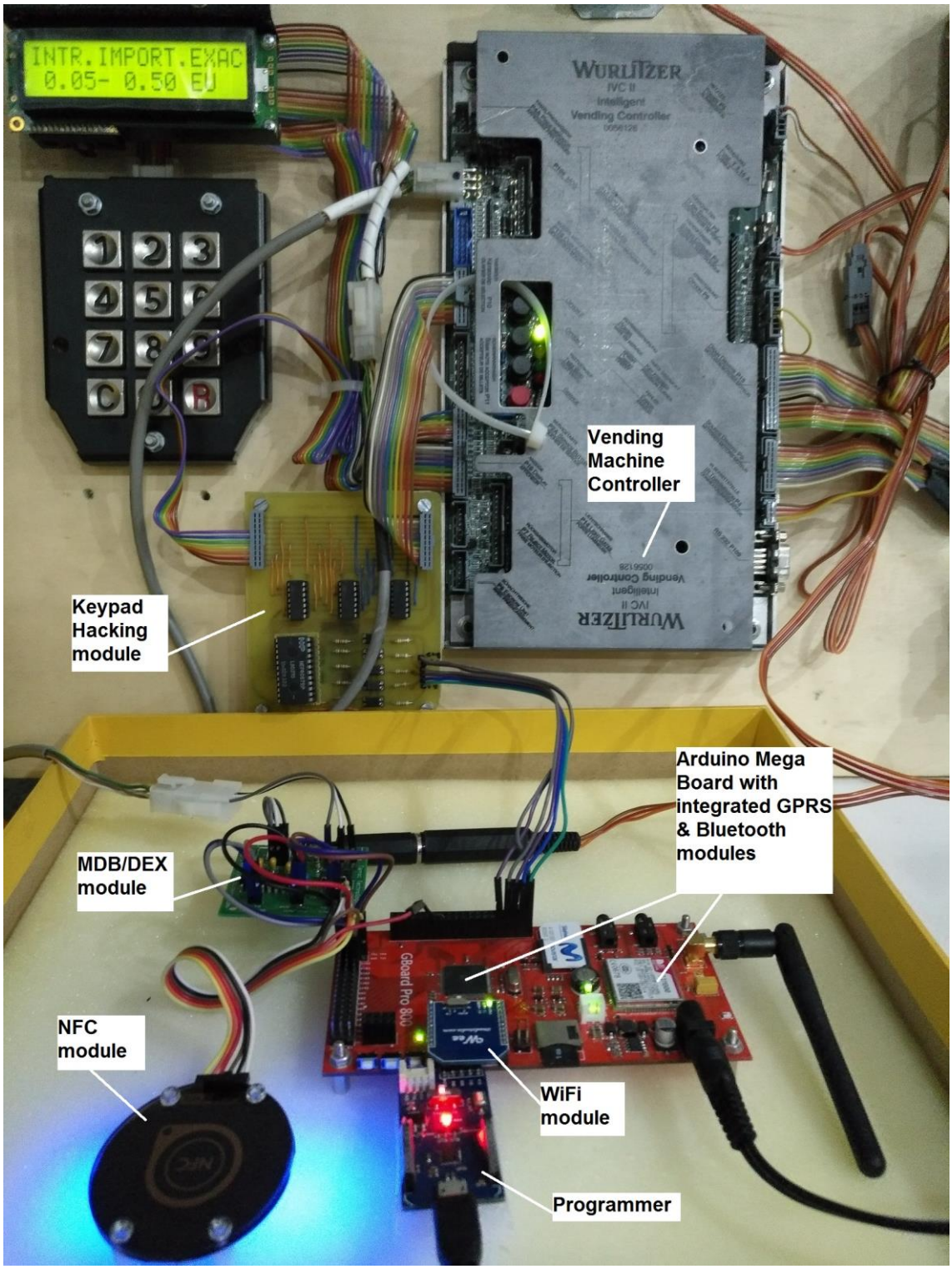


Fig. 5-17 Open hardware prototype

From the end-user perspective, the presented approach based on URLs changes completely the paradigm of having multiple native applications (apps) installed on smartphones which have to be downloaded previously. In the next section, how to interact with nearby things without needing the installation of

native apps is presented. As a key competitive contribution, this paper demonstrates that it is possible to provide webapps based on HTML5 with similar functionality than native apps. Indeed, as described in Section 5.4.6.2, the end-user purchasing experience is enhanced providing a universal method to grant access to any service provided by the system.

5.4.3 Proximity Scenarios Based on Short Range Wireless Technologies

The aim of this section is to define a user-friendly scenario for mobile users to discover and interact with connected things based on pervasive radio-frequency technologies available on smartphones such as WiFi, Bluetooth or NFC. Fig. 5-18 despites how to add communication modules to a simple 8 bits microcontroller.

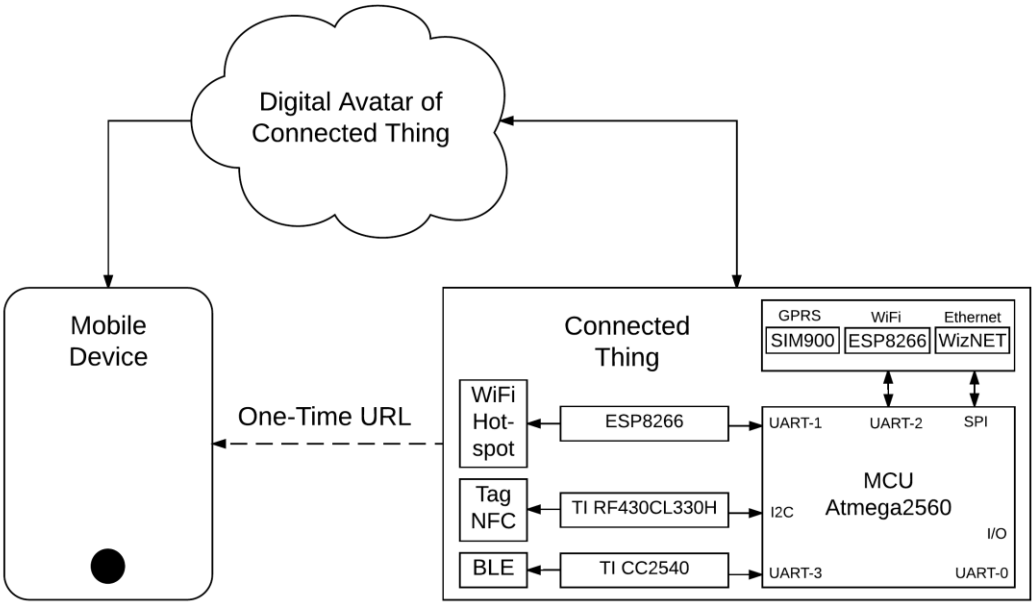


Fig. 5-18 Short range wireless technologies available on smartphones

In the following subsections, these three short-range radio-frequency technologies are introduced to achieve the same scenario: initiating a mobile

browsing session to a landing web page to interact with our surrounding connected things.

5.4.3.1 The Tap & Go Reference Model

Near field communications (NFC) is a radio-frequency identification (RFID) technology operating at 13.56 MHz that enables communication between devices that are held in close proximity. NFC forum introduced their first standardized technology architecture and standards for NFC compliant devices in June 2006.

NFC devices may be active or passive. A passive device, such as an NFC tag, contains information that other devices can read but does not read any information itself. An active NFC device, like a smartphone, would not only be able to read NFC tags, but configured in peer to peer (P2P) mode, it would also be able to exchange information with other compatible phones or devices and could even write or alter the information on the NFC tag, if authorized to make such changes. A NFC tag is a passive device with an EEPROM (electrically erasable programmable read-only memory) memory attached to the RFID inlays of the antenna. The power the tag needs to operate and communicate the information stored in the EEPROM, comes from the electromagnetic field generated by the emitter, usually a smartphone acting as NFC reader.

A dynamic NFC tag features an SRAM (static random access memory) bank that can be also accessed through a low-power Inter-Integrated Circuit (I^2C) interface and therefore easily be overwritten from any low cost microcontroller. A NFC-enabled smartphone, by default, is configured to initiate P2P communications or to read NFC tags. Therefore, taping a dynamic NFC Tag with a recorded URL will initiate a browsing session on the mobile device. For the proof of concept of our research we have implemented prototypes based on the cost-effective RF430CL330H chipset from Texas Instruments (Dallas, TX, USA) as it is shown in Fig. 5-17 and Fig. 5-18.

5.4.3.2 The Beacon Reference Model

iBeacon is a protocol standardized by Apple (Cupertino, CA, USA) and introduced in 2013. iBeacons uses Bluetooth low energy (BLE) proximity technology to broadcast their identifier to notify nearby devices of their presence. The technology enables smartphones, tablets and other devices to perform actions when in close proximity to an iBeacon.

In October 2014 Google (Mountain View, CA, USA) created UriBeacon, an open specification to connect Bluetooth low power beacons to the Web. Instead of broadcasting an identifier as iBeacon does, UriBeacons broadcast a URL. In 2015, UriBeacon evolved to become part of the Eddystone-URL (https://developers.google.com/beacons/), which was launched by Google in July 2015.

The TI CC2540 (Texas Instruments, Dallas, TX, USA) is a cost-effective, low-power, true system-on-chip (SoC) for Bluetooth low energy applications used in our open hardware prototypes as it is shown in Fig. 5-18.

5.4.3.3 The Free Hotspot Reference Model

Welcome web pages are commonly used when connecting to free WiFi hotspot. Nowadays we can also find cost-effective system-on-chip (SoC) for WiFi, for example the ESP8266 module, which may be used to create a bespoke hotspot as shown in Fig. 5-17 and Fig. 5-18.

5.4.3.4 Alternative Models Not Based on Short Range Wireless Technologies

It is worth mentioning an alternative mechanism to read URLs using a smartphone´s cameras. Dynamic URLs can be easily encoded on quick response (QR) codes or augmented reality (AR) makers that can be displayed on screens. But nowadays, cameras on smartphones are not natively processing the images to discover the QR codes or AR markers. So, it is necessary to use

an application to performance these actions. In a future, if these technologies based on image and patterns recognition are becoming pervasive technologies on smartphones, the presented approach on this paper should also apply.

5.4.4 Defining the Live Cycle of Dynamic URLs

The main idea behind our research goal is that one-time tokenized URLs, which are announced by connected things, may be used to determine the physical proximity of mobile communication devices. To tokenize a URL is just enough to add a token as parameter to the URL. The Fig. 5-19 depicts the life cycle of the token.

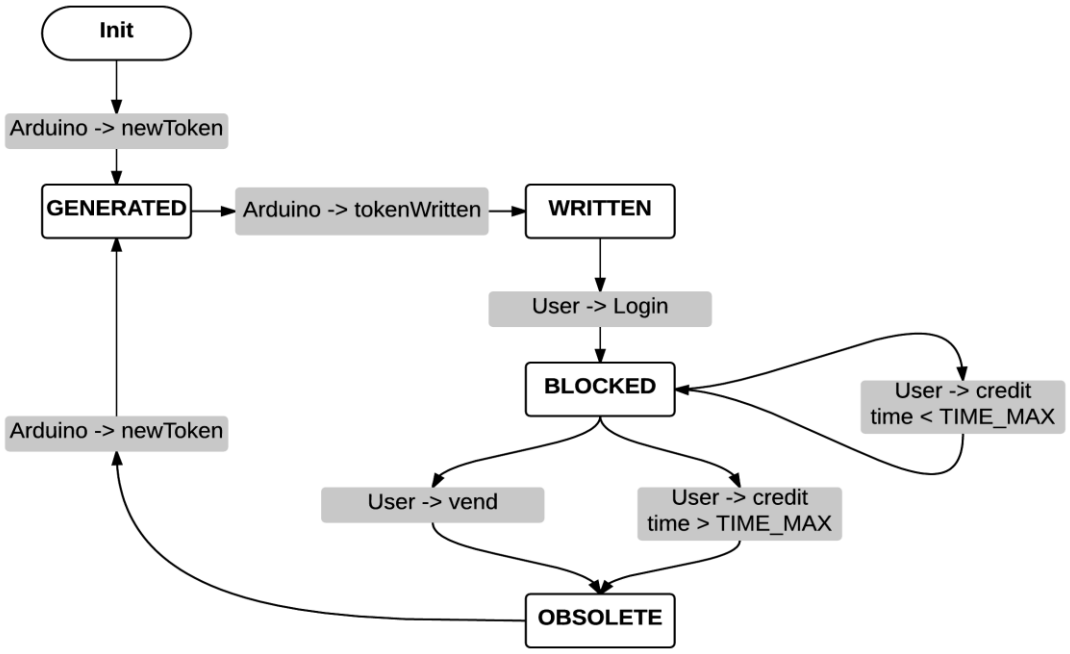


Fig. 5-19 Token life cycle

As it is shown, the life cycle of the token follows a state machine workflow in response to events coming either from the connected thing or from the end user mobile device. Furthermore, this approach allows to detect unusual situations and to implement easily corrective actions. For example, if there is a communication problem with the connected thing, the token in the server will be different than the one stored in the NFC tag. At this point, a cross-platform two-

factor authentication mechanism may be introduced as it will be described in Section 5.4.6.3. Tokens can be found in the following states:

- GENERATED: The token has been generated and sent to the connected thing but no confirmation of dynamic URL publication is yet received.
- WRITTEN: Connected thing has positively confirmed that the token has been written, for example, on the NFC tag.
- BLOCKED: A situation to avoid if a request for a new token arrives to the server while the end user is still interacting with the connected thing. For example, when a user initiates an e-commerce transaction with a connected point of sale, the current token (which will be in WRITTEN status) will change to BLOCKED state, not accepting new token requests until the end of the transaction.
- OBSOLETE: If a logged user does not finish the expected action or does not log out after a reasonable time, the token expires and becomes obsolete.

5.4.5 Underlying Cloud Based Architecture

Recently, many research works are interested in combining Cloud Computing and IoT [11]. Fig. 5-20 shows our end-to-end cloud based architecture. The approach is similar to the one followed in [12], but it benefits from the proximity-layer security mechanism presented in this paper.

The overall infrastructure consists of a private cloud, which uses COTS (Commercial Off-The-Shelf) bare-metal servers running OpenStack (https://www.openstack.org/) open source. A detailed description of the main building blocks is presented in the following subsections.

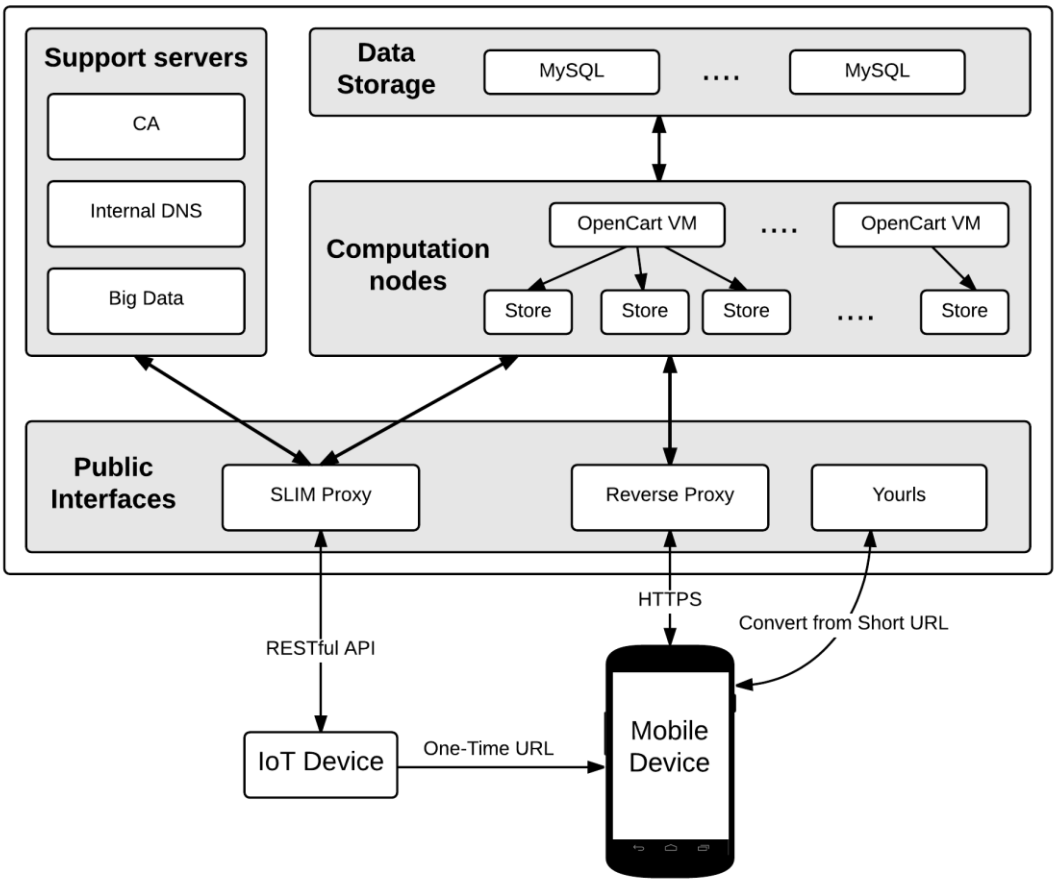


Fig. 5-20 Cloud based architecture

5.4.5.1 Slim Proxy

This is the interface to connect things with the platform. It is based on SLIM (http://www. slimframework.com/), a Representational State Transfer (RESTful) open source framework. It provides a REST-based interface to exchange JavaScript Object Notation (JSON) messages between IoT devices and the platform. The main roles of this server are described below.

a) Support a Plug-and-Play Mechanism

The very first time a physical object is connected, an ApiKey identifier is created and linked with the corresponding digital avatar. This digital avatar will be instantiated by the platform and being accessible through the aforementioned dynamic URL.

The process of building a digital avatar, by means of a virtual machine instance in OpenStack, consists of a series of steps executed in a specific order. The details of such plug-and-play mechanism can be found in [13].

b) Route Communications between Things and Their Digital Avatars

The SLIM proxy enables a translation table to map ApiKeys with assigned floating IPs in OpenStack. The ApiKey header is embedded in every HTTP request defined as a RESTful interface.

The JSON messages are found in the body of the RESTful interface. JSON messages will be routed from IoT devices to the right floating IP using the translation table and finally reach the virtual representation of the thing.

5.4.5.2 Reverse Proxy

On the other hand, in the proposed design, the fact of having to virtualize potentially any physical object requires that each connected thing has its own URL. Using different domains or subdomains could be an option. For example, https://mything.mydomain.com.

This option has to be configured in public Domain Name Servers (DNS) and requires some synchronization time before the URL can be reached from the Internet.

For the sake of simplicity, the proposed approach defines URLs for things based on a domain followed by a unique path to address the virtual representation of the thing. For example, https://mydomain.com/mything.

A standard Apache reverse proxy is used to route the URLs to the right floating private IPs and private subdomains where reside the web servers that serve these URLs. However, this system has as drawback that requires a script to be running during the initialization of the virtual machines in OpenStack as described in [13].

5.4.5.3 URL Shortener

This service is required for the tokenization of the URL and for the verification of physical proximity between connected things and mobile devices, which is our main contribution in this paper. Other motivations to introduce a shortener in the proposed design were to reduce the payload in the communications with our IoT devices and therefore the memory required to store and broadcast dynamic URLs.

Our solution is based on an open source called YOURLS (http://yourls.org/), which stands for Your Own URL Shortener. It is a small set of PHP (hypertext processor) scripts that allows running our own URL shortening service. The internal architecture of a shortener is quite simple. Any URL of any size generates a fixed size code that is stored in the database and it is associated with the URL. Some shorteners allow choosing the short code while others simply generate a pseudorandom sequence of fixed size. In our design, the generated tokens are pseudorandom sequences of 36 characters that are added to the URL, for example, https://mydomain.com/myoperator/mything/?route=module/openvend&token=HPLoMLNzQuQdVFe-wSv9Pg.

YOURLS codes the long URLs in a sequence of six characters by taking the first characters of the generated token. As an outcome of this process, the SLIM server composes a JSON message, like {“u”: “HPLoML”}, which will be populated to the IoT device to compose a short URL for example, https://myshortener.cc/HPLoML.

When YOURLS server receives this request, it extracts the short code “HPLoML” and it searches in its database for the long URL that corresponds to this code. Then, it performs a redirection to the page pointed by this URL in a completely transparent manner for the end user. YOURLS has many advantages, but the most determining factors when selecting it for using on this work are the following:

- It is a free and open source, which is a prerequisite for all software used in our design.
- Easy installation on our own private cloud, without limits on number of shortened URLs.
- It generates sequential keys or allows customized keys, allowing us to generate keys randomly by a hash function.
- Availability of plugins to enhance functionality. In our design, it is necessary to eliminate short URLs from the database once they have been used (as they are for one time use). This feature is not available in the default installation of YOURLS, but can be added using the plugin API (application programming interface).

5.4.5.4 Other Servers

In previous sections the key servers needed in our architecture to implement the proposed proximity mechanism have been detailed. In this section it is introduced, at a very high level, the rest of servers presented in Fig. 5-20 to despite the overall architecture. These are the following:

- Application Web Servers: They provide the virtual representation of the connected things. As shown in Fig. 5-20 OpenCart (http://www.opencart.com/) is the open source selected in our real deployment to virtualize unattended point of sales.
- MySQL (https://www.mysql.com/) Servers: This is a good design practice in order to dissociate the compute nodes and the data storage nodes in a Cloud infrastructure based on OpenStack.
- Certificate Authority (CA): Security requirements force to perform communications using HTTPS which require digital certificates in our servers. In order to maintain a low-cost system, we create our own Certificate Authority using OpenSSL (https://www.openssl.org/).
- DNS Server: Currently URL addressing is done through the reverse proxy, but to provide high scalability, an internal DNS it is considered in the architecture. The selected open source is BIND (http://www.isc.org/).

- Big Data: One of the new areas being followed now in this research project is the massive analysis of the data generated. To do so, we will require new servers whose task will be the collection and analysis of all transactions taking place on the platform.

5.4.6 Security by Design

Designing a secured proximity layer system to interact with connected things from mobile devices requires the analysis of security from different perspectives as described below.

5.4.6.1 Unauthorized Users: Encryption of User Credentials on Smartphones

To guarantee a good end-user experience our design requires a minimum number of interactions between the user and the terminal. Unfortunately, this requirement collides with the security requirements.

In order to achieve this goal, the local Storage of the HTML5 web browser is used to store the user credentials to allow a silent logging mechanism. The drawback of this solution is that anyone who has access to the smartphone can get these credentials unless they are encrypted. The solution includes the concept of one time token which is consumed after each logging as described below.

1. The username and password are stored in the server encrypted.
2. The encryption key is stored in the localStorage of the smartphone browser, along with one-time token generated by the server. Fig 5-21 is a simplified sequence diagram of the registration process which only requires as input a valid end-user email.

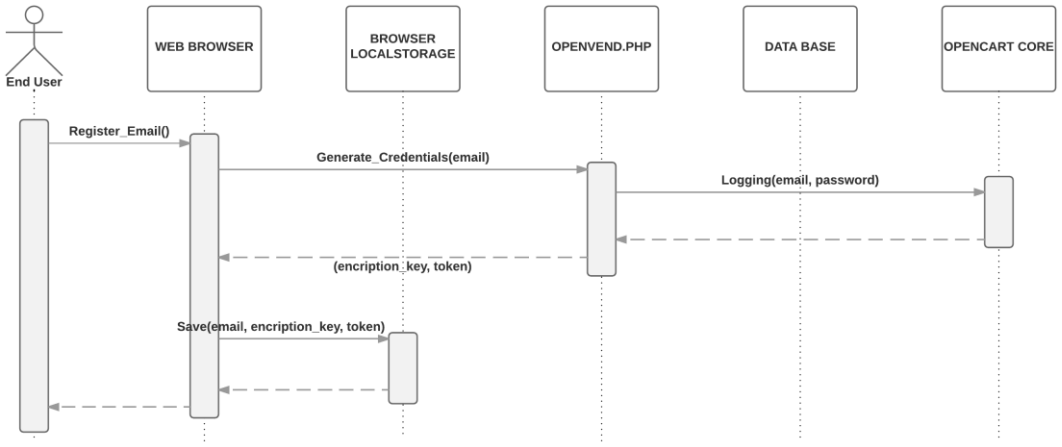


Fig. 5-21 Registration process with end-user email and one-time token

3. Next time the user is going to perform a purchase; a silent login mechanism will retrieve the credentials as show in Fig. 5-22. Both, the encryption key and the token, are sent to the server. The token is checked and if it is correct, the key is used to decrypt the user credentials to log into the server. The encryption key is not kept in the server after it is used.
4. A new token is generated and sent to the user's smartphone.

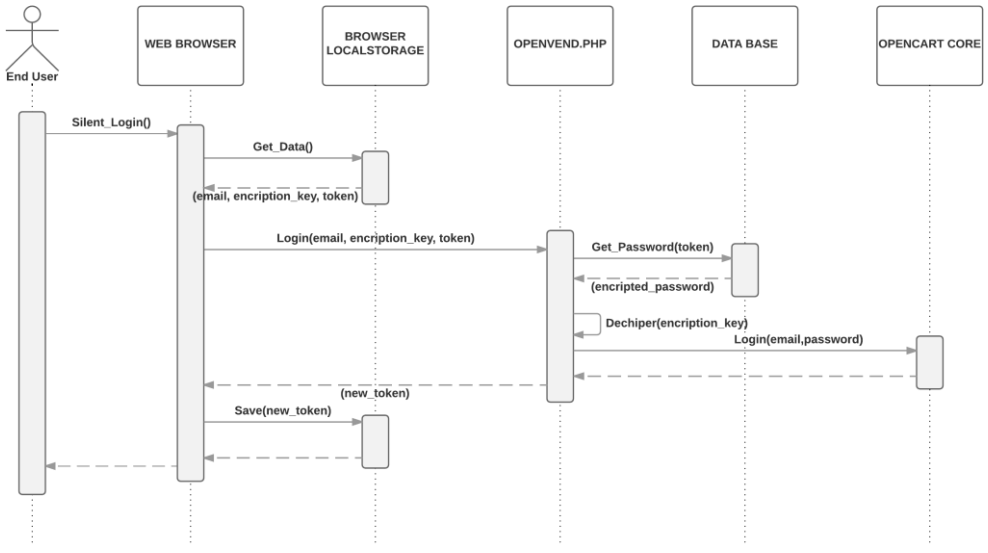


Fig. 5-22 Silent logging process and credentials tokenization

The advantage of this mechanism is that, if the smartphone is compromised, the attacker can only get access to the key to decipher the

credentials, but not to the credentials themselves (which are stored and encrypted in the server). The one-time token guarantees that if an attacker gets the key, she/he can only access the system once, but still, this access can be identified as an intrusion and be notified to user´s email. A functional demo to show the registration process and silent logging mechanism is available on [14].

5.4.6.2 Cross Origin Resource Sharing (CORS)

For security and privacy reasons Web browsers prevent documents in different domains from affecting each other; that is, cross-site scripting is disabled. While this is an important security feature, it prevents pages from different domains or subdomains access to data stored on the local storage of the HTML5 web browser. In Section 5.4.6.1 how to store the user credentials encrypted on the localStorage with the aim to have a silent logging mechanism for all connected things is described, but if connected things are addressed by URLs using subdomains, as OpenCart internally does, the silent logging mechanism will not work properly. The method used to circumvent this problem is based in CORS, a cross-document messaging HTML5 API that provides a mechanism that allows the interchange of messages between documents hosted in different domains or subdomains.

The main idea behind CORS is to create an HTML document hosted in a public domain, such as the one configured in our SLIM proxy that acts as a server. The only job of this document is to handle all the operations related to the localStorage. Every webapp, regardless of the domain in which it is hosted, sends a message to the server requesting the read or write operation over the localStorage and the server is the one that performs it, returning to the webapp the result of the operation. This means that the owner of the localStorage is the domain that hosts the server document and the only way to access it is through the server document.

In order to avoid unauthorized requests from other domains, the server stores a whitelist of domains that are allowed to perform operations in the localStorage. The other file needed for the implementation of CORS provides the

JavaScript supporting the functionality needed in the webapp side. Its first task is to create an HTML iframe that points to the server file.

Once this operation has been done, this file provides the functions to read and write in the localStorage. These functions, invoked from the webapp, resend the request to the server file using the cross-document messaging. That is, the function is called from a webapp hosted in any domain included in the whitelist, but it is redirected to the server which is the owner of the localStorage and the one that performs the operations over it.

In other words, using both the CORS mechanism and the silent logging mechanism presented in Subsection 5.4.6.1, results in the possibility to have a whitelist of URLs in different domains which may be accessed by the same end user. For instance, if domain A belongs to a service provider and domain B belongs to a second service provider, the end user will access both domains if previously service providers agreed to share the end-user credentials. Potentially, the presented work provides a universal method to grant access to any service provided by the system.

5.4.6.3 Enhancing Security with a Cross-Platform Two-Factor Authentication Mechanism

The silent logging and dynamic URL mechanisms are happening in the background to enhance the end-user experience, but sometimes it may be required to request explicit actions to end user, for example when performing in-store high value ecommerce transactions or when dynamic URLs are corrupted.

On such scenarios additional proximity mechanism may be implemented at check-out time. For instance, a personal identification number (PIN) may be generated randomly for each transaction and shown in the display of the connected point of sale. The end user has to enter this PIN into the webapp to validate the transaction.

This mechanism reinforces the collocation of the user with our nearby-connected things, but also permits the usage of other “discovery” technologies that use static URLs such as: quick response (QR) codes (see demo [14]), augmented reality (AR) markers, webapp shortcuts on mobile devices, etc.

Alternatively, the presented two-factor authentication mechanism can be used to pre-order products remotely on unattended point of sales. In such scenarios the consumer will receive a PIN code on his mobile phone. To dispense the pre-ordered product, consumer only has to type the PIN on the keypad of the unattended point of sales, which becomes a proof of the consumer´s presence.

5.4.6.4 Secure Web Communications from Mobile: Adding Secure Socket Layer (SSL)

As our main interface to interact with the thing is a smartphone provided with a modern web browser, the digital representation of the thing will be in the form of a webapp. Nowadays, from an end-user experience perspective, webapps are at same level of native apps for many scenarios thanks to HTML5 standards. Our webapps use HTTPS to secure all information exchanged online.

5.4.6.5 Secure IoT Communications: Following Lightweight Machine-to-Machine (LWM2M) Security Model

To secure the IoT device communications, our designis guided by LWM2 (http://openmobilealliance.hs-sites.com/lightweight-m2m-specification-from-oma) recommendations and uses pre-shared keys (PSKs). PSK is preferred to other mechanisms also proposed by LWM2M for very constrained devices with limited computing and memory resources. This design decision implies that the same PSKs and PSK IDs need to be generated and installed on the IoT device and on the backend. LWM2M proposes two modes:

- TLS_PSK_WITH_AES_128_CCM_8,
- TLS_PSK_WITH_AES_128_CBC_SHA256.

Both options provide all the requirements of a cryptographic system: confidentiality, authentication and data integrity. In our proposal it is selected the second mode considering the availability of open source libraries from Arduino (https://www.arduino.cc/) community. Table 5-3 shows some benchmarks of test performed with an Arduino Mega board and open source libraries detailed in Appendix (5.4.10).

Table 5-3 AES Processing time with Arduino Mega.

Text to Encript (bytes)	Key Length	Operation	Processing Time (s)
16	128	Encryption	0.51
16	128	Decryption	0.59
16	192	Encryption	0.65
16	192	Decryption	0.69
16	256	Encryption	0.72
16	256	Decryption	0.78

When a message is encrypted the result is a random sequence of bits. This result has to be encoded into printable characters to be sent in the message. There are two options for this: hexadecimal or American Standard Code for Information Interchange (ASCII) characters. Hexadecimal encodes each byte of information as two bytes. ASCII uses the Base64 encoding where 3 bytes are encoded in 4 bytes. Due to the resource-constrained memory of our IoT devices the Base64 encoding (lower overhead) has been selected in our design.

In our API RESTful design, sensitive information is only transmitted in the body of the HTTP requests so it has been decided to encrypt the JSON messages located on the body. After making these decisions, all the steps to encrypt the original message and validate it with the HMAC function are showed in the Fig. 5-23. In Fig. 5-23 the following protocol is established to protect communications between Arduino and the SLIM server:

1. SLIM Arduino server shares two private keys: one for the HMAC function and one for the encryption algorithm. Our IoT devices are resource-constrained. It has been discarded to implement a key exchange protocol session. Therefore, these keys are pre-set in advance, both in the SLIM Proxy and in the IoT device.

2. Any message sent must attach a timestamp in the message body JSON. This decision is made for two reasons: first, it allows our IoT device to be synchronized with an external time reference despite it not always having an internal clock, and second and more importantly, it avoids man in the middle (MiM) attacks via a message replacement. To explain this type of attack let's suppose, for example, an http request to transfer money. The body of this call (if we do not add a timestamp) will have a JSON message, such as: {"credit": 2000}, which indicates 20 Euros. By encrypting this body, since the key is always the same, the encrypted message will be always the same. This will invite to a hypothetical hacker to perform the following operation:

   - Make a monetary transaction for a very high amount and intercepts the encrypted message (eavesdropping).
   - Back and perform an operation for a very low amount, then intercept the message and replace it by the one intercepted previously with a higher amount.
   - The SLIM server will receive the second message with the body of the first, and once correctly decrypted, the monetary transaction will be executed with the high amount.

   As it can be seen for this type of attack, it is not necessary that the attacker be able to decrypt the message, it is enough to replace it. By adding a timestamp, we pseudo-randomize the encrypted message and once decrypted we know at what time the message was sent, so in the above example, the SLIM server will discard the second message for being too old.
3. Because messages can be of variable length, a fill (padding) to reach a length which is a multiple of the encryption algorithm block size has to be applied; in this case, a multiple of 16 bytes. There are different standards that determine how to apply padding to a plain text:

   - Space padding, which consist to fill with spaces.
   - ISO/IEC 7816 which determines that the first byte of the fill should be 0x80 (128 in decimal value) and the remaining bytes are filled with the value 0x00.

- Public Key Cryptography Standards (PKCS7), which states that each padding bytes should have, as value, the number of bytes to be filled. For example, if five bytes are needed, those 5 bytes will have the value 0x05.
- ANSI X.923, indicating that all the padding bytes must have the value 0x00, except the last, whose value is the number of bytes to be filled.
- ISO/IEC 9797 is not set as a standard for encryption algorithms but used for hash and MAC functions. In this case, all the padding bytes have the value 0x00.

4. The use of either mechanism can influence depending on the original text. It is important to avoid ambiguity decrypting the message because there may be problems in distinguishing padding from message bytes. This is especially important when the original message is in binary as any value is possible in a byte. In our particular case the messages to encrypt are in plain text, and they have a clear defined structure: a JSON string begins with left brace and ends with right brace, so the choice of any standard is not particularly relevant. Finally, although in the SLIM server all available methods are deployed, the ISO/IEC 9797 has been chosen because is the easiest to implement on the IoT device.
5. Once the message has filled, the encryption algorithm is applied to the message using the key shared between sender and receiver. The encryption mode (Cipher Block Chaining or CBC) requires an initialization vector (IV), which is generated randomly.
6. The resulting encrypted text message is sent to the HMAC (hash message authentication code) function together with the key shared between the IoT device and the SLIM server.
7. To the output of the HMAC function the initialization vector is added, which is needed to decrypt the message at destiny and also the encrypted message.
8. Finally, since the generated message is in binary format, a Base64 encoding is applied to represent it by ASCII characters. This process consists of dividing the binary message in blocks of 6 bits and assigning to each of these blocks (which may have 64 different values) a character. The characters

used are uppercase letters, lowercase letters, numeric digits and the symbols (+) and slash (/). On the receiver side it is necessary to perform the same steps in reverse order to obtain the original message.

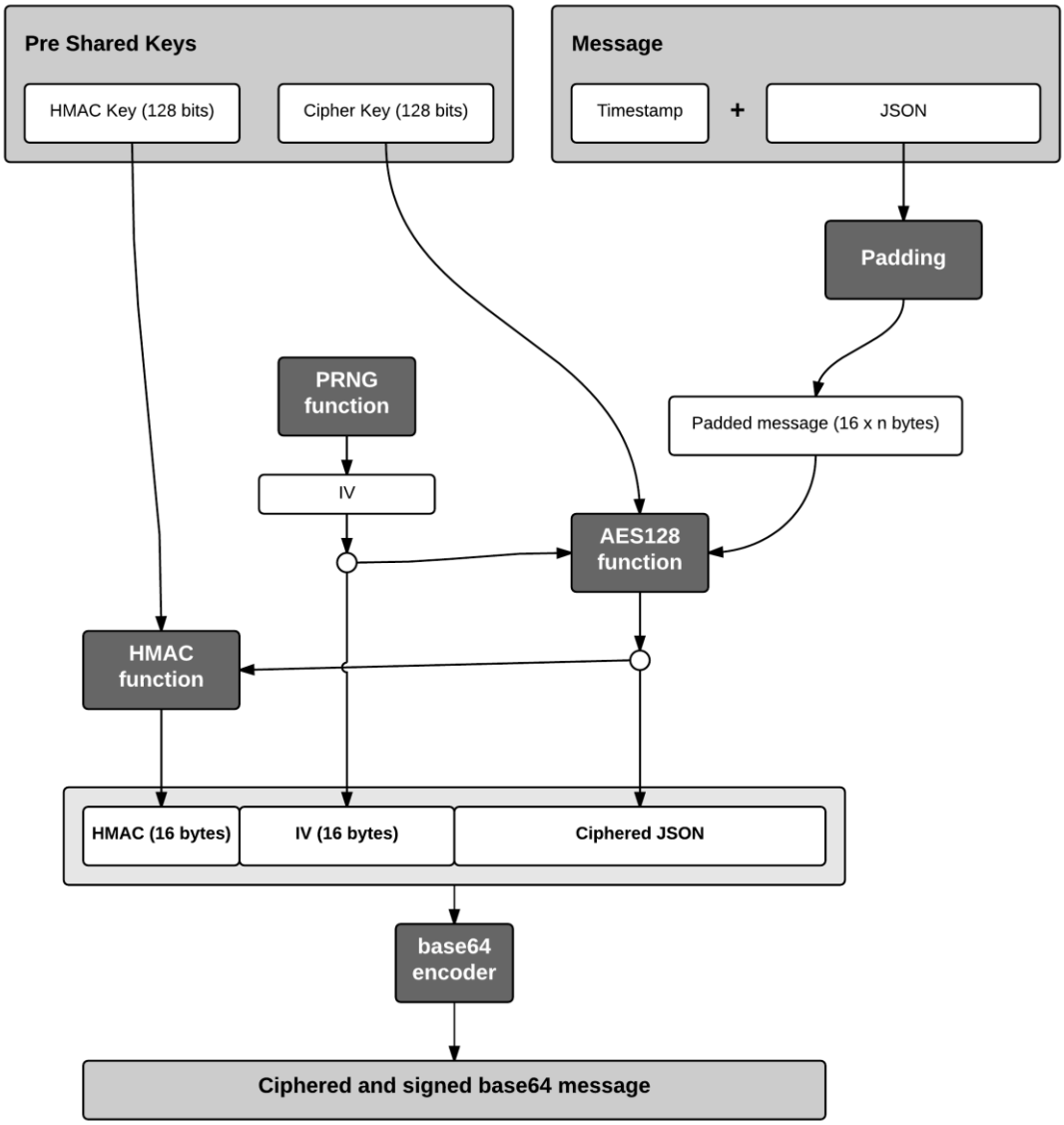


Fig. 5-23 Generation of the encrypted message

5.4.7 Call Flow Scenario

In the Section 5.4.5, the underlying cloud architecture is presented with the aim to describe the main building blocks involved in the tokenization of URLs. Besides, in the Section 5.4.6, a silent multi-domain logging mechanism was introduced to secure the access to the generated one-time URLs. Fig. 5-24 shows

a call flow diagram of the process of ordering a product in our retrofitted vending machine (see Fig. 5-17). This diagram combines the different mechanisms introduced in this paper.

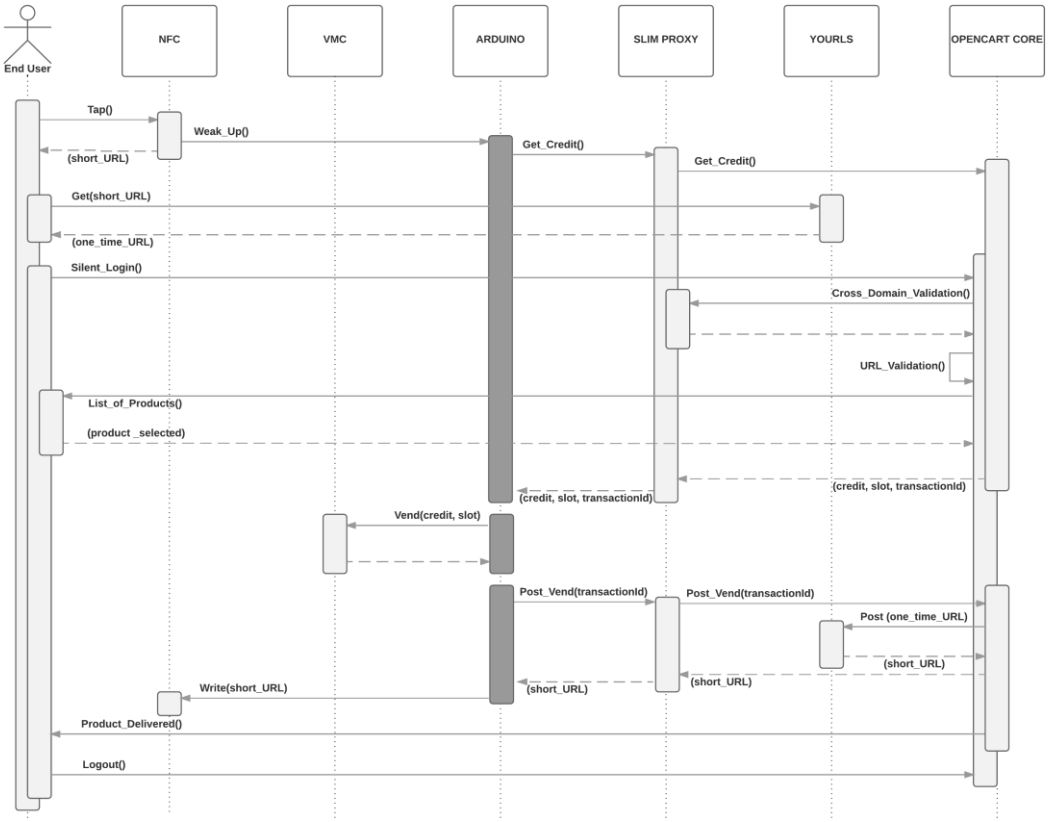


Fig. 5-24 Call flow example of usage of one-time URL in a vending machine

The scenario starts when the end user provided with a NFC-enabled smartphone taps the NFC module (see Fig. 5-17). The NFC module will launch an interruption to Arduino (see Fig. 5-17) to let him know that someone is reading the short URL. Then, Arduino will query the shopping cart of Opencart (see Fig. 5-20) and wait for the details of the ordered product. Section 5.4.5.1 presents all the communications between Arduino and Opencart. These communications are routed through the SLIM Proxy (see Fig. 5-20).

Coming back to the first step, when the end user taps the NFC module, the browser of the smartphone will retrieve the one-time URL from YOURLS, the URL shortener (see Fig. 5-20) and automatically, logging into the vending machine webapp as it is described in the Fig. 5-22. The access to the webapp

will be granted with the cross-domain mechanism described in the Section 5.4.6.2.

In the second step, the end user will buy online the desired product selecting it on the touch screen of his smartphone (see demo [14]) and OpenCart will reply to Arduino, which was waiting for the ordered product details. Arduino will start a dialogue with the vending machine controller (VMC) (see Fig. 5-17) and dispense the ordered product. Arduino will confirm to OpenCart that the product has been delivered and will retrieve a new one-time URL from the server which will be finally announced through the NFC module. As the final step, the end user will be notified on his smartphone that the product has been delivered and the web session will be closed.

5.4.8 Open Hardware Prototype and Preliminary Benchmarks

Arduino compatible electronic components introduced in Section 5.4.3 have been selected to build our open hardware prototype.

Fig. 5-25 Arduino Mega compatible prototype for vending machines

As reference, Fig. 5-25 shows one of our bespoke Arduino Mega open hardware design. This board is powered by the Multidrop Bus Standard (MDB)

interface of the vending machine and it is able to communicate with our cloud solution via WiFi, and to retrieve the one-time URL which is announced via NFC.

To retrofit a vending machine, we need a plug-and-play device that can be turned ON and OFF at any time. Arduino is a simple microcontroller which does not have to be properly shut down before turning OFF the power. Arduino is also well suited for repeated type of works like keep alive the slave-master synchronization mechanism, which requires sending every 200 ms acknowledge messages to the vending machine controller. As disadvantage, Arduino is not capable of doing multiple tasks at a time, like a computer as Raspberry-Pi does. To overcome such limitations, we programed a pseudo multitask mechanisms consisting of:

(a) Interrupts every 200 ms, to keep alive the master-slave synchronization with the VMC.

(b) Finite-State Machine (FSM) computation models, to handle the call flows with the back-end servers and to interact with other peripheral devices such as NFC modules, hacked vending machine keypad, etc.

To improve the response time using FSM handlers, it is important not to lose cycles of microcontroller, therefore we avoided the usage of the directive delay(). Unfortunately, this is a directive commonly used in Arduino libraries, forcing us to discard them and code at a very low level. These obstacles become benefits thanks to Arduino community best practices. We learned how to optimize and control the memory of our code, resulting in a firmware below the 60 kilobytes, and a stable dynamic memory allocation of 6331 bytes. These facts are observed after running a stress test of three hundred consecutive orders, as shown the Table 5-4. These achievements ensured the robustness of our open hardware design.

Table 5-4 Performance test result.

Test	Count	Minimum (ms)	Maximum (ms)	Average (ms)
Cycles	300	7918	13,457	9049
GET	295 *	2235	5189	2972
VEND	295 *	1499	1501	1499
POST	295 *	4140	6994	4565
FREE Memory Allocation	300	6331	6331	6331

* Difference due to 5 timeouts from server.

The GET, VEND and POST rows of Table 5-4 refer respectively to Get_Credit(), Vend(credit,slot) and Post_Vend(transactionId) calls initiated by Arduino as Fig. 5-24 shows. Two main calls are required to complete an order. First, Arduino performs a GET RESTful call and once the product has been selected on the smartphone, Arduino gets the ordered product details, including the price and the slot on which the product is located in the vending machine. Gathering such info takes 2972 ms. At this moment the vending machine performs the VEND in 1499 ms and the product is delivered. In other words, the product is delivered in 4471 ms providing a good purchasing experience. To complete the ordering cycle, a second POST RESTful call is performed to update the system with the renewed URL and to inform the user that the order has been successfully been delivered. The POST query requires an additional 4565 ms, which overlaps with the time used by the consumer to collect the product from the vending machine. As outcome, the overall cycle is completed in 9049 ms and the system is ready to serve new orders with the renewed one-time URL.

5.4.9 Conclusions

The great potential of the Internet of Things (IoT) is widely known [15]. To unlock its full potential in order to develop IoT solutions it is necessary to bring together connected devices and web standards, conforming the new paradigm of the Web of Things (WoT).

The initial motivation of this research was to provide an alternative mobile payment for unattended point of sales just using web standards technologies, without the need to install any native application or mobile wallet. As payment systems deal with sensitive and private data, it makes security to be an integral part of presented work. This paper presents a proximity application layer security

mechanism for the WoT applied to unattended point of sales. It is shown the complete design of the secured proximity layer system to interact with connected things from mobile devices. This approach includes cryptography and tokenization for low-resource devices to ensure its security. Specifically the mechanism to tokenize the URLs, which uniquely identify the thing with its digital avatar. These URLs are broadcasted through pervasive short range radio-frequency technologies available on smartphones.

It has also been detailed how the system developed has been integrated as a part of an end-to-end cloud architecture for unattended point of sales to enable proximity mobile payments. At this point, it is worth it to mention a successful payment mechanism based on tokenized QRs that WeChat [16] has introduced recently in China. Conceptually the presented one-time URLs can be converted to tokenized QRs and be easily displayed in any machine with a screen. Reading these QRs will point consumers to our one-time URLs, so possibilities are endless for disrupting the mobile payment ecosystem of unattended point of sales and by extension to any point of sales.

Moreover, other aspects of the security system developed have been analyzed. These aspects include encryption of the user credentials on mobiles, the verification of multiple origins and a two factor authentication mechanism among others. Standardized secured mechanisms of communication have been used always looking for availability of open source libraries.

Finally, some performance benchmarks have been presented to validate the robustness of our open hardware designs and the overall purchasing experience. The next step in this project would be to demonstrate the benefits of our approach in a real case scenario.

5.4.10 Appendix

Below is the source code for the AES performance test showed in Table 5-4 on Arduino Mega. The library used can be downloaded from github [17].

Algorithm 1: Source code for the AES algorithm on Arduino Mega

```
#include <AESLib.h>
1: void setup(){
2: Serial.begin(57600);
3: uint8_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
4: char data[] = "0123456789012345"; //16 chars == 16 bytes
5: aes128_enc_single(key, data);
6: Serial.print("encrypted:");
7: Serial.println(data);
8: aes128_dec_single(key, data);
9: Serial.print("decrypted:");
10: Serial.println(data);
11: }
12: void loop() {}
```

5.4.11 References

[1] Miorandi, D.; Sicari, S.; De Pellegrini, F.; Chlamtac, I. Internet of things: Vision, applications and research challenges. *Ad Hoc Netw.* **2012**, *10*, 1497–1516. [CrossRef]

[2] Want, R.; Schilit, B.N.; Jenson, S. Enabling the Internet of Things. Available online: http://citeseerx.ist.psu. edu/viewdoc/download?doi=10.1.1.698.6666&rep=rep1&type=pdf (accessed on 11 September 2016).

[3] Tanyeri, G.; Messiter, T.; Beckett, P.; Matthews, G. Short Range Wireless Technologies BLE, Bluetooth and WiFi Are an Essential Part of Any IoT Effort. Available online: http://www.clarinox.com/docs/whitepapers/ Short_Range_Wireless_in_IoT.pdf (accessed on 11 September 2016).

[4] Coskun, V.; Ozdenizci, B.; Ok, K. A survey on near field communication (NFC) technology. *Wirel. Pers. Commun.* **2013**, *71*, 2259–2294. [CrossRef]

[5] Burzacca, P.; Mircoli, M.; Mitolo, S.; Polzonetti, A. “iBeacon” technology that will make possible Internet of Things. In Proceedings of the International Conference on Software Intelligence Technologies and Applications & International Conference on Frontiers of Internet of Things 2014, Hsinchu, Taiwan, 4–6 December 2014; pp. 159–165.

[6] Fisher, R.; Ledwaba, L.; Hancke, G.; Kruger, C. Open hardware: A role to play in wireless sensor networks? *Sensors* **2015**, *15*, 6818–6844. [CrossRef] [PubMed]

[7] Nguyen, K.T.; Laurent, M.; Oualha, N. Survey on secure communication protocols for the Internet of Things. *Ad Hoc Netw.* **2015**, *32-C*, 17–31. [CrossRef]

[8] Roman, R.; Najera, P.; Lopez, J. Securing the Internet of things. *Computer* **2011**, *44*, 51–58. [CrossRef]

[9] Atzori, L.; Iera, A.; Morabito, G.; Nitti, M. The social Internet of things (SIoT)—When social networks meet the Internet of things: Concept, architecture and network characterization. *Comput. Netw.* **2012**, *56*, 3594–3608. [CrossRef]

[10] Google. The Physical Web. Available online: https://google.github.io/physical-web/ (accessed on 31 July 2016).

[11] Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]

[12] Le Vinh, T.; Bouzefrane, S.; Farinone, J.M.; Attar, A.; Kennedy, B.P. Middleware to integrate mobile devices, sensors and cloud computing. *Procedia Comput. Sci.* **2015**, *52*, 234–243. [CrossRef]

[13] Solano, A.; Dormido, R.; Duro, N.; Sánchez, J.M. A self-provisioning mechanism in OpenStack for IoT devices. *Sensors* **2016**, *16*, 1306. [CrossRef] [PubMed]

[14] OpenVend Demo. Available online: http://openvend.es/demo/ (accessed on 31 July 2016).

[15] Atzori, L.; Iera, A.; Morabito, G. The Internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]

[16] Huang, W.; Tang, J. Explore the development of WeChat payment from user behavior. In Proceedings of the 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Singapore, 6–9 December 2015.

[17] Arduino AES Library. Available online: https://github.com/DavyLandman/AESLib (accessed on 31 July 2016).

6 CONCLUSIONES Y APORTACIONES

Iniciar una tesis a más de cuatro años vista y plantearse el uso de tecnologías pervasivas como prerrequisito para realizar un prototipo mínimamente funcional, tiene un alto riesgo de obsolescencia. Hoy en día la tecnología evoluciona exponencialmente, y el mero hecho de tener que discriminar entre varias opciones a la hora de implementar una solución, plantea serios dilemas.

Esta tesis se gestó en el 2012, como se ha presentado en el capítulo 4, sobre la base de tres pilares tecnológicos: *Internet of Things*, *Cloud Computing* y *Mobile Computing*. Desde entonces, dichas tecnologías han evolucionado a su ritmo, tal y como se ha subrayado en las conocidas gráficas del *hype cycle* de Gartner mostradas en la Fig. 6-1.

En este capítulo de conclusiones, las gráficas de Gartner se van a usar para justificar con cierta retrospectiva la elección de tecnologías clave empleadas en esta tesis.

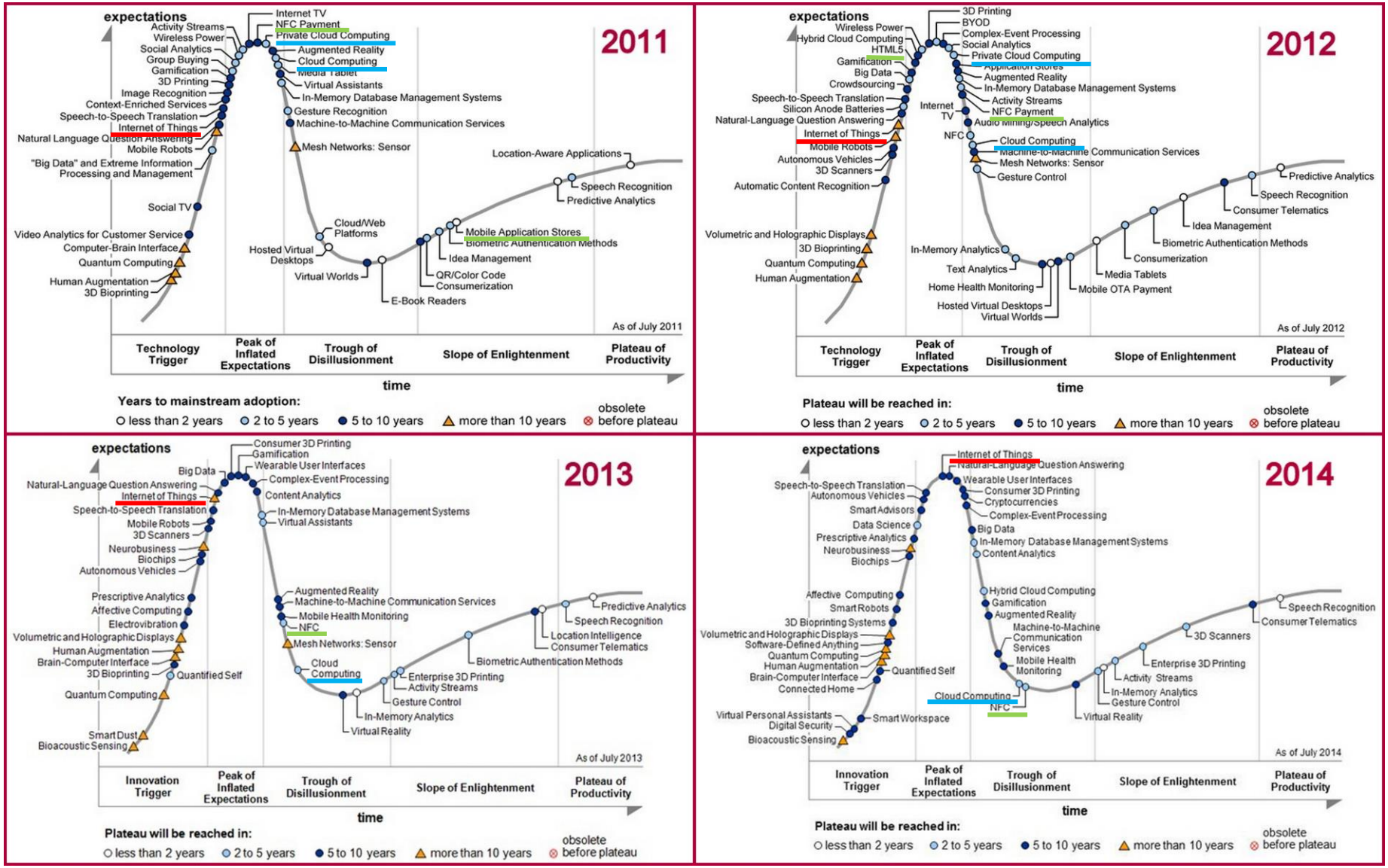


Fig. 6-1 Gráficas del *hype cycle* de Gartner de 2011 a 2014

6.1 Internet de las Cosas: Un problema de taxonomía

Internet of Things apareció en las gráficas de Gartner en 2011 y en 2014 alcanzó el pico en la generación de expectativas, manteniéndose en la misma posición en 2015 y 2016. Es curioso observar que en 2011 se proyectó que IoT tardaría en consolidarse entre 5 y 10 años y en 2012 se pasó a una proyección de más de 10 años. En 2012, se produjo también un leve retroceso en la gráfica tal y como se muestra en la Fig. 6-2.

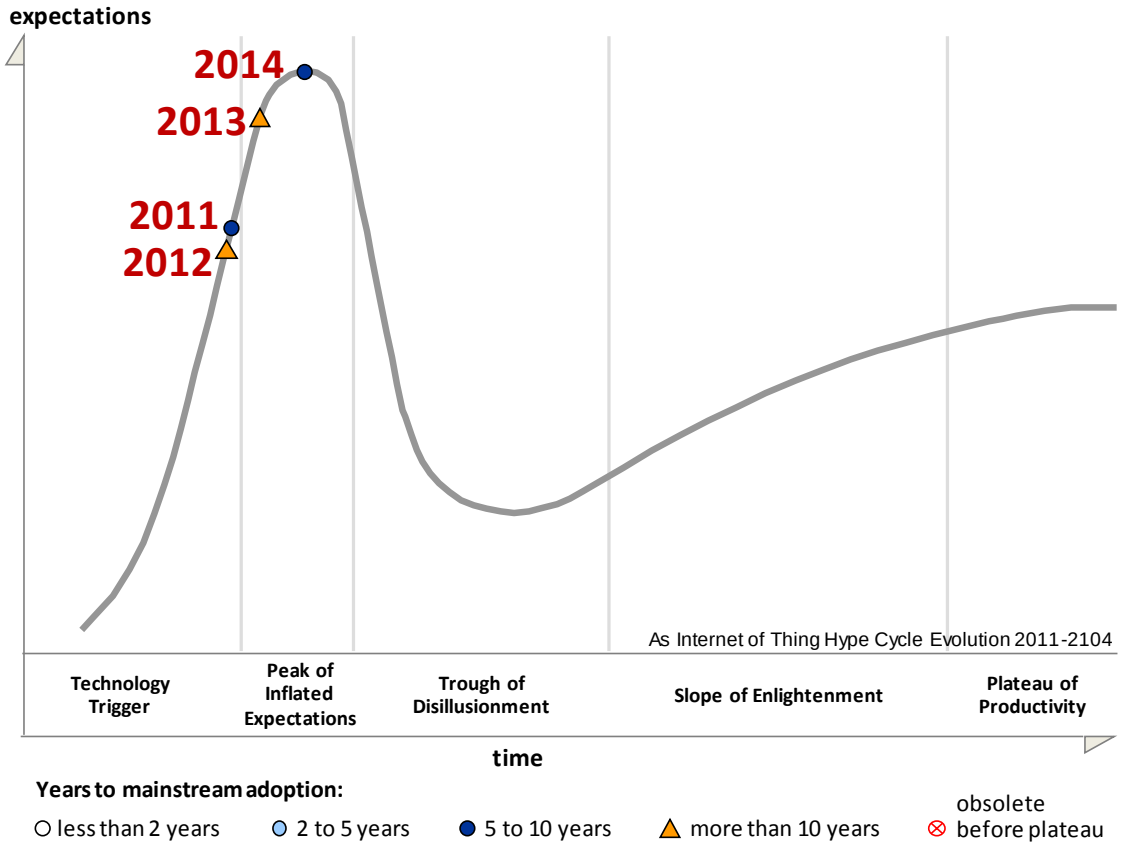


Fig. 6-2 Evolución de IoT en las gráficas del *hype cycle* de Gartner entre 2011 y 2014

Internet of Things ha sido, y sigue siendo un término difícil de explicar y que ha llevado a muchas interpretaciones. Podría pensarse que la falta de consenso en definir su ámbito de actuación ha llevado a Gartner a alargar los tiempos para su consolidación y a ampliar el espectro de conceptos y tecnologías, llegando a dedicarle una gráfica de *hype cycle* como la publicada en 2016, ver Fig. 6-3.

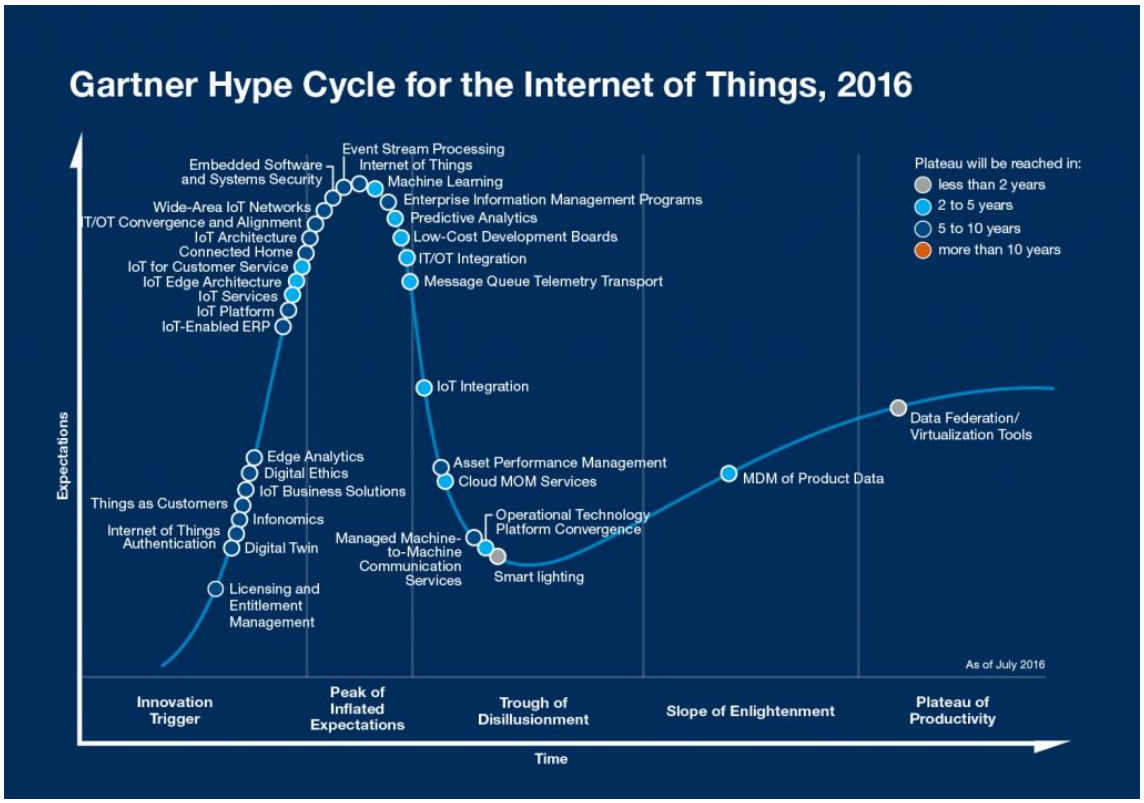


Fig. 6-3 Gráfica del *hype cycle* de Gartner para IoT en 2016

Entre las siete tecnologías clave que Gartner referencia en su *hype cycle* de IoT para 2016 [39], Gartner destaca el concepto de "*Digital Twin*" como un "*software* dinámico que modela una cosa física o sistema y que se basa en los datos de sus sensores para entender su estado y responder a los cambios, mejorar las operaciones y añadir valor". El primer artículo [12] en esencia describe exactamente ese mismo concepto al introducir el término "*ghost Vending Machine*" y el segundo artículo [17] está dedicado a dar los detalles de cómo se ha creado una arquitectura de referencia para una plataforma IoT que en principio permitiría instanciar "*Digital Twins*", en nomenclatura de Gartner. Pero no sólo términos como "*Digital Twin*" o "*IoT Platform*" aparecen como detonantes de la innovación en la curva de Gartner en 2016, también aparecen términos como "*IoT Architecture*" y "*IoT Services*" que se explican en el segundo artículo. El tercer artículo [17] direcciona temas de seguridad y también encontramos en la curva de Gartner "*IoT Authentication*" justo encima de "*Digital Twin*".

Por último, en los artículos se presenta el diseño de placas electrónicas de bajo coste basado en Arduino, que se mantienen en la zona pico de las expectativas de la gráfica de Gartner como "*Low-Cost Development Boards*". Para mantener el bajo coste de las placas, se ha trabajado con microprocesadores muy sencillos de 8 bits, que tienen recursos de memoria y procesado muy limitados. Eso ha forzado a minimizar y optimizar los mensajes y flujos de llamadas porque cada bit cuenta. Este ejercicio puede tener mucha importancia en un futuro, por la emergente aparición de tecnologías LPWA (*Low-Power Wide-Area*), referenciadas en la gráfica de Gartner como "*Wide Area IoT Networks"*. En concreto, la introducción de "*Narrow Band Internet of Things*" (NB-IoT) [40] en la transición de las redes móviles de 4G a 5G, va a permitir conectar objetos a gran distancia alimentados por baterías. El consumo de energía de los dispositivos es proporcional a la información que debe transmitirse, por lo que nuestro diseño en principio podría encajar con el uso de esta tecnología y permitir conectar cualquier objeto alimentado con baterías para convertirse así en un punto de venta.

Como mayor conclusión respecto a la aplicación del paradigma de IoT en esta tesis, se puede afirmar que los objetivos y resultados obtenidos encajan perfectamente con las innovaciones que los analistas de Gartner apuntan como necesarias en IoT.

6.2 *Mobile Computing*: Aplicaciones nativas vs HTML5

Tal y como se desprende de las gráficas de Gartner, ver Fig. 6-1, en 2011 la economía de las apps se había consolidado con la App Store de Apple y Google Play. A finales de 2011, la Fig. 6-4 (fuente: https://www.statista.com) muestra como Android crecía fuerte, alcanzado una cuota de mercado del orden del 50% y expulsando del mercado a otros sistemas operativos que no fueran el iOS de Apple. Por aquel entonces, en febrero de 2012, Mozilla anunció Firefox OS, un nuevo sistema operativo basado en HTML5. Firefox OS fue avalado inicialmente por operadores de telecomunicaciones, si bien finalmente el programa fue cancelado a finales de 2015.

Durante ese periodo, hubo un debate intenso sobre si era mejor desarrollar aplicaciones nativas, o en HTML5 para dispositivos móviles, pero el constante incremento de las capacidades en computación y memoria de los móviles ha terminado por dejar dicho debate estéril, ya que simplemente direccionan segmentos distintos. Actualmente, la tendencia apunta por el desarrollo de aplicaciones híbridas que aúnen lo mejor de cada tecnología.

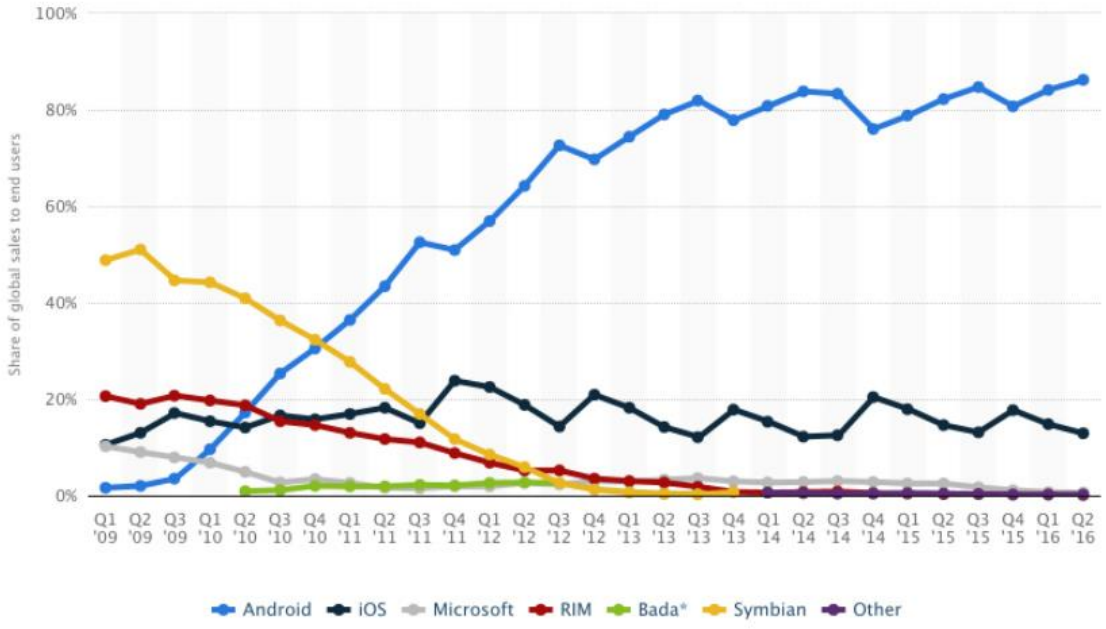


Fig. 6-4 Cuota de sistemas operativos móviles

Por tanto, nuestra apuesta por HTML5 para tener una webapp sencilla y universal, e independiente del dispositivo, se ha mantenido válida a lo largo del tiempo, pero requería que los móviles incorporasen modernos navegadores HTML5 del mundo PC, como Chrome de Google o Safari de Apple. La decisión de Google de incorporar Chrome como navegador de base en Android y su imparable crecimiento con cuotas de mercado que superaban a finales del 2016 el 87% (Fig. 6-4), además del hecho de que el resto de mercado lo cubre prácticamente Apple con Safari en iOS, no hace más que ratificar nuestra premisa de partida sobre el uso de tecnologías ubicuas.

Por último, los recientes anuncios de dejar de soportar flash como opción por defecto en Chrome para reproducir multimedia, pone también en valor las soluciones basadas en estándares abiertos como HTML5.

Concluimos que se ha conseguido implementar un mecanismo de login transparente multidominio basado en HTML5, que permite ofrecer una solución universal de pagos que transciende del vending y es extensible a cualquier sector, permitiendo transformar cualquier objeto en un punto de venta, siempre y cuando este esté conectado a Internet.

6.3 *Mobile Payments*: Evolución de NFC en pagos por móvil

La fragmentación con la que partíamos a principios de 2012 en la aplicación de tecnologías NFC para pagos móviles no ha hecho más que aumentar. Por un lado, tenemos a los principales fabricantes de terminales como Apple y Samsung imponiendo soluciones propietarias, mientras Google sigue buscando encontrar una solución tras su inicial fiasco con Google *Wallet*. A los operadores de telecomunicaciones tampoco les ha ido muy bien, pues no han conseguido posicionarse incluso buscando alianzas en su sector [41]. Sin embargo, sí que hemos asistido a la sustitución y renovación de los terminales en el punto de venta que son capaces de leer tarjetas NFC, porque los bancos han adoptado dicha tecnología como una más a incorporar en sus tarjetas físicas, conjuntamente con la banda magnética y el chip [42]. Los bancos al incorporar NFC, han creado sus propias aplicaciones nativas y han hecho frente a nuevos competidores del mundo móvil que quieren entrar en su negocio. Es curioso ver como la potencia de computación de los *smartphones* ha permitido a multitud de *startups* y empresas fabricantes de datafonos o TPVs (Terminal Punto de Venta) crear accesorios para el teléfono, que son capaces de leer las tarjetas de crédito y usar el teléfono para validar las transacciones contra pasarelas de pago. Como resultado, estas soluciones han desintermediado a los operadores de telecomunicaciones ya que ahora las transacciones no se hacen por conexiones específicas M2M (Machine to machine).

Por tanto, se observan dos tendencias que realmente no son disruptivas, sino más bien continuistas de los modelos actuales de pagos con tarjeta. Por un lado, se pretende eliminar el plástico de las tarjetas y pasarlas al móvil, y por otro lado, se intenta sustituir los TPV físicos por un móvil con un accesorio para leer

tarjetas. Ambos modelos no han hecho más que educar a la gente a usar la tecnología NFC y confiar en los móviles como medio de pago. De hecho, los operadores de telefonía celular siempre han argumentado que se reporta antes la pérdida, o robo, de un móvil que el de una tarjeta de crédito. Llegados a este punto, se puede dar el caso que estemos realizando una transacción sin tarjetas físicas, ni TPV físicos, utilizando dos teléfonos, uno con un *wallet* NFC, y otro con un accesorio NFC para leer tarjetas. Asumiendo que los usuarios están cambiando sus usos y costumbres respecto al uso de los móviles, dicho escenario nos lleva a preguntarnos si no hay maneras más sencillas de usar los teléfonos para realizar transacciones electrónicas.

El modelo alternativo de pagos por móvil propuesto pretende ofrecer una experiencia similar a los pagos por NFC, pero sin requerir TPV físicos o accesorios en los móviles para leer tarjetas. Opcionalmente el modelo puede utilizarse con QRs, siempre y cuando el usuario tenga instalada una aplicación que soporte la lectura de los mismos, como “Wechat”, una aplicación de mensajería que en China ha conseguido irrumpir en el mercado *fintech*.

6.4 *Cloud Computing*: Nube privada con *Big Data*

Desde 2008, el nexo de las fuerzas que define Gartner y que combina las tecnologías: *Cloud*, *Data*, *Mobile* y *Social*, están impactando en nuestra sociedad y acelerando su trasformación digital, pero si bien el 90% de las grandes empresas están siguiendo iniciativas de transformación digital, se estima que menos del 30% sociedad se han subido a la ola de la cuarta revolución industrial porque han vislumbrado la gran escala y el gran impacto global que va a tener en la sociedad. Dos tendencias emergentes surgen para impulsar la trasformación digital. Por un lado, una innovación no tecnológica, es decir, las empresas se tienen que reinventar, y rehacer sus procesos y modelos de negocio adaptándose al uso de las nuevas tecnologías. Y por otro lado, las organizaciones tienen que prepararse para la gestión del dato en tiempo real.

Según ha publicado Gartner en agosto de 2016, ver Fig. 6-5, existen 4 mega tendencias que marcan la evolución de la digitalización de la industria:

- *Cloud, Mobile, Social, Information Technologies.*
- *Digital Business Technologies.*
- *New Design and Innovation Aproaches.*
- *Artificial Intelligence.*

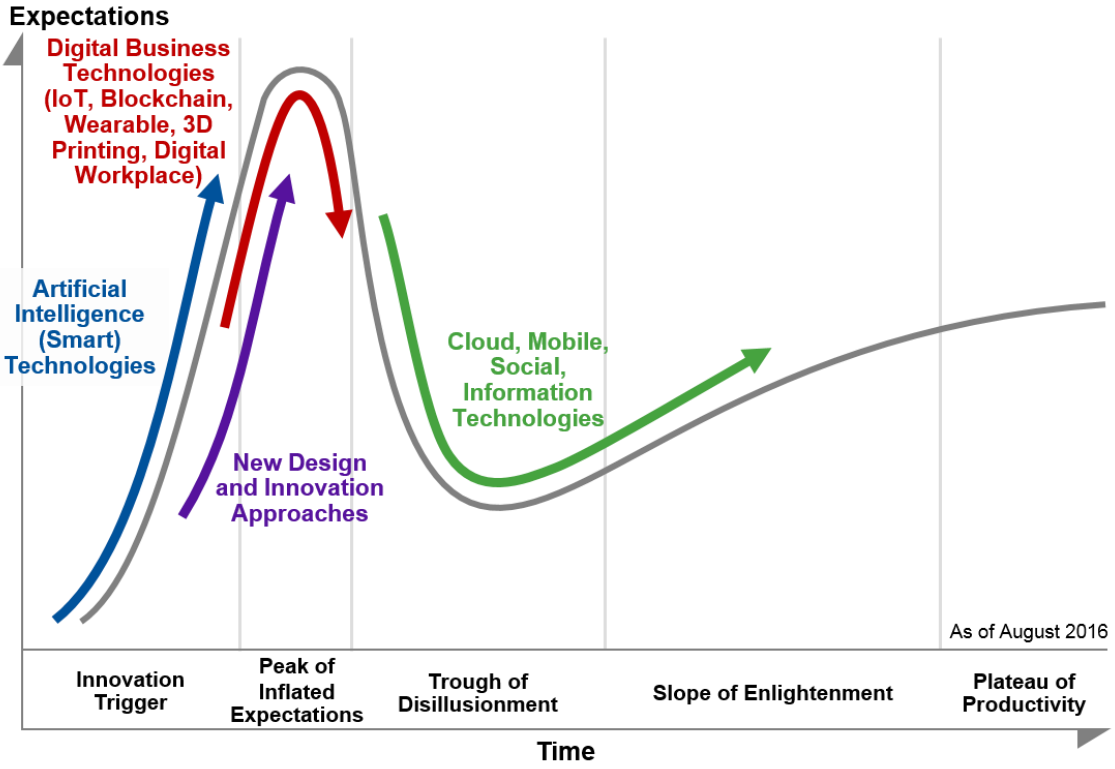


Fig. 6-5 Mega tendencias de Gartner para 2016.

Para concluir, analicemos por qué esta tesis encaja bien en las mega tendencias presentadas en la Fig. 6-5. Partiendo de una necesidad de negocio para un sector como el vending: la búsqueda de eficiencias operacionales y la mejora de la experiencia de compra de los usuarios, se ha construido una solución basada en nube privada, preparada para escalar y procesar gran cantidad de datos, que usa los móviles de los usuarios como principal manera de interactuar con los objetos conectados y cercanos, incorporándolos a nuestra red social de objetos con los que poder interactuar. Por tanto, el diseño de plataforma propuesto encaja perfectamente en la primera mega tendencia de

Gartner (*Cloud, Mobile, Social, Information Technologies*), ya que las elecciones tecnológicas realizadas han llegado al mercado durante este periodo de cuatro años. La segunda mega tendencia (*Digital Business Technologies*) es la aplicación de los paradigmas de IoT para facilitar la trasformación digital. Se han construido dispositivos capaces de auto provisionarse en Internet y anunciarse a los usuarios con mecanismos de proximidad securizados. Los prototipos y pruebas de concepto realizadas auguran una buena aceptación por parte de los usuarios, si bien para poder probar el sistema es necesario convencer a la industria del vending que tiene que innovar en sus procesos internos, que justamente es la tercera mega tendencia identificada por Gartner (*New Design and Innovation Aproaches*). Por último, la adopción de esta solución permitiría agregar mucha información de los usos y costumbres de los usuarios del vending ya que la arquitectura se ha diseñado para albergar capacidades de *Big Data*. La cuarta tendencia (*Artificial Intelligence*) en la gráfica de Gartner pasaría por añadir más inteligencia a la solución, a medida que se vayan incorporando nuevas tecnologías. En este sentido, esta tesis también ha hecho una aproximación necesaria al *Big Data* como se menciona en las secciones 7.5.3 y 7.6.4.

En todas las publicaciones presentadas se concluye que como siguiente paso es necesario demostrar con pruebas de campo que la solución diseñada es bien aceptada por los usuarios finales. Pero para ello, es necesario que el sector del vending esté abierto a adoptar cambios en sus procesos y se reinvente a sí mismo. Uno de los mayores obstáculos en la realización de esta tesis ha sido la propia incredulidad del sector del vending respecto a que es posible conectar todas sus máquinas de vending a bajo coste y generar las suficientes eficiencias operacionales e incremento de ventas para acometer dichas inversiones. Si las predicciones de Gartner se cumplen, se necesitarán entre 5 y 10 años para que IoT finalmente llegue al vending, ya que es un sector extremadamente conservador.

6.5 Sumario: 6 contribuciones a recordar

A continuación, se enumeran a modo de resumen las contribuciones claves que se han ido comentando a lo largo de este capítulo:

1. Enfrentarnos a un caso real de negocio como el vending, nos ha forzado a implementar una arquitectura y unos mecanismos de seguridad para IoT, que se vislumbran como una necesidad para la materialización del propio IoT. El loging silencioso multidominio presentado trasciende el ámbito del vending y puede aplicarse a multitud de escenarios en distintos sectores.

2. A fecha de publicación de esta tesis, la penetración de HTML5 en los navegadores de los nuevos móviles es prácticamente total, por lo que la decisión de implementar una webapp como mecanismo de interacción entre las personas y los objetos ha sido muy acertada, ya que proporciona una solución universal, no vinculada a ningún actor de la cadena de valor.

3. El uso de tecnologías web permite realizar transacciones de comercio electrónico típicas de las tiendas *online* desde el móvil en el propio punto de venta físico, y acto seguido informar al punto de venta de la misma. La solución presentada se puede aplicar a la eliminación de las colas para pagar en TPV físicos, por ejemplo, en las cajas de los supermercados.

4. La tokeninzación de las URLs es un mecanismo sencillo y transparente para los usuarios, que ya se están acostumbrando a usar NFC en sus transacciones electrónicas. La emergencia de Web física, con *beacons* publicando URLs con tecnología BLE (*Bluetooth Low Energy*), no hará más que constatar que este mecanismo es extensible a cualquier tecnología inalámbrica de corto alcance.

5. La restricción autoimpuesta de desarrollar dispositivos IoT de muy bajo coste, deja la puerta abierta a integrar tecnologías de comunicación de LPWA como NB-IoT, eMTC, SigFox, LoRA etc. Estas tecnologías podrían ser válidas incluso en escenarios donde las transacciones electrónicas

admitiesen latencias de varios segundos, lo que permitiría convertir en un punto de venta cualquier objeto desatendido alimentado con baterías, por ejemplo, una mesa de un restaurante.

6. Diseñar una nube privada tiene la ventaja de controlar la utilización de los recursos y poder instanciar sistemas muy complejos ("*Digital Twins*") de forma automática. El mecanismo de "*plug-and-play*" diseñado para nuestros módulos IoT, plantea un diseño de plataforma IoT alternativo al de la multitud de plataformas que están emergiendo y que están desapareciendo casi a la misma velocidad por no diferenciarse entre ellas.

6.6 Líneas de investigación futuras

En las tres publicaciones se menciona en las conclusiones que como siguiente paso debería hacerse una prueba piloto en campo para evaluar tanto la aceptación por parte de los operadores de vending como por parte de los consumidores. La captura y procesado de datos reales aplicando las analíticas implementadas de Big Data deberían darnos muestras empíricas de los beneficios generados por OpenVend en términos de:

- Ahorros en costes por eficiencias operacionales.
- Incrementos en ventas por mejora de la experiencia de compra.

Por otro lado "OpenVend" debería transformarse en una comunidad de código y *hardware* abierto con la visión de conectar "TODAS" las máquinas de vending a "*LOW COST*" y con la misión de crear la Internet de las máquinas de vending con "Venduino", un open *hardware* 100% compatible con Arduino. En este sentido se han reservado los dominios openvend.eu y venduino.com para crear dichas comunidades.

Otro camino pendiente de explorar es lanzar una campaña de micro-mecenazgo para llevar los prototipos mínimamente funcionales a un primer producto comercial. Para ello es necesario realizar una intensa campaña de marketing y educar al sector del vending, para que pueda afrontar con éxito su

transformación digital. A tal fin, en Julio de 2015 y como anticipo a los resultados presentados en esta tesis, se publicó el artículo “Open Hardware & Software para el Vending en la era de la Internet de las Cosas” [43] en una revista especializada y dirigida al sector del vending español.

Si bien la primera publicación [12] (ver sección 5.2.6) justifica el uso de microcontroladores como Arduino [1] frente al uso de microcomputadores como Raspberry-Pi [20] por su simplicidad y bajo coste, para la implementación de un prototipo de cartelería digital (*Digital Signage*) sí se ha llegado a utilizar un sistema embarcado con *Linux* como Sistema Operativo corriendo en una Raspberry-Pi A+ (ver sección 7.6.2). Dicha implementación deja la puerta abierta a un rediseño de la arquitectura Cloud aplicando los nuevos paradigmas de *Fog Computing* o *Edge Computing*. De ese modo, la arquitectura Cloud podría distribuirse siendo posible correr las instancias a nivel de SaaS en los propios sistemas embarcados de las máquinas de vending.

7 OTRAS APORTACIONES CIENTÍFICAS

Para la realización de un prototipo mínimamente funcional, se ofertaron proyectos multidisciplinares de fin de carrera en cuatro cursos académicos consecutivos de 2012 a 2016 dirigidos por mí y supervisados por las directoras de esta tesis. Como resultado, se realizaron cuatro ciclos de innovación de prototipado rápido, uno por curso académico, que fueron subscritos por un total de 17 alumnos (ver Fig. 7-1). En cada ciclo los estudiantes se auto organizaron siguiendo la metodología *Scrum/Agile* [2]. Es de destacar que todos ellos obtuvieron la máxima calificación en sus respetivas defensas, en reconocimiento a su esfuerzo y dedicación.

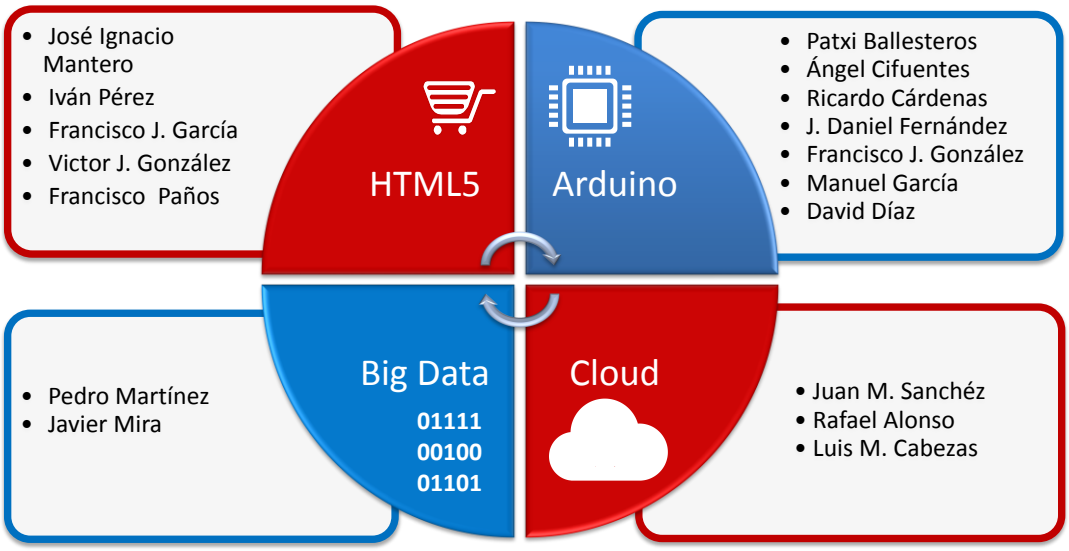


Fig. 7-1 Alumnos involucrados en esta tesis

Hasta ahora, en esta disertación no se ha hecho referencia explícita a sus memorias de proyecto de fin de carrera, ni tampoco a su contribución. Este capítulo tiene una finalidad doble. Primero, dar la posibilidad al lector de

profundizar en los detalles de las soluciones implementadas por los alumnos, como respuesta a los retos planteados en el ámbito de esta tesis, y segundo, presentar una experiencia docente donde un grupo de estudiantes se auto-organizan y realizan conjuntamente sus proyectos fin de carrera siguiendo la metodología *Scrum/Agile* [2].

Para conseguir la involucración de los alumnos en este programa ha sido clave partir de un escenario muy concreto como el propuesto en esta disertación. La visión y misión de OpenVend [44] ha servido como fuente de inspiración para que cada alumno encontrase sus propias áreas de interés en las cuatro áreas tecnológicas propuestas: Mobile Computing con HTML5, Internet of Things con Arduino, *Cloud Computing* y *Big Data*.

7.1 Adaptación de la Metodología *Scrum/Agile*

Scrum es un método de desarrollo ágil iterativo e incremental, propuesto por Hirotaka Takeuchi y Ikujiro Nonaka (1986) [2] como un nuevo método para mejorar la rapidez y flexibilidad en el desarrollo de productos comerciales. Scrum se basa en entregas parciales y regulares del producto final, comenzando por aquellas funcionalidades más importantes para el cliente. Por ello Scrum está especialmente indicado para proyectos donde se requieren resultados tempranos, donde los requisitos son cambiantes y donde la competitividad, flexibilidad y productividad son cruciales.

Por tanto, un principio clave de *Scrum* es el reconocimiento de que durante un proyecto los “clientes” pueden cambiar de idea sobre lo que quieren y necesitan. Por lo tanto, *Scrum* adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes. *Scrum* permite la creación de equipos auto-organizados y potencia la comunicación verbal entre todos los miembros involucrados en el proyecto.

La metodología *Scrum* encaja bien en entornos de innovación donde los equipos están co-localizados y se premia el diálogo entre ellos de manera que se auto organicen. Podríamos decir que se elimina el rol del director de proyectos y se sustituye por la figura del *Scrum master*. El *Scrum master* vela por los intereses del equipo y prioriza las actividades para poder entregar un prototipo mínimamente funcional en un corto periodo de tiempo. El prototipo se presenta al responsable de producto y/o cliente final para su validación, en lo que se denominan los *Sprints* que, en entornos de desarrollo de *Software* suelen durar de una a varias semanas.

Adaptar la metodología *Scrum* a la UNED no ha sido inmediato. Primero, porque la UNED es una universidad a distancia, y por tanto no podemos partir de la hipótesis que los equipos estarán localizados en una misma ubicación donde poderse reunir. Segundo, por la propia heterogeneidad de los estudiantes de la UNED, ya que muchos estudian segundas carreras, suelen trabajar y no siempre en sectores relacionados con sus estudios. Bajo estas premisas, parecería imposible adaptar la metodología *Scrum* al entorno de la UNED, sin embargo, nada más lejos de la realidad. Para ello, las reuniones se han realizado fuera del horario laboral. Las reuniones diarias típicas de *Scrum* han pasado a ser reuniones semanales y los *Sprints,* que típicamente duran una semana, han pasado a ser mensuales. Por lo general, un *Sprint* empieza con un *brainstorming* de ideas que termina en una lista de actividades que el grupo se compromete a realizar durante la duración del *Sprint,* y termina con una presentación/demostración de lo conseguido.

Para ser admitidos en cualquiera de los ciclos del programa, los alumnos tenían que comprometerse a participar en 8 *Sprints* de octubre a junio. El ANEXO B es un ejemplo de la solicitud de adscripción al programa con la planificación detallada de la adaptación de la metodología *Scrum/Agile.*

Ha sido justamente la necesidad de interactuar con los compañeros lo que ha permitido motivar a los alumnos a realizar las tareas asignadas por el mismo grupo. Por otro lado, el rol de *Scrum master* ha ido rotando entre los participantes de cada ciclo. El rol de *product owner* ha recaído en mi persona, como director

de sus proyectos. El respeto mutuo y el interés por conocer los problemas a los que cada uno se enfrentaba en sus respectivas áreas tecnológicas, ha llevado a los participantes, incluso entre distintos ciclos, a colaborar para llevar a buen término sus respectivos proyectos de fin de carrera. Cada estudiante, en su memoria, ha evaluado la experiencia y si bien hay opiniones dispares, el programa ha dado buenos resultados, sin sufrir ningún abandono por parte de los estudiantes.

7.2 Planificación del trabajo y memorias

Cada uno de los cuatro ciclos consecutivos se ha estructurado en 8 *Sprints* de octubre a junio. Cada *Sprint* se cerraba con la presentación de un prototipo mínimamente funcional. Cada dos o tres *Sprints* se presentaban los avances a las directoras de esta tesis, como muestra del avance de los proyectos. Las presentaciones de los alumnos a las directoras se limitaban a 15 minutos siguiendo las pautas de una defensa de proyecto de fin de carrera. El alumno tenía hasta la primera convocatoria de octubre para completar la documentación necesaria y poder presentar su proyecto, preferentemente de manera colectiva con el resto de participantes. Mi participación activa como director del proyecto y la colaboración con el resto de estudiantes, sólo se aseguraba hasta la primera convocatoria. De no presentarse en primera convocatoria, el alumno disponía de tres convocatorias más: diciembre, marzo y junio para presentar su proyecto. Pero pasado la primera convocatoria no se garantizaba ni la colaboración, ni la defensa conjunta con el resto de participantes.

7.3 Primer ciclo de innovación: Curso 2012-2013

En el primer ciclo correspondiente al curso 2012-13, se propusieron proyectos basados en tres áreas tecnológicas:

- Arduino como open *hardware*.

- OpenStack para crear la infraestructura necesaria para soportar el proyecto en una nube privada.
- HTML5 para crear webapps.

Tres alumnos mostraron interés en el área de HTML5. Por lo que la parte Cloud y de Arduino fue soportada por mí para cohesionar el grupo durante las reuniones semanales del *Scrum*. La parte de infraestructura se solventó recurriendo a un proveedor de servicios, y se optó por OpenCart como *software* libre sobre el cual desarrollar el proyecto y las webapps en HTML5. En la parte de Arduino, tomé personalmente el rol de desarrollador de *software* para realizar las interacciones con el resto de los miembros del equipo.

El primer ciclo se destinó a explorar las capacidades que HTML5 podía proporcionar a nuestro proyecto. Se abrieron las líneas de investigación que se recogen a continuación.

7.3.1 Webapp de compra (Ignacio Mantero)

Con este trabajo [45] se logra implementar un mecanismo de login transparente multidominio en OpenCart que permite iniciar una sesión en la webapp de compra, sin requerir introducir las credenciales cada vez que se accede al sistema. Dicho mecanismo fue integrado en el resto de desarrollos de sus compañeros de ciclo y mejorado en el segundo ciclo. El detalle del mecanismo es descrito en el artículo "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [14], capítulo 5.4.

http://yourls.openvend.es/2014imr

7.3.2 Webapp de fidelización (Iván Pérez)

Este trabajo [46] profundiza en cómo implementar mecanismos de fidelización con tecnologías HTML5, que hacen uso de la cámara de los móviles. Acceder a un recurso del móvil, como es la cámara desde una webapp en

HTML5, no es obvio [36]. En este trabajo se consigue leer QRs y marcadores de "Realidad Aumentada" sin necesidad de instalar aplicación nativa. El proyecto implementa cuatro escenarios sobre OpenCart: cupones de descuento, compra mediante realidad aumentada, monitorización del estado de máquinas de vending y reserva de productos.

http://yourls.openvend.es/2014ipm

7.3.3 Webapp de inventarios y planificación de rutas (Francisco J. García)

Partiendo de ficheros de telemetría de máquinas de vending, este trabajo [47] plantea los problemas típicos de gestión inventarios y optimización de rutas. Este proyecto se adentra en las APIs de localización para la planificación y visualización de rutas de reposición. Este proyecto es el primero que introduce el *SLIM framework* [31] para implementar interfaces RESTful que será adoptado y ampliado en los posteriores ciclos. La telemetría de la máquina de vending se envía regularmente a Opencart para sincronizar el *stock online*. Si en el momento de la reposición se reconfigura la máquina, ya sea cambiando el precio o los productos, el operario dispone de una webapp para reconfigurar OpenCart. Dicha webapp explora las capacidades "*drag-and-drop*" en HTML5.

http://yourls.openvend.es/2015fjg

7.4 Segundo ciclo de innovación: Curso 2013-2014

En el segundo ciclo, curso 2013-2014, se siguieron ofertando los mismos proyectos que en el primer ciclo basados en las tres tecnologías ya referenciadas: HTML5, Cloud y Arduino. Esta vez se captaron para el programa

cinco nuevos alumnos. En este ciclo tres alumnos se interesaron por Arduino y un cuarto en OpenStack. El quinto alumno siguió con las tecnologías HTML5.

7.4.1 Pagos por móvil con Arduino (Patxi Ballesteros)

Este proyecto [48] consiste en el desarrollo de un sistema sin efectivo (*cashless*) con comunicaciones GRPS. La mayor contribución de este proyecto fue mejorar las librerías públicas que implementan el protocolo MDB, y conseguir optimizar los recursos del microcontrolador de Arduino para que fuese capaz de atender varias tareas a la vez con distinta prioridad. Para ello se implementó un mecanismo basado en varias máquinas de estado que no hacían uso de la directiva "*delay()*". Dicha implementación sirvió de referencia para el resto de proyectos de Arduino.

http://yourls.openvend.es/2014pbf

7.4.2 Pagos por móvil con Arduino & NFC (Ángel Cifuentes)

Este proyecto [49] implementa un sistema cashless con tecnología NFC, que introduce un lector NFC y proporciona un método de pago con móvil y también, el tradicional método de pago con tarjetas de prepago con opción de recarga de saldo desde la máquina. En este proyecto se profundiza en el estado del arte de la tecnología NFC en el ecosistema Arduino.

http://yourls.openvend.es/2014acs

7.4.3 Telemetría y concurrencia con Arduino (Ricardo Cárdenas)

A partir de librerías en "C" de código abierto, en este proyecto [50] se crea una librería en Arduino capaz de extraer la Telemetría de la máquina de vending.

Otra gran contribución de este proyecto es el uso de interrupciones para simular la multitarea y relajar así las restricciones de diseño, previamente impuestas, de no utilizar la directiva "*delay()*", presente en multitud de librerías de Arduino.

http://yourls.openvend.es/2015rcm

7.4.4 Construyendo un PaaS con Openstack (Juan M. Sánchez)

En este segundo ciclo se empezó a trabajar con la tecnología OpenStack. En el trabajo [51] se detalla cómo construir una nube privada donde instalar los servidores de OpenCart utilizados en el primer ciclo. Tomar el control del IaaS permite diseñar el mecanismo de *plug-and-play* descrito en el artículo "*A self-provisioning mechanism in OpenStack for IoT devices*" [13], sección 5.3.

http://yourls.openvend.es/2014jms

7.4.5 Webapps con HTML5/SS3/Javascript (Víctor J. González)

Este proyecto [52] da continuidad a los proyectos del primer ciclo de HTML5 y mejora la encriptación del sistema de login silencioso sobre OpenCart tal y como se describe en el artículo "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [12], sección 5.4. También se realiza una labor importante de integración al desarrollar toda la capa del PaaS sobre el IaaS que comunica los Arduinos con Opencart, en la capa de aplicación.

http://yourls.openvend.es/2015vjg

7.5 Tercer ciclo de innovación: Curso 2014-2015

El segundo ciclo sentó las bases de la arquitectura de referencia lo que permitió atraer a nuevos estudiantes interesados en la parte de sistemas de computación. Se incluyó *Big Data* como nueva área en la oferta de proyectos fin de carrera publicada para ese curso. En consecuencia, en el tercer ciclo se unieron al proyecto 5 nuevos estudiantes, dos interesados en *Cloud Computing*, otro estudiante en *Big Data* y dos estudiantes más continuando los desarrollos en Arduino.

7.5.1 IaaS en alta disponibilidad con Openstack (Rafael Alonso)

El objetivo alcanzado en este proyecto [53] es la construcción de una infraestructura de virtualización plenamente operativa en un *data center* real, partiendo de la arquitectura de referencia diseñada en el ciclo 2. Para esta infraestructura se requiere una configuración que opere conforme a las exigentes condiciones de la alta disponibilidad, esto es, disponibilidad superior a los “cinco nueves”, el 99,999% del tiempo.

http://yourls.openvend.es/2015raa

7.5.2 Construyendo PaaS con Hadoop (Luis M. Cabezas)

Solventadas las limitaciones de infraestructura al disponer de un *data center* con varios servidores físicos, se abrió la posibilidad de instalar en el mismo el ecosistema Hadoop y plantear escenarios de *Big Data*. El enfoque principal de este trabajo [54] es explorar la tecnología proporcionada por el

ecosistema de Hadoop. Éste está formado por un conjunto de programas, que trabajando de forma colaborativa, otorgan la posibilidad de realizar almacenamiento masivo (*data warehouse*) y prospección de datos (*data mining*) dotando de la capacidad de obtención y control de variables que ayuden a mejorar la competitividad y la toma de decisiones.

http://yourls.openvend.es/2015lmc

7.5.3 *Big Data* en PaaS con Hadoop (Pedro Martínez)

El primer objetivo de este proyecto [55] es poner en marcha un subsistema dedicado al almacenamiento y procesamiento de datos con Hadoop, y tener la capacidad de concentrar todos los datos que se generen en la plataforma, independientemente, del número de servicios de vending en la nube que se desplieguen o sistemas que lo compongan. El segundo objetivo es ofrecer un portal con un cuadro de mando (*Dashboard*), que permita presentar información de valor que ayude a supervisar la actividad a bajo nivel entre módulos, la productividad del servicio de vending y favorezca la toma de decisiones de negocio de los diferentes propietarios.

http://yourls.openvend.es/2015pml

7.5.4 Pagos por móvil con Arduino & NFC (J. Daniel Fernández)

El objetivo base de este proyecto [56] se focaliza en plantear nuevos escenarios alternativos usando mecanismos de doble nivel de autentificación enviando un PIN a la máquina de vending. Dichos escenarios se pueden dar ante eventuales fallos en la resolución de la URL dinámica, o ante escenarios de uso de URL estáticas para anunciar la máquina. También se estudió la actualización del firmware de Arduino.

http://yourls.openvend.es/2015jdf

7.5.5 Redes malladas con Arduino (Francisco J. González)

Retomando y ampliando los desarrollos en Arduino, este proyecto [57] sigue nuevas líneas de investigación para minimizar el coste asociado a las comunicaciones creando redes de WiFi que agrupasen varias máquinas de vending de forma que compartiesen una única salida a la red celular.

http://yourls.openvend.es/2015fgs

7.6 Cuarto ciclo de innovación: Curso 2015-2016

En el cuarto ciclo, correspondiente al 2015-2016 se ofertaron los mismos proyectos que en el ciclo anterior, pero por causas diversas no se pudo disponer de una infraestructura de *data center* como la usada en el tercer ciclo, lo que condicionó el desarrollo de los proyectos. Este hecho unido a que los alumnos nuevos procedían del nuevo plan de estudios de Grado, hizo relajar el nivel de interdependencia entre los proyectos. Cuatro nuevos alumnos se unieron al programa, en este caso dos alumnos siguieron desarrollando proyectos alrededor de Arduino, otro estudiante se interesó en la parte de HTML5, y por último un estudiante se decantó por el desarrollo de algoritmos de *Big Data*.

La falta de una infraestructura de *data center* sobre la que seguir construyendo nuestra arquitectura de referencia limitó las líneas de actuación de los estudiantes dedicados a evolucionar la plataforma. Por tanto, se decidió plantear proyectos que no tuviesen gran dependencia con los anteriores ciclos y no requiriesen de una infraestructura de nube privada. En la parte de Arduino se iniciaron dos nuevas vías de investigación si bien se decidió que fuesen independientes. Para dar soporte a los nuevos alumnos, hubo que recodificar el

software a nivel de PaaS y SaaS, para hacerlo funcional sobre un entorno hosteado como el utilizado en el primer ciclo.

7.6.1 Pagos por móvil con Arduino & NFC (David Díaz)

En este trabajo [58] se profundizó en el diseño de circuitos integrados y se fabricó una placa base específica para el proyecto que aunaba todas las necesidades de los proyectos previos. También se hizo una actualización del estado del arte respecto a nuevos chipsets, y librerías de Arduino que permitían reducir el coste y tamaño del diseño resultante.

http://yourls.openvend.es/2016ddp

7.6.2 Digital Signage con Raspberry-Pi (Manuel García)

En este trabajo [59] se acomete la inclusión de cartelería digital como una posible extensión al proyecto y la posibilidad de convertir a las máquinas de vending en un *hostpot* de WiFi, cuya página de inicio fuese un portal cautivo que nos llevase a la URL de la máquina de vending. Dichos requerimientos requerían un *hardware* abierto más potente que Arduino por lo que se pasó a trabajar con Raspberry-Pi y programar en Python.

http://yourls.openvend.es/2016mgj

7.6.3 Construyendo un PaaS con WordPress (Francisco Paños)

En cuanto a la parte de HTML5, en este proyecto [60] se opta por investigar una línea nueva y utilizar en la capa de aplicación WordPress en vez de OpenCart. Si bien el principal uso de WordPress es la publicación de blogs, no es menos cierto que la comunidad que soporta esta iniciativa de código

abierto ha desarrollado en los últimos años multitud de extensiones, incluyendo también soporte a tiendas virtuales con la extensión Woo Commerce.

http://yourls.openvend.es/2016fpm

7.6.4 *Big Data* para Internet de las Cosas (Xavier Mira)

En lo que respecta a la parte de *Big Data*, con este trabajo [61] se profundiza en el tratamiento y análisis de los datos. Este proyecto tiene como objetivo principal el análisis, diseño e implementación de un sistema de predicción de ventas usando técnicas de *Machine Learning*. Se adopta la programación en R y se analizan varias técnicas de predicción, algunas clásicas como los modelos de regresión lineal múltiple o regresiones de *Poisson*, y otras más actuales como *Support Vector Machines* (SVM) con el objetivo final de predecir la demanda y por ende, optimizar las rutas y las reposiciones (temas que ya fueron planteados en el primer ciclo).

http://yourls.openvend.es/2016xmf

7.7 Defensas conjuntas y sinergias entre ciclos

Como ya se ha mencionado, los ciclos de innovación propuestos estaban vinculados a años académicos y compuestos por 8 *Sprints* de octubre a junio con demostraciones parciales trimestrales. Una vez finalizados los *Sprints*, los alumnos tenían como objetivo leer en cualquiera de las cuatro convocatorias de octubre, diciembre, marzo o junio. La incorporación al programa requería la participación en las reuniones semanales de los citados *Sprints*. Una vez finalizados los 8 *Sprints* era responsabilidad de los alumnos escribir su memoria

y coordinarse con sus compañeros para intentar realizar una defensa conjunta, compartiendo un mismo entorno para realizar las demostraciones oportunas.

La Fig. 7-2 muestra las defensas realizadas a lo largo de los cuatro cursos académicos (ciclos).

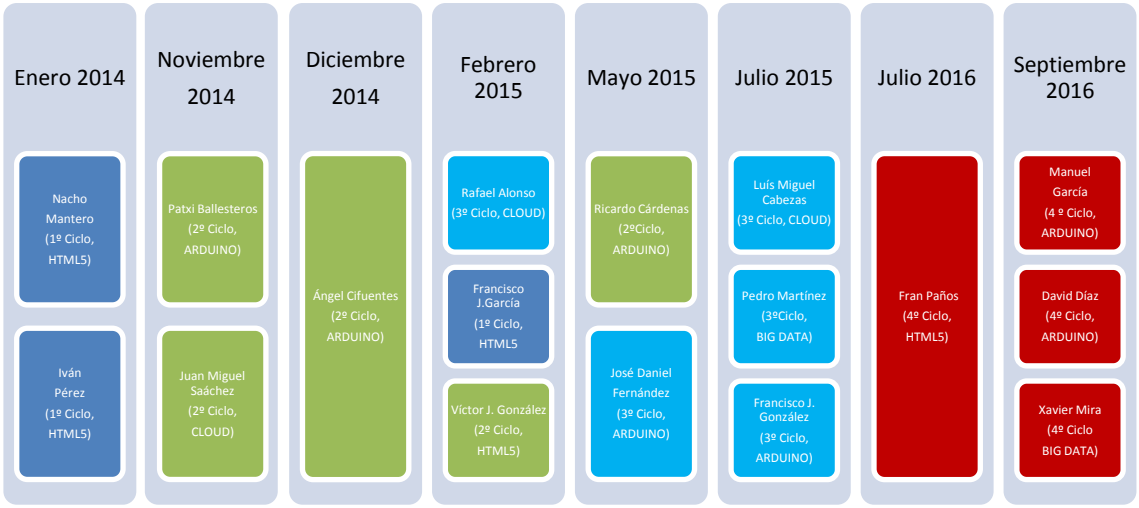


Fig. 7-2 Defensas proyectos fin de carrera

A finales del primer ciclo, en septiembre de 2013 sólo un alumno había escrito la memoria y estaba en disposición de leer en primera convocatoria. No obstante, decidió esperar a su compañero y hacer una defensa conjunta en segunda convocatoria. Así, la primera defensa se realizó en enero de 2014.

La primera lectura del segundo ciclo correspondiente al curso académico 2013-2014, se hizo en primera convocatoria, noviembre de 2014.

En la cuarta defensa confluyeron los tres ciclos de innovación, siendo la primera demostración sobre un entorno de nube privada corriendo en un *data center* real llevándose a cabo la migración del diseño del entorno Cloud del segundo ciclo. Fue en esta cuarta defensa conjunta donde se integraron los desarrollos del ciclo 1 relativos a HTML5, rediseñándose el mecanismo de login transparente para integrarlo en la nueva arquitectura del PaaS.

En mayo de 2015 se realizó la quinta defensa, está vez se disponía de un entorno estable en una nube privada. Los proyectos de Arduino presentados

hasta la fecha, fueron cohesionados, e integrados, requiriendo un gran esfuerzo por aunar tanto lógica en un dispositivo tan limitado como el Arduino Mega con un microcontrolador AVR de 8 bits y una RAM de 256K.

El ciclo 3, correspondiente al curso 2014-2015 se concluyó en primera convocatoria en Julio de 2015.

El ciclo 4 se realizó ya con estudiantes de Grado. Desde un principio se les planteó a los alumnos que para participar en el proyecto se debía seguir la planificación planteada en los ciclos anteriores y empezar las reuniones semanales en octubre.

7.8 Conclusiones

El hecho que todos los participantes hayan superado con la máxima puntuación sus respectivos proyectos de fin de licenciatura, o de grado, no deja lugar a dudas que la metodología *Scrum/Agile* es aplicable a entornos educativos *online*, o a distancia. Pero también es cierto que es necesario adaptarla, y como la propia metodología promulga, ser flexible ante los cambios e imprevistos y la situación personal e intereses de cada alumno.

Se remite al lector a las memorias de cada alumno ya que todos ellos incluyeron un apartado sobre su interpretación de esta metodología aplicada a su proyecto.

A continuación, se enumeran doce recomendaciones para poder replicar esta experiencia en entornos académicos:

1. Partir de un caso concreto, tangible y real que permita a los alumnos adentrarse en las problemáticas de un sector en concreto, en nuestro caso, el sector del vending, y plantear por si mismos soluciones a tales problemas.
2. El proyecto debe descomponerse en módulos y tecnologías que despierten el interés de los alumnos con el fin de formar un equipo multidisciplinar pero

reducido, de unas 6 personas. El ANEXO A incluye, a modo de ejemplo, los proyectos ofertados en el último ciclo.

3. El director del proyecto debe actuar como el "*product owner*" y facilitar el contacto con el cliente final y los alumnos para que los alumnos prioricen por si mismos las necesidades del cliente.
4. Si no se consigue formar el equipo multidisciplinar necesario para llevar a cabo el proyecto, es responsabilidad del director de proyecto tomar parte activa como miembro del equipo para intentar suplir por cualquier otro medio las necesidades para la consecución de los *Sprints*.
5. Aun estando en entorno académico, es muy importante intentar involucrar al cliente final en la validación de hipótesis de partida y prototipos. En nuestro caso, contamos con la empresa Automatic S.A., que nos facilitó datos reales de telemetría, así como simuladores fabricados con controladores y monederos de máquinas de vending.
6. Realizar las reuniones de *Scrum* de forma semanal y plantear la duración de los *Sprints* a un mes (4 o 5 semanas).
7. Limitar la duración de los ciclos de innovación al curso académico.
8. Realizar presentaciones trimestralmente con formato de defensa (presentación y demo en 15 minutos). La participación de los supervisores y directores de los proyectos, o tesis, debe ser obligatoria, siendo muy recomendable invitar a actores externos afines a los temas presentados.
9. Las reuniones deben adaptarse a la disponibilidad de los alumnos y consensuarse de mutuo acuerdo tras cada reunión. La reiterada no participación en las reuniones debe ser considerada como causa de salida del programa.
10. El rol de "*Scrum master*" debe rotar entre los alumnos para facilitar la cohesión del grupo.
11. Los *Sprints* son principalmente para programar y prototipar. Los errores o malas decisiones a la hora de elegir tecnologías son bienvenidos, si se identifican y descartan al final de cada *Sprint*.
12. La memoria debe escribirse en el último *Sprint*, alentando al alumno a ir tomando notas a lo largo de los *Sprints*.

8 FACTOR DE IMPACTO

El primer artículo: *"Smart vending machines in the era of Internet of things"* [12] está publicado en la revista *"FUTURE GENERATION OF COMPUTER SYSTEMS"* (ISSN: 0167-739X), de la editorial Elselvier.

FUTURE GENERATION OF COMPUTER SYSTEMS figura en el 2015 *JCR Science Edition* con un factor de impacto de 2.430. Es una revista situada en el Q1. En concreto se encuentra la número 11 de 105 dentro de la categoría de COMPUTER SCIENCE, THEORY & METHODS del JCR 2015 (ver Fig. 8-1).

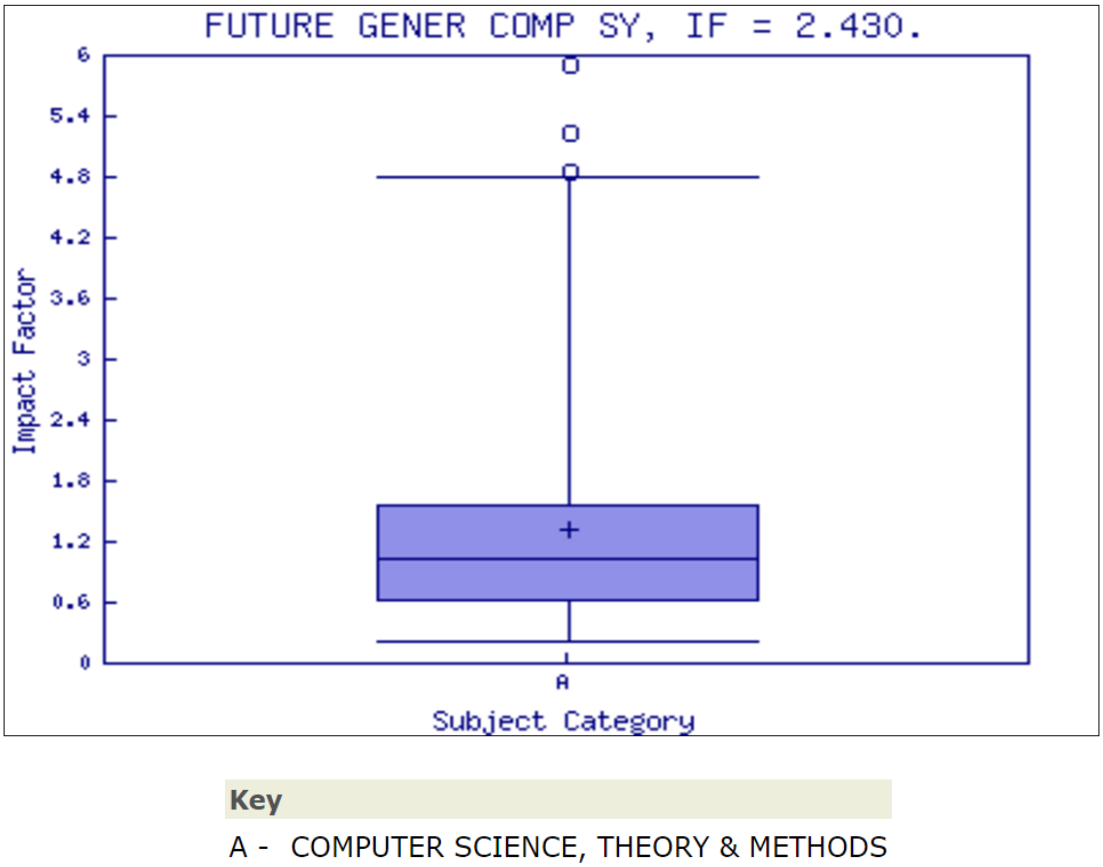


Fig. 8-1 Factor de impacto de la revista "Future Generation Computer Systems"

El artículo es accesible bajo suscripción en el siguiente enlace (Fig. 8-2):

http://dx.doi.org/10.1016/j.future.2016.10.029

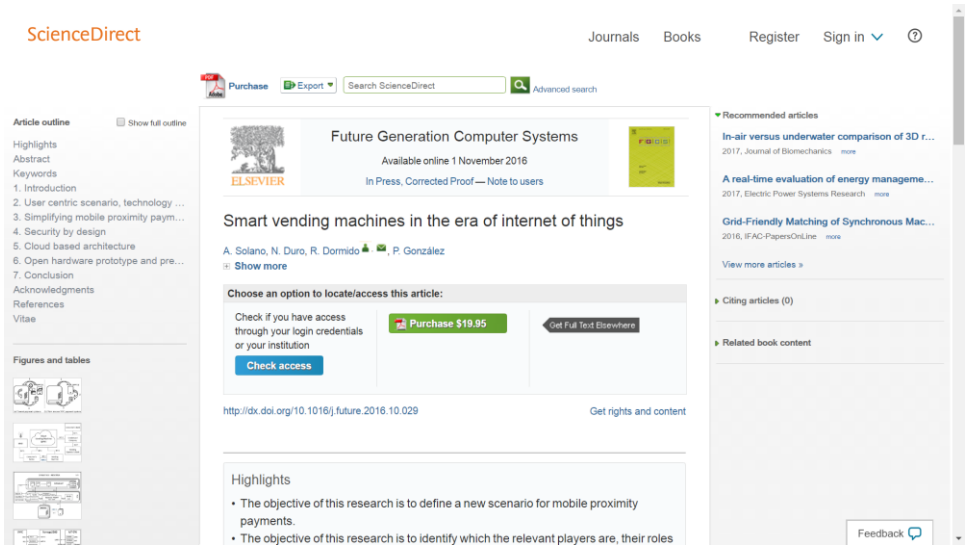
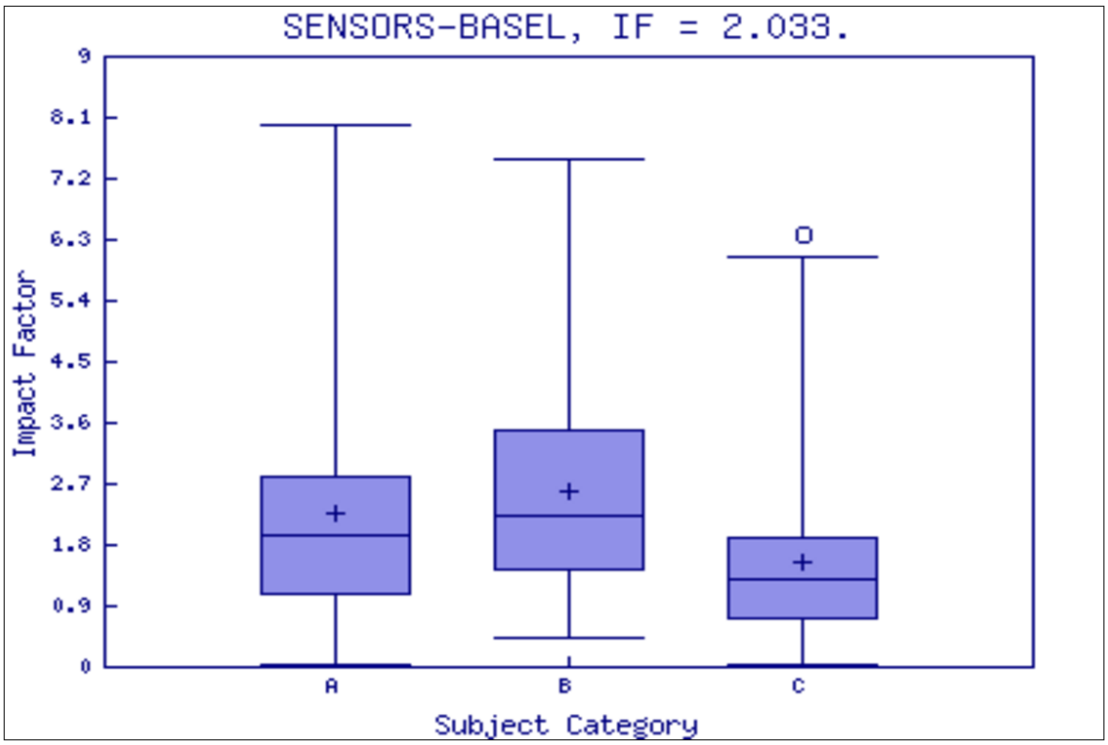


Fig. 8-2 Publicación en SienceDirect del primer artículo

El segundo y tercer artículo, "*A self-provisioning mechanism in OpenStack for IoT devices*" [17] y "*One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices*" [14] respectivamente, están publicados en la revista "SENSORS" (ISSN 1424-8220; CODEN: SENSC9), de la editorial MDPI.

SENSORS figura en el 2015 *JCR Science Edition* con un factor de impacto de 2.033. Es una revista situada en el Q1. En concreto se encuentra la número 12 de 56 dentro de la categoría de INSTRUMENTS and INSTRUMENTATION del JCR 2015 (ver Fig. 8-3).



Key

A - CHEMISTRY, ANALYTICAL
B - ELECTROCHEMISTRY
C - INSTRUMENTS & INSTRUMENTATION

Fig. 8-3 Factor de impacto de la revista “Sensors”

Ambos artículos son de libre acceso en los siguientes enlaces (ver Fig. 8-4 y Fig. 8-5):

http://dx.doi.org/10.3390/s16081306

http://dx.doi.org/10.3390/s16101694

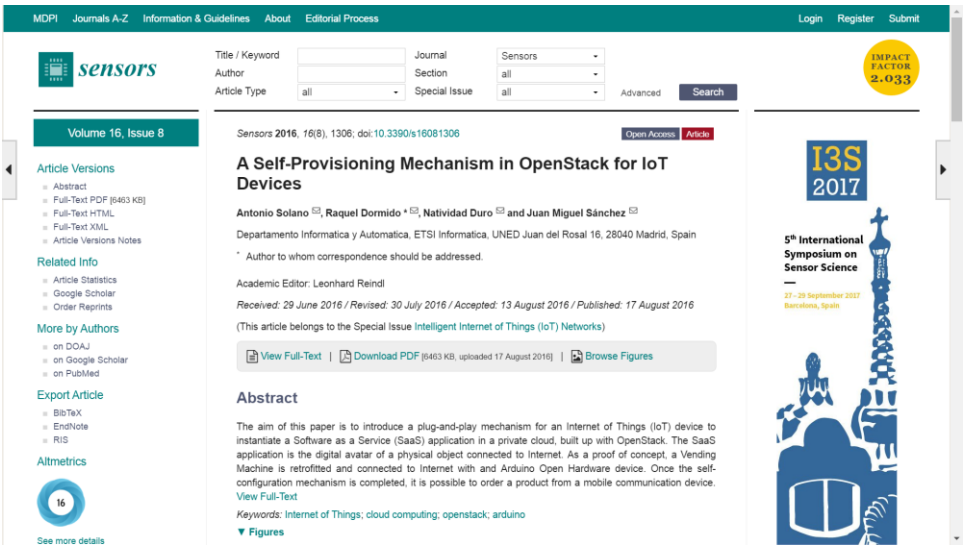


Fig. 8-4 Publicación en Sensors del segundo artículo

Fig. 8-5 Publicación en Sensors del tercer artículo

9 BIBLIOGRAFÍA

[1] "Arduino," [Online]. Available: http://arduino.cc. [Accessed 31 01 2017].

[2] K. Imai, I. Nonak and H. Takeuchi, "Managing the new product development process: how Japanese companies learn and unlearn," *Division of Research, Harvard Business School,* 1984.

[3] O. J. Stache, A. S. Tarroc, A. N. W. Cronstrom, S. M. C. Cabaco, T. A. D. D. N. Pergira and D. Czernous, "Remote vending machine controller". Patent U.S. Patent Application No. 13/151,446., 2012.

[4] O. J. Stache, A. S. Tarroc, A. N. W. Cronstrom, S. M. C. Cabaco, T. A. D. D. N. Pergira and D. Czernous, "Vending machine information". Patent U.S. Patent No. 8,788,359. Washington, DC: U.S. Patent and Trademark Office., 2011.

[5] O. J. Stache, A. S. Tarroc, A. N. W. Cronstrom, S. M. C. Cabaco, T. A. D. D. N. Pergira and D. Czernous, "Vending machine ordering". Patent U.S. Patent Application No. 13/151,402., 2011.

[6] "VendMe," [Online]. Available: http://www.vendme.net/. [Accessed 31 01 2017].

[7] "GTB presents Innovation Awards 2012," 13 06 2012. [Online]. Available: http://www.globaltelecomsbusiness.com/Article/3045679/GTB-presents-Innovation-Awards-2012.html#/.WHpEpFMrLX5. [Accessed 31 01 2017].

[8] H. Derhamy, J. Eliasson, J. Delsing and P. Priller, "A survey of commercial frameworks for the internet of things," in *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), (pp. 1-8). IEEE.*, 2015.

[9] A. Solano Tarroc, "Smart Vending Machines: el papel activo de las Telcos en Vending.," *BIT,* no. 187, pp. 57-60, 2011.

[10] "La UNED y HUAWEI presentan la 'Cátedra Cloud Computing y Big Data UNED-HUAWEI' para impulsar el I+D de tecnologías de vanguardia entre estudiantes de Ingeniería," 24 11 2015. [Online]. Available: http://portal.uned.es/portal/page?_pageid=93,52836153&_dad=portal&_schema=PORTAL. [Accessed 31 01 2017].

[11] "Vodafone IoT Barometer 2016," [Online]. Available: http://www.vodafone.com/business/iot/the-iot-barometer-2016. [Accessed 31 Enero 2017].

[12] A. Solano, N. Duro, R. Dormido and P. González, "Smart vending machines in the era of internet of things," *Future Generation Computer Systems,* 2016.

[13] T. Pflanzner and A. Kertesz, "A survey of IoT cloud providers," *Information and Communication Technology, Electronics and Microelectronics (MIPRO),* vol. 39th International Convention on. IEEE, pp. 730-735, 2016.

[14] A. Solano, R. Dormido, N. Duro and V. González, "One-Time URL: A Proximity Security Mechanism between Internet of Things and Mobile Devices," *Sensors,* vol. 16, no. 10, p. 1694, 2016.

[15] "Módulo de cómputo Intel® Edison," [Online]. Available: Módulo de cómputo Intel® Edison. [Accessed 31 01 2017].

[16] "Servilleteros digitales con pantalla táctil para los bares," *cincodias.com,* 15 12 2014.

[17] A. Solano, R. Dormido, N. Duro and J. M. Sánchez, "A self-provisioning mechanism in OpenStack for IoT devices," *Sensors,* vol. 16, no. 8, p. 1306, 2016.

[18] "OpenVend Demo," [Online]. Available: http://openvend.es/demo. [Accessed 31 01 2017].

[19] "Google Trends," [Online]. Available: https://www.google.es/trends/. [Accessed 31 01 2017].

[20] "Raspberry Pi," [Online]. Available: https://www.raspberrypi.org/. [Accessed 31 01 2017].

[21] E. S. Raymond, " The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary.," O'Reilly Media, Inc., 2001.

[22] "OpenStack: Open source software for creating private and public clouds.," [Online]. Available: https://www.openstack.org/. [Accessed 31 01 2017].

[23] "Hadoop, open-source framework that allows to store and process big data," [Online]. Available: http://hadoop.apache.org/. [Accessed 31 01 2017].

[24] F. Samie, L. Bauer and J. Henkel, "IoT technologies for embedded computing: A survey," *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2016 International Conference on. IEEE, 2016. p. 1-1,* 2016 International Conference on. IEEE, 2016. p. 1-1.

[25] "OpenCart - Open Source Shopping Cart Solution," [Online]. Available: https://www.opencart.com/. [Accessed 31 01 2017].

[26] A. Botta, W. De Donato, V. Persico and A. Pescapé, "Integration of cloud computing and internet of things: a survey," *Future Generation Computer Systems, 56, 684-700,* no. 56, pp. 684-700, 2016.

[27] M. Díaz, C. Martín and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing," *Journal of Network and Computer Applications,* vol. 67, pp. 99-117, 2016.

[28] "The Apache HTTP Server Project," [Online]. Available: https://httpd.apache.org/.

[29] "MySQL, the world's most popular open source database," [Online]. Available: https://www.mysql.com/.

[30] R. T. Fielding, "Architectural styles and the design of network-based software architectures," *Doctoral dissertation, University of California, Irvine,* 2000.

[31] "Slim a micro framework for PHP," [Online]. Available: https://www.slimframework.com/. [Accessed 31 01 2017].

[32] A. F. Mohammed, V. T. Humbe and S. S. Chowhan, "A review of big data environment and its related technologies," *Information Communication and Embedded Systems (ICICES),* vol. International Conference on, pp. 1-5, 2016 International Conference on. IEEE, 2016. p. 1-5..

[33] F. Alam, R. Mehmood, I. Katib and A. Albeshri, "Analysis of Eight Data Mining Algorithms for Smarter Internet of Things (IoT). Procedia Computer Science, 98, 437-442," *Procedia Computer Science,* no. 98, pp. 437-442, 2016.

[34] "HTML5," [Online]. Available: https://www.w3.org/TR/html5/. [Accessed 31 01 2017].

[35] H.-C. Lin and G. Lee, "Building a Secure Cross Platform Enterprise Service Mobile Apps Using HTML5," *Network-Based Information Systems,* vol. 18th International Conference on, pp. 162 - 166, 2015.

[36] K. Takasu, T. Saito, T. Yamada and T. Ishikawa, "A Survey of Hardware Features in Modern Browsers: 2015 Edition," *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS),* no. 9th International Conference on Innovative Mobile and Internet Services , 2015 .

[37] "LocalStorage," [Online]. Available: http://www.w3.org/TR/webstorage.

[38] "CORS," [Online]. Available: http://www.w3.org/TR/cors/.

[39] "7 Technologies Underpin the Hype Cycle for the Internet of Things, 2016," [Online]. Available: http://www.gartner.com/smarterwithgartner/7-technologies-underpin-the-hype-cycle-for-the-internet-of-things-2016/. [Accessed 31 01 2017].

[40] Y.-P. E. Wang, X. Lin, A. Adhikary, A. Grövlen, Y. Sui, Y. Blankenship, J. Bergman and H. S. Razaghi, "A Primer on 3GPP Narrowband Internet of Things (NB-IoT)," 2016.

[41] M. De Reuver and J. Ondrus, "When technological superiority is not enough: The struggle to impose the SIM card as the NFC Secure Element for mobile payment platforms," *Telecommunications Policy,* 2017.

[42] O. Sajid and M. Haddara, "NFC mobile payments: Are we ready for them?," *SAI Computing Conference (SAI), IEEE,* pp. 960-967, 2016.

[43] A. Solano Tarroc, "Open Hardware & Software para el Vending en la era de la Internet de las Cosas," *Mundo Vending,* vol. nº 144, Jul-Ago. 2015.

[44] "OpenVend, building the Internet of Vending Machines," [Online]. Available: http://openvend.es/vision-y-mision-de-openvend. [Accessed 31 01 2017].

[45] J. I. Mantero Ruiz, "Web-app de compra para smartphones aplicado a máquinas de vending," UNED - PFC INF 0432 00002212404, 2014.

[46] I. Pérez Martínez, "Software de fidelización en máquinas de vending para smatphones y HTML5," UNED PFC INF 0429 00002212401, 2014.

[47] F. J. García Mateu, "Software de inventarios y planificación de rutas en máquinas de vending y HTMLS," PFC INF 0600 00002440954, 2015.

[48] P. Ballesteros Fernández, "Pagos por móvil con Arduino & NFC para la Internet de las Cosas," UNED PFC INF 0501 00002213533, 2014.

[49] Á. Cifuentes Sabio, "Pagos con móvil con arduino de NFC para internet de las cosas," UNED PFG INF 0014 00002214470, 2014.

[50] R. J. Cárdenes Medina, "Internet de las cosas para máquinas expendedoras," UNED PFC INF 0536 00002442686, 2015.

[51] J. M. Sánchez García, "Construyendo PaaS con Ubuntu y OpenStack para Internet de las Cosas," PFC INF 0502 00002213534, 2014.

[52] V. J. González Rodríguez, "Creando Mobile WebApps con HTML5/CSS3/JavaScript para Internet de las Cosas," 2015 PFC INF 0525 00002442675.

[53] R. Á. Alonso Álvarez, "Construyendo PaaS con Ubuntu y OpenStack para internet de las cosas," UNED PFC INF 0545 00002442695, 2015.

[54] L. M. Cabezas Clavo, "Construyendo paas con Ubuntu y Openstack para internet de las cosas," UNED PFG INF 0015 00002214471, 2015.

[55] P. Martínez Luque, "Big Data para un Cloud Paas en internet de las cosas," UNED PFC INF 0569 00002440923, 2015.

[56] J. D. Fernández Rijo, "Pagos por móvil con Arduino & NFC para la internet de las cosas," UNED PFC INF 0555 00002442705, 2015.

[57] F. J. González Saíz, "Redes malladas con Arduino para internet de las cosas," UNED PFC INF 0577 00002440931, 2015.

[58] D. Díaz Pérez, "Pagos por el móvil con Arduino y NFC para el internet de las cosas," UNED PFG, 2016.

[59] M. García Jiménez, "Pago por móvil & NFC para la internet de las cosas," UNED PFG, 2016.

[60] F. J. Paños Merino, "OpenVend el internet de las máquinas de vending," UNED PFG, 2016.

[61] X. Mira Fernández, "BigData para Internet de las Cosas," UNED PFG, 2016.

10 GLOSARIO Y ACRÓNIMOS

A

API

Application Programming Interface, - 72 -, - 73 -, - 76 -, - 78 -, - 79 -, - 102 -, - 105 -, - 108 -

B

BLE

Bluetooth Low Energy, - 48 -, - 89 -, - 96 -, - 117 -, - 129 -, - 161 -

C

CORS

Cross Origin Resource Sharing, - 37 -, - 91 -, - 105 -, - 106 -

E

eMTC

enhanced Machine Type Communication, - 129 -

I

IaaS

Infraestructure as a Service, - 3 -, - 25 -, - 41 -, - 54 -, - 55 -, - 62 -, - 66 -, - 73 -, - 76 -, - 140 -, - 141 -, - 162 -, - 165 -, - 167 -, - 173 -

L

LPWA

Low-Power Wide-Area, - 129 -

M

M2M

Machine to Machine, - 7 -, - 43 -, - 125 -

MDB

Multi Drop Bus, - 43 -, - 71 -, - 92 -, - 113 -, - 139 -, - 173 -

N

NB-IoT

Narrow Band Internet of Things, - 123 -, - 129 -

NFC

Near Field Communication, - 3 -, - 4 -, - 13 -, - 17 -, - 37 -, - 38 -, - 42 -, - 46 -, - 47 -, - 48 -, - 49 -, - 50 -, - 51 -, - 56 -, - 57 -, - 58 -, - 62 -, - 89 -, - 90 -, - 94 -, - 95 -, - 97 -, - 98 -, - 112 -, - 113 -, - 114 -, - 117 -, - 125 -, - 126 -, - 129 -, - 139 -, - 142 -, - 144 -, - 161 -, - 164 -, - 173 -

P

PaaS

Platform as a Service, - 25 -, - 41 -, - 54 -, - 56 -, - 66 -, - 72 -, - 73 -, - 76 -, - 81 -, - 88 -, - 140 -, - 141 -, - 142 -, - 144 -, - 146 -, - 159 -, - 160 -, - 162 -, - 163 -, - 164 -, - 165 -, - 167 -, - 168 -, - 173 -

PYMES

Pequeñas y Medianas Empresas, - 11 -, - 12 -, - 162 -, - 167 -

Q

S

T

U

ANEXO A. PROYECTOS FIN DE CARRERA OFERTADOS

A.1 *Digital signage* con Arduino para IoT

La evolución de las tecnologías de comunicación está permitiendo la conexión de todo tipo de dispositivos a Internet a bajo coste, con un menor consumo y con tamaños reducidos, conformándose lo que se ha denominado la "Internet of Things" (IoT), que en otras palabras, facilita que el mundo digital interactúe con el mundo físico.

Arduino es un dispositivo programable que nos permite interactuar con nuestro entorno. Es básicamente un microcontrolador sobre una plataforma abierta de *hardware* extensible con módulos de comunicaciones, sensores y actuadores. La comunidad de Arduino no deja de crecer gracias a la publicación de librerías de código abierto en C/C++.

Este proyecto consiste en una implementación concreta de este tipo de tecnología en un escenario de digital signage para distribuir imágenes desde un nodo central a multitud de Arduinos provistos de una pantalla táctil. El reto será integrar la parte de digital signage con desarrollos de otros Proyectos Fin de Carrera basados en Arduino que incluyen sistemas de pago y encriptado de las comunicaciones y todo ello corriendo en el mismo microprocesador con recursos muy limitados.

Este proyecto se engloba dentro de una iniciativa para proporcionar un Cloud PaaS ("Platform as a Service") de Código Abierto para la "Internet of Things" aplicando los nuevos paradigmas de "Cloud Computing". Por tanto, será necesario cooperar con otros proyectos para su posible integración.

Requisitos:

- Conocimientos de C/C++, Linux
- RESTful/JSON, PHP, MySQL

A.2 Redes malladas con Arduino para IoT

La evolución de las tecnologías de comunicación está permitiendo la conexión de todo tipo de dispositivos a Internet a bajo coste, con un menor consumo y con tamaños reducidos, conformándose lo que se ha denominado la "Internet of Things" (IoT), que en otras palabras, facilita que el mundo digital interactúe con el mundo físico.

Arduino es un dispositivo programable que nos permite interactuar con nuestro entorno. Es básicamente un microcontrolador sobre una plataforma abierta de *hardware* extensible con módulos de comunicaciones, sensores y actuadores. La comunidad de Arduino no deja de crecer gracias a la publicación de librerías de código abierto en C/C++.

Este proyecto consiste en una implementación concreta de este tipo de tecnología para crear redes malladas con salida a Internet a través de un nodo concentrador. Como aplicación práctica se pretende cubrir edificios con máquinas de vending distribuidas en distintas plantas donde puede haber problemas de cobertura. Para el envió de la información a la nube, se evaluarán y prototiparán el mayor número de tecnologías de comunicaciones disponibles: Mesh networks con RF24, Zigbee, Bluetooth 4, GPRS/3G/LTE, Wifi , Ethernet, PLC, Neul/Sigfox etc. El reto será integrar las comunicaciones con desarrollos de otros Proyectos Fin de Carrera basados en Arduino que incluyen telemetría, sistemas de pago y encriptado de las comunicaciones y todo ello corriendo en la mismo microprocesador con recursos muy limitados.

Este proyecto se engloba dentro de una iniciativa para proporcionar un Cloud PaaS ("Platform as a Service") de Código Abierto para la "Internet of Things" aplicando los nuevos paradigmas de "*Cloud Computing*". Por tanto, será necesario cooperar con otros proyectos para su posible integración.

Requisitos:

- Conocimientos de C/C++, Linux
- RESTful/JSON, PHP, MySQL

A.3 Pagos por móvil en proximidad con Arduino para IoT

La evolución de las tecnologías de comunicación está permitiendo la conexión de todo tipo de dispositivos a Internet a bajo coste, con un menor consumo y con tamaños reducidos, conformándose lo que se ha denominado la "Internet of Things" (IoT), que en otras palabras, facilita que el mundo digital interactúe con el mundo físico. En este sentido la llegada de módulos electrónicos de comunicaciones de muy bajo coste: "Near Field Communication" (NFC), Bluetooth Low Power (BLE) o WiFi entre otras tecnologías, nos permite vincular usuarios y máquinas a través del móvil, impulsando la Internet de las Cosas.

Por otro lado, Arduino es un dispositivo programable que nos permite interactuar con nuestro entorno. Es básicamente un microcontrolador sobre una plataforma abierta de *hardware* extensible con módulos de comunicaciones, sensores y actuadores. La comunidad de Arduino no deja de crecer gracias a la publicación de librerías de código abierto en C/C++.

Este proyecto consiste en una implementación concreta de este tipo de tecnologías en distintos puntos de venta: restaurantes, pequeño comercio, parkings, vending etc. El módulo de Arduino deberá comunicarse con el punto de venta, confirmar el pago y actuar sobre distintos periféricos, por ejemplo: abrir barrera en parking, imprimir ticket en restaurantes, entregar producto en vending, etc. El objetivo será validar en tiempo real transacciones electrónicas iniciadas principalmente desde el navegador del móvil (HTML5) para pagos por proximidad. Se prototiparán pagos con sistemas cerrados *online* o contra tarjetas de prepago. También se evaluarán los pagos con sistemas abiertos de tarjetas de crédito. Se analizará la viabilidad de certificación contra Visa y Mastercard y el uso de web services contra las pasarelas de comercio electrónico: Paypal, TPV virtuales de RedSys, etc.

El reto será integrar el sistema de pago de Arduino con desarrollos de otros Proyectos Fin de Carrera basados en Arduino que incluyen telemetría y encriptado de las comunicaciones, y todo ello, corriendo en un mismo microprocesador con recursos muy limitados.

Este proyecto se engloba dentro de una iniciativa para proporcionar un Cloud PaaS ("Platform as a Service") de Código Abierto para la "Internet of Things" aplicando los nuevos paradigmas de "*Cloud Computing*". Por tanto, será necesario cooperar con otros proyectos para su posible integración.

Requisitos:

- Conocimientos de C/C++, Linux
- PHP, MySQL, REST/JSON
- HTML5/CSS3, Javascript

A.4 Construyendo un PaaS con OpenStack para IoT

Rackspace y la NASA crearon en 2010 un proyecto conjunto de *software* libre para montar plataformas de almacenamiento y computación en la nube. Le pusieron el nombre de OpenStack y se ha convertido en un estándar para IaaS, siendo la alternativa comercial más cercana y conocida del servicio AWS (Amazon Web Service) de Amazon.

Este proyecto consiste en la implementación concreta de este tipo de tecnología para crear un OpenStack Cloud con Ubuntu 14.04 LTS. Sobre el IaaS resultante se implantará se diseñará e implementará los cimientos de una Plataforma como un Servicio (PaaS) y para ellos se tomarán como modelos de referencia plataformas como Xively.com, Nimbits.com o similares. Es decir, se almacenarán y procesarán en tiempo real datos proporcionados por redes de sensores y/o módulos m2m, y preferiblemente, se hará uso de la tecnología Hadoop *Big Data* que permita procesar los datos y crear nuevos servicios. Una vez diseñada la PaaS, se le dotará de una capa de aplicaciones que se ofertará como SaaS. Como resultado final se implantará un orquestador que con un solo clic proporcione un Software de Gestión en modelo SaaS orientado a PYMES.

Este proyecto está relacionado con otros proyectos de diseño de módulos m2m y sus aplicaciones SaaS pensadas para distintos sectores verticales como coches conectados, contadores de suministros, máquinas de vending, etc. Por tanto, el foco no está en dichos componentes, sino más bien en como desacoplar los módulos del aplicativo mediante a una plataforma horizontal que con un diseño innovador de PaaS, permita a desarrolladores y a terceros acceder a un entorno remoto de desarrollo e integración, que al estar pre-configurado aceleré el "time to market" de nuevos servicios en la Internet de las Cosas.

Requisitos:

- Conocimientos de C/C++, Linux
- PHP, MySQL, REST/JSON
- HTML5/CSS3, Javascript

A.5 Creando mobile webapps con HTML5 para IoT

La tecnología HTML5 es un estándar que ya está disponible para desarrolladores y empresas y ofrece al usuario aplicaciones muy similares a las aplicaciones nativas con las ventajas que ofrecen las aplicaciones Web.

Este proyecto pretende aplicar dichas tecnologías para crear webapps para móviles. El diseño se hará pensando en pantallas táctiles haciendo uso extensivo de técnicas de "*drag-and-drop*" y "*gestures*" y se generaran páginas web dinámicas ("*responsive*") que se adapten al tamaño de las pantallas de los smartphones y tablets.

En este sentido, se pretende reescribir las páginas PHP de paquetes de Software Abierto de comercio electrónico como OpenCart que siguen el patrón MVC (Modelo Vista Controlador) y crear extensiones HTML5 que proporcionen innovadoras interfaces gráficas. El objetivo es transformar cualquier smartphone que soporte HTML5 en un TPV móvil que soporte transacciones de comercio electrónico "*onsite*", es decir, en el punto de venta físico donde se está realizando la transacción de bienes y/o servicios.

Este proyecto se engloba dentro de una iniciativa para proporcionar un Cloud PaaS ("Platform as a Service") de Código Abierto para la "Internet of Things" aplicando los nuevos paradigmas de "*Cloud Computing*", y por otro lado con dispositivos Arduino Open Hardware a instalarse en el punto de venta. Por tanto, será necesario cooperar con otros proyectos para su posible integración y posible ampliación del *Dashboard* del PaaS. Para el *dashboard*, se ha seleccionado el framework SPRING por seguir el patrón MVC (Modelo Vista Controlador) y por soportar el desarrollo de los servicios RESTFUL de integración entre con el PaaS.

Requisitos:
- HTML5/CSS3, Javascript, Ajax, jQuery
- PHP, MySQL, REST/JSON

A.6 *Big Data* para un *Cloud* PaaS en IoT

Hadoop se creó en el año 2006 a raíz de la necesidad de sistemas nuevos para gestionar la explosión de datos de la web. De descarga gratuita, y libre para potenciarlo y mejorarlo, Hadoop es un método de código abierto para almacenar y procesar los datos que «permite el procesamiento en paralelo distribuido de enormes cantidades de datos en servidores estándar del sector, económicos, que almacenan y procesan los datos, y que pueden escalarse sin límite» (Fuente: HADOOP).

Este proyecto se engloba dentro de una iniciativa para proporcionar un Cloud PaaS ("Platform as a Service") de Código Abierto para la "Internet of Things" aplicando los nuevos paradigmas de "*Cloud Computing*" y "*Big Data*". Este proyecto consiste en una implementación concreta de este tipo de tecnologías en máquinas auto expendedoras (vending) que previamente han sido conectadas con dispositivos open *hardware* (Arduino) para telemetría y para poder realizar pagos con móviles con tecnología NFC ("Near Field Communications").

El procesado de los datos de telemetría tendrá como objetivo la generación de eficiencias operativas. Para ello se propondrán modelos de optimización de inventarios, rutas de reposición, product-mix en las máquinas, etc. Por otro lado, gracias a la vinculación entre máquinas de vending y sus usuarios, se pretende mejorar la experiencia de compra y crear programas de fidelización que permitan por ejemplo modificar dinámicamente los precios de los productos de la máquina de vending para un usuario en concreto, crear promociones personalizadas, etc.

Para la presentación de los datos, se complementará el proyecto con el diseño de un Portal basado en webapps. Las webapps se desarrollarán en HTML5, CSS3 y JavaScript y estarán optimizadas para dispositivos móviles con pantallas táctiles. Se ha seleccionado el framework SPRING por seguir el patrón MVC (Modelo Vista Controlador) y por soporta el desarrollo de los servicios RESTFUL de integración entre los módulos del PaaS para el Vending y los propios módulos de Hadoop.

Este proyecto es una continuación de otros proyectos y se dispone de un IaaS/PaaS para Hadoop.

Requisitos:

- Conocimientos de C/C++, Linux, REST/JSON
- HTML5/CSS3, Javascript, Ajax, jQuery

ANEXO B. MODELO DE PREINSCRIPCIÓN

Título: **Construyendo PaaS con Ubuntu y OpenStack para Internet de las Cosas**

Breve descripción:

Rackspace y la NASA crearon en 2010 un proyecto conjunto de *software* libre para montar plataformas de almacenamiento y computación en la nube. Le pusieron el nombre de OpenStack y se ha convertido en un estándar para IaaS, siendo la alternativa comercial más cercana y conocida del servicio AWS (Amazon Web Service) de Amazon.

Este proyecto consiste en la implementación concreta de este tipo de tecnología para crear un OpenStack Cloud con Ubuntu 12.04 LTS. Sobre el IaaS resultante se diseñará, e implementará, los cimientos de una Plataforma como un Servicio (PaaS), y para ellos se tomarán como modelos de referencia plataformas como Cosm.com, Nimbits.com, o similares. Es decir, se almacenarán y procesarán en tiempo real datos proporcionados por redes de sensores y/o módulos m2m, y preferiblemente, se hará uso de la tecnología Hadoop *Big Data* que permita procesar los datos y crear nuevos servicios. Una vez diseñada la PaaS, se le dotará de una capa de aplicaciones que se ofertará como SaaS. Como resultado final se implantará un orquestador que con un solo click proporcione un Software de Gestión en modelo SaaS orientado a PYMES.

Este proyecto está relacionado con otros proyectos de diseño de módulos m2m y sus aplicaciones SaaS pensadas para distintos sectores verticales como coches conectados, contadores de suministros, máquinas de vending, etc. Por tanto, el foco no está en dichos componentes, sino más bien en como desacoplar los módulos del aplicativo, mediante a una plataforma horizontal que con un

diseño innovador de PaaS, permita a desarrolladores y a terceros acceder a un entorno remoto de desarrollo e integración, que al estar pre-configurado acelere el “time to market” de nuevos servicios en la Internet de las Cosas.

Objetivos

Este PFC se caracteriza por estar vinculado a otros PFC que tutelados bajo el mismo Director de PFC, y conforman áreas de intereses de dicho Director de PFC para la realización de su tesis doctoral en el Departamento de Informática y Automática de ETSI Informática de la UNED.

Por tanto, su objetivo es doble. Por un lado, se espera que los estudiantes sean capaces de implementar un prototipo funcional por sus propios medios, y por otro lado, que trabajen en equipo con la metodología “*Agile*” para integrar su prototipo con otros módulos funcionales, con el objetivo final de demostrar conjuntamente una solución extremo a extremo aplicada al sector del Vending.

Método de desarrollo, fases del trabajo y fechas de realización:

En lo que respecta al desarrollo y la gestión del *software* se van a utilizar un marco de trabajo basado en la metodología “*Scrum*” (http://es.wikipedia.org/wiki/*Scrum*), que básicamente es un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo de *software* “*Agile*”.

Un principio clave de *Scrum* es reconocer que durante un proyecto los “clientes” pueden cambiar de idea sobre lo que quieren y necesitan. Por lo tanto, *Scrum* adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes. *Scrum* permite la creación de equipos auto-organizados y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.

Roles en *Scrum*

***Product owner*:** Representa la voz del cliente y será asumido por el Director de PFC. Se asegura de que el equipo *Scrum* trabaja de forma adecuada desde la perspectiva del negocio. El *Product owner* escribe historias de usuario, las prioriza, y las coloca en el *Product Backlog*.

***Scrum Master* (o Facilitador):** El *Scrum* es facilitado por un *Scrum Master* cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del *Sprint*. El *Scrum Master* no es el líder del equipo (porque ellos se auto-organizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El *Scrum Master* se asegura de que el proceso *Scrum* se utiliza como es debido. El *Scrum Master* es el que hace que las reglas se cumplan. El *Scrum Master* rotará entre los participantes de los distintos PFC vinculados.

Equipo de desarrollo. En este caso el equipo estará formado por los alumnos realizando el PFC y que tienen la responsabilidad de entregar el producto. En principio estará compuesto por un pequeño equipo de 4 a 6 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc).

Supervisores del PFC. Son los profesores de la UNED encargados de la tutela del PFC y que participan activamente durante las revisiones del *Sprint*.

***Stakeholders* (Clientes, Proveedores, Vendedores, etc)**. Un aspecto importante de una aproximación ágil es la práctica de involucrar en el proceso a los usuarios, expertos del negocio y otros interesados. Es importante que esa gente participe y entregue retroalimentación con respecto a la salida del proceso a fin de revisar y planear cada *Sprint*. Se refiere por tanto a la gente que hace posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su producción. Sólo participan directamente durante las revisiones del *Sprint*.

Reuniones en *Scrum*

***Weekly Scrum*.** Si bien estas reuniones son diarias para equipos co-localizados, al tratarse de la UNED se adaptarán para ser reuniones por teleconferencia y semanales.

El *scrum* tiene unas guías específicas:

- La reunión comienza puntualmente a su hora.
- La reunión tiene una duración fija de 30 minutos, de forma independiente del tamaño del equipo.
- Durante la reunión, cada miembro del equipo contesta a tres preguntas:

  1. ¿Qué has hecho desde la semana pasada?
  2. ¿Qué es lo que estás planeando hacer esta semana?
  3. ¿Has tenido algún problema que te haya impedido alcanzar tu objetivo? (Es el papel del *Scrum Master* recordar estos impedimentos).

Reunión Mensual de Planificación del *Sprint* (*Sprint* Planning Meeting). Al inicio del ciclo *Sprint* (cada mes), una "Reunión de Planificación del *Sprint*" se lleva a cabo, donde se debe:

- Seleccionar qué trabajo se hará.
- Preparar, con el equipo completo, el *Sprint Backlog* que detalla el tiempo que tomará hacer el trabajo.
- Identificar y comunicar cuánto del trabajo es probable que se realice durante el actual Sprint.
- 1 hora como límite.

Al final del ciclo *Sprint*, dos reuniones se llevarán a cabo: la “Reunión de Revisión del *Sprint*” y la “Retrospectiva del *Sprint*”.

Reunión de Revisión del *Sprint* (*Sprint* Review Meeting). En ella se debe:

- Revisar el trabajo que fue completado y no completado.
- Presentar el trabajo completado (alias “demo”) a los interesados con participación del Director de PFC y supervisoras de la UNED.
- El trabajo incompleto no puede ser demostrado.
- 30 min como límite.

Retrospectiva del *Sprint* (*Sprint Retrospective*)

Después de cada *Sprint*, se lleva a cabo una retrospectiva del *Sprint*, en la cual todos los miembros del equipo dejan sus impresiones sobre el *Sprint* recién superado. El propósito de la retrospectiva es realizar una mejora continua del proceso. Esta reunión tiene un tiempo fijo de 15 minutos.

Planificación

El PFC constará de 8 *Sprints* de octubre a junio que se materializarán con prototipos funcionales. El alumno tendrá hasta la primera convocatoria de octubre 2014 para completar la documentación necesaria y poder presentar el PFC de manera colectiva con el resto de participantes. Recalcar que sólo se asegura la participación activa del director del PFC y la colaboración del resto de participantes hasta la primera convocatoria. De no presentarse en la convocatoria de octubre, el alumno dispondrá de tres convocatorias: diciembre, marzo y junio tal y como marca la normativa, si bien su defensa se hará de manera aislada ya que no se podrá garantizar la colaboración y defensa conjunta con el resto de participantes.

Se proponen las siguientes fechas y horarios para realización de las multi conferencias, si bien se podrá acordar cambios de día y horario si existe quórum entre los participantes:

Reunión semanal de *Scrum*: Todos los lunes de 20h30 a 21h00, 30 minutos máximo, independiente de los participantes.

Reunión Mensual de Planificación del *Sprint*: Primer Lunes de cada mes de 20h30 a 21h30 (primer *Sprint*: 7 de octubre).

Reunión de Revisión del *Sprint*: Último Jueves de cada mes de 20h30 a 21h00.

Retrospectiva del *Sprint*: Último Jueves de cada mes de 21h00 a 21h15.

Nota 1: La reiterada e injustificada falta de asistencia a las reuniones en remoto implicará la expulsión del programa y la necesidad por parte del alumno de buscarse un nuevo Director de PFC. En la medida de lo posible se flexibilizará las fechas y horarios de las reuniones siempre y cuando se notifique con la suficiente antelación para que haya quórum entre los participantes.

Nota 2: Se calcula en unas 3 horas mensuales la participación en reuniones de Scrum, por tanto, a lo largo de los 8 Sprints los alumnos habrán acumulado 24 horas de dedicación al PFC. En este sentido los tutores intercederán ante la comisión de evaluación del PFC para que se tenga en consideración en la nota final.

Medios a utilizar y breve justificación de la pertinencia de los mismos:

Se trabajará preferente con *Open Source*.

El director de proyecto pondrá a disposición de los participantes:

- Acceso sftp y ssh a un servidor Linux con PHP/MySql y subdominios en 1and1.es donde poder ejecutar y personalizar paquetes open source como OpenCart, SugarCRM, etc.

- Acceso online a Trello como herramienta sencilla de Scrum y uso de Teamviewer para las conferencias remotas.
- Un simulador HW de una Máquina de Vending Wurlitzer con protocolo DEX y MDB.
- Módulos y Shells Arduino compatibles con Wifi, Gprs, Ethernet, Zigbee y NFC para realizar test de integración y demostrador extremo a extremo.
- Ubuntu Cloud/OpenStack para validar tecnologías Coud IaaS/PaaS/SaaS y Big Data.
- La UNED facilitará el acceso a sus laboratorios, servidores, centros de datos, etc, siempre y cuando el profesorado de la UNED lo estime pertinente para alcanzar los objetivos del PFC.
- Se asume que los alumnos dispondrán de su propio ordenador con capacidad para ejecutar instancias locales del entorno de desarrollo y validarán, en el mayor número de dispositivos móviles, ya sean suyos o de conocidos, las webapps desarrolladas en HTML5/CS3/JS.

Firmado: el Estudiante

Firmado: el Director

Firmado: el Supervisor