EMS in Cloud

Departamento de Arquitectura de Computadores y Automática.
Facultad de Informática
Universidad Complutense de Madrid

Proyecto de Sistemas Informáticos

Curso 2013/2014

Autores:
Alejandra González Reyes
Claudia Montero Aneiros
Ricardo Champa Bujaico

Director:
José Luis Vázquez Poletti

Alejandra González Reyes, Claudia Montero Aneiros y Ricardo Champa Bujaico, autores del presente documento y del proyecto “EMS in Cloud” autorizan a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado.

23 de Junio de 2014

Alejandra González Reyes Claudia Montero Aneiros Ricardo Champa Bujaico

Agradecimientos

La realización de este proyecto y los objetivos alcanzados no serían posibles sin el apoyo de las personas que nos rodean. Por eso, queremos dedicar este trabajo en primer lugar a nuestro director de proyecto José Luis Vázquez Poletti, por sus consejos, interés y colaboración. A Manuel Vázquez, por prestarnos su ayuda y sus conocimientos durante este año de trabajo para conseguir nuestros objetivos.
A cada uno de los profesores que nos han guiado durante estos años.
Y por último, también a nuestras familias,
sin ellas no estaríamos hoy aquí, a nuestros amigos, compañeros y parejas por todo el apoyo y ánimo recibido de ellos.

Resumen

En el mundo de las emergencias médicas, cada segundo es valioso. Los equipos médicos tienen la tarea fundamental y a su vez nada sencilla de salvar vidas. Para llevar a cabo su trabajo, necesitan asistencia tanto para la consulta de los protocolos a seguir como para la realización de cálculos con la finalidad de suministrar los medicamentos necesarios.

Las herramientas de las que se disponen no son cómodas ni tampoco personalizables. Estos aspectos se podrían mejorar haciendo uso de los recursos tecnológicos de las que disponemos hoy en día, como por ejemplo ordenadores y dispositivos móviles. Es por esto que, siguiendo la petición de un profesional sanitario, decidimos cambiar esta situación aprovechando los recursos de las aplicaciones software basadas en sistemas de Cloud.

La solución propuesta es la que presentamos en este documento: EMS in Cloud. Se trata de un software como servicio desplegado en múltiples plataformas. Este proyecto ofrece una solución económica y escalable que permitirá poner a disposición de equipos sanitarios una herramienta profesional.

Palabras clave: Medicamentos, Protocolos, Parser, Nube, Multiplataforma, Android, Servicios Sanitarios, Bootstrap

Abstract

In the world of medical emergencies, every second matters. The medical teams main target is save lives. To accomplish their work, medical teams need check protocols to follow and performing calculations in order to provide the necessary medicines.

The tools available are not helpful enough and customizable neither. These aspects could be improved taking advantage from technological resources available nowadays, such as computers and mobile devices. Because of this, and following the request of a medical professional, we decided to change this approach using the resources of software applications based on cloud systems.

The solution proposed in this document is EMS in Cloud. We are using a software as a service approach to develop a cross platform project.
This proyect offers an economic and scalable solution, making available to healthcare teams a professional work tool.

Keywords: Medicines, Protocols, Parser, Cloud, Cross platform, Android, Medical Service, Bootstrap.

Tabla de contenido

Índice de Ilustraciones

Índice de tablas

Capítulo 1

Descripción del problema

1.1 Introducción

En este capítulo presentamos una visión general de la cuestión. Primero abordaremos el trabajo de campo de los servicios sanitarios con el objetivo de acercar al lector a su sistema de trabajo y los problemas que éste presenta. A continuación, explicaremos más en detalle esos inconvenientes y las soluciones adoptadas para ellos, con sus correspondientes desventajas, lo que para nosotros constituye nuestra motivación. Por último expondremos cómo hemos decidido solventar estos problemas, presentando nuestro sistema de asistencia a las emergencias médicas.

1.2 Las urgencias médicas: una visión general

Una urgencia médica es toda situación que requiera una actuación médica inmediata, independientemente del lugar y el momento. Los servicios sanitarios están preparados para atenderlas, pero los miembros de esos equipos, realmente, no saben a lo que se van a enfrentar hasta llegar al lugar de los hechos. En el desplazamiento desde su centro de trabajo al lugar de la urgencia, los profesionales repasan los protocolos de actuación que consideran con mayor posibilidad de aplicación en el lugar al que se dirigen.

Una vez han llegado al sitio, los miembros del equipo sanitario deben poner en práctica los protocolos adecuados. Esto no es una tarea fácil. Los protocolos son documentos que describen la secuencia del proceso de atención de un paciente. De forma muy general, están formados por diagnósticos y por criterios a tener en cuenta para elegir el diagnóstico adecuado. Un protocolo puede ser un documento de más de cien páginas, por lo que todos ellos se resumen en un diagrama de flujo. Son estos diagramas los que siguen los médicos y enfermeros en la urgencia. Muchos de los criterios de toma de decisiones en los protocolos son valores concretos, difíciles de memorizar en una situación de estrés. Además, en los diagnósticos que se van adoptando también se pueden encontrar medicamentos. En ese caso, si se quiere consultar algo sobre un fármaco en concreto, no queda más remedio que consultar el vademécum.

Además de seguir los protocolos, los profesionales deben ejecutarlos. Su trabajo no es sencillo, y a menudo hay vidas humanas dependiendo de ellos. Es necesario que tengan la información necesaria para llevar a cabo los protocolos y consultar los medicamentos necesarios de forma eficiente, rápida y sencilla.

1.3 Motivación

Los protocolos de actuación médicos se presentan en libros, pero tienen muchas páginas y no son cómodos de llevar. Si los profesionales no se quieren llevar esos grandes tomos, existen compendios disponibles. Obviamente, los compendios presentan la selección que considera el autor del libro.

En cuanto a los medicamentos, los médicos y enfermeros tienen disponible, como ya hemos dicho antes, el vademécum. Este libro está constituido por toda la colección de fármacos disponibles, no para ellos, si no en el mercado, por lo que presenta mucha información innecesaria que hace de la búsqueda de fármacos una labor más lenta y tediosa.

Los medicamentos no siempre se presentan en la posología acorde con la administración de los diagnósticos del protocolo. Para solucionar este problema, los miembros del equipo de urgencias tienen varias opciones, que van desde coger un papel y un lápiz, usar una calculadora o usar algunas aplicaciones de móvil o tablet específicas para estos cambios de unidades.

Como la medicina no es una ciencia exacta y no existe una urgencia idéntica a otra, hay muchos datos de importancia que deben ser recordados de cara a futuras emergencias. Estos datos son muy valorados por los profesionales, tanto es así que existen libros editados que constan únicamente de estas notas, a modo de recomendaciones. En general, muchos trabajadores de las urgencias llevan algún ejemplar de estos libros, o al menos un cuaderno de notas personal donde guardar estos datos.

Las soluciones existentes no son eficaces. Los documentos y/o libros son voluminosos y pesados. La información en ellos no es concisa. Si cada miembro del equipo sanitario lleva consigo al lugar de la urgencia la información necesaria, debe llevar muchos documentos. Si los lleva sólo un miembro del equipo es posible que en la urgencia se retrase la consulta de cierta información, por sólo haber un recurso disponible. En conclusión, es necesario presentar un modelo de ayuda a la información en la urgencia más rápido, sencillo y efectivo.

1.4. Planteamiento de la solución

Actualmente la circulación como el procesamiento de datos se hacen a altas velocidades gracias a las nuevas tecnologías de la información, y esto se puede aprovechar en los servicios de emergencias para salvar vidas.

EMS in Cloud unifica la información de los protocolos, notas y medicamentos en una única solución disponible y personalizable para cada usuario. Nos basamos en las ventajas de la nube para tener disponibilidad total de los datos. Además, para no depender de la conectividad de los dispositivos, la solución permite la descarga de los datos para su utilización offline.

La solución consta de dos partes principales y bien diferenciadas. Una parte web y otra parte móvil desarrollada en Android. La web está orientada al trabajo de los doctores y enfermeros desde su oficina. Podrán crear y modificar los diagramas de los protocolos de actuación, también tendrán a su disposición la lista de fármacos que suele utilizar y sus notas. Los medicamentos tendrán toda la información que les caracterice, es decir, la del prospecto. Las notas serán del contenido que elija el usuario, funcionando como un cuaderno de anotaciones. De esta forma, toda esta información queda presentada de una forma más eficiente y personalizable.

El objetivo de EMS in Cloud es facilitar la labor de los sanitarios en el momento de la urgencia. Es en este escenario en el que entra en juego la parte Android del proyecto. Como hemos comentado, en el lugar de la urgencia los profesionales deben ejecutar las instrucciones del protocolo en cuestión. Desde la aplicación se le irá mostrando al usuario el fragmento de las instrucciones que debe hacer en cada momento, según la toma de decisiones que se vaya ejecutando. Además, se podrá cambiar las unidades de medida que aparezcan en los fármacos de

forma muy rápida e intuitiva. En cualquier momento el usuario podrá consultar su lista de fármacos, así como leer o crear sus notas.

Los datos que sean introducidos a través de la web estarán siempre disponibles, ya que se aloja en la nube. La información de la aplicación móvil se obtiene a partir de la web, sincronizando los datos.

EMS in Cloud propone un tratamiento de la información más eficaz, rápido, intuitivo y siempre disponible para facilitar la tarea más importante de los miembros de los equipos de urgencias, salvar personas.

Capítulo 2

Estado del arte

2.1. Cloud Computing

2.1.1. Historia

Antes de la llegada del cloud computing existían y existen en la actualidad alternativas para aquellos que deseen llevar a cabo actividades informáticas en la red por medio de internet, como por ejemplo el servicio de alojamiento web. El alojamiento web o web hosting[1] es el servicio por el cual se brinda al usuario la posibilidad de almacenar cualquier tipo de información accesible desde la red, típicamente a través de un navegador de internet. En este tipo de servicio, hay distintos tipos de alojamiento dependiendo de las necesidades del desarrollador, gratuitas, compartidas, de pago, de imágenes, revendedores, servidores virtuales, servidores dedicados, hosting administrado y no administrado.

Ilustración 1: El mundo conectado a la red.

Por supuesto, el alojamiento web ha jugado un papel muy importante en el desarrollo de aplicaciones web, pero su poca elasticidad y su tarifa típicamente mensual hacen que la disponibilidad de recursos que desea ofrecer el desarrollador no sea la óptima en un entorno como internet dónde la demanda de recursos informáticos es muy variable.

Como ya se ha mencionado, el alojamiento web ha sido y es parte importante de la historia de internet, jugó un papel importante en el boom de las .com, y es uno de los modelos más usados aún en la actualidad.

El creciente uso de la web dio paso al auge de nuevas tecnologías.

Cloud computing nace de la necesidad de solventar las carencias del alojamiento web tradicional, entre las carencias más importantes se encuentran la elasticidad y la escalabilidad.

2.1.2. Propiedades de la computación en la nube

A continuación explicaremos las dos propiedades fundamentales que caracterizan la computación en la nube

1. Elasticidad

La elasticidad consiste en la potencia de escalar los recursos informáticos ampliando y reduciendo la disponibilidad de los recursos con una fricción mínima, es decir, un entorno en el cuál las aplicaciones ceden y solicitan los recursos bajo demanda.

A diferencia de los servicios de alojamiento tradicionales, que también tienen la posibilidad de redimensionar la disponibilidad de los recursos que ofrecen, la propiedad de elasticidad de la nube le permite hacerlo en tiempo real sin impactos de paradas en tiempos de producción.

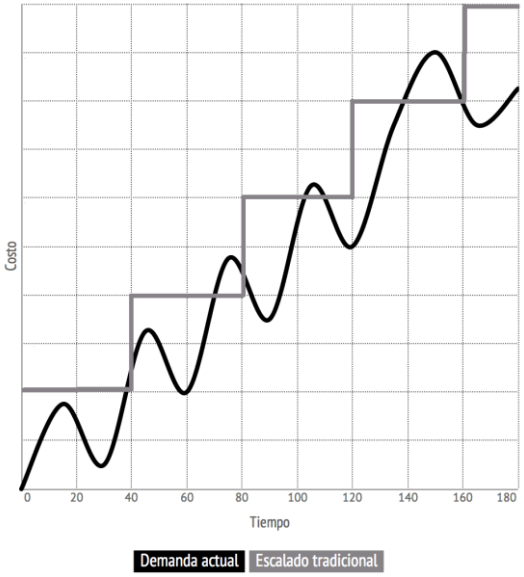


Ilustración 3: Gráfica sobre el escalado tradicional.

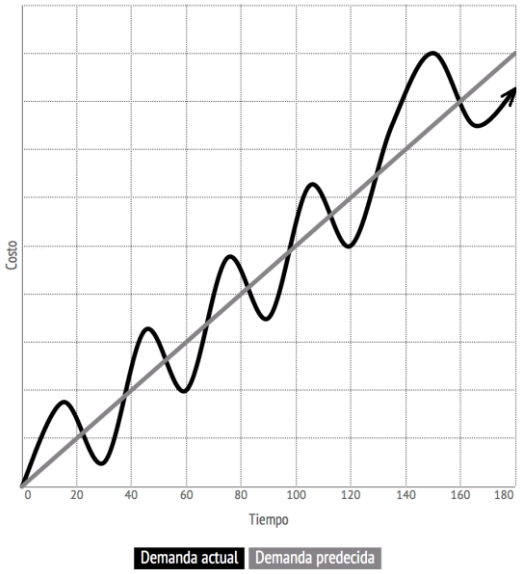


Ilustración 2: Gráfica sobre la demanda predicha.

2. Escalabilidad

La escalabilidad, es la característica que permite a la nube reaccionar y adaptarse a las nuevas demandas de disponibilidad sin comprometer su funcionamiento y tampoco perder su calidad.

Con el hosting tradicional la demanda de usuarios predicha era simplemente una aproximación. El escalado también era posible, pero a la larga el escalado tradicional siempre es insuficiente o demasiado, y como consecuencia de ello una pérdida de dinero sustancial. Con el cloud computing y su característica de elasticidad el escalado automático consigue adaptarse a la demanda de los usuarios y que se pague sólo lo consumido.

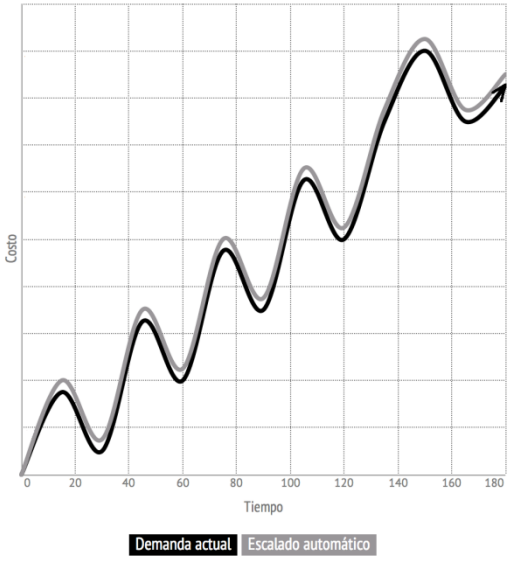


Ilustración 4: Gráfico de escalado automático

2.1.3. Tecnologías en las que se apoya la nube

1. Virtualización

Virtualización[2] en informática es la creación de una versión virtual de algún recurso tecnológico a través de software, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red.

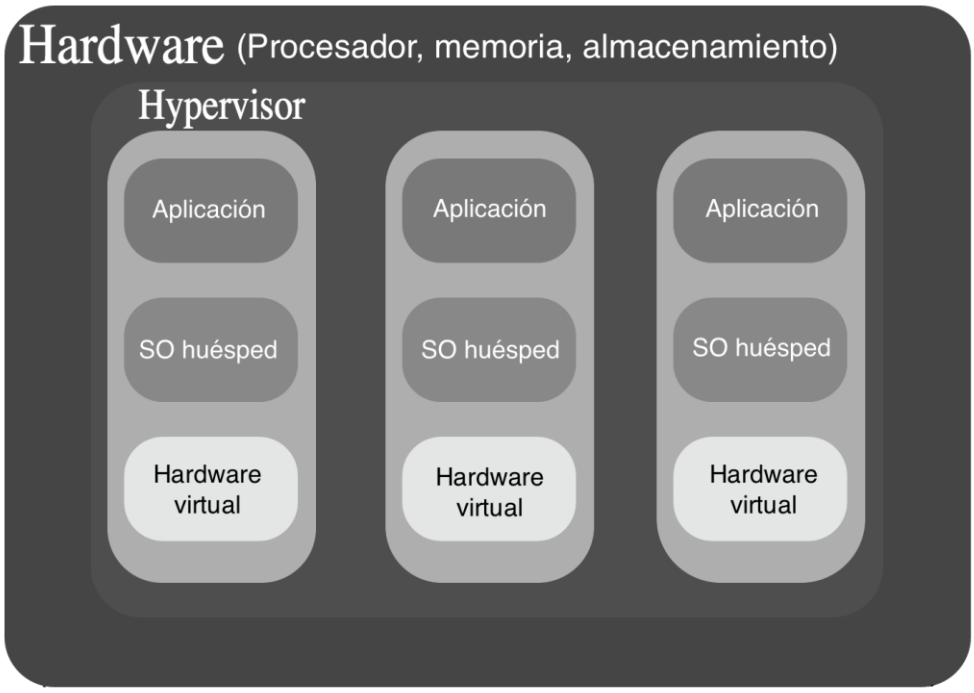


Ilustración 5: Virtualización hardware.

2. Tenencia múltiple

Tenencia múltiple[3] o multitenencia en informática es el modelo en el cual una sola instancia de la aplicación se ejecuta en el servidor, pero ofreciendo el servicio a múltiples clientes. Esto evita tener diferentes instancias de una misma aplicación para organización o cliente. Además permite fragmentar virtualmente datos y configuraciones para que cada cliente u organización tenga una instancia adaptada a sus necesidades y requerimientos. Esta característica es sin duda un factor clave en el paradigma de la computación en la nube.

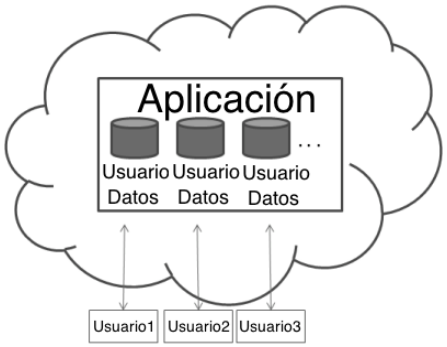


Ilustración 6: Tenencia múltiple

La tenencia múltiple se apoya en tecnologías como virtualización y segmentación, así como estrategias para gobernar y el aislar las aplicaciones entre sí.

2.1.4. Tipos de nubes

El cloud computing o computación en la nube es una plataforma que se caracteriza por ser altamente escalable y que permite un acceso rápido a los recursos hardware o software.

La computación en nube, también conocida como "la nube", es la provisión bajo demanda de recursos de computación, desde aplicaciones hasta centros de datos por medio de Internet, y/o de acuerdo a un plan de pagos.

Para emplear mejor este paradigma es necesario conocer los tres modelos[4] de nube que existen en la computación en la nube, los cuales se desarrollan a continuación.

Ilustración 7: Nube pública vs. nube privada.

1. La nube pública

En este modelo, los servicios como por ejemplo aplicaciones y/o almacenamiento se ponen a disposición de los usuarios a través de Internet y ajustadas a un plan gratuito ó de pago que dependerá exclusivamente del proveedor de servicios.

Una de las ventajas de este modelo es que permite una economía de escala, es decir, el coste de este servicio está estrechamente ligado al coste de disponibilidad del servicio, de manera que a mayor disponibilidad del servicio mayor será el coste del servicio y si por el contrario la disponibilidad del servicio disminuye el coste del servicio también decrecerá.

La economía de escala es una de las características más atractivas que se puede implantar en un sistema basado en la nube.

E

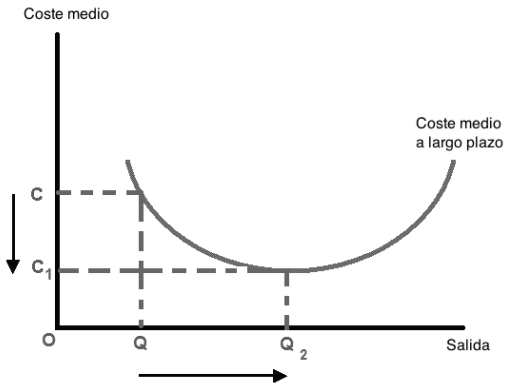


Ilustración 8: Gráfica de economía de escala.

Existen tres tipos principales de nubes públicas:

- **Infraestructura como servicio (IaaS):** Como su nombre sugiere, el servicio que ofrece es del tipo infraestructura, de manera que proporciona acceso a recursos situados en un entorno físico o por lo general virtualizado, al cual se accede típicamente mediante una interfaz web o mediante el protocolo SSH[7] explicado más adelante.

  La definición de IaaS abarca aspectos como el espacio en servidores virtuales, ancho de banda, conexiones de red, direcciones IP y balanceadores de carga. Además también proporciona un repertorio de recursos de hardware disponibles procedentes de multitud de granjas de servidores, de cuyo mantenimiento se encarga el proveedor del servicio.

  El cliente, como por ejemplo un desarrollador, obtiene acceso a los componentes virtualizados para construir con ellos su propia plataforma informática, algunos ejemplos son: Amazon EC2, Windows Azure, Rackspace, Google Compute Engine.

- **Plataforma como servicio (Paas):** El servicio de plataformas consiste en ofrecer plataformas computacionales que por lo general incluyen un sistema operativo, entornos y APIS para lenguajes de programación, bases de datos y servidores web, al cual se accede típicamente mediante una interfaz web o mediante el protocolo SSH, algunos ejemplos son: AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Openshift.

- **Software como servicio (SaaS):** Es un modelo que proporciona acceso a aplicaciones web, donde el usuario final no tiene que preocuparse por la instalación, configuración y ejecución en el sistema, puesto que es tarea del proveedor de dicho servicio, algunos ejemplos son: Google Apps, Microsoft Office 365.

  En nuestro proyecto, Openshift será la plataforma encargada de administrar la gestión de nuestra aplicación multiplataforma, y albergará en él un SaaS que hemos llamado EMS, el cual se discutirá más adelante.

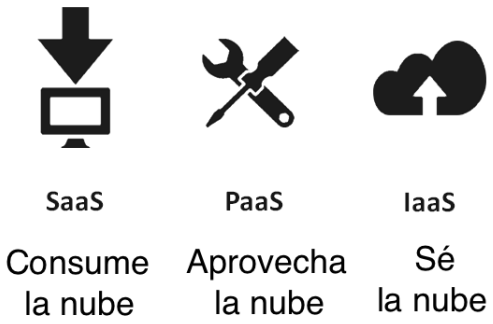


Ilustración 9: Tipos de nube pública.

2. La nube privada

Una nube es privada si el mantenimiento y costes de la infraestructura se asume por cuenta propia, por lo general interesan a empresas con normativas estrictas y aplicaciones que exigen tener una disponibilidad muy variada. Este tipo de nube siguen un modelo in house y requieren un nivel de compromiso importante entre la virtualización del entorno empresarial, y la evaluación, elección y asignación de los recursos existentes en la nube. Además deben implementar soluciones de seguridad avanzada, alta disponibilidad y tolerancia a los fallos que no tienen tanta relevancia en una nube de tipo pública.

3. La nube híbrida

Como su nombre indica, este modelo mezcla las soluciones de una privada (interna) y una nube pública (externa). Es muy común que las empresas ejecuten una aplicación principal en su nube privada y utilicen una nube pública para enfrentarse a los distintos altibajos que pueda tener la demanda de la aplicación. La seguridad y la infraestructura subyacente están reguladas por reglas, protocolos y directivas específicas.

2.1.5. Seguridad

La seguridad de la nube es tan buena o mejor que los sistemas tradicionales, porque ahora son los proveedores del servicio quienes se encargan de dicha tarea. Sin embargo, no deja de ser un asunto importante cuando la sensibilidad de acceso los datos es de carácter confidencial.

Las tres características principales de la seguridad en la nube son:

1. Integridad de los datos

Es necesario realizar la autenticación de que quienes puedan acceder y modificar los datos a fin de prevenir la intrusión de entidades no autorizadas, y de asegurar que la información proporcionada por los que sí son usuarios autorizados sea la original

2. Evitar pérdidas de información

Una de las características que más preocupan a los clientes es que sus datos pasan a estar ubicados en servidores de la nube, de esta manera existe la probabilidad de pérdida de información en el caso de fallo en el sistema ajeno al usuario.

3. Confidencialidad y privacidad

El proveedor de servicios debe ofrecer la garantía de que sólo puedan acceder a los datos usuarios autorizados. El usuario, en principio, no tiene control completo del acceso del proveedor a sus datos

2.1.6. Cloud computing como resultado de la convergencia de tecnologías

A día de hoy la computación en la nube se ha convertido en un modelo a seguir imprescindible por quienes busquen optimizar la accesibilidad de sus recursos, resumiendo la computación en la nube es el resultado de aunar algunas de las siguientes tecnologías:

- **Virtualización**: consiste en la abstracción de un sistema hardware.
- **Tenencia múltiple**: una instancia de la aplicación ofrecida a muchos clientes.
- **Elasticidad en tiempo real**: consiste en adaptarse a los picos de demanda en tiempo real.
- **Arquitectura orientada a servicios:** permite crear de sistemas de información altamente escalables.
- **Marcos de trabajo orientado a aplicaciones web**: consiste en un conjunto de conceptos, prácticas y criterios orientados a resolver ciertos problemas de las aplicaciones web, que sirven como referencia para resolver nuevos problemas de índole similar.
- **Servicios web**: consiste en el uso de servicios web que sigan los estándares establecidos, siendo la solución en modo de canal de comunicación a sistemas compatibles e incompatibles, y además eliminando interdependencias.
- **Computación grid**: permite utilizar de forma coordinada recursos de tipos muy diversos (como recursos cómputo, de almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado.

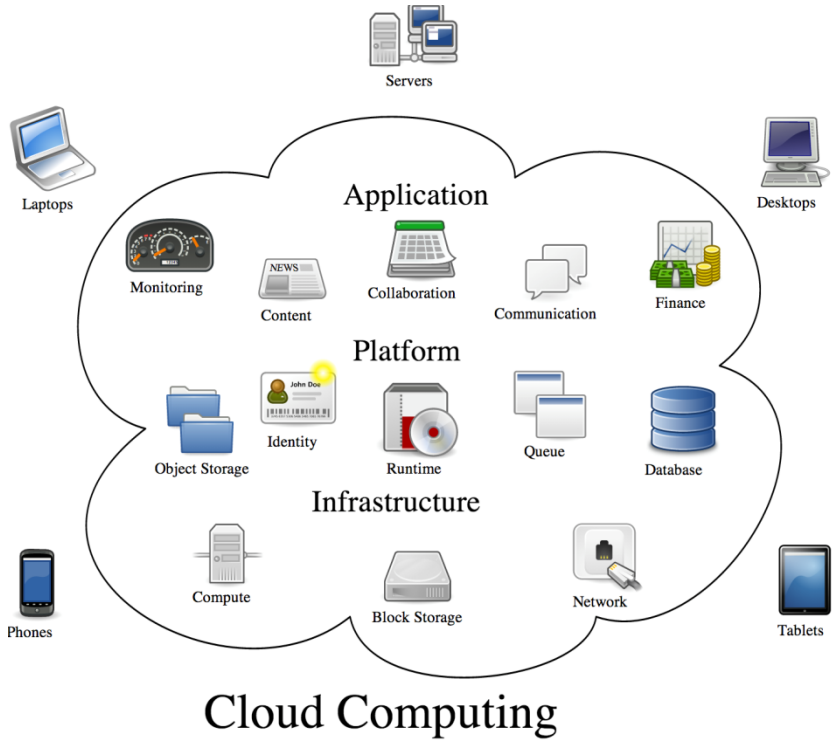


Ilustración 10: La computación en la nube. Fuente: Wikipedia

2.2. Sistema de control de versiones

El carácter de este proyecto busca ser serio y profesional, para esto es imprescindible un sistema de control de versiones.

Cuando nos encontramos desarrollando cualquier producto, llamamos versión del producto al estado en el que se encuentra dicho producto en un momento concreto de su desarrollo. Un buen desarrollo exige que estemos totalmente seguros de guardar y de poder volver a recuperar cualquiera de esas versiones que se van sucediendo a medida que el proyecto va madurando.

Se llama control de versiones a la gestión de esos cambios y los sistemas de control de versiones son programas que nos facilitan esa gestión encargándose de administrar las distintas versiones de cada producto.

Este control se ha consolidado en la industria informática para controlar el desarrollo y el código fuente de los programas, pero también es aplicable a cualquier otro ámbito de desarrollo, desde documento, creación de imágenes, o escritura de libros.

2.2.1. Características

Estas son algunas de las características principales de los sistemas de versiones:

- Se utilizan como **modos de almacenamiento** para archivos de texto, imágenes, documentación, código fuente, etc.
- **Posibilidad de cambios**: además también permite hacer sobre ellos todo tipo de modificaciones parciales o completas.
- **Registro de modificaciones**: cuenta también con la conservación de un registro histórico de todas las acciones realizadas con cada uno de los elementos y con el conjunto de los mismos, pudiendo en cualquier momento volver a extraerlos y revertir el producto a una fase anterior de su desarrollo.
- **Crear informes**: algunos de estos sistemas generan informes de los cambios entre dos versiones, informes de estado, identificación nominal, identificación de un conjunto de ficheros, etc.

Algunos sistemas conocidos son Concurrent Version System (CVS), Subversion (SVN) y Git. Este último se ha convertido en el más popular

2.2.2. Tipos de sistemas de control de versiones

1. Centralizado

Es este tipo de sistema existe un repositorio centralizado de toda la información con un único administrador. Este enfoque facilita mucho las tareas administrativas pero también reduce la flexibilidad, ya que el acceso a algunas funcionalidades importantes requiere la aprobación del responsable. Algunos ejemplos son CVS y SVN.

2. Distribuido

En estos sistemas de control cada usuario tiene su propio repositorio, no es necesario tomar decisiones centralizadamente, sino que los distintos repositorios pueden intercambiar y mezclar información entre ellos. Algunos ejemplos son Mercurial y git.

2.2.3. Git

El éxito y la popularización de git se debe ser un sistema distribuido[5], ya que convierte a cada uno de los clientes en un auténtico espejo que almacena una copia idéntica del repositorio, de manera que si cualquiera de los clientes falla otro cliente podrá devolver la información y que recupere todo lo necesario.

Este sistema permite configurar muchos tipos de dinámicas de trabajo que no son posibles en un sistema centralizado, que tienen una organización mucho más jerarquizada.

Es ideal para un desarrollo no lineal.

2.3. Openshift

Openshift[6] es un producto de computación en la nube de plataforma como servicio (PaaS) de la empresa Red Hat.

Una de las características más atractivas es que los desarrolladores pueden usar Git para desplegar sus aplicaciones Web en los diferentes lenguajes de la plataforma.

Openshift también soporta programas binarios que sean aplicaciones Web, con tal de que se puedan ejecutar en Red Hat Enterprise Linux (RHEL Linux). Esto permite el uso de lenguajes arbitrarios y frameworks.

Openshift se encarga de mantener los servicios subyacentes a la aplicación y la escalabilidad de la aplicación como se necesite.

2.3.1. Componentes de Openshift

- **Red Hat Enterprise Linux:** Sistema operativo en el cual se sustenta todo la infraestructura de Openshift.
- **Infraestructura**: Openshift puede ejecutarse en cualquier lugar donde RHEL pueda ejecutar un hypervisor, servidores privados, y nubes públicas, privadas o híbridas.
- **Instancias**: Un conjunto de instancias RHEL llamados nodos Openshift donde el desarrollador despliega su aplicación.
- **Broker o hypervisor**: Los nodos Openshift son controlados por los brokers que conforman el motor del PaaS coordinando y automatizando tareas.
- **Gear o engranaje**: Cada nodo Openshift es un entorno de tenencia múltiple para las aplicaciones de los desarrolladores, los engranajes son seguros contenedores de estos nodos.

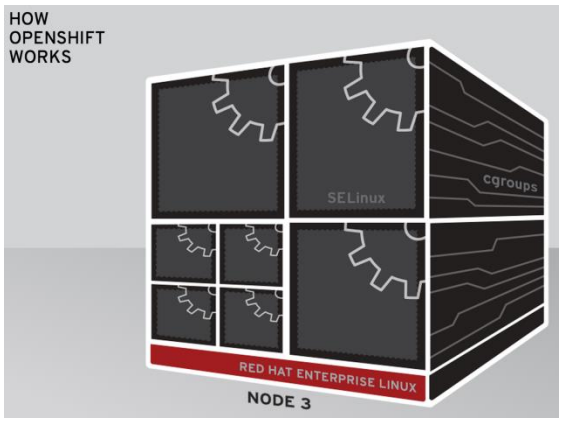


Ilustración 11: Gear de Openshift.

- **Security-Enhanced Linux (SELinux)**: Módulo de seguridad para el kernel Linux que funciona como capa de seguridad de los engranajes.
- **CGroups**: Openshift usa grupos de control para asignar recursos computacionales tales como CPU, RAM y almacenamiento.
- **Alta densidad**: El modelo de contenedores de engranajes Openshift permite seguridad con una alta densidad.

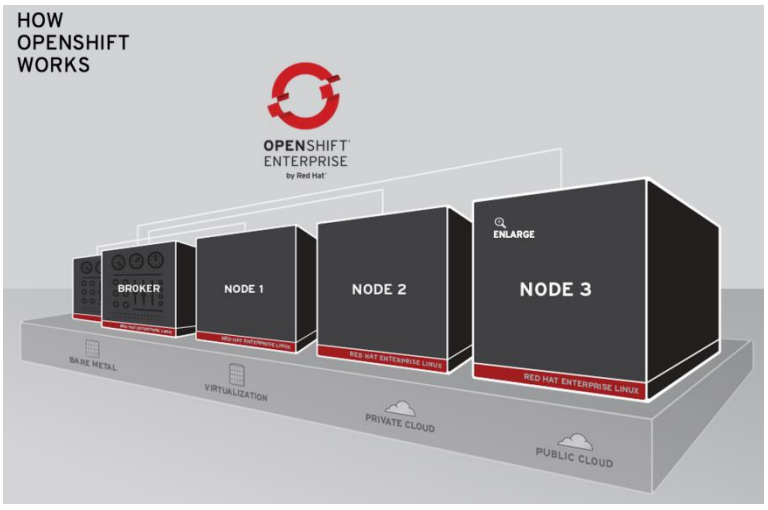


Ilustración 12: Alta densidad de Openshift.

2.4. Secure Shell (SSH)

Secure Shell o mejor conocido como SSH es un protocolo que permite acceder a computadores remotos a través de una red. De esta manera se puede manejar una computadora remota por medio del intérprete de comandos, cifrar datos y gestionar claves RSA entre otras características.

2.4.1. Seguridad

Una de las ventajas de usar el SSH es que nos protege de ataques conocidos como Man in the Middle, ARP, ARP Spoofing, entre otros. Los ataques mencionados siempre un común denominador y es que todos buscan robar, interceptar y/o modificar la transmisión de los datos en nuestras comunicaciones.

2.4.2. Tunel SSH

Este es el mecanismo por el cual el sistema cifra todos los datos de una conexión gracias a un cliente SSH, de esta manera el cliente inicia una conexión al servidor tunelizado, y finalmente este servidor se conectará al servidor final.

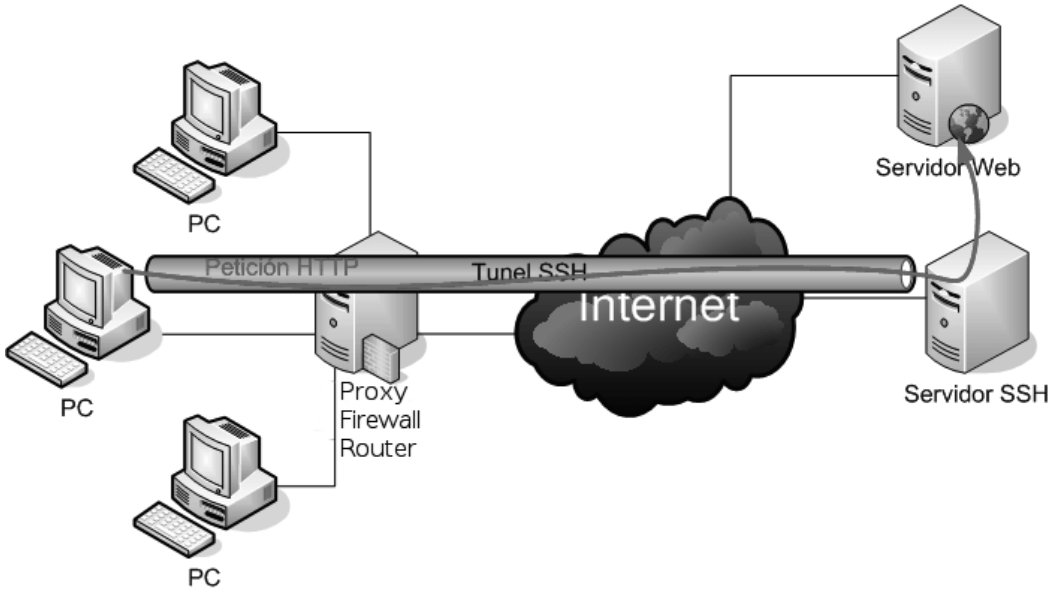


Ilustración 13: Túnel SSH.

Lo atractivo de este protocolo es que se consigue una conexión segura sin alterar el sistema del cliente o del servidor.

2.5. Bootstrap

Es el framework de software libre CSS que la empresa Twitter pone a disposición de los desarrolladores. En este proyecto, bootstrap nos permite relajar la responsabilidad y decisiones de una interfaz de usuario que gracias a este framework es intuitiva y adaptada a la mayor parte de monitores y pantallas de los dispositivos más usados en la actualidad.

2.5.1. Ventajas

- **Fácil e intuitivo**: la curva de aprendizaje es perfectamente afrontable en el tiempo que disponemos para el desarrollo de este proyecto en parte gracias a la gran cantidad de ejemplos y documentación que encontramos en su web oficial.
- **Libre de hacks**: no más de los conocidos "workarrounds" (la traducción más aproximada en castellano sería: apaños) debido a las tan variadas implementaciones de HTML que tienen la gran cantidad de navegadores, haciendo que el enfoque del proyecto sea independiente del tipo de navegador. De esta manera el proyecto se ejecuta correctamente en cualquier navegador aceptado por la W3Schools.
- **Optimizado para dispositivos móviles**: Bootstrap ofrece todas las reglas CSS necesarias para hacer que la aplicación web se adapta dinámicamente a la gran mayoría de pantallas y resoluciones existentes en el mercado.
- **Soporte por parte de twitter**: esto da una confianza añadida al framework, ya que en caso de encontrar bugs, sabemos que hay un equipo detrás solucionando los problemas.

2.6. JQuery

JQuery es una librería gratuita y de software libre, que simplifica el trabajo en la creación de páginas web altamente interactivas. Funciona en todos los exploradores de internet aceptados por la W3Schools y abstrae características específicas de cada uno de estos, permitiendo enfocar el esfuerzo en el diseño y resultado final, en lugar de tratar de desarrollar funciones complejas en exploradores individuales.

2.6.1. Ventajas

- **Fácil búsqueda y manipulación** de contenido en una página HTML
- Permite trabajar con el **modelo de eventos** de los exploradores modernos
- Permite añadir **efectos y transiciones** sofisticadas que vemos en páginas modernas, como animaciones disparadas por eventos.

2.7. Aplicaciones Médicas

Queríamos hacer un pequeño inciso antes de continuar, a lo largo de este proyecto hemos contado con el asesoramiento de un experto experto: Manuel Vázquez, Enfermero que forma parte del equipo sanitario en una Unidad de Emergencias Medicalizada del 061 ARAGON en la Comunidad Autónoma de Aragón, miembro de la tripulación sanitaria en el helicóptero medicalizado de emergencias perteneciente al 112 SOS ARAGON durante cuatro años y Master en Medicina de Urgencia en Montaña por la Universidad de Zaragoza, entre otras cosas.

Ilustración 14: Manuel Vázquez, experto colaborador con EMS in Cloud.

Con su ayuda, hemos elaborado una lista con las aplicaciones para dispositivos móviles que contienen aspectos interesantes de cara a nuestro proyecto.

2.7.1. Listado de aplicaciones

1. IV Drips v3.2

- **Enlace**: https://play.google.com/store/apps/details?id=appinventor.ai_jonsaputo.IVDrips
- **Descripción**:
  - La más útil para dosis y perfusiones.
  - Permite configurar medicamentos nuevos, pero no los existentes (como cambiarles nombre). Parece ser que la última versión permite copiar medicamentos a la zona personalizada.
  - Permite indexarlos por nombre genérico y comercial.
  - Las dosis y diluciones aparecen de forma previa al pasar a la calculadora.

2. IV Infusion Calc v1.4

- **Enlace**: https://play.google.com/store/apps/details?id=com.jonsap.ivinfusioncalc
- **Descripción**:
  - Aporta una calculadora completa que permite:
    - Dosis a ml/hr
    - ml/hr a Dosis
  - Para bomba de perfusión y goteo por gravedad en equipos de gotero.

3. Infusion pump v1.2

- **Enlace**: https://play.google.com/store/apps/details?id=es.miguelcpal.panel
- **Descripción**:
  - No permite tener ningún fármaco almacenado
  - Las unidades de dosificación solo se contemplan de 5 maneras
  - Sí que permite modificar kg, mg y ml de uno en uno.

4. Pedi safe v3.3

- **Enlace**: https://play.google.com/store/apps/details?id=org.pedisafe
- **Descripción**:
  - La aplicación agrupa por rangos de peso desde 2 a 120 kg y también por grupos de fármacos a utilizar en una emergencia.
  - Debería ajustarse a tablas de kg en kg.
  - No permite configurar nada, ni fármacos ni diluciones.

5. Safedose v3.5

- **Enlace**: https://play.google.com/store/apps/details?id=com.ebroselow.safedoseandroidfree
- **Descripción**:
  - Bastante completa, encaminada a tratamientos con fármacos agrupados por utilidad y en rangos de peso.
  - Muestra un cuadro de coordenadas, con los valores.
  - No pone la presentación del fármaco.
  - Debería ajustarse a tablas de kg en kg.
  - Viene con tablas para perfusiones pero no deja modificar la tabla.
  - Error de combinar mcg y µg.

6. DTsEMT Menu v1.0

- **Enlace**: https://play.google.com/store/apps/details?id=com.dtsemt.menu

7. DTsEMT Drip Rate Calculator v1.4

- **Enlace**: https://play.google.com/store/apps/details?id=com.DTsEMT.DripRateCalc
- **Descripción**:
  - Calculadora de velocidad de infusión en la que se introducen dosis, volúmenes, dosificación, y peso para obtener velocidad de infusión, administración con cámara de goteo y sumario de los datos introducidos

8. DTsEMT O2 v1.3

- **Enlace**: https://play.google.com/store/apps/details?id=dtsemt.o2
- **Descripción**:
  - Calculadora del tiempo restante y volumen de oxigeno según modelo de tanque y aparato utilizado en la aplicación de oxigeno

9. DTsEMT Drip Timer v2.0

- **Enlace**: https://play.google.com/store/apps/details?id=com.DTsEMT.DripTimer
- **Descripción**:
  - Calculadora de volúmenes en tiempo con la originalidad de tener un contador pulsátil.

10. DTsEMT RSI Calculador v2.0

- **Enlace**: https://play.google.com/store/apps/details?id=com.dtsemtrsicalc
- **Descripción**:
  - Calculadora de dosis y utilización de fármacos

11. Tom's Ambulance App v1.4

- **Enlace**: https://play.google.com/store/apps/details?id=com.tommy.CPGApp
- **Descripción**:
  - Integra una serie de guías/protocolos para adultos y pediátrico, sobre fármacos, una calculadora que yo no he conseguido que funcione, permite crear una lista de contactos para llamar, y realizar búsquedas desde la propia aplicación.

12. Ems Notes v4.4.0

- **Enlace**: https://play.google.com/store/apps/details?id=air.emsnoteemsoperations&hl=es
- **Descripción**:
  - Quizás sea la más completa, es como un report de trabajo, pero también contiene herramientas para fármacos, dosis y toma de constantes, va dirigida a paramédicos y otros sistemas de atención sanitaria, pero podría ser un modelo de trabajo interesante.

Capítulo 3

Arquitectura del sistema

3.1. Introducción

En este capítulo presentaremos una visión general de la arquitectura de EMS in Cloud. Nuestro objetivo es proporcionar al lector una idea clara y amplia del funcionamiento del sistema, por lo que evitaremos una explicación del código muy detallada.

También expondremos en este capítulo las decisiones de diseño adoptadas para llevar a cabo la realización de nuestro proyecto, ya que, al ser cloud computing, depende en muchos aspectos de las tecnologías utilizadas.

3.2. Organización del capítulo

Hemos decidido dividir el capítulo en los siguientes bloques:

- **Objetivos y restricciones**: en él expondremos la finalidad de EMS in cloud, las necesidades que pretende cubrir así como las restricciones que conlleva este sistema.
- **Decisiones de diseño**: se explicará el porqué de las elecciones de las tecnologías utilizadas de cara a los objetivos declarados anteriormente
- **Aproximación de alto nivel**: en el detallaremos los principales elementos que forman EMS in Cloud, así como sus iteraciones, proporcionando una visión global
- **Vista lógica**: Es esta sección entraremos en detalle en el diseño a nivel de clases y paquetes de la solución y sus relaciones. Lo complementaremos con diagramas de clases para clarificar las explicaciones. Para hacer este apartado más sencillo, no serán mostradas todas y cada una de las relaciones existentes, sino únicamente aquellas que hemos considerado más relevantes.

3.3. Objetivos y restricciones

3.3.1. Objetivos

El propósito de este proyecto es el desarrollar una herramienta que facilite la labor del servicio de rescate de Aragón. Esta herramienta debe ser:

- Configurable: proporcionar la posibilidad al usuario de incluir nueva información. En concreto, poder configurar:
  - Nombre del fármaco
  - Presentaciones del fármaco
  - Dosis
- Permitir el cambio de unidades inmediato.
- Tener como idioma principal el castellano
- Disponer de los protocolos, cuadros de decisión

Se pretende entonces informatizar el sistema actual con ayuda de las nuevas tecnologías y a un precio que esté lo más cerca posible de la gratuidad. Veremos a lo largo de los siguientes apartados que esta no es una tarea fácil de conseguir debido a que los servicios en los que nos apoyamos para hacer este proyecto están en una constante evolución tanto tecnológica como económica.

El primer objetivo es hacer una aplicación que permita gestionar protocolos, fármacos y notas de intervención. Este requisito se puede conseguir a través de una aplicación web. Se permitirán las siguientes funcionalidades:

- **Introducir nuevos protocolos**: el usuario será capaz de crear nuevos protocolos generando su diagrama de flujo correspondiente. Esta creación debe ser lo más intuitiva posible, alejándonos así de modelos basados en la introducción de código para la generación de un diagrama.
- **Gestionar una lista de fármacos**: teniendo en cuenta que la información debe ser editable por el usuario.
- Crear **anotaciones** personales para su consulta posterior.

El segundo objetivo consiste en tener una aplicación móvil que pueda servir de herramienta a quién esté en el campo de actuación intentando diagnosticar. Este requisito se puede conseguir a través de una app Android que se sincronice con la aplicación web. La aplicación Android deberá permitir:

- **Ver el listado** de protocolos, fármacos y notas del usuario
- **Seguir paso a paso el contenido del protocolo**: cuando el usuario inicie la consulta de un nuevo protocolo, se le mostrará la primera caja de texto que lo forme. Si el contenido implica una decisión, tendrá que dar una respuesta a dicha decisión para ver la siguiente caja de texto. En caso contrario bastará con que solicite mostrar la caja siguiente. Este proceso de navegación a través del diagrama que conforma el protocolo se seguirá hasta llegar a la finalización del mismo.
- **Detectar los valores numéricos y las unidades de medida**: siguiendo el objetivo de permitir el cambio de unidades inmediato se plantea este nuevo objetivo. Para realizar la conversión entre unidades de medida es necesario detectar la el valor numérico y la unidad que lo acompaña. También se podrán detectar los valores numéricos que no incluyan unidad métrica si están en una caja de decisión.

Del segundo objetivo se deriva un tercer objetivo y es la creación de un servicio web, este servicio web es imprescindible para asegurar la consistencia de datos en el dispositivo móvil, es importante que la aplicación puede ejecutarse en una tablet o móvil.

Para llevar a cabo estas tareas hay que tener en cuenta los siguientes apartados:

- **Tiempo**: Para gestionar mejor el tiempo disponible para este proyecto se propone un desarrollo iterativo. Considerando que la asignatura tiene un valor de 15 créditos lo normal es que dediquemos como mínimo un tiempo de 5 horas a la semana hasta mediados de Mayo.
  En total disponemos de 15 horas a la semana para el desarrollo del proyecto. Por supuesto, se descuentan de este tiempo los periodos de vacaciones y exámenes.

- **Conocimientos**: Debido a que el tiempo es escaso con respecto a nuestras pretensiones, se intenta aprovechar al máximo la experiencia adquirida a lo largo de nuestra estancia en la carrera y poder evitar en la medida de lo posible lidiar con nuevos lenguajes de

programación y aprovechar la destreza que ya tenemos en otros que nos resultan más familiares.

Los lenguajes y tecnologías con los que más cómodos nos sentimos son:

- Java
- Android SDK
- J2EE
- J2SE
- PHP
- Javascript
- HTML
- CSS
- XML
- JSON
- MySQL

Partiendo de este conjunto de tecnologías, en los apartados posteriores exploraremos la mayor cantidad de alternativas posibles para cada desafío que se nos presente.

3.3.2. Restricciones

Finalmente, el conjunto de herramientas que vayamos adoptando para resolver los diversos desafíos técnicos deberán no alejarse mucho de las tecnologías que ya conocemos y no suponer una inversión muy elevada en tiempo de aprendizaje.

Por último, en medida de lo posible se intentará elegir siempre herramientas de software libre para conseguir acercarnos al objetivo de gratuidad.

3.4. Decisiones de diseño

Una vez decidido seguir el paradigma Cloud, lo primera decisión importante a tomar es que servicio PaaS se ajusta mejor a nuestras necesidades, es importante tener muy presente los objetivos y restricciones del punto anterior.

3.4.1. Eligiendo el PaaS más adecuado

Entre las opciones barajadas tenemos:

1. **Google app engine**

Una de las opciones más populares y de las que más fuerza toma al inicio del desarrollo de este proyecto, entre sus características más atractivas era su soporte a los lenguajes:

- Java 2 platform enterprise edition
- Python DJango
- Go
- PHP (en fase beta hasta enero de este año)

2. Gestor de bases de datos

Otro aspecto importante es el sistema gestor de base de datos, en este caso google app engine ofrece:

- Google Cloud Storage: este gestor de base de datos no relacional se toma en cuenta y se realizan una serie de pruebas funcionales y de viabilidad.
- Google Cloud SQL a partir de Abril de 2014: este gestor de bases de datos relacionales se apoya el ya conocido de lenguaje de consulta estructurado SQL.

3. Pruebas con google app engine

Estas pruebas arrojan resultados positivos pero no excelentes, por una parte la disponibilidad de usar el lenguaje Java en el entorno de desarrollo integrado Eclipse nos da la confianza de poder afrontar los nuevos desafíos de implementación que surjan en el futuro con una base sólida de conocimientos de programación e ingeniería del software adquiridos en asignaturas tales como programación orientada a objetos (POO), ingeniería del software (IS) y bases de datos y sistemas informáticos (BDSI). Por otro lado Google cloud storage se antoja arriesgado dado el poco tiempo que tenemos para el desarrollo del proyecto, por lo que de usar Google App Engine se usará con su alternativa de almacenamiento Google Cloud SQL.

4. Evaluación de costes

Servicio de red	RAM	G.C. Storage	G.C. SQL
Gratuito	512MB	Gratuito	0.02$/hora/GB
0.07€/instancia	1GB/instancia	Gratuito	0.02$/hora/GB

Tabla 1: Los costes de google app engine (antes de abril de 2014)

Llegados a este punto, google app engine es un PaaS más que interesante pero no termina de satisfacer nuestras necesidades de gratuidad.

Actualización: en Noviembre de 2013 era un servicio aún gratuito, pero a partir de Abril de 2014 este servicio es descatalogado por google en pos de la nueva versión de este servicio bautizado con el mismo nombre.
A continuación se muestra un resumen de los precios del paquete más económico que pudo habernos interesado. También dispone ahora de una calculadora https://cloud.google.com/products/calculator/

5. Servicio de red

Uso mensual	Red (salida): América y EMEA* (por GB)	Red (salida): Asia y Pacífico (por GB)	Red (entrada)
Primeros 0-1 TB	0,12 USD	0,21 USD	Gratis
Siguientes 9 TB	0,11 USD	0,18 USD	Gratis
Siguientes 90 TB	0,08 USD	0,15 USD	

Tabla 2: Costes de almacenamiento Google App Engine I

Procesadores virtuales	RAM	GCEUs	Precio
1	3.75	2.75	0.02$/hora
2	7.5	5.5	0.14$/hora
4	15	11	0.28$/hora
8	30	22	0.56$/hora
16	60	44	1.12$/hora

Tabla 3: Costes de almacenamiento Google App Engine II

3.4.2. Amazon EC2

El mayor atractivo del servicio de Amazon es su elasticidad, desde el principio se nos ofrece la posibilidad de tener todo del poder de un ordenador virtual a nuestra disposición. Se puede configurar un ordenador a medida con casi cualquier característica que deseemos, sistema operativo, entornos de programación, múltiples servicios para diferentes puertos, etc.

Con esta flexibilidad, problemas como la elección de lenguaje de programación y sistema gestor de bases de datos dejan de serlo.

El poder de este servicio es increíble, y por eso también es caro, a continuación la tabla de precios de las distintas y posibles configuraciones más económicas que ofrece Amazon:

LINUX	CPU Virtual	ECU	RAM	Almacenamiento	Precio
m3.medium	1	3	3.75	1 x 4 SSD	$0.077 por hora
m3.large	2	6.5	7.5	1 x 32 SSD	$0.154 por hora
m3.xlarge	4	13	15	2 x 40 SSD	$0.308 por hora
m3.2xlarge	8	26	30	2 x 80 SSD	$0.616 por hora

Tabla 2: Costes Amazon para Linux.

WINDOWS	CPU Virtual	ECU	RAM	Almacenamiento	Precio
m3.medium	1	3	3.75	1 x 4 SSD	$0.133 por hora
m3.large	2	6.5	7.5	1 x 32 SSD	$0.266por hora
m3.xlarge	4	13	15	2 x 40 SSD	$0.532 por hora
m3.2xlarge	8	26	30	2 x 80 SSD	$1.064 por hora

Tabla 3: Costes Amazon para Windows.

WINDOWS con SQL	CPU Virtual	ECU	RAM	Almacenamiento	Precio
m3.medium	1	3	3.75	1 x 4 SSD	$0.356 por hora
m3.large	2	6.5	7.5	1 x 32 SSD	$0.711 por hora
m3.xlarge	4	13	15	2 x 40 SSD	$1.280 por hora
m3.2xlarge	8	26	30	2 x 80 SSD	$2.560 por hora

Tabla 4: Costes Amazon para Windows con SQL

Si bien es cierto, EC2 ofrece un nivel de customización muy elevado, se antoja caro para nuestro proyecto. Es importante destacar que Amazon es el único que ofrece almacenamiento SSD a día de hoy y eso es algo muy a tener en cuenta en caso de proyectos que requieran altas velocidades de lectura.

Por suerte, Amazon también piensa en los desarrolladores con menos presupuesto y ofrece un plan llamado “Capa gratuita”. Los nuevos clientes de AWS pueden empezar de forma gratuita con Amazon EC2. Al registrarse, los nuevos clientes de AWS recibirán cada mes y durante un año los siguientes servicios de EC2:

- 750 horas de uso para la ejecución de instancias Linux/Unix Micro en EC2
- 750 horas de uso para la ejecución de microinstancias Microsoft Windows Server en EC2
- 750 horas de Elastic Load Balancing más 15 GB de procesamiento de datos
- 30 GB de almacenamiento del volumen estándar de Amazon EBS más 2 millones de E/S y 1 GB de almacenamiento de instantáneas
- Se añaden 15 GB de ancho de banda saliente en todos los servicios de AWS
- 1 GB de transferencia de datos regional

Esta capa gratuita incluso con sus restricciones se adapta mejor a nuestras necesidades. El desarrollo del proyecto debería darse durante el curso, por lo que un año sería suficiente, y en el caso de que el proyecto coja una masa de usuarios importante y requiera más demanda, se tendría muy en cuenta el pago mensual que sugiere Amazon para servicios más potentes.

3.4.3. Openshift

Ya se ha mencionado como funciona Openshift, por lo que ahora pasamos a definir sus principales características pues es el que se impone frente a las dos alternativas anteriores por ajustarse más a los requerimientos que buscamos.

En cuanto a tipos de aplicación y lenguajes, Openshift ofrece un amplio abanico de posibilidades pre-construidas.

- Gestores de contenido
  - Drupal
  - Wordpress
- Java
  - JBoss Application Server
  - Tomcat 6
  - Tomcat 7
  - Vertx 2.1
  - CapeDwarf
  - Wildfly 8
- Python
  - Python 2.6
  - Python 2.7
  - Python 3.3
  - DJango
- PHP
  - PHP 5.3
  - PHP 5.4
  - PHP 5.4 con ZendServer 6.1
  - Cake
  - Reveal.js

- Ruby
  - Ruby 1.8
  - Ruby 1.9
  - Ruby on Rails
- Node
- Pearl
- Go language

Si alguna de las anteriores opciones no fuera suficiente, siempre es posible configurar el nodo (máquina virtual que ofrece Openshift) con la combinación de lenguaje o gestor de bases de datos que deseemos, siempre teniendo en cuenta que trabajamos sobre el sistema operativo Red Hat Linux Enterprise.

La flexibilidad de Openshift en cuanto a customización se ajusta a la que necesitamos, por tanto sólo queda considerar el plan de precios.

1. Evaluación de costes

➢ **Plan Gratuito:**
- Tres engranajes pequeños
- Cada engranaje con 1GB de almacenamiento
- Posibilidad de migrar a otro plan cuando se desee y de manera inmediata.

➢ **Plan Bronce:**
- Desde 0$ al mes, dependiendo de si se tienen o no más engranajes de los que ofrece el plan gratuito.
- Posibilidad de añadir hasta 16 engranajes más.
- Certificados SSL
- Posibilidad de aumentar el almacenamiento.

➢ **Plan Plata:**
- Desde 20$ al mes, siendo más si se contrata más engranajes extras.
- Todas las ventajas del plan bronce.
- Soporte técnico por parte de la empresa Red Hat

El plan de precios de Openshift también se ajusta a nuestras necesidades pues también tiene un plan gratuito de precios que se ajusta a nuestras necesidades y es incluso mejor que el de Amazon porque la gratuidad se puede prolongar tanto como se desee, y cuando el balanceador automático necesite más potencia siempre podremos migrar a otros planes de Openshift con mejores servicios.

3.4.4. Conclusiones y elección del PaaS

Finalmente elegimos a Openshift como servicio PaaS para alojar y escalar nuestro proyecto por los siguientes motivos:

- Gratuito en tiempo de desarrollo
- Gratuito en tiempo de producción tanto tiempo como se desee
- Potencia ajustada pero suficiente a la cantidad de usuarios que se espera tener

- Almacenamiento SQL gratuito
- Posibilidad de usar lenguajes de programación ya conocidos
- Posibilidad aumentar la potencia migrando a otro plan

Una vez cerrada la decisión del servicio PaaS, pasamos a elegir los marcos de trabajo (framework) que vamos a usar.

3.4.5. Lenguajes de programación y nuestras capacidades técnicas

Elegir los lenguajes de programación fue una tarea no muy complicada, ya que tenemos entre todos los integrantes del proyecto un fuerte conocimiento en lenguajes como Java, PHP y Javascript. Estos lenguajes nos permitirán llevar a un siguiente nivel nuestras expectativas del proyecto.

Ventajas de un framework cualquiera:

- Mejorar la organización de código y estructura de ficheros
  - Evita el aislamiento de código
  - Elimina ambigüedades
  - Aumenta la legibilidad
  - Estandarización de términos y contenidos
- Aplicar el patrón MVC
  - Permite una correcta separación de las tareas
  - Reutilización del código
  - Mejora la abstracción separando los conceptos de presentación y lógica de negocios.
  - Más fácil de mantener
  - Desarrollo escalar
- Seguridad
  - Protege a la aplicación de los ataques más comunes
    - Inyección SQL: usando PDO
    - Ataques XSS:
    - Ataques DDoS: evitando escribir en los formularios los nombres de los scripts PHP de nuestro servicio.
- Encriptación de cookies

1. Frameworks candidatos de PHP

Una vez se ha decidido que se usará un framework PHP nos disponemos a elegir el que mejor que se ajuste a nuestras necesidades.

- **Cake PHP:** Este framework sigue el patrón de programación modelo vista controlador pero en la web oficial se sugiere que la curva de aprendizaje es elevada, además de establecer muchos convenios que no hay en ningún otro framework.
- **Yii PHP:** Muy similar a Cake PHP, pero a diferencia de Cake, la inversión de tiempo de aprendizaje de este framework para el desarrollo del proyecto es es bastante más alcanzable para nuestros propósitos.

3.4.6. Alcance del proyecto

Llegados a este punto es importante plantear cómo se estructura el desarrollo del proyecto. Se determina pues, que se dispondrá de tres subproyectos que en su conjunto componen todo el proyecto de la asignatura de Sistemas Informáticos.

- **Aplicación web** bautizado como Rescue in Cloud, y luego finalmente renombrado como Emergency Medical Services. Esta aplicación permite a los usuarios dar de alta protocolos de actuación, fármacos e informes o notas de intervención
- **Servicio web**: este servicio mediante un API permite sincronizar la información del cliente Android con la información mostrada en la aplicación web.
- **Cliente Android**: esta aplicación esta enfocada para el tipo de usuario que haga un uso productivo de información almacenada y customizada por el propio usuario en la aplicación web.

  Además, un aspecto interesante de esta aplicación es que dispone de un módulo de procesamiento de texto para poder inferir y sugerir equivalencias con los distintos tipos de unidades encontradas en los protocolos.

A continuación se describe la estrategia seguida en cada subproyecto y las decisiones tomadas en ella durante su fase de desarrollo.

1. **Emergency Medical Service (aplicación web)**

➢ **Protocolos**

Estos protocolos ofrecen una herramienta para hacer diagramas de flujo de manera que a través de una serie de pasos y decisiones se pueda diagnosticar el cuadro clínico de una persona.
Para desarrollar una herramienta de estas características, la primera dificultad residía en cómo crear los protocolos.
Las características deseadas, es que sea intuitivo, fácil de usar y aprender y que visualmente sea agradable y fácil de entender, para esta tarea, se encontraron algunas de las siguientes librerías de grafos:

- **arbor.js**: herramienta muy sofisticada pero con más dinamismo de lo que necesitamos.
- **D3.js**: Poderosísima herramienta Javascript para presentar datos de manera clara y dinámica.
- **Processing.js**: una librería Javascript artística que no se ajustaba para nada a lo que su descripción en la web oficial señala.
- **RaphaelJS's Graffle**: esta librería al principio gustó mucho, pero enseguida se nos hizo muy limitada. Esta librería no permitía colocar los nuevos nodos del grafo en vertical y horizontal, de manera que si bien la librería colocaba los nuevos nodos de forma inteligente, esta inteligencia podía llegar a ser el lugar menos intuitivo para el usuario.
- **Graph JavaScript framework**: librería que es capaz de dibujar grafos de manera dinámica, pero con el problema de no poder albergar textos de longitud muy grandes sin que ello afectase a la presentación visual.
- **Cytoscape.js**: esta librería soluciona el problema del colocado inteligente de los nuevos nodos, pero le sucede lo mismo que a la librería RaphaelJS's, la presentación visual se ve mermada cuando se introducen textos muy largos.

- **JS Graph It**: librería muy prometedora, incluso formó parte del desarrollo al principio del mismo, pero finalmente se descartó debido a la poca documentación que había disponible para modificar el código. Este fue un punto importante en el desarrollo ya que afinamos más nuestras exigencias, y nos dimos cuenta de que en realidad necesitábamos un software de diagramas de flujo.
- **Flowchart UI**: esta librería es un proyecto independiente pero en el que el objetivo es satisfacer muchas de las funcionalidades que el proyecto EMS necesita.
  Esta librería fue lanzada el 4 de Abril del 2014, y es una herramienta con licencia de pago pero al que los autores (Ricardo Champa y Adriano Raiano) han concedido la gratuidad por colaborar con la fase de pruebas de la misma.

  A continuación vemos un ejemplo de la herramienta:

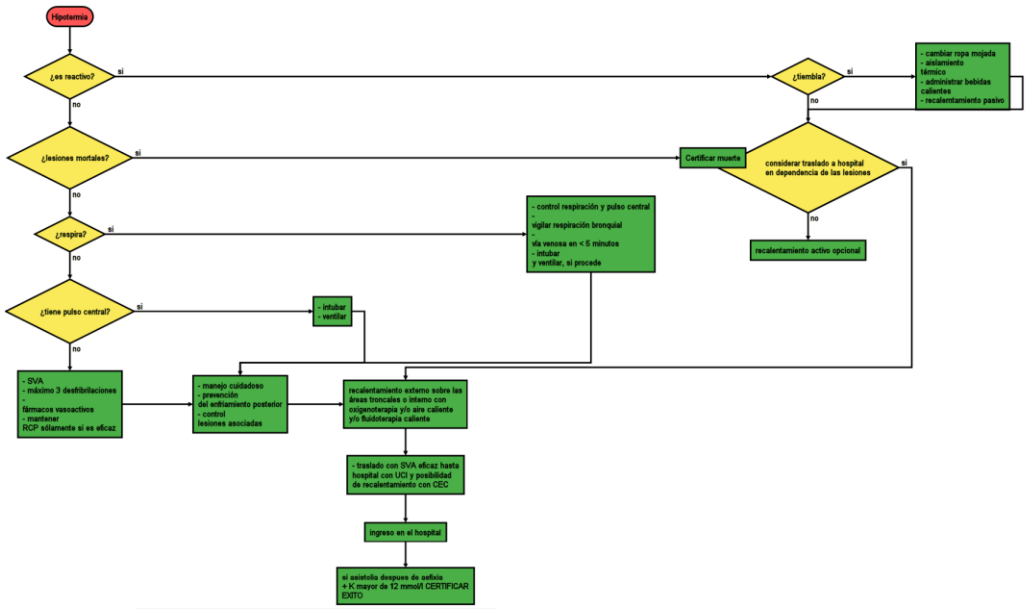


Ilustración 15: Ejemplo de la herramienta.

➢ Fármacos y notas de intervención

Una vez decidido cómo se van a representar los protocolos, queda por resolver los fármacos y las notas o informes de intervención.

En un principio se busca un diseño homogéneo para las operaciones de protocolos, fármacos y notas. En una primera versión se intenta impregnar un estilo propio para la aplicación web, se consigue un estilo intuitivo y agradable, pero resulta muy difícil que todos los integrantes del grupo apliquen un estilo homogéneo.

2. Buscando un diseño elegante, homogéneo e intuitivo

Se procede a buscar inspiración para un diseño elegante e intuitivo[8], es decir, librerías que nos ayuden a centrarnos más en la lógica de negocio y menos en el diseño de una interfaz que bien puede llevar gran parte del tiempo de desarrollo intentando encontrar

una armonía de colores y colocar los elemento de manera que no se sobrecargue al usuario con información a la par de mostrar los contenidos de forma clara y ordenada.

Algunas de las librerías que formaron que pudieron ser usadas en el proyecto:

- **Yui library**: librería Javascript y CSS con un diseño soberbio y serio.
- **HTML5 Boiler plate**: librería HTML5 y CSS3 con un diseño bonito y poca documentación.
- **Pure library**: librería que muy agradable visualmente y con diseño responsive.
- **Bootstrap Twitter**: librería CSS y Javascript, muy agradable visualmente, con diseño responsive y con gran variedad de ejemplos y documentación.

Finalmente, Bootstrap y Pure eran las opciones que más gustaron

➢ **Pure**

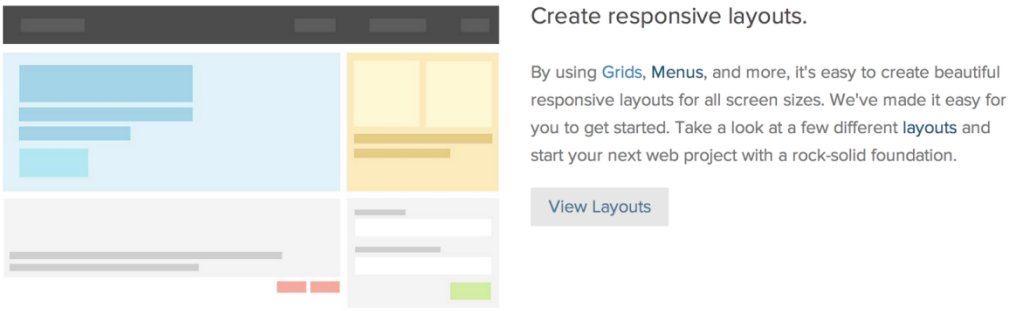


Ilustración 16: Pure.

➢ **Bootstrap**

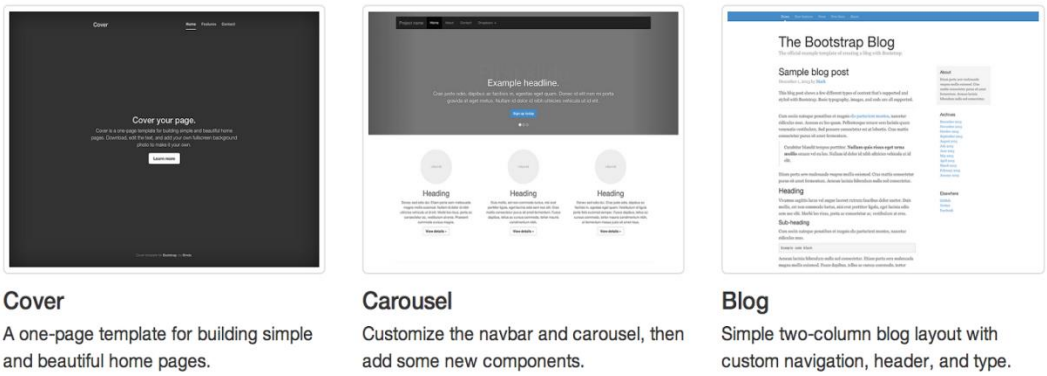


Ilustración 17: Bootstrap.

Finalmente se decide usar Bootstrap debido a su gran variedad de ejemplos y flexibilidad para ampliar las hojas de estilos CSS.

3. Servicio web

El servicio web[9] es el encargado de sincronizar los contenidos de la aplicación Android con la aplicación web, y se debe definir si el servicio es RESTFul o SOAP.

➢ **SOAP**

Es el paradigma de comunicación a través de mensajes sin estado y por lo general el transporte de datos se realiza en formato XML.

Un mensaje SOAP es un documento XML con una estructura definida en la especificación del protocolo:

- **Envelope** (obligatoria): raíz que de la estructura, es la parte que identifica al mensaje SOAP como tal.
- **Header**: esta parte es un mecanismo de extensión ya que permite enviar información relativa a cómo debe ser procesado el mensaje. Es una herramienta para que los mensajes puedan ser enviados de la forma más conveniente para las aplicaciones. El elemento "Header" se compone a su vez de "Header Blocks"que delimitan las unidades de información necesarias para el header.
- **Body** (obligatoria): contiene la información relativa a la llamada y la respuesta.
- **Fault**: bloque que contiene información relativa a errores que se hayan producido durante el procesado del mensaje y el envío desde el "SOAP Sender" hasta el "Ultimate SOAP Receiver"

➢ **REST**

Por otro lado un servicio de tipo REST que aporta escalabilidad como resultado de una serie de diseños fundamentales clave:

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)
- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos (ABMC en castellano: Alta, Baja, Modificación y Consulta) que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.
- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML, XML o JSON. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

➢ **Operaciones GET y POST**

Al ser de carácter procedural y orientado a servicios sólo queda la decisión de definir las operaciones mínimas que tendrá la API que dispondrá las operaciones GET y POST.

- **listar_protocolos**: esta función ha de recibir un usuario y su contraseña y como respuesta, se devuelve la lista de protocolos asociados a un usuario en formato JSON.
- **listar_farmacos**: esta función ha de recibir un usuario y su contraseña y como respuesta, se devuelve la lista de fármacos asociados a un usuario en formato JSON.
- **listar_notas**: esta función ha de recibir un usuario y su contraseña y como respuesta, se devuelve la lista de notas asociados a un usuario en formato JSON.
- **login_android**: esta función ha de recibir un usuario y su contraseña y como respuesta, se devuelve la lista de protocolos, fármacos y notas asociados a un usuario en formato JSON.

4. Cliente Android

Actualmente hay muchas alternativas en el desarrollo del mundo móvil, por un lado tenemos aplicaciones nativas y por otro, aplicaciones multiplataforma.

Las ventajas de aplicaciones multiplataforma[10], se basan sobretodo en ahorrar tiempo de desarrollo, es decir, desarrollando la aplicación una vez y desplegándola en las muchas otras plataformas tales como Android, IOS, Windows Phone, Firefox OS, Ubuntu OS y Blackberry.

➢ **Phonegap**

Este kit de desarrollo permite realizar con Javascript, HTML y CSS un desarrollo y a través de modificaciones pequeñas desplegar la aplicación en las distintas plataformas móviles.

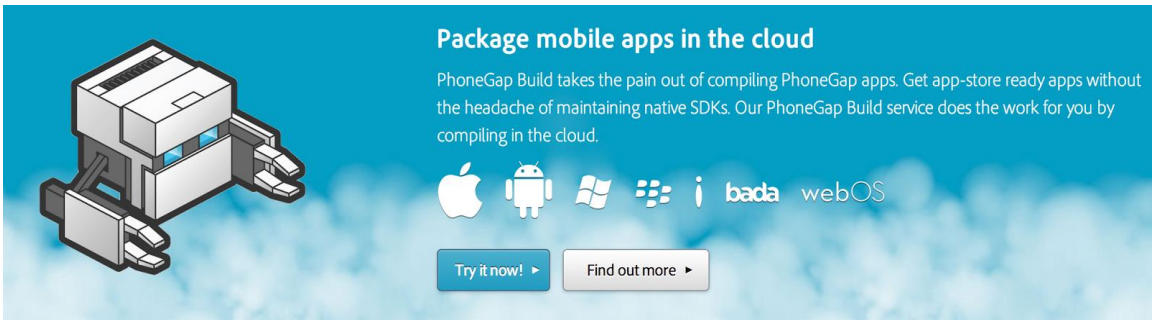


Ilustración 18: Phonegap.

➢ **Android SDK**

El desarrollo nativo[11] por otro lado, juega su principal basa en aprovechar toda la potencia que ofrece el kit de desarrollo software que pone Google a disposición de los desarrolladores. Muchas de las funcionalidades sólo están disponibles en Android SDK.

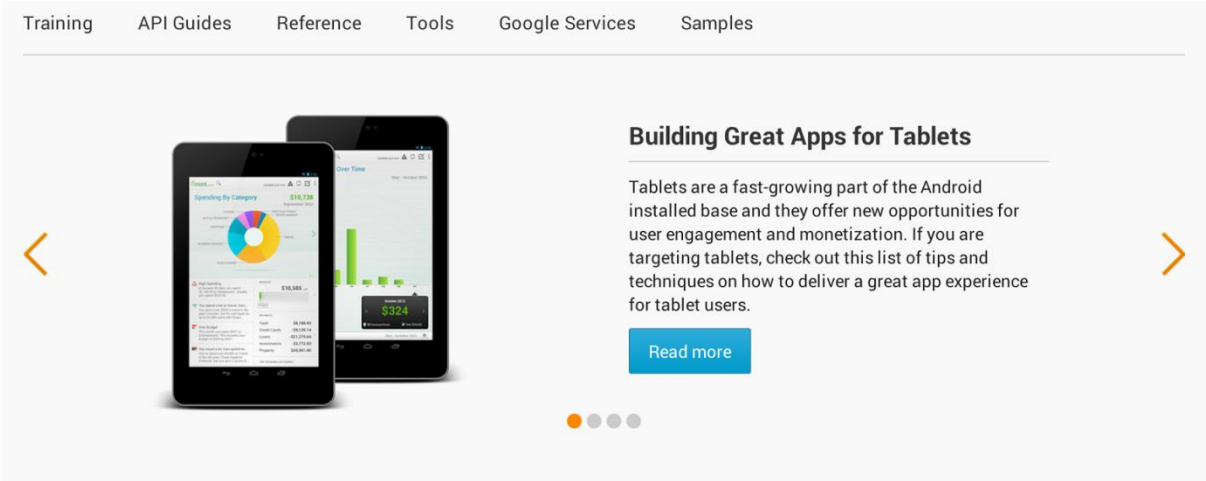


Ilustración 19: Android sdk

Las ventajas del desarrollo multiplataforma son tentadores, pero finalmente se decide apostar por una aplicación nativa en Android para aprovechar muchas de las atractivas funcionalidades de las que dispone.

5. Balanceador de carga (HAProxy)

El balanceador de carga[12] juega un papel importante en el escalado automático. En el caso de Openshift el balanceador distribuye las peticiones sobre los engranajes disponibles y son ellos quienes construyen la respuesta efectiva al cliente.

Algunas de las características de este balanceador son:

- Soporte de sesiones, de manera que el usuario no pierda la sesión.
- Detectar engranajes caídos para no dirigir peticiones a ellos.
- Algoritmo de balance round robin para elegir un balance por peso, origen, etc.
- Escalabilidad horizontal, de manera que añadir nuevos engranajes no suponga hacer cambios y sea transparente de cara al desarrollo.

HAProxy por tanto es imprescindible en el escalado automático de engranajes y la decisión de emplearlo en el proyecto es obligatoriamente positiva.

3.5. Vista lógica

3.5.1. Visión general

Mediante el siguiente diagrama resumimos la estructura general de EMS in Cloud.

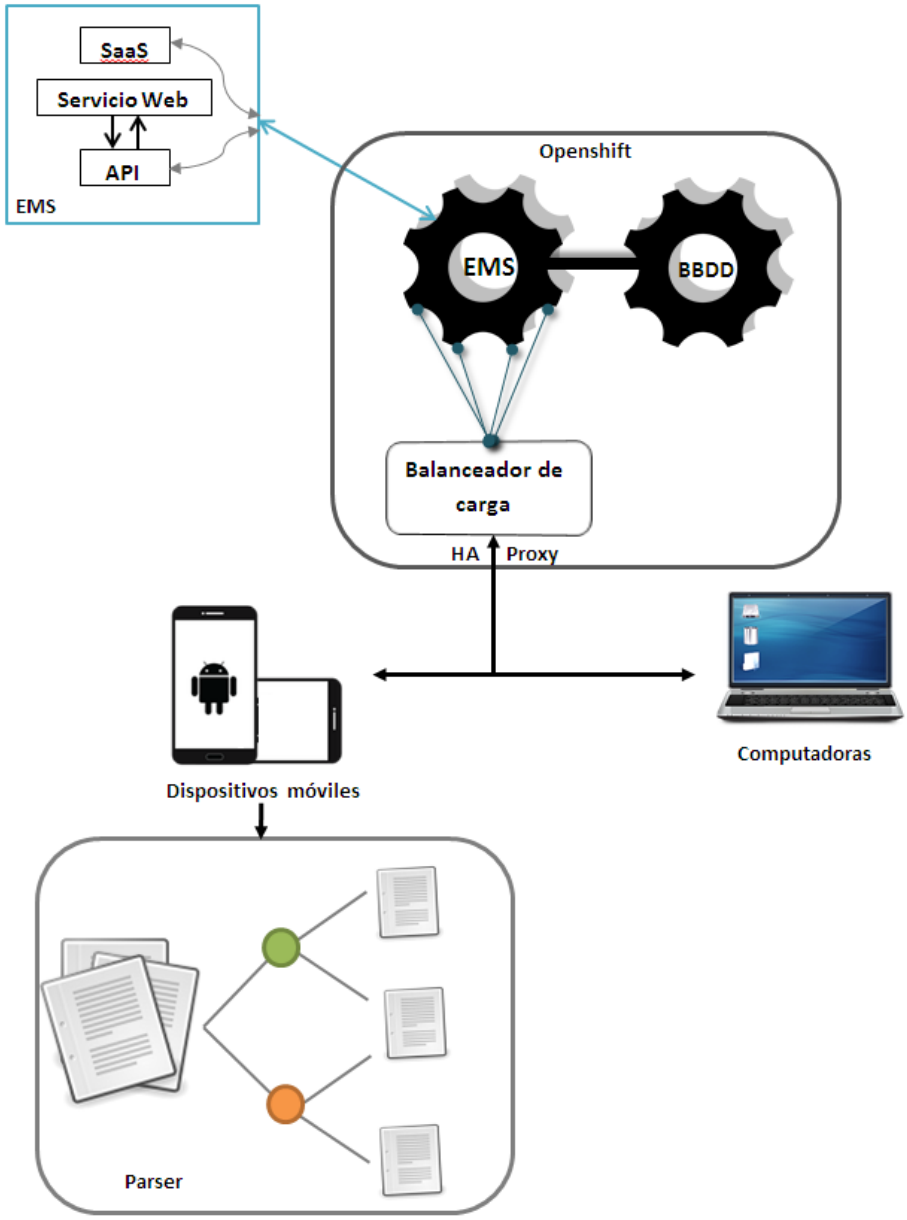
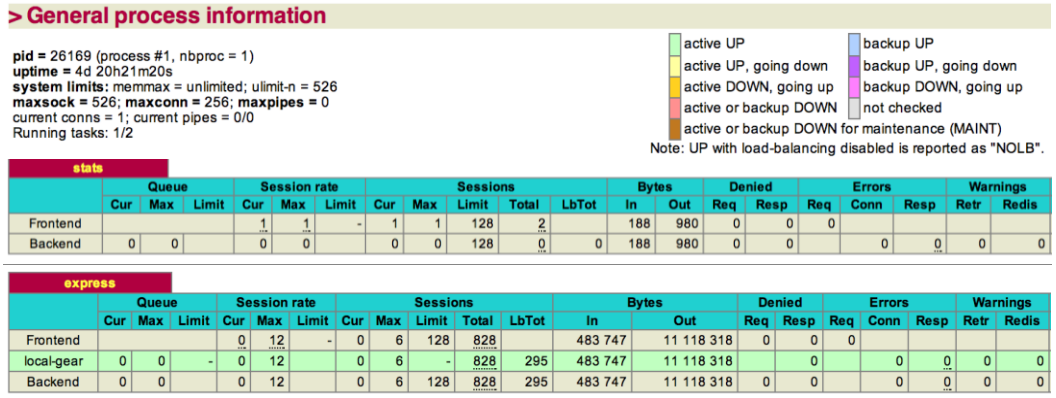


Ilustración 20: Diagrama de visión general.

Openshift es el servicio PaaS que sustenta nuestro proyecto, de esta manera dentro de él tenemos:

- **EMS**:
  EMS es el nombre elegido para la aplicación web y que ahora mismo dispone dos engranajes para balancear la carga si fuese necesario.

- **El balanceador de carga**:
  El elegido para este proyecto es High Availability Proxy (HAProxy), a continuación se pueden ver algunas de las estadísticas, entre ellas, las conexiones y sesiones actuales, el balanceo a distintas IP's si lo hubiera, y el número de engranajes usados.



> General process information

pid = 26169 (process #1, nbproc = 1)
uptime = 4d 20h21m20s
system limits: memmax = unlimited; ulimit-n = 526
maxsock = 526; **maxconn =** 256; **maxpipes =** 0
current conns = 1; current pipes = 0/0
Running tasks: 1/2

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
Note: UP with load-balancing disabled is reported as "NOLB".

stats

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis
Frontend				1	1	-	1	1	128	2		188	980	0	0	0				
Backend	0	0		0	0		0	0	128	0	0	188	980	0	0		0	0	0	0

express

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis
Frontend				0	12	-	0	6	128	828		483 747	11 118 318	0	0	0				
local-gear	0	0	-	0	12		0	6	-	828	295	483 747	11 118 318		0		0	0	0	0
Backend	0	0		0	12		0	6	128	828	295	483 747	11 118 318	0	0		0	0	0	0

Ilustración 21: Información general de funcionamiento de HAProxy.

De esta manera todas las solicitudes web son atendidas primero por el balanceador de carga y son los servidores los que generan la respuesta efectiva al cliente, dentro de los servidores se accede al gestor de bases de datos MySQL.

Dentro de EMS tenemos el software como servicio http:/ems-rczone.rhcloud.com/ que permite a un usuario acceder a la gestión de protocolos, fármacos y notas.

Dentro de EMS también tenemos un API que permite acceder al servicio web http://ems-rczone.rhcloud.com/ws . A esta API hay que acceder por medio de solicitudes del protocolo HTTP get o post. Siempre que se realice una solicitud hay que especificar el recurso a solicitar, nombre de usuario y contraseña.

- **Bases de datos**:
  El motor de almacenamiento para la base de datos es InnoDB, este motor es de tipo transaccional por lo que las propiedades atomicidad, consistencia, aislamiento y durabilidad están garantizadas.
  La codificación y juego de caracteres para este proyecto en general y no sólo para la base de datos es UTF-8.

  Finalmente, las definiciones de relaciones se pueden ver en el siguiente modelo relacional.

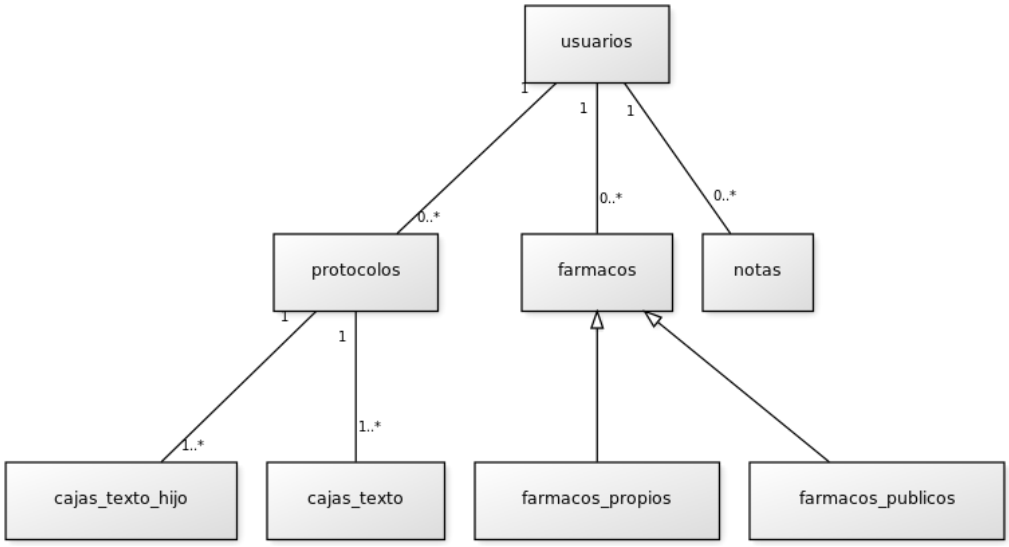


Ilustración 22: Diagrama ER del sistema EMS in Cloud.

- **Android**:
  El enfoque seguido en la aplicación Android es que su uso sea de sólo lectura, por eso la funcionalidad más importante es la sincronización con el servicio web.
  En la comunicación con el servicio web lo importante es la autenticación en cada solicitud, por eso cuando el usuario se autentica con éxito estos datos son guardados en las preferencias de la aplicación, de manera que antes de cada solicitud se recuperen los datos necesarios.
  Una de las características más atractivas de la aplicación es la detección de unidades.

1. Administrador de sistemas Openshift

Antes de adentrarnos en la visión general del cliente Web y Android, debemos hablar previamente de Openshift.

Openshift nos ofrece una serie de herramientas por la cual podemos acceder a todas las características de sus servicios como Paas.

- **Desplegar la aplicación**:
  Openshift ofrece una manera muy sencilla de desplegar contenidos en su plataforma. Esta manera es usando el sistema de control de versiones git. Teniendo en cuenta los conceptos básicos de git, si en algún momento del desarrollo se desee desplegar nuevo contenido en Openshift debemos primero instalar el intérprete de comandos de Openshift llamado RHC.

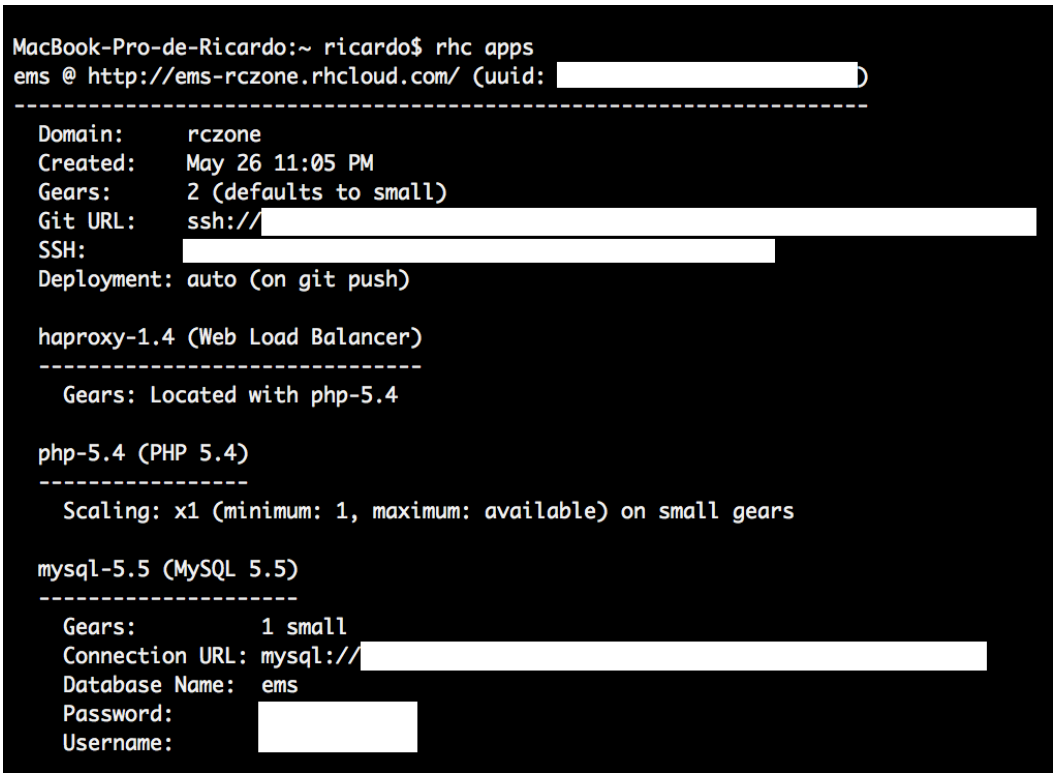


Ilustración 23: Representación de las aplicaciones alojadas en Openshift.

En esta imagen se han omitido los datos más sensibles por razones de seguridad.

Si se necesita acceder a la gestión de MySQL, Openshift permite conectarse a su servicio por medio de un túnel (redirección de puertos) para acceder a la IP privada que Openshift dispone para nosotros.

Para conseguir esto, sólo hay que escribir en el intérprete de comandos la siguiente instrucción: $~ rhc port-forward -a ems
Con este comando se consigue la redirección de puertos a la IP que tiene asignada nuestro proyecto llamado ems.

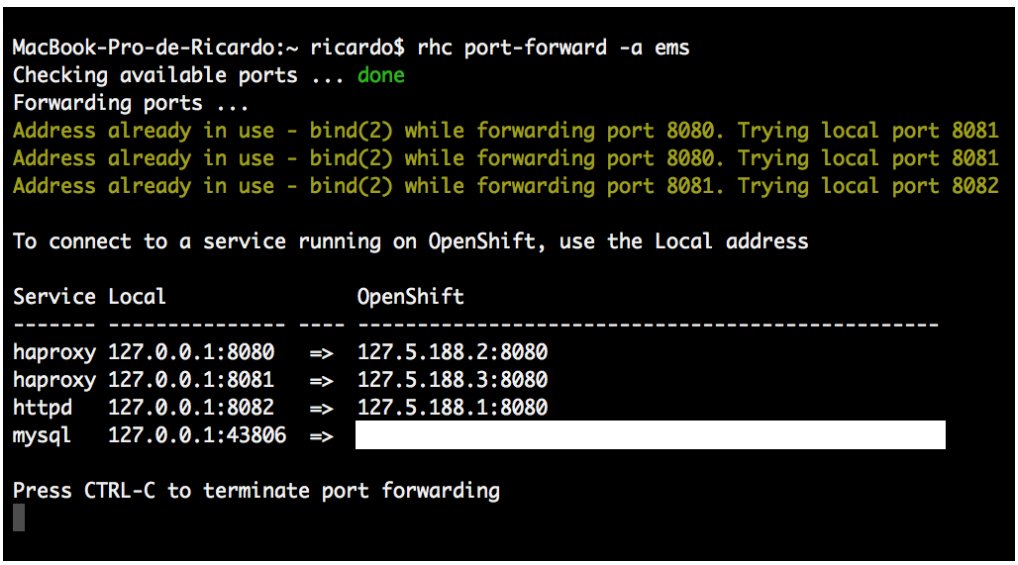


Ilustración 24: Representación del acceso a la gestion de MySQL con Openshift.

Típicamente la dirección 127.0.0.1 es conocida como localhost, una vez considerado esto vemos que en este caso Openshift va a utilizar en este caso el puerto 43806 (podría ser otro) para la comunicación entre un equipo y el servidor que nos permitirá acceder algunas características, entre ellas el acceso a la base de datos.

Finalmente ya tenemos toda la información que necesitamos para conectarnos a la base datos de manera remota:

```
MacBook-Pro-de-Ricardo:~ ricardo$ mysql -u[redacted] -p[redacted] -h 127.0.0.1 -P 43806
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 286
Server version: 5.5.37 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Ilustración 25: Representación de conexión a la BBDD de forma remota mediante Openshift.

A partir de aquí, los conocimientos adquiridos en la asignatura de Base de Datos y Sistemas de la Información nos serán de gran ayuda.

```
mysql> select * from farmacos_publicos;
+------------+---------------+---------------------+------------------------+---------------------+
| id_farmaco | nombre_farmaco | nombre_fabricante    | presentacion_farmaco    | tipo_administracion |
+------------+---------------+---------------------+------------------------+---------------------+
|          1 | Aspirina       | Bayer                | 500mg                   | comprimidos         |
|          2 | Diazepam       | Bayer                | sobres                  | oral                |
|          3 | Paracetamol    | Cinfa                | Comprimidos             | oral                |
|          4 | Gelocatil      | Bayer                | Comprimidos             | oral                |
|          5 | Frenadol       | Johnson&Johnson      | Sobres                  | Oral                |
|          6 | Lizipaina      | Boehringer Ingelheim | Comprimidos masticables | Bucofaringeo        |
|          7 | Neobrufen      | Abbott               | Comprimidos             | Oral                |
+------------+---------------+---------------------+------------------------+---------------------+

+---------------------+---------------------+--------------------+-------------+---------+
| creado_en           | modificado_en       | descripcion_farmaco | sincronizado | borrado |
+---------------------+---------------------+--------------------+-------------+---------+
| 2014-01-13 17:46:18 | 0000-00-00 00:00:00 |                    |           0 |       0 |
| 2014-01-13 18:27:03 | 0000-00-00 00:00:00 |                    |           0 |       0 |
| 2014-01-22 17:07:31 | 0000-00-00 00:00:00 |                    |           0 |       0 |
| 2014-01-22 17:31:51 | 0000-00-00 00:00:00 |                    |           0 |       0 |
| 2014-03-31 17:32:57 | 0000-00-00 00:00:00 |                    |           0 |       0 |
| 2014-03-31 18:50:39 | 0000-00-00 00:00:00 |                    |           0 |       0 |
| 2014-03-31 17:35:45 | 0000-00-00 00:00:00 |                    |           0 |       0 |
+---------------------+---------------------+--------------------+-------------+---------+
 7 rows in set (0.20 sec)
```

Ilustración 26: Ejemplo de consulta SQL usando Openshift.

2. Cliente web

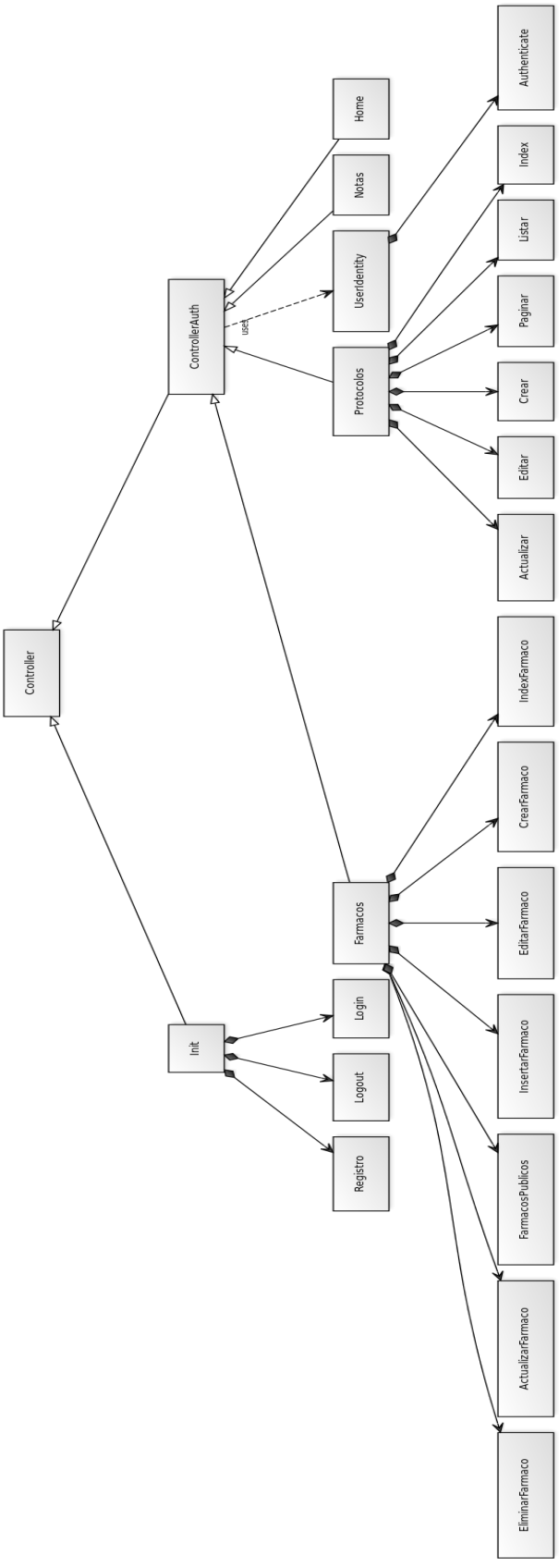


Ilustración 27: Diagrama UML de clases – Web.

Recordemos que la aplicación web a la que hemos llamado EMS, está basada en el framework Yii de PHP para el backend y Bootstrap para el frontend.
Para describir este cliente web, se empieza por describir la estructura de directorios.

Dentro de la raíz del proyecto hay dos tipos de directorios:

- públicos
- protegidos

Además también se puede encontrar el fichero index.php que es el entry-point (el main de java o C) de la aplicación. Es importante no olvidar que este fichero debe ser conocedor de la ubicación del directorio framework.

Adicionalmente, también se puede usar el fichero index-test.php como entry-point para la gestión de pruebas.

- **Directorios públicos:**

En el framework Yii y particularmente para nuestro proyecto los directorios públicos, es decir, los recursos que pueden ser accedidos mediante una petición HTTP y ejecutadas desde el cliente (en el caso de javascript) son:

- assets
- css
- images
- js
- themes

- **Directorio protected:**

En Yii y para nuestro proyecto existe un directorio llamado *protected* al que no se puede acceder por medio de una petición HTTP gracias al fichero *.htaccess* que en su contenido tiene la sentencia "deny from all"

Si se intenta acceder a este directorio lo que se obtendrá es el siguiente mensaje:

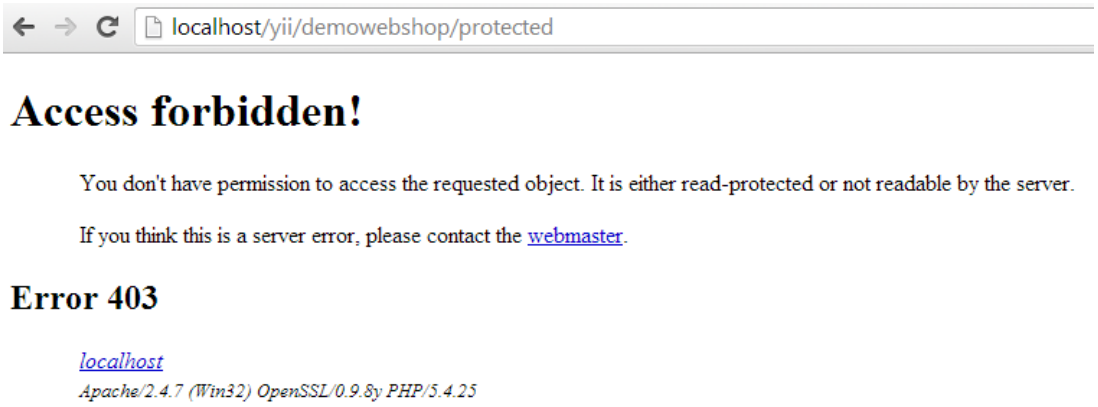


Ilustración 28: Ejemplo de acceso al directorio *protected* de Yii.

Por eso este directorio es el encargado de albergar toda la lógica de la aplicación.

Dentro del directorio protected encontramos los siguientes directorios:

- **Directorio components:**

Alberga todos los componentes que se quieran redefinir del core de Yii. Esto se puede configurar desde el fichero config/main.php que se explicará más adelante.
Dentro de este directorio hay tres clases importantes:

  - Controller: esta es la clase base que permite gestionar todos los enlaces de la aplicación.
  - ControllerAuth: permite la verificación y autentificación de usuarios de la aplicación.
  - UserIdentity: se encarga de almacenar los métodos necesarios para la identificación de un usuario a partir de los datos provistos.

- **Directorio config:**

Aquí está almacenado el fichero main.php, es el único archivo de configuración que se basa en componentes. Esto es una gran ventaja de cara a la reutilización.
En este fichero está configurado:

  - la base de datos
  - el gestor de url's
  - el gestor de errores
  - el log

- **Directorio controllers:**

Albergará todas las clases que extiendan de la clase CController.php.

- **Directorio extensions:**

Directorio creado para contener las extensiones creadas por la comunidad.

- **Directorio messages:**

Este directorio almacena los mensajes de traducción de la aplicación. Sirve para hacer la aplicación multiidioma.

- **Directorio models:**

Este directorio contiene los modelos de la aplicación. En este proyecto hay dos tipos de modelos, modelos de bases de datos (DAOs) y modelos de datos (ValueObjects).

- **Directorio runtime:**

Este directorio almacena el log de la aplicación.

- **Directorio test:**

Sirve para realizar las pruebas unitarias y funcionales con ayuda de Selenium.

- **Directorio vendor:**

Directorio que almacena herramientas de third partys.

- **Directorio views:**

Este directorio se encarga de almacenar las vistas de la aplicación.

3. Cliente Android

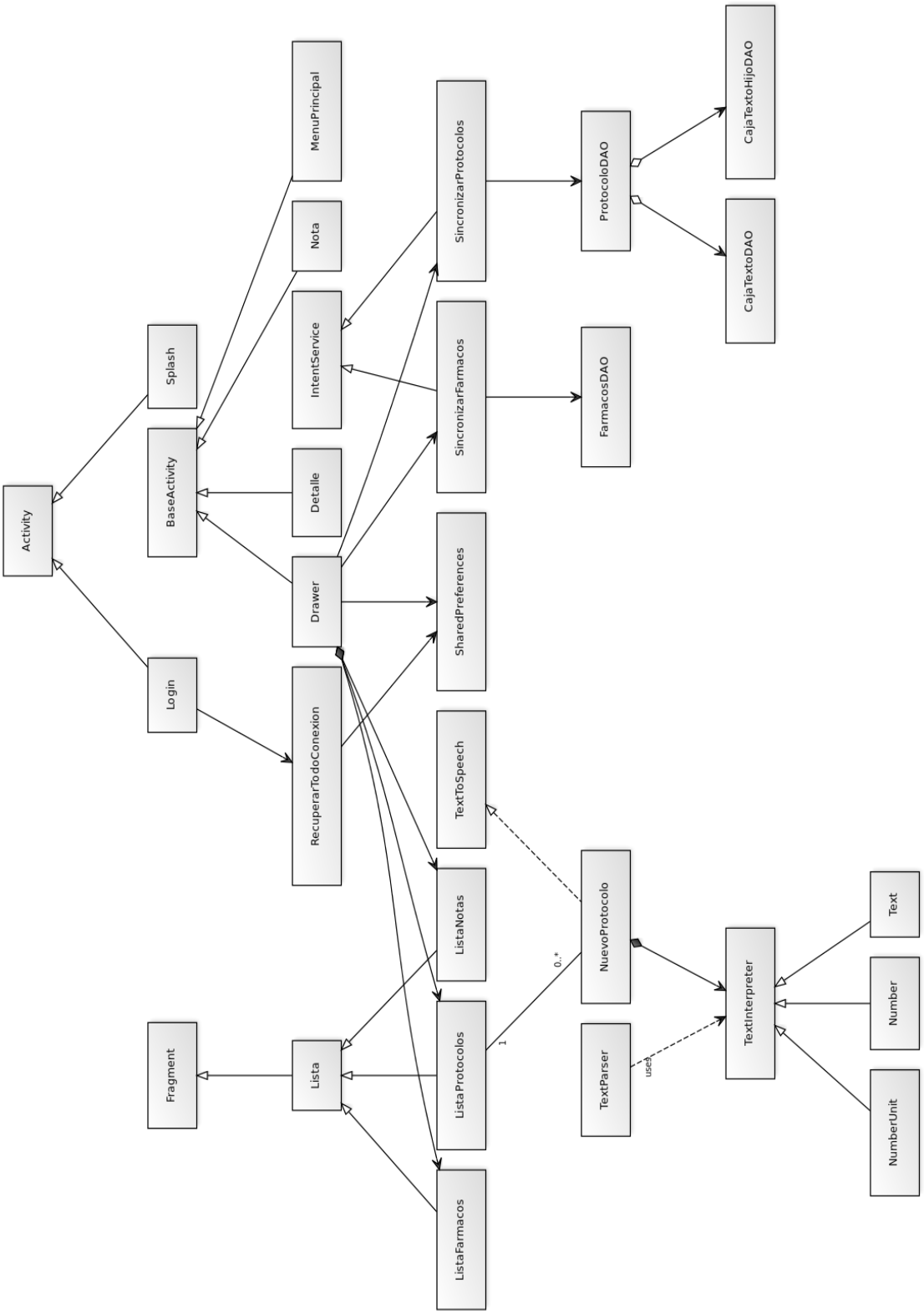


Ilustración 29: Diagrama UML de clases – Android.

Este diagrama muestra los paquetes principales que constituyen la aplicación Android para EMS in Cloud.

A continuación se describe la estructura de directorios y paquetes Java dentro de él:

- **Directorio src**:
  En esta carpeta se encuentra toda la lógica de negocio. A continuación un resumen de los paquetes y clases más importantes.

  - **MyApp**: esta clase ubicada en la raíz del directorio src se encarga de cargar las definiciones de la base de datos sqlite desde un fichero predefinido y almacenado en res/raw/rescue_lite_db, es decir las sentencias que permiten crear las tablas. Además, si en el futuro las definiciones de estas tablas necesitan ser actualizadas, el código en esta clase permitirá migrar el contenido que el usuario final mantiene a las nuevas definiciones a través de una actualización de la versión del proyecto.

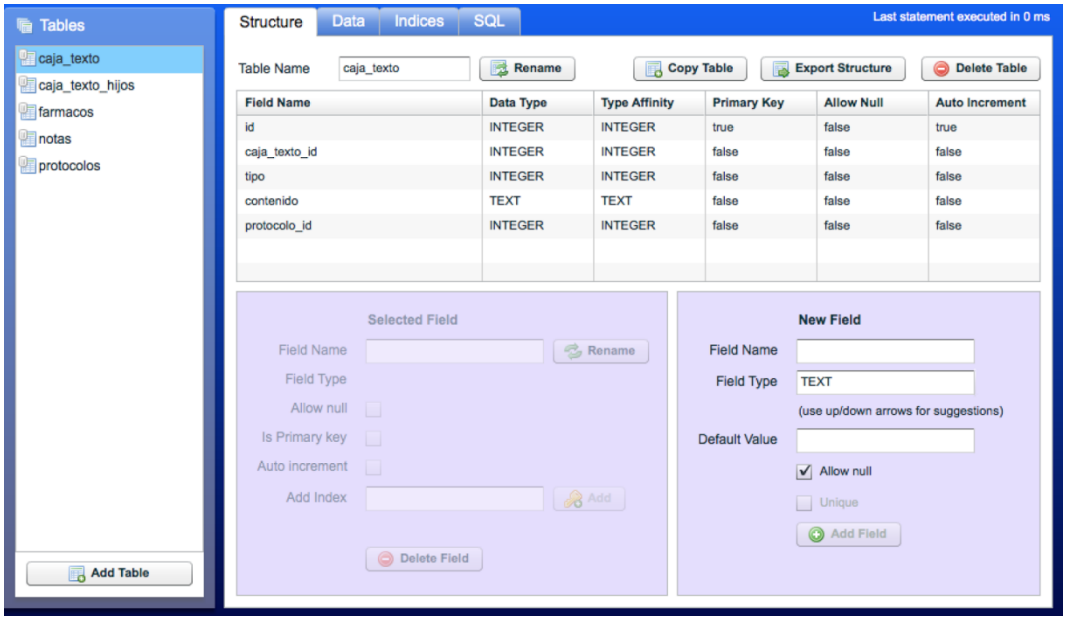


Ilustración 30: Ejemplo de interacción con la base de datos sqlite.

- **Paquete dao**: en este paquete se encuentra el CRUD de la aplicación, los nombres de las clases dentro de él describen bien las consultas SQL que se realizan en ellas.

- **Paquete peticiones**:
  - Operaciones: esta clase gestiona el tratamiento de la información que se recibe del servicio web.
  - SinocronizarFarmacosIntentService y SincronizarProtocolosIntentService: ambas clases definen cada una de ellas un servicio para comunicarse con el servicio web por cada solicitud web del tipo get. Lo importante de estos servicios es que se ejecutan en un segundo plano a la aplicación y no a la

actividad, esta característica permite que los procesos que desatan estos servicios podrán sobrevivir a cualquier interacción del usuario.

- RecuperarTodoConexion: cuando el usuario se autentica por primera vez, se realiza una conexión con el servicio web y se recupera toda la información. La diferencia de este proceso con los servicios descritos antes es que esta operación es bloqueante, y hasta que no se valide la autenticación el usuario no puede acceder a ninguna funcionalidad más de la aplicación.

- **Paquete Tools**: este paquete contiene clases como HttpPostConnector, AsyncConnect, BaseActivity, IDialogOperations que son abstracciones que permiten estandarizar el código y seguir patrones de programación.

- **Paquete Activities**:
  En este paquete se encuentran las clases de tipo:
  - Activity: clase que implementa una ventana independiente a otras.
  - Fragment: componentes visuales usada en distintos fragmentActivity.
  - FragmentActivity: clase similar a la clase Activity pero con la posibilidad de estar compuesto por uno o muchos fragments.
  - DrawerActivity: esta clase es la más destacable ya que es el contenedor de la pantalla principal que da la posibilidad de elegir con una lista desplegable desde la izquierda la posibilidad de acceder rápidamente a las opciones de protocolos, fármacos y notas.

➢ **Directorio res:**
En este directorio se encuentra todos los diseños de la interfaz visual. El convenio de nombres activity_ para las activitys y fragment_activitys, y fragment_ para los fragment java.

3.5.2. El paquete Parser

Está formado por las clases encargadas de parsear el texto cuando se accede al protocolo por primera vez y un paquete secundario que utilizamos en caso de que sea necesario modificar el contenido ya parseado debido, por ejemplo, a un cambio de unidades producido por el usuario.

1. La clase TextInterpreter

Es la clase abstracta que caracteriza a las clases Text, Number y NumberUnit por tener éstas el método abstracto *textInterpreterParseText()*. Este método se utiliza para tratar el texto contenido en una caja del diagrama del protocolo y determinar qué tipo de caja es. Cada una de las clases implementará este método con la finalidad de reconocer si el texto es adaptable a las características de la clase en cuestión.

También cuenta con el método que permite reconocer una relación de cantidad (mayor o menor) en un texto.

2. La clase TextParser

Constituye la clase principal del parser, ya que desde ella se accede a tratar el contenido del protocolo. Toma el contenido de cada una de las cajas y ejecuta los métodos de parseado de las clases Nuber, NumberUnit y Text. Cuando una de las clases reconoce el texto de acuerdo a sus características, desde la clase TextParser se instancia una nueva clase del tipo reconocido. Al finalizar el protocolo devuelve la relación de todas las cajas instanciadas a una de las clases que heredan de TextInterpreter, manteniendo toda la información relativa a ellas más la añadida por el tratamiento del lenguaje.

3. **La clase Number**

Caracteriza a las cajas de texto en cuyo contenido se encuentra un número sin estar seguido de una unidad**.** Se caracteriza por el identificador original de la caja, el contenido, el valor numérico de ese contenido y la relación de cantidad en caso de que exista esta última.

4. **La clase NumberUnit**

A diferencia de la anterior, se instanciarán objetos de esta clase cuando las cajas de texto contengan número y una unidad métrica**.** Se caracteriza por el identificador original de la caja, el contenido, el valor numérico de ese contenido, la unidad de medida y la relación de cantidad, otra vez, en caso de que exista esta última.

5. **La clase Text**

Esta clase determina a las cajas en las que no se encuentra ningún valor numérico. Sus atributos serán el identificador original de la caja y el texto contenido en ella.

6. **La clase Unidades**

Incluye todas las unidades de medida que reconoce el parser. De esta forma, si se quisieran añadir nuevas unidades, sólo sería necesario añadirlas en esta clase.

7. **Las clases RelacionCantidadMayor y RelacionCantidadMenor**

Contienen todas las palabras que reconoce el parser para determinar si una relación es del tipo “mayor que” o “menor que” respectivamente. Al igual que en las unidades, si se quisieran añadir nuevas palabras, sólo sería necesario añadirlas en esta clase.

8. **El paquete parser.ui**

Lo forman las interfaces IFuncion y SFuncion, que caracterizan los métodos que modifican el valor numérico de las clases parseadas y la unidad de medida (esto último sólo si es una clase de tipo NumberUnit).

También se encuentra la clase Conversiones que implementa las interfaces antes mencionadas para realizar los cambios de unidades solicitados por el usuario.

3.5.3. El paquete Model

Contiene las clases que modelan la información del protocolo, es decir, sus cajas de texto y las relaciones entre ellas. Se utiliza tanto antes como después de parsear el contenido. Está relacionado con el paquete DAO, desde el que se instancian los nuevos objetos de las clases pertenecientes al paquete de este apartado.

1. **La clase Protocolo**

En ella se almacena la información relevante del protocolo, es decir, su nombre, su identificador numérico, las cajas de texto que lo forman y sus relaciones. Además, se almacena el conjunto de cajas que son de toma de decisiones.

2. **La clase ProtocoloParseado**

Extiende de la clase Protocolo. Contiene toda la información del protocolo y el conjunto de cajas parseadas.

3. **La clase CajaTexto**

Contiene los atributos que caracterizan a cada una de las cajas: su identificador, el identificador del protocolo al que pertenece, el contenido de la caja y el tipo de la caja, esto último determina si es de decisión o no.

4. **La clase CajasHijos**

Relaciona cada caja de texto con todos sus hijos directos. Cada caja de texto será de tipo Tupla, explicado a continuación.

5. **La clases Tupla y TuplaParseada**

Representan una caja de texto mediante su identificador numérico y el tipo de relación que tiene con su caja “padre”. El tipo de relación dependerá de si su padre es una caja de tipo decisión o no. Si es de tipo decisión, dependerá de si en el camino del padre al hijo se llega respondiendo “sí” o “no”, y si no será una relación directa.

En el caso de la clase TuplaParseada, para cada caja, no sólo se almacena su identificador sino toda la información de la caja ya parseada.

6. **La clase Fármaco**

Contiene toda la información propia de los fármacos: su identificador, su nombre, el nombre del fabricante, la presentación del fármaco, el tipo de presentación y su descripción.

7. **La clase Nota**

Incluye la información propia de los fármacos: su identificador, su nombre y su descripción.

Capítulo 4

Conclusiones

4.1. Principales conclusiones.

EMS in Cloud es un proyecto que cubre una necesidad real. Hemos puesto en práctica nuestros conocimientos para detectar las necesidades de un cliente, obtener a partir de ellas nuestros objetivos, elaborar un plan para conseguirlos y, por último, realizar el proyecto.
Después de someter el producto a la valoración del cliente, obtuvimos nuestro mayor logro, su satisfacción.

Hemos obtenido nuevos conocimientos en cuanto al Cloud Computing así como la programación en Android. También hemos conocido muchas tecnologías al tener que hacer una valoración inicial, como ya explicamos en el apartado de decisiones de diseño.

Trabajar de la mano de un experto ha sido positivo para nosotros dado que le ha dado un toque de realidad a nuestra experiencia y ha contribuido a construir una herramienta que resuelve un problema real.

Durante estos diez últimos meses, hemos aprendido a superar desafíos, a mejorar el trabajo colaborativo y encontrar vías de comunicación que nos permita mejorar nuestra organización personal y grupal.

Hemos aprendido a conocernos más y gracias a ello ahora podemos tomar mejores decisiones y asesorar a quién participe en proyectos de índole similar.

4.2. Conocimientos Adquiridos.

4.2.1. Computación en la nube

La velocidad a la que va la migración de proyectos software a la nube es de vértigo, por eso la experiencia adquirida durante el desarrollo de este proyecto es vital para nuestra futura inserción en el mundo laboral. Ahora hemos ganado la capacidad de tener una visión global y el entendimiento de las filosofías que encierran los nuevos enfoques tecnológicos.

Por otro lado, esta experiencia también sirve para animar a alguno de nosotros a entrar en el mundo empresarial aprovechando algunos de las grandes ventajas que ofrece la computación en la nube:

- eliminar los costes de adquisición y mantenimiento de infraestructura.
- eliminar los costes de licencias software privativas en favor del software libre.
- permitir que los usuarios finales formen parte activa del desarrollo de manera que ayuden a mejorar la calidad del software reportando fallos y errores.
- establecer un modelo de suscripción para que los usuarios que deseen una mejor experiencia puedan acceder a ella.

4.2.2. Infraestructura, plataforma y software como servicio

Infraestructuras con la potencia como la que tiene Amazon EC2 nos ha enseñado que tenemos a nuestro alcance posibilidades de ofrecer servicios y soluciones totalmente customizados y que se pueden adaptar a las necesidades más exigentes.

Plataformas como Openshift, permiten a los desarrolladores tener una herramienta profesional para empezar proyectos de cualquier envergadura.

Finalmente el software como servicio es la última capa de la cebolla de la computación en la nube, gracias a esto podemos administrar correctamente las responsabilidades, eliminar las interdependencias, ofrecer soluciones escalables, y sobretodo que hacen un mejor control de la economía de un proyecto software.

4.2.3. Balanceador de Carga

Este concepto es uno de los pilares en los que se basa la computación en la nube, y nos ha resultado particularmente atractivo el hecho de poder gestionar los recursos hardware bajo demanda, algo que hasta antes de la aparición de la computación de la nube significaban una tarifa plana mensual de pagos que muchas veces podía perjudicar a aquellos proyectos que no pueden afrontar grandes gastos.

4.2.4. Android y Java como herramienta de desarrollo

La evolución del sistema operativo Android es innegable, prueba de ello es la presencia que tiene en el día a día de muchas personas.

Aún con esto, de cara al desarrollo de aplicaciones Android el entorno que ofrece Google (empresa propietaria del sistema operativo Android) está aún pagando el precio de la fragmentación que hay presente en el mercado.

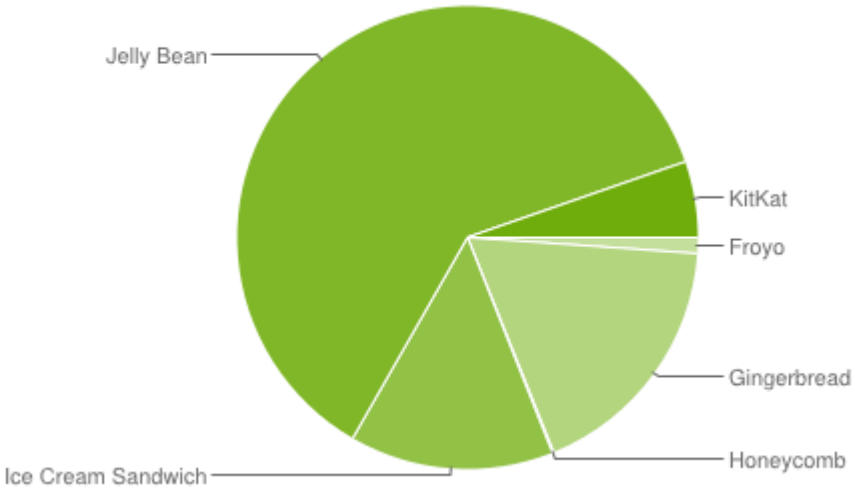


Ilustración 31: Fragmentación del mercado competente a Android.

Versión	Nombre en código	Fecha de distribución	Nivel de API	Cuota (1 de abril, 2014)
4.4	*Kit Kat*	31 de octubre de 2013	19	5.3%
4.3	*Jelly Bean*	24 de julio de 2013	18	8.9%
4.2.x	*Jelly Bean*	13 de noviembre de 2012	17	18.1%
4.1.x	*Jelly Bean*	9 de julio de 2012	16	34.4%
4.0.x	*Ice Cream Sandwich*	16 de diciembre de 2011	15	14.3%
3.2	*Honeycomb*	15 de julio de 2011	13	0.1%
2.3.3–2.3.7	*Gingerbread*	9 de febrero de 2011	10	17.8%
2.2	*Froyo*	20 de mayo de 2010	8	1.1%

Tabla 5: Fragmentación del mercado competente a Android.

Esta fragmentación ha hecho el proceso de desarrollo muy engorroso y la gestión de pruebas un proceso más complejo.

Por suerte, hay avistamientos de que esta fragmentación está cerca de llegar a su fin. Se espera pues que los usuarios que representan la versión Gingerbread con un 17.8% den el paso a adquirir la nueva versión Jelly Bean que si posee una amplia documentación y un gran entorno de desarrollo como Android Studio.

4.2.5. Tecnologías para el desarrollo de aplicaciones profesionales

Llegados a este punto entendemos que la experiencia universitaria no pasa por aprender las últimas tecnologías. Sino, se trata de que tener la capacidades de entender las decisiones tomadas hasta el día de hoy y que han llevado a la aparición de estas nuevas tecnologías.

Gracias a esto, hemos sido capaces de usar productivamente sin casi ayuda externa herramientas que pueden suponer una preparación extra para otras personas.

4.3. Repercusiones.

Todo el trabajo invertido en llevar a cabo este proyecto, se ha visto reflejado en un artículo de *HPCwire* gracias a su autor y nuestro director: José Luis Vázquez Poletti.

Ilustración 32: Artículo HPCwire.

El artículo titulado *Cloud Bolsters HPC/HTC Student Research* recoge las ideas básicas que constituyen EMS in Cloud.

Se puede consultar accediendo a través del siguiente enlace:

http://www.hpcwire.com/2013/12/09/cloud-bolsters-hpchtc-student-research/

With EMS in Cloud, the user can easily define and manage medical protocols to be used during emergency operations. These protocols use different metrics and provide action plans according the available medical supplies. If a given medical product is depleted or its composition changes, the user is then requested to modify the affected protocols, increasing the effectiveness of service.

Even if they can be shared, each user can have his own set of protocols. By using the cloud at a Platform as a Service level, both scalability and availability are guaranteed.

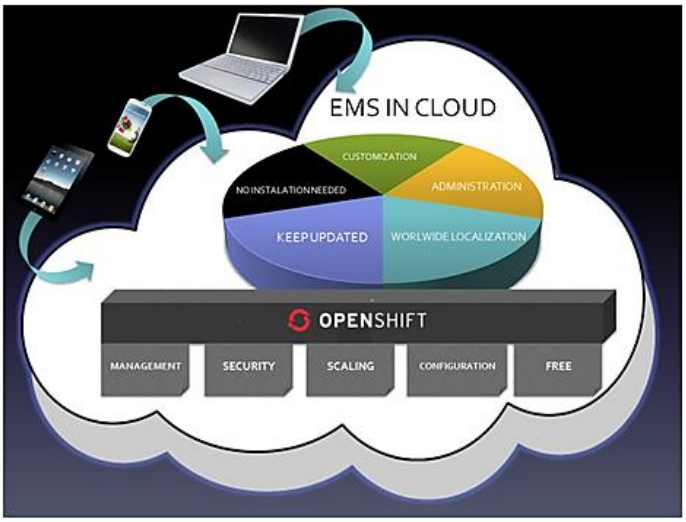


Diagram showing EMS in Cloud features.

During the emergency service, the user accesses the protocols through a mobile client, which is able to work offline with most of its functionality when no Internet is available.

The team is currently working in partnership with Manuel Vazquez, medical staff at the Spanish EMS. He is not only providing his knowledge, he will be also testing EMS in Cloud on the field.

Ilustración 33: EMS in Cloud en HPCwire.

Capítulo 5

Trabajo futuro

5.1. Áreas de trabajo

EMS in Cloud podría utilizarse en numerosos campos, no sólo en el de la medicina. Es una herramienta de ayuda a la toma de decisiones y altamente configurable, por lo que podría servir para cualquier ciencia que apoye su trabajo en diagramas de flujo.

También puede ser de ayuda a la emergencia realizada por personal no sanitario, en condiciones como expediciones de montaña o estancias en otros países.

Otro ámbito aplicable serían las empresas especializadas en desarrollo de software, en las que EMS in Cloud podría ayudar a sus empleados a seguir y compartir los diagramas de flujo.

5.2. Tipos de diagramas

Existe la posibilidad de dar soporte a más tipos de diagramas como diagramas de secuencias. Se podría extender su aplicación a la Ingeniería del software y aprovecharlo también para dar soporte a los casos de uso.

5.3. Aplicación Android para la medicina

Se podrían incluir las siguientes mejoras:

- Para los fármacos, diferenciar entre administración en bolo y perfusión.
- Permitir introducir el peso al inicio de la aplicación y generar las dosis para todos los fármacos
- Permitir cambio de unidades con respecto al denominador de las mismas.

5.4. Posibles mejoras

5.4.1. Cliente iOS

La primera ampliación a este proyecto es claramente el cliente iOS. Algo a tener en cuenta es que esta aplicación es del tipo SaaS y debido a esto las licencias iOS tienen claro la segmentación de clientes que hay en su ecosistema.

En la actualidad, existen cuatro opciones de licencia de desarrollo Apple:

- **Apple Developer**: esta opción es completamente gratuita, y únicamente requiere registrarse como desarrollador desde la propia Web de Apple. Nos da a acceso a:
  - Documentación técnica (versión final del SO y iOS)
  - Herramientas de desarrollo (versiones finales)
  - Solo podemos probar las aplicaciones en el Simulador de XCode.

- **iOS University Program**: esta opción es completamente gratuita, aunque solo aplica a determinadas universidades que estén incluidas dentro del programa. Obliga a que no haya ánimo de lucro, y que todo se destine a la formación de alumnos en las herramientas de Apple. Ofrece así mismo la opción de probar nuestras aplicaciones en nuestros dispositivos.
- **iOS Developer Program:** esta opción tiene un coste de 99 dólares al año, y nos ofrece adicionalmente las siguientes opciones**:**
  - Documentación técnica (de versiones beta)
  - Herramientas de desarrollo (de versiones beta)
  - Descarga de firmwares beta para probar nuestras app.
  - Permite instalar aplicaciones en nuestros dispositivos registrándolos previamente en la Web de Apple.
  - Permite publicar en la AppStore nuestras aplicaciones y comercializarlas con unos beneficios del 70% sobre el precio de venta que establezcamos.
- **iOS Developer Enterprise Program:** esta opción tiene un coste de 299 dólares al año, y está orientada al desarrollo de aplicaciones de tipo In-house (aplicaciones corporativas privadas). Nos ofrece las siguientes opciones**:**
  - Permite publicar de forma privada a la empresa, aplicaciones que puedan ser instaladas en los dispositivos de los diferentes empleados.

Lamentablemente, para publicar nuestra aplicación en la Apple Store se necesita la iOS Developer Enterprise Program, un coste elevado sino se tienen claras las expectativas o un modelo económico en el que no se recupera la inversión.

5.4.2. Ampliar la API del servicio web

Actualmente la API es bastante pequeña y sólo permite acceso a usuarios que puedan identificarse.

Sería interesante poder ofrecer servicios estadísticos, tales como los fármacos más usados, los protocolos más populares, etc.

5.4.3. OAuth2

Una de las principales características de un servicio web es la confianza que ofrece a sus clientes. Este protocolo soluciona el problema de la confianza entre un usuario y aplicaciones de terceros, y a su vez nos permitiría facilitar nuestros servicios a aplicaciones de terceros. De esa manera el usuario que use nuestro servicio revocar o facilitar acceso a sus datos, a quién lo desee y cuándo lo desee, creando así un ecosistema de aplicaciones alrededor nuestro servicio.

Capítulo 6

Manual de usuario

6.1. Requisitos mínimos

El usuario de EMS in Cloud necesitará disponer del siguiente software:

6.1.1. Aplicación web

Para empezar a disfrutar de todos los servicios de EMS in Cloud es necesario que el usuario disponga de un ordenador o dispositivo móvil con conexión a internet para interactuar con la aplicación y generar la sincronización de datos una vez efectuados cambios en la web y verlos reflejados en el resto de dispositivos.

6.1.2. Aplicación móvil

EMS in Cloud presta servicio para terminales que dispongan de una versión de Android 2.3.3 o superiores.

En cuanto a la calificación del contenido, se puede determinar apta para todos los públicos.

Finalmente, los permisos que debe otorgar el usuario a nuestra aplicación son los siguientes:

- acceso completo a la red
- ver conexiones de red

6.2. Instrucciones de instalación

Este apartado servirá de guía para los nuevos usuarios de EMS in Cloud. En él encontraremos dos secciones, correspondientes a las dos partes bien diferenciadas del proyecto. Primero explicaremos lo correspondiente al área de la sección web y a continuación lo correspondiente a la parte móvil.

6.2.1. Sección web

Para poder acceder a EMS in Cloud tan solo es necesario disponer de un navegador web. Ya sea desde un ordenador o desde el navegador de un dispositivo móvil, en este último caso, recomendamos el uso de la propia aplicación móvil cuyas instrucciones de instalación se encuentran en el siguiente apartado.

6.2.2. Sección móvil

Esta instalación es similar a la de cualquier aplicación con la que el usuario esté familiarizado. Necesitaremos disponer del dispositivo móvil con conexión a internet, y que cumpla los requisitos mínimos de software arriba mencionados.

1. Ejecución de Google Play

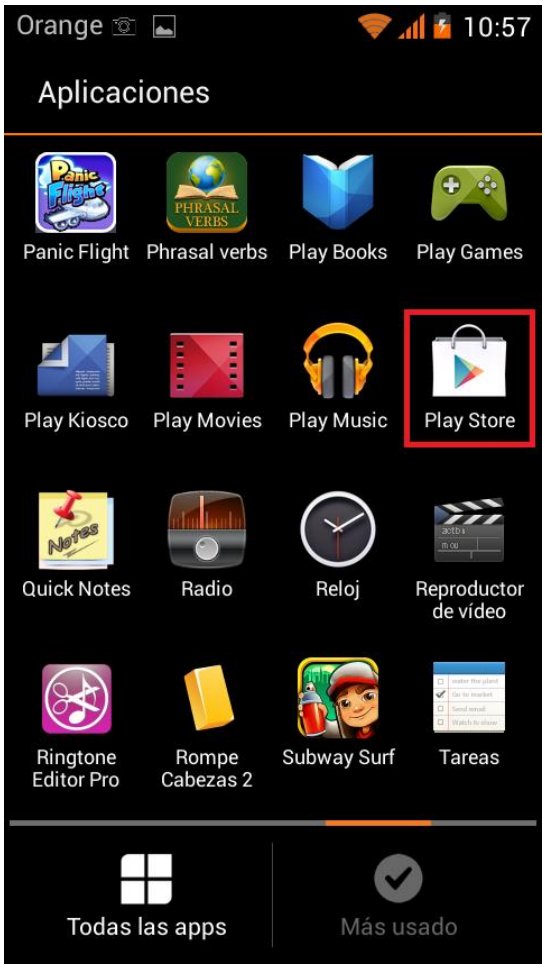


Ilustración 35: Icono de Google Play en el menú principal

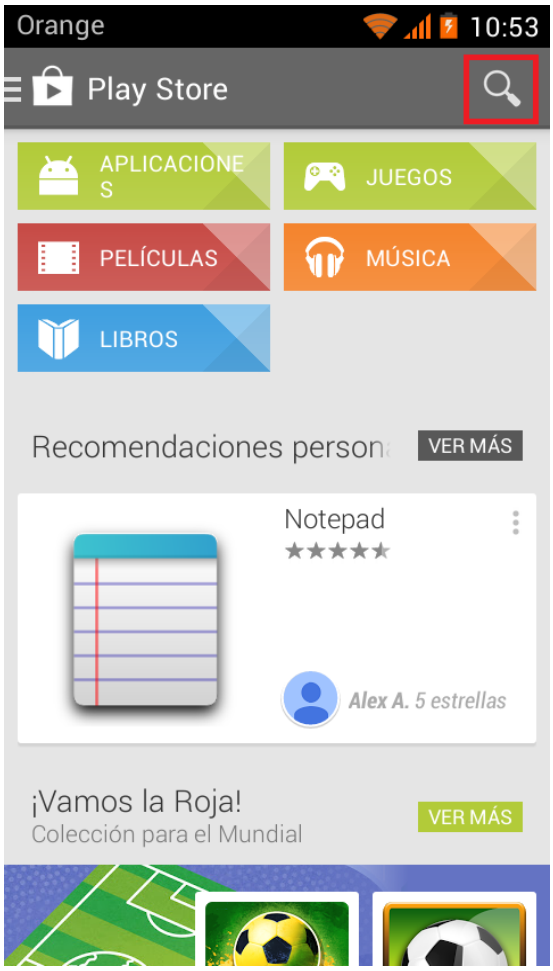
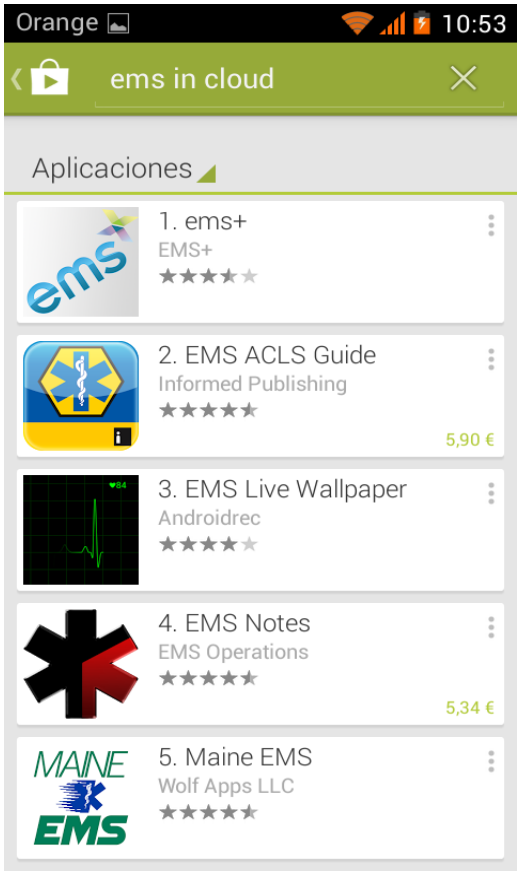


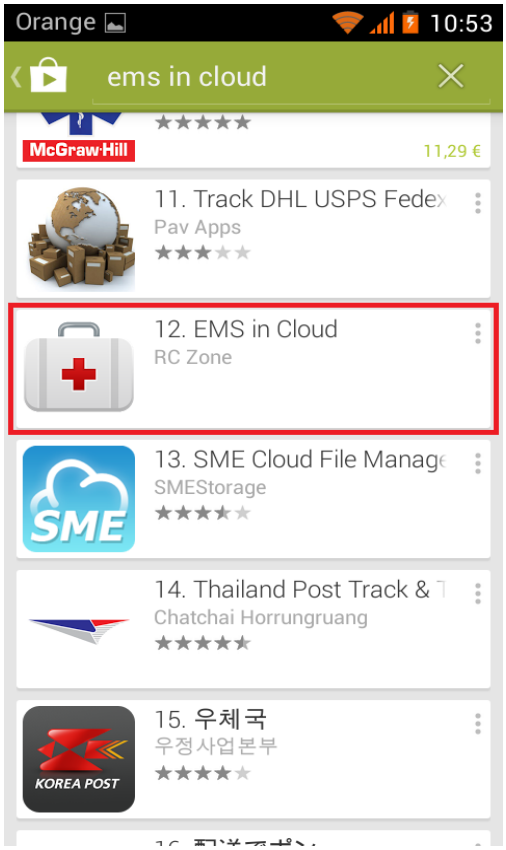
Ilustración 34: Página principal de Google Play

Desde el menú de nuestro terminal pulsar el icono de Play Store, lo que nos llevará a la página principal de Google Play. A continuación pulsaremos el icono de búsqueda para introducir el nombre de nuestra aplicación

2. Búsqueda de la aplicación

Introduciremos el nombre EMS in Cloud en el cuadro de búsqueda y elegiremos nuestra





aplicación de entre todas las de la lista resultante

Ilustración 37: Resultado de la búsqueda EMS in Cloud en Google Play Store I.

Ilustración 36: Resultado de la búsqueda EMS in Cloud en Google Play Store II.

3. Instalación

Pulsaremos el botón *Instalar* y a continuación nos aparecerá una ventana emergente con la información de los permisos que necesita la aplicación para ser ejecutada en nuestro terminal. Una vez aceptemos se iniciará automáticamente la instalación en nuestro terminal.

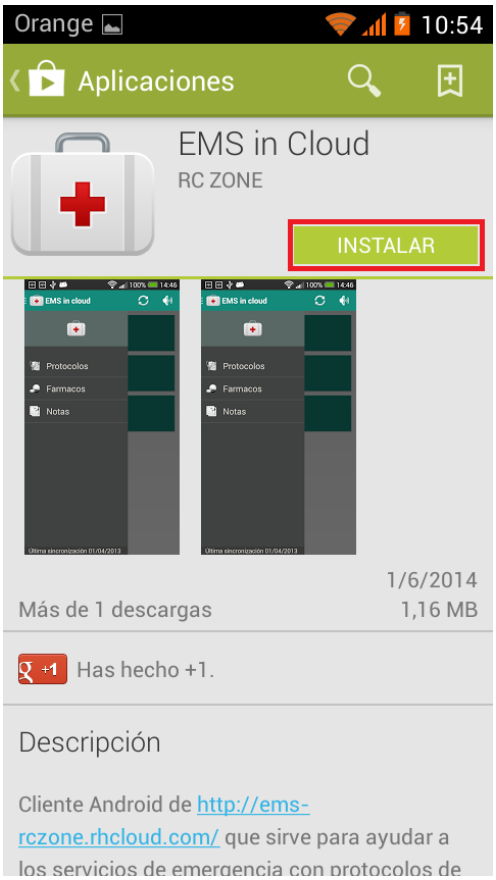


Ilustración 39: Pantalla principal de instalación

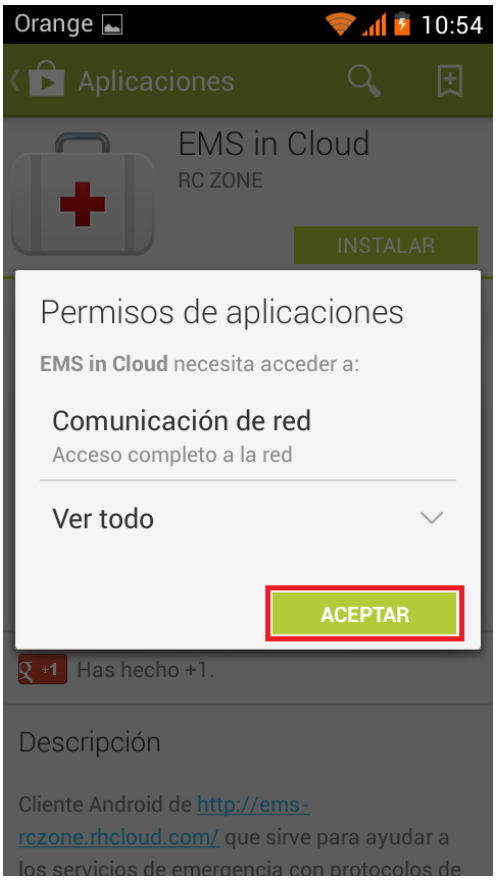


Ilustración 38: Ventana emergente con los permisos que necesita EMS in Cloud

4. Finalización y apertura

Una vez finalizada la instalación, podremos abrir la aplicación desde la página actual o desde el icono que encontraremos en el escritorio y en el menú principal de nuestro dispositivo.

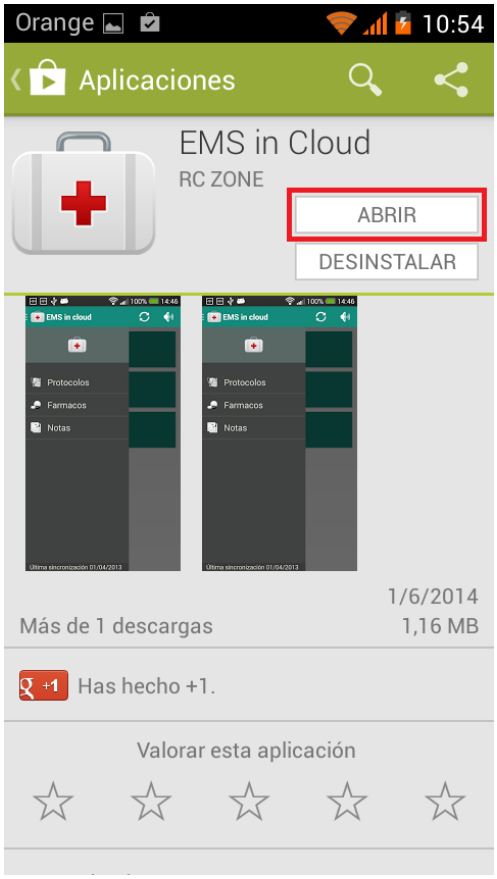


Ilustración 41: Página principal en Google Play después de la instalación.

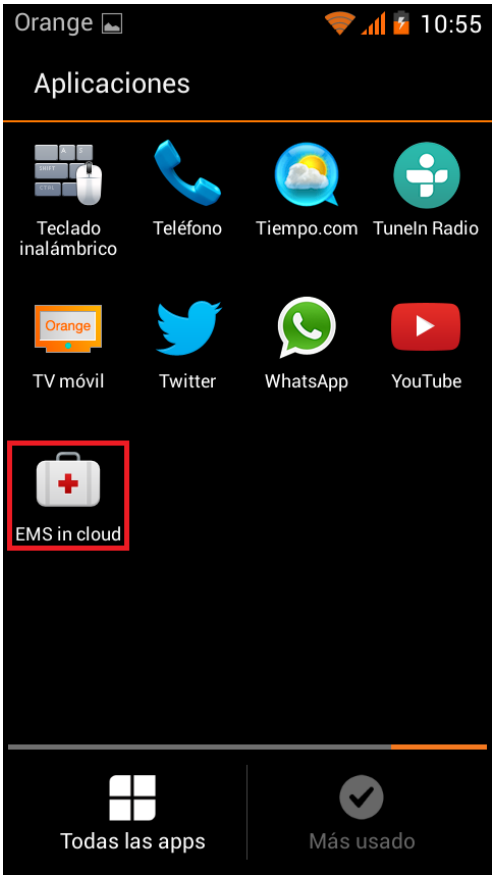


Ilustración 40: Icono de la aplicación en el terminal

6.3. Guía de utilización

En este apartado explicaremos cómo usar EMS in Cloud una vez que ya se encuentra instalado en el equipo del usuario. Seguiremos la estructura del apartado anterior, es decir, que primero nos dedicaremos a la sección web para continuar con la móvil.

6.3.1. Sección web

1. Acceso a la página web.

Para localizar nuestra aplicación tecleamos en la barra del navegador la siguiente URL: *http://ems-rczone.rhcloud.com/*. A continuación, el usuario puede acceder directamente a los servicios de EMS in Cloud.

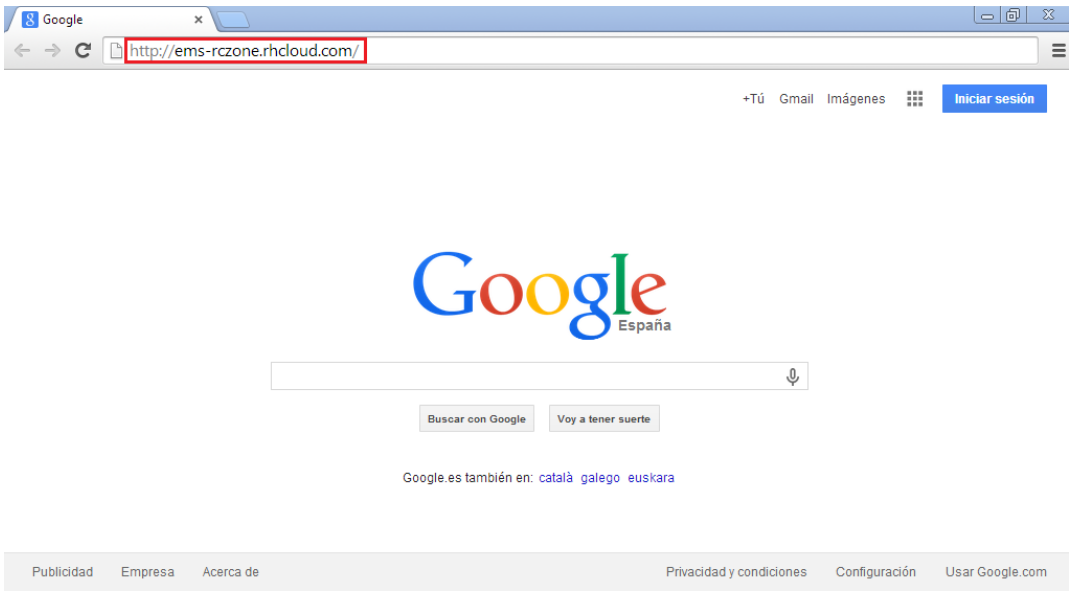


Ilustración 42: búsqueda de EMS in Cloud en el navegador.

2. Acceso al sistema.

El usuario podrá acceder a los servicios si presiona sobre el botón *Iniciar sesión.*

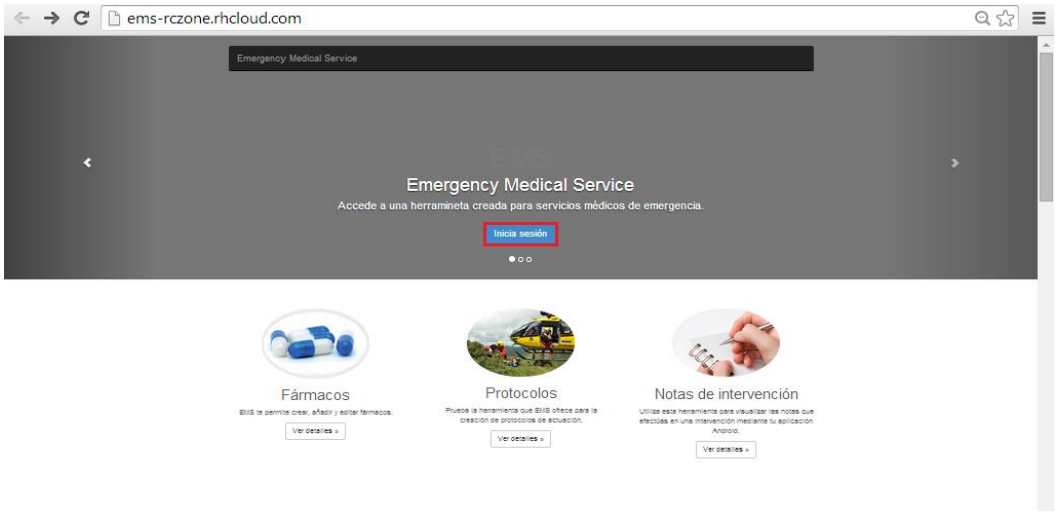


Ilustración 43: página principal de EMS in Cloud.

3. Usuario registrado.

Tras completar debidamente los campos de usuario y contraseña y pulsar el botón *Iniciar sesión*, el usuario accederá al sistema como muestran las dos siguientes ilustraciones:

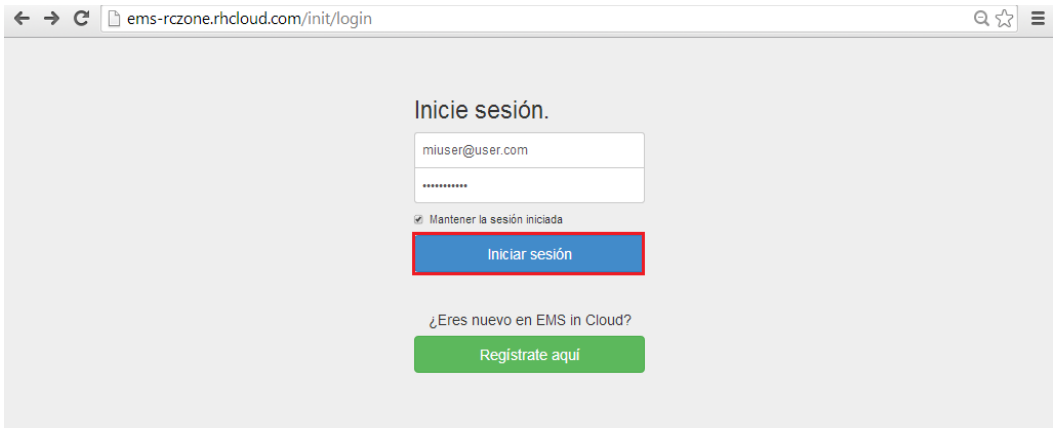


Ilustración 44: Formulario de inicio de sesión.

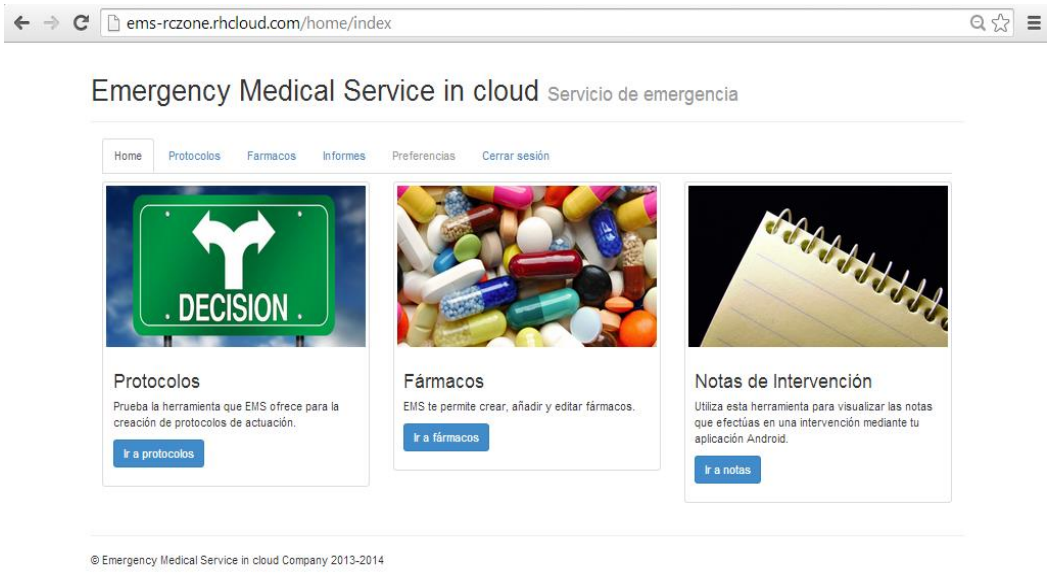


Ilustración 45: Página de inicio de EMS in Cloud.

4. Usuario nuevo.

El usuario debe pulsar en el botón *Regístrate aquí* y posteriormente rellenar un formulario de registro.

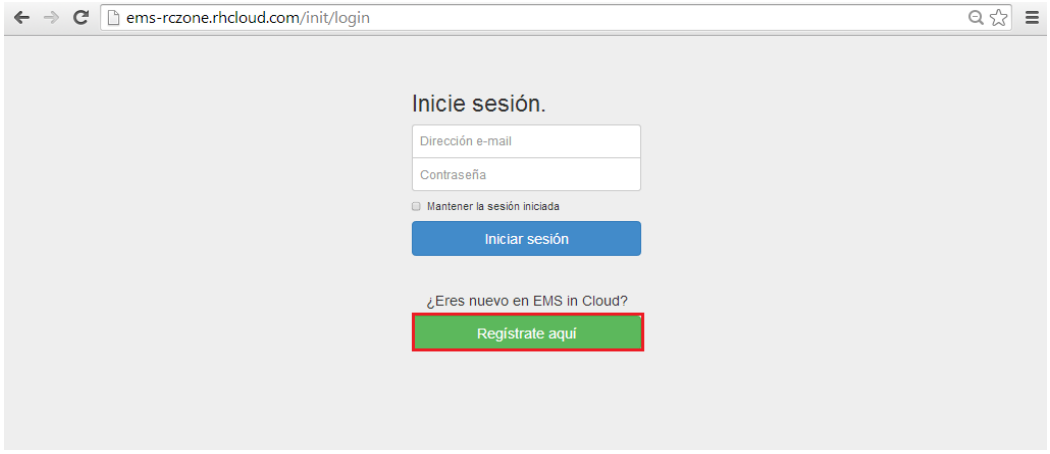


Ilustración 46: Formulario de inicio de sesión.

5. Registro.

El formulario de registro que debe rellenar el usuario es el que se muestra en la siguiente ilustración:

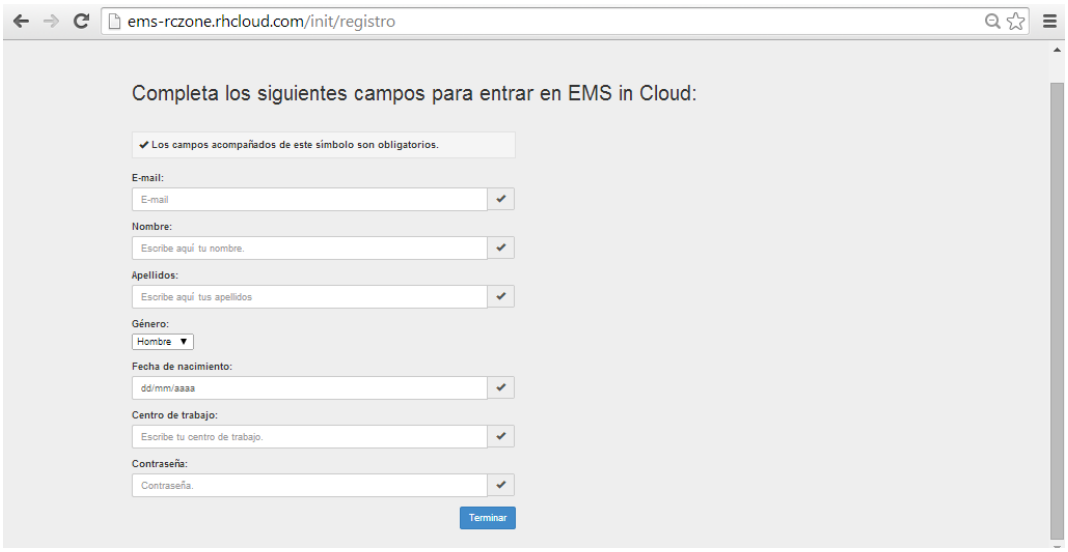


Ilustración 47: Formulario de registro.

El usuario deberá completar los campos: E-mail, Nombre, Apellidos, Género, Fecha de Nacimiento, Centro de Trabajo y Contraseña. Para finalizar el proceso de registro el usuario deberá pulsar en el botón *Terminar* y automáticamente será redirigido a la vista de inicio de sesión que viene representada en la ilustración 44, donde deberá introducir correctamente sus credenciales (Usuario: e-mail y Contraseña) para acceder a la vista principal de la aplicación.

6. Protocolos.

Una de las funcionalidades básicas de EMS in Cloud reside en la creación y edición de Protocolos de actuación.

Existen dos formas de acceder al menú de Protocolos desde la vista principal de la aplicación. Son las que están remarcadas en la ilustración 45.

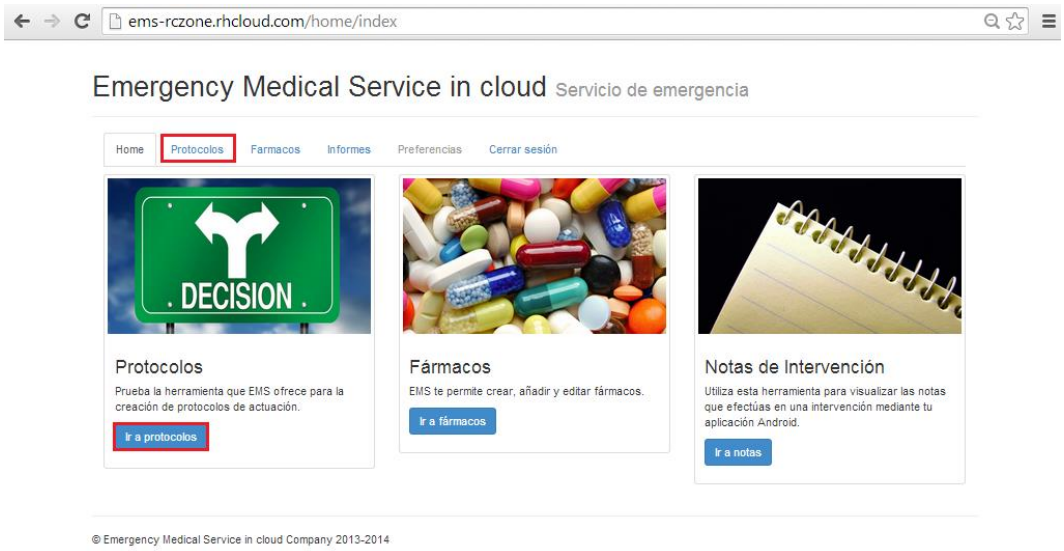


Ilustración 48: Página de inicio de EMS in Cloud

La ilustración 46 muestra el resultado de pulsar el botón anterior.

El usuario accederá a todos los protocolos que haya creado y a través del menú localizado en la parte izquierda podrá:

- Crear protocolos.
- Eliminar protocolos.
- Buscar protocolos.

Si el usuario pulsa en el valor que aparece a la izquierda del nombre del protocolo listado, también conocido como identificador, podrá acceder a la vista de edición de protocolos como se muestra en la ilustración 47.

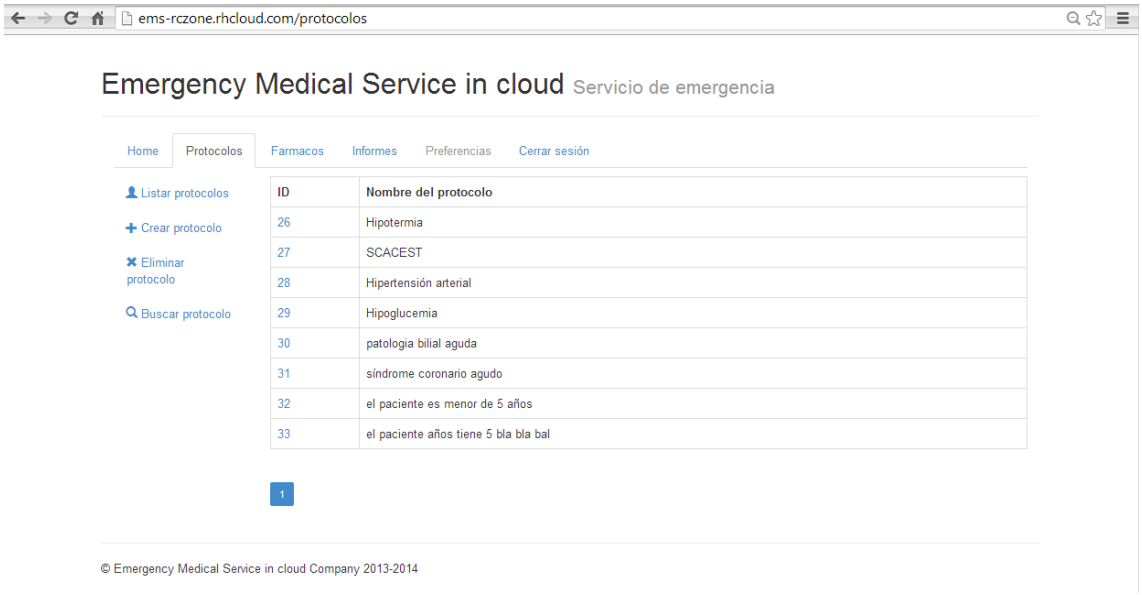


Ilustración 49: página principal de protocolos.

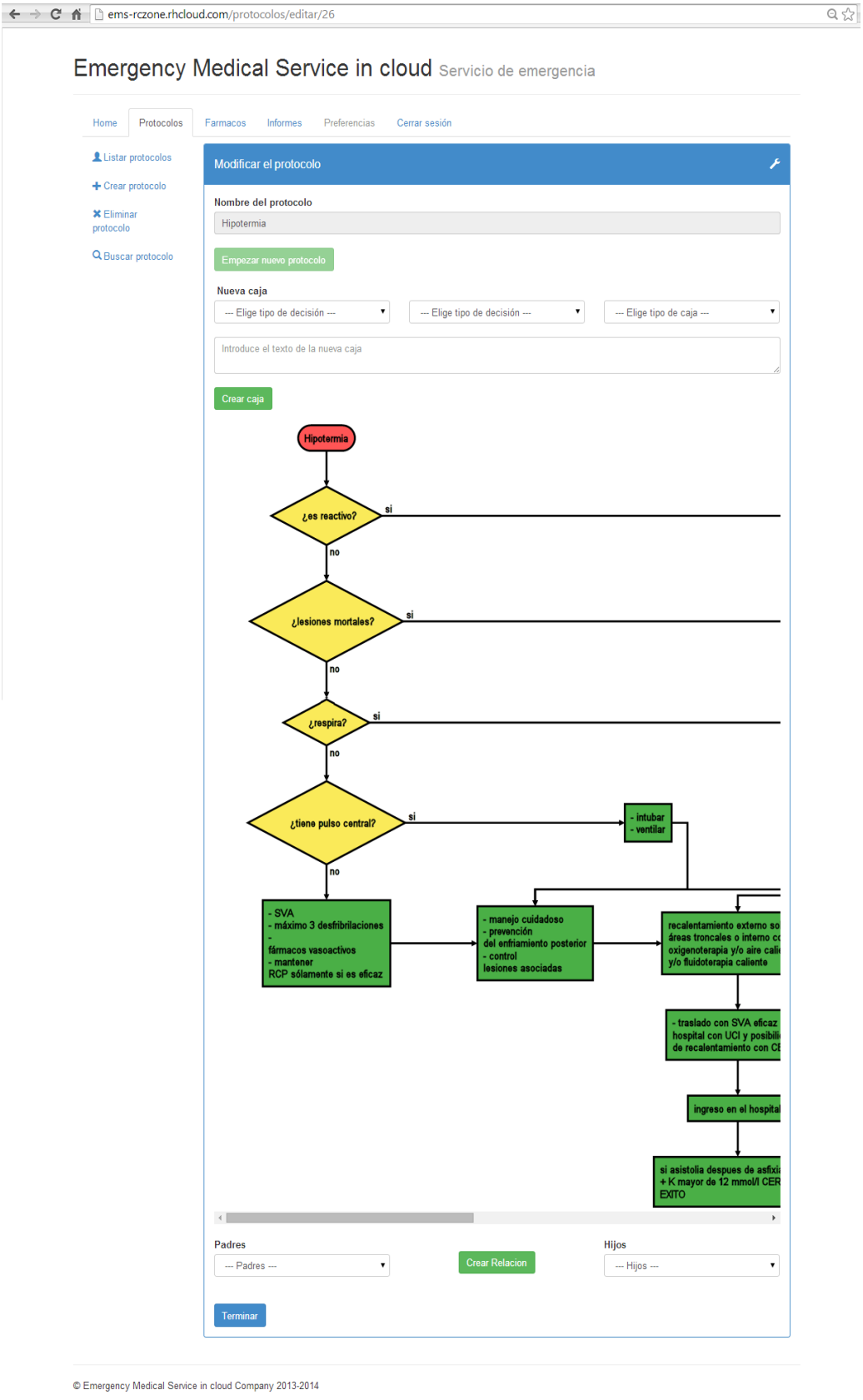


Ilustración 50: edición de un protocolo.

La creación de un protocolo se realizaría de la siguiente manera (ver como referencia la ilustración 48):

Para poder empezar, el usuario deberá rellenar el campo de nombre de protocolo. A continuación deberá pulsar el botón *Empezar nuevo protocolo.* En todo momento tendrá habilitado el botón *Reset* para reiniciar el proceso.

En este proceso de creación el usuario creará *cajas* y tendrá que elegir:

- En caso de que no sea la primera caja, tendrá que definir la procedencia de dicha caja nueva con la funcionalidad *--Es hijo de---*
- El tipo de decisión mediante la funcionalidad *--Elige el tipo de decisión--*:
  - Si/No en caso de que el tipo de la caja sea de decisión
  - Directa si es una caja de tipo normal
- El tipo de la caja a través de la funcionalidad *--Elige tipo de caja--*:
  - Normal: se verá representada como un rectángulo y de color verde.
  - Decisión: se verá representada como un rombo y de color amarillo.

El usuario deberá escribir un texto identificativo de la caja que esté creando y al acabar, para ver el resultado de cada caja deberá pulsar sobre el botón *Crear caja*.

Existe también la funcionalidad *Crear Relación* en la que el usuario puede elegir dicha relación en los campos *--Padres--* e *--Hijos--*.

Para guardar los cambios y acabar, el usuario deberá pulsar en el botón *Terminar*.

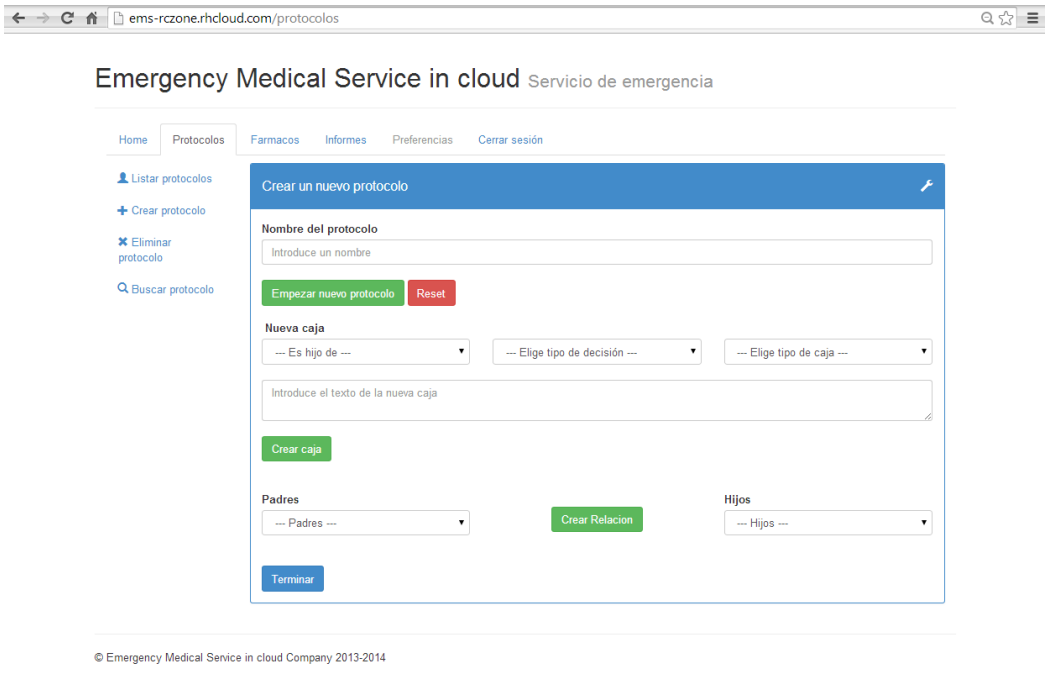


Ilustración 51: creación de un protocolo.

7. Fármacos.

Otra de las funcionalidades que proporciona EMS in Cloud es la creación y edición de fármacos.

Al igual que en protocolos, existen dos formas de acceder al menú de Fármacos desde la vista principal de la aplicación. Son las que están remarcadas en la ilustración 49.

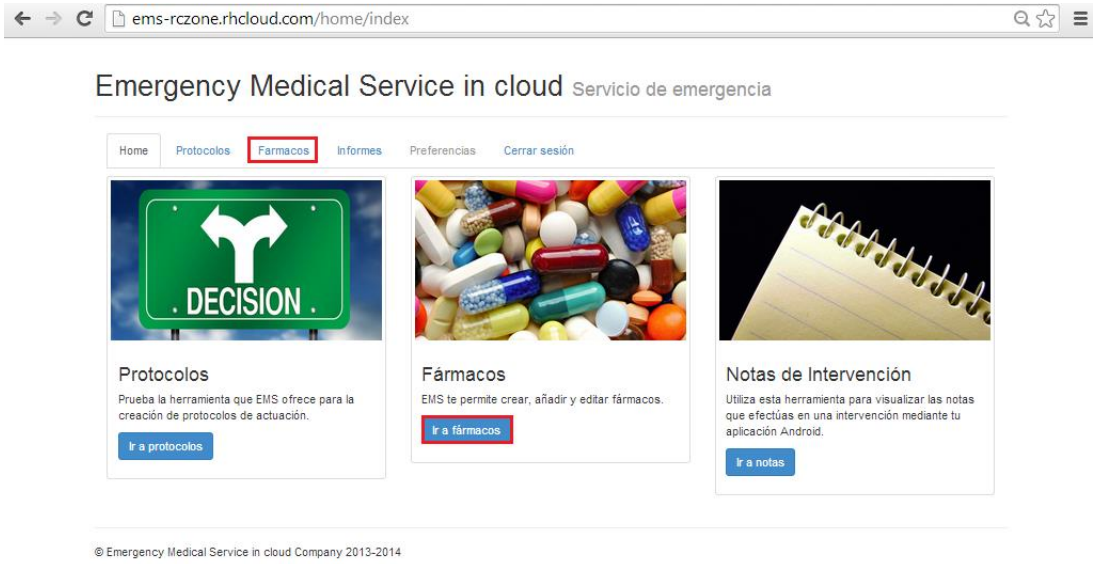


Ilustración 52: Página de inicio de EMS in Cloud

La ilustración 50 muestra el resultado de pulsar el botón anterior.

El usuario accederá a todos los fármacos que tenga asociados y a través del menú localizado en la parte izquierda podrá:

- Añadir fármacos.
- Eliminar fármacos.
- Buscar fármacos.

En caso de tener fármacos asociados, si el usuario pulsa en el icono que aparece en la columna de *Opciones* podrá acceder a la vista de edición de fármacos como se muestra en la ilustración 51.

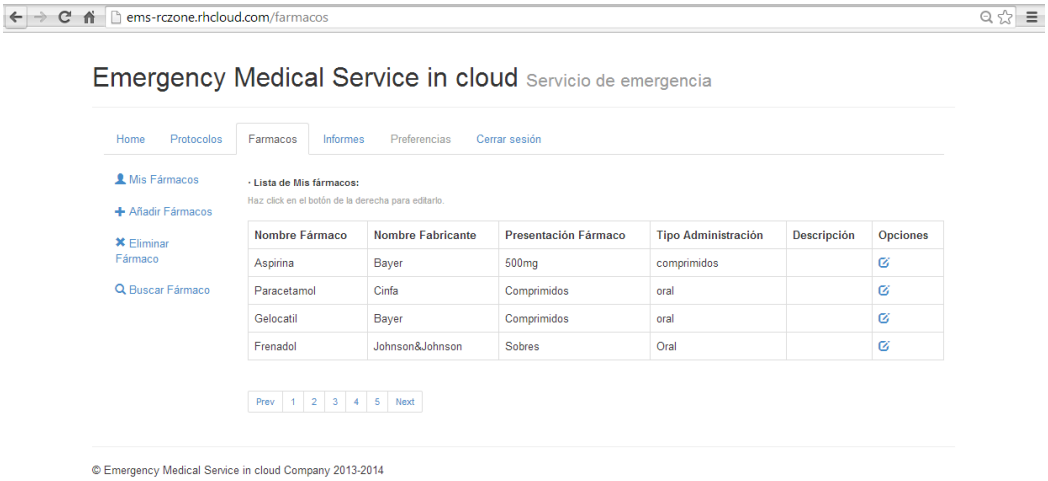


Ilustración 53: página principal de fármacos.

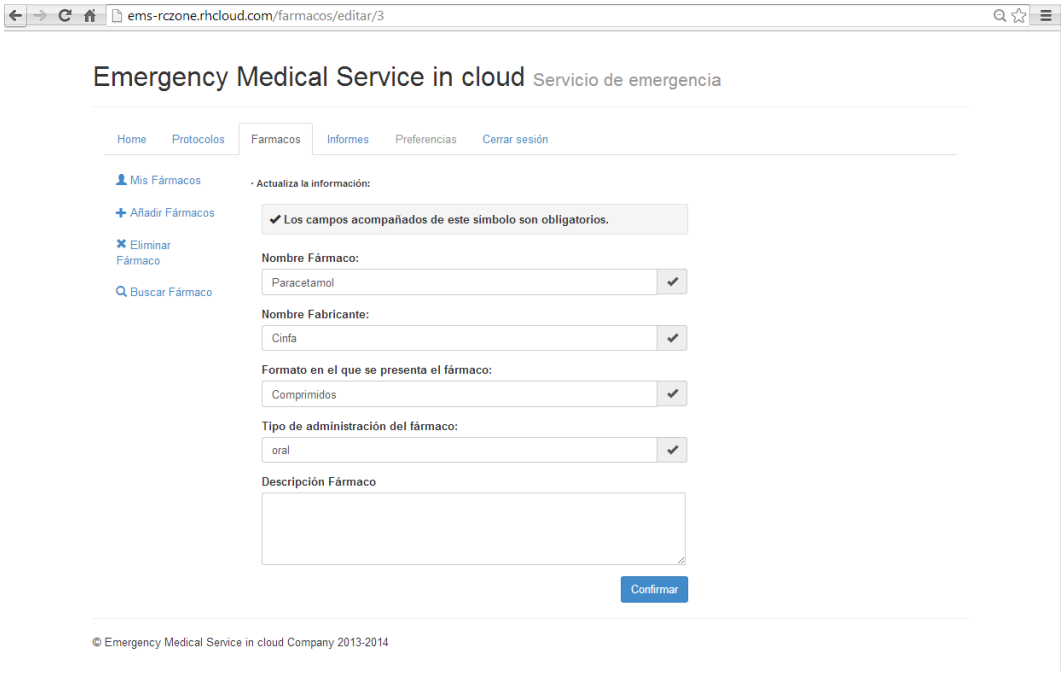


Ilustración 54: edición de un fármaco.

Si el usuario desea añadir un fármaco a su lista de fármacos tendrá dos opciones (ver ilustración 52):

- Buscando en fármacos públicos.
- Creándolo desde cero.

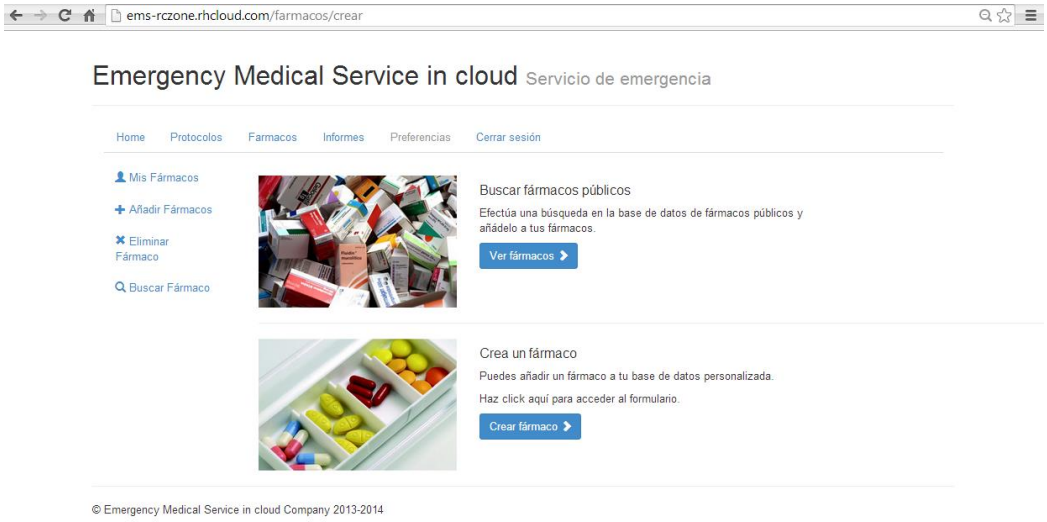


Ilustración 55: Añadiendo un fármaco.

Mediante la opción *Buscar fármacos públicos* el usuario accederá a la vista que se representa en la ilustración 53. En la que si el usuario desea incorporar un fármaco a su colección de fármacos propia, deberá pulsar en el icono correspondiente a dicho fármaco, situado en la columna Opciones. Automáticamente se redirigirá a la vista principal de fármacos, donde se verá actualizada la información.

Ilustración 56: Búsqueda de fármacos públicos.

Por otro lado, si el usuario desea crear un fármaco, deberá rellenar los campos que se muestran en la ilustración 54 y presionar en el botón *Insertar*.

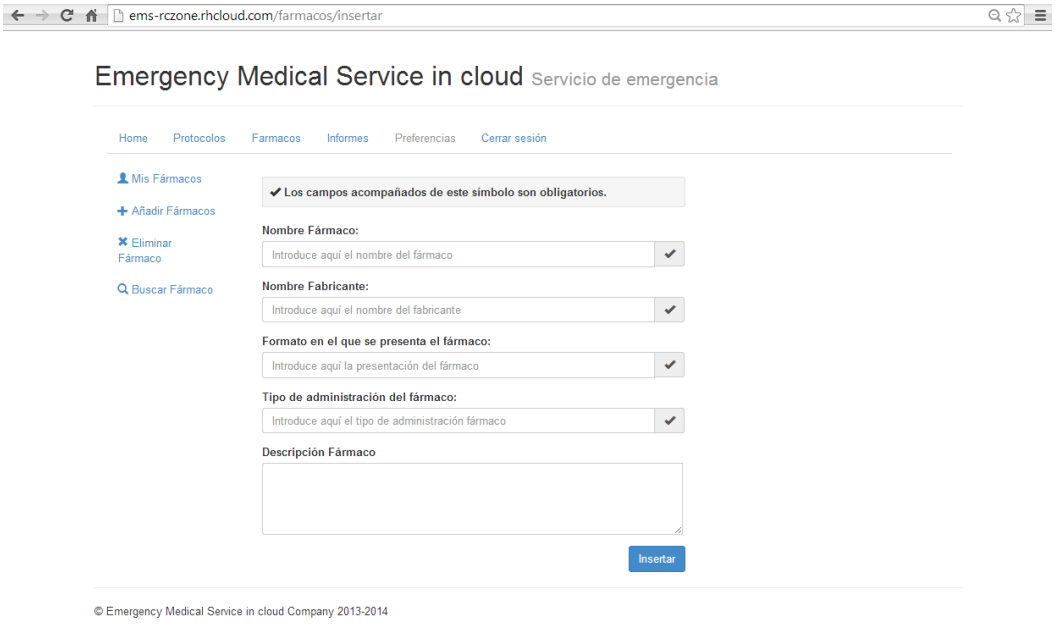


Ilustración 57: creación de un fármaco.

El usuario podrá también acceder al menú y eliminar un fármaco pulsando en la opción *Eliminar Fármaco*. Una vez se ha accedido a esta opción, obtendremos algo similar a lo que se puede apreciar en la ilustración 55. Tan solo es necesario pulsar en el icono de la izquierda para quitarlo de la colección de fármacos. Automáticamente se refrescará la vista y el usuario verá reflejados los cambios.

Ilustración 58: Eliminando un fármaco.

8. Notas.

Otra de las funcionalidades que proporciona EMS in Cloud es la creación y edición de notas asociadas a una intervención o un protocolo de actuación.

Al igual que en protocolos y fármacos, existen dos formas de acceder al menú de Notas (también conocidas como *Informes*) desde la vista principal de la aplicación. Son las que están remarcadas en la ilustración 56.

Ilustración 59: Página de inicio de EMS in Cloud.

La ilustración 56 muestra el resultado de pulsar el botón anterior.

Ilustración 60: página principal de notas de intervención.

6.3.2. Sección móvil

1. Registro

La primera vista del usuario es la pantalla de registro. Desde ella podrá acceder a sus datos personales así como darse de alta en el sistema si es la primera vez que accede a EMS in Cloud.

Si el usuario no forma parte del sistema, se le conducirá al formulario de registro de la web, ya que, como hemos dicho antes, esta parte de la aplicación es dependiente de los datos que el usuario introduzca desde la página.

Ilustración 62: Login de la aplicación Android.

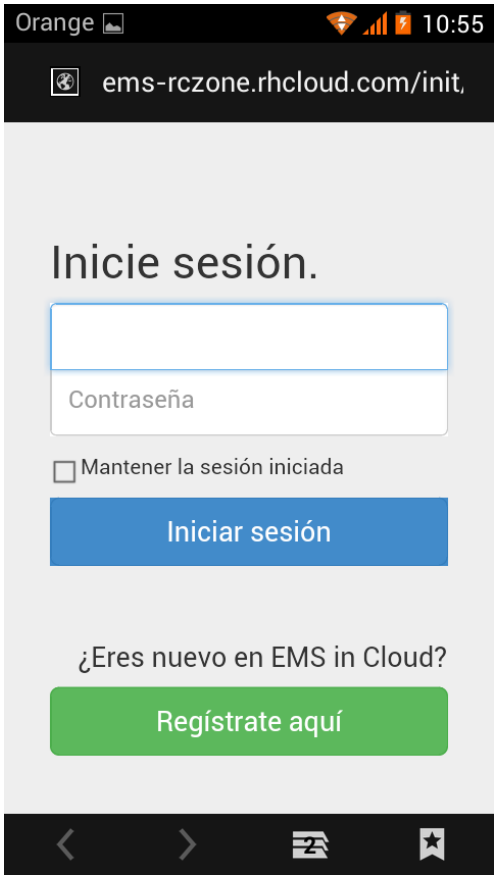


Ilustración 61: Login de la aplicación web

2. Navegación

Una vez completado el registro se mostrará como pantalla principal el listado de protocolos del usuario. Si el usuario añadió nuevos datos a su perfil, y no aparecen en esta vista, puede pulsar el botón *Sincronizar* para que se complete la información. Dicho botón está remarcado en el cuadro rojo en la ilustración 61 que se muestra a continuación y estará disponible tanto para protocolos como para fármacos y notas.

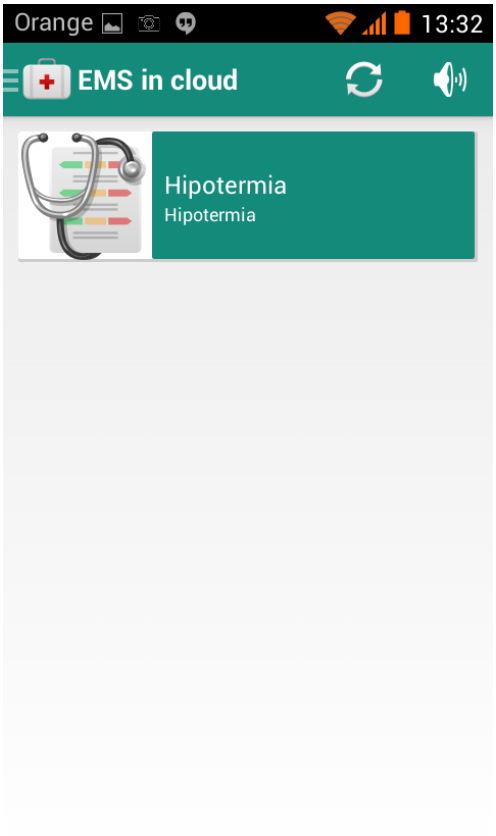


Ilustración 63: Página principal de listado de protocolos.

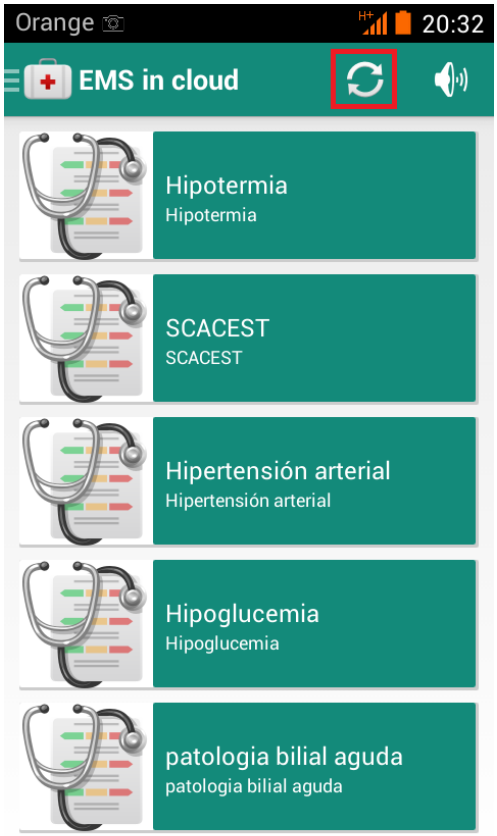


Ilustración 64: Listado de protocolos después de actualizar

Siempre que esté visible la barra principal superior, se podrá acceder al menú para dirigirnos a los protocolos, fármacos o notas pulsando el botón que está a la izquierda, el cual remarcamos en la Ilustración 63. Cuando se pulsa ese botón, se despliega el menú de navegación que permite al usuario elegir la opción que prefiera. También existe la posibilidad de acceder al menú realizando un desplazamiento de izquierda a derecha sobre la pantalla, y ocultarlo realizando el movimiento inverso.

En la vista del menú tenemos las siguientes opciones a las que podremos acceder pulsando sobre ellas:

- Protocolos: vuelve a la vista principal, es decir, al listado de los protocolos.
- Fármacos: accede a la lista de fármacos del usuario.
- Notas: entra en la lista de notas editadas del usuario.

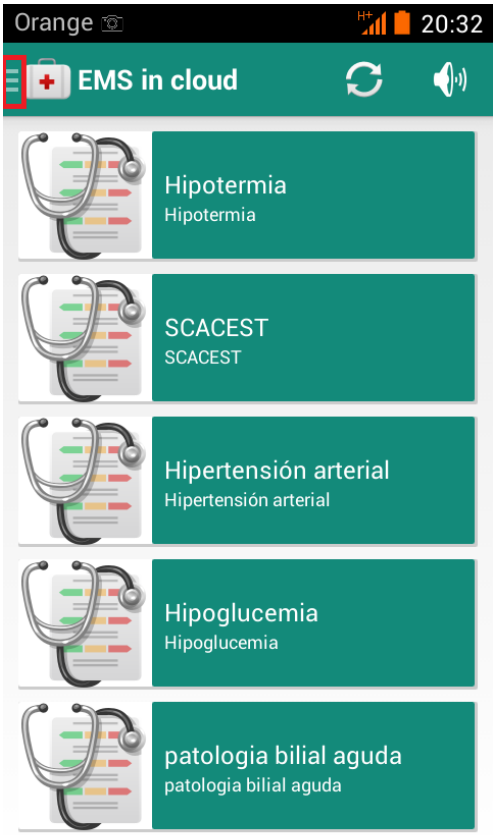


Ilustración 66: Vista del listado de protocolos con el botón del menú destacado.

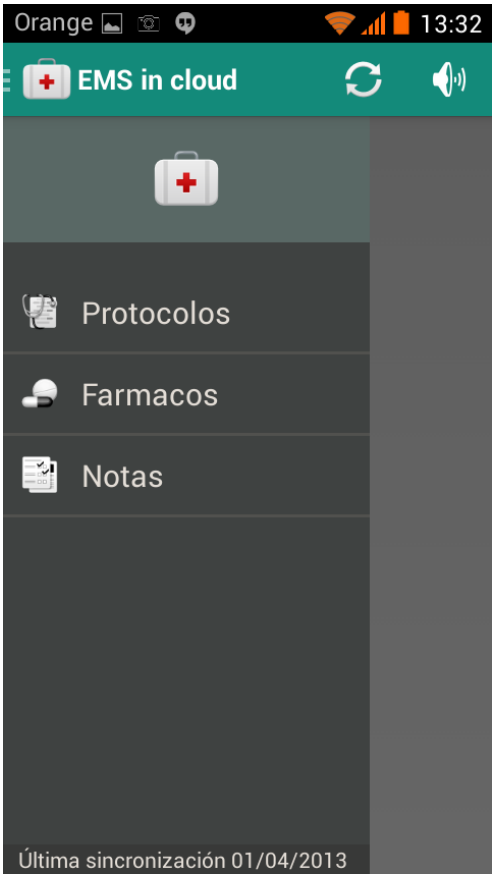


Ilustración 65: Menú de navegación desplegado.

3. Protocolos

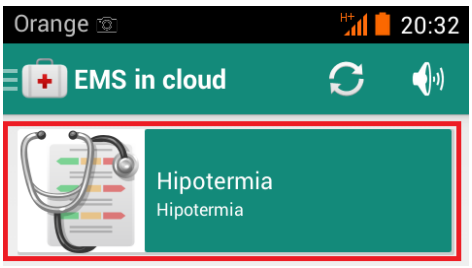


Ilustración 67: Vista del listado de protocolos con la casilla de protocolo destacada.

Desde la lista de protocolos accedemos a uno en particular pulsando sobre su casilla como se muestra en la figura de la izquierda. En este momento la aplicación mostrará la pantalla de exploración del protocolo, exponiendo el contenido de sus cajas de texto.

Todo lo comprendido en cada una de las cajas se escuchará a través del terminal, pudiendo desconectar esta función en cualquier momento pulsando sobre el icono con forma de altavoz de la barra principal. De la misma forma, si el usuario quiere volver a activar la lectura del texto sólo tiene que volver a pulsar el mismo icono.

Como se explicó anteriormente, las cajas de texto pueden conllevar o no la toma de una decisión. Dependiendo de la información contenida en la caja que se esté consultando en cada momento, se tendrán distintas funcionalidades, las cuales vamos a detallar a continuación.

Si la información no se refiere a la toma de una decisión, se mostrará el texto acompañado del botón *Continuar.* En el caso contrario en lugar del botón continuar se encontrarán dos botones *Si* y *No*, que indican la respuesta del usuario a la cuestión que plantea el contenido de la caja. (Ver ilustraciones 65 y 66 respectivamente) Una vez el usuario pulse uno de los botones, se le añadirá a la pantalla el contenido de la caja que sigue a la actual.

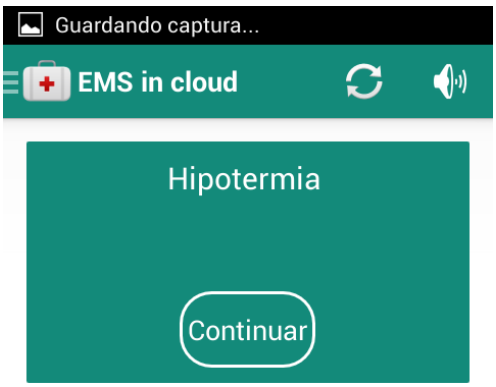


Ilustración 69: Vista de una caja del protocolo.

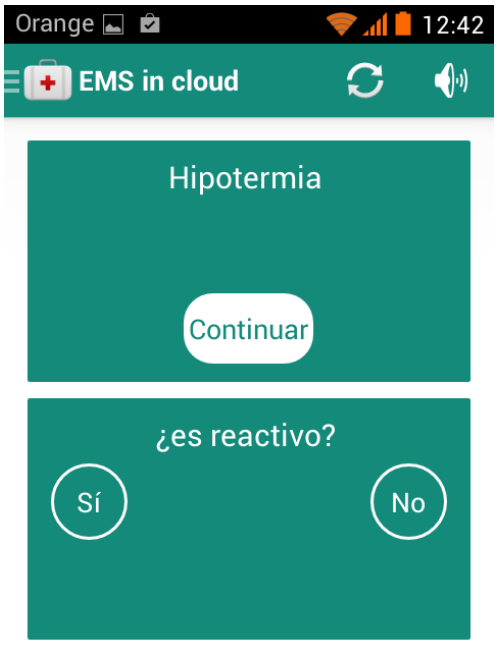


Ilustración 68: Vista de una caja cuyo contenido implica una decisión y se su predecesora.

Si en cualquiera de los casos se encuentra en el contenido de la caja un valor numérico seguido de una unidad métrica reconocida por el sistema, la interfaz resultante permitirá al usuario realizar el cambio de unidad que desee oportuno, retornando a la misma caja con el valor y la unidad actualizados según la elección de cambio de unidad seleccionada previamente.

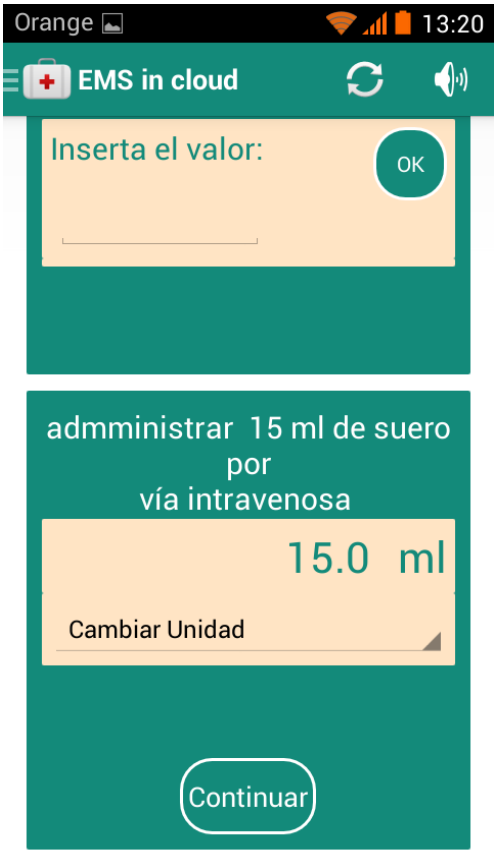


Ilustración 71: caja con unidad métrica en su contenido.

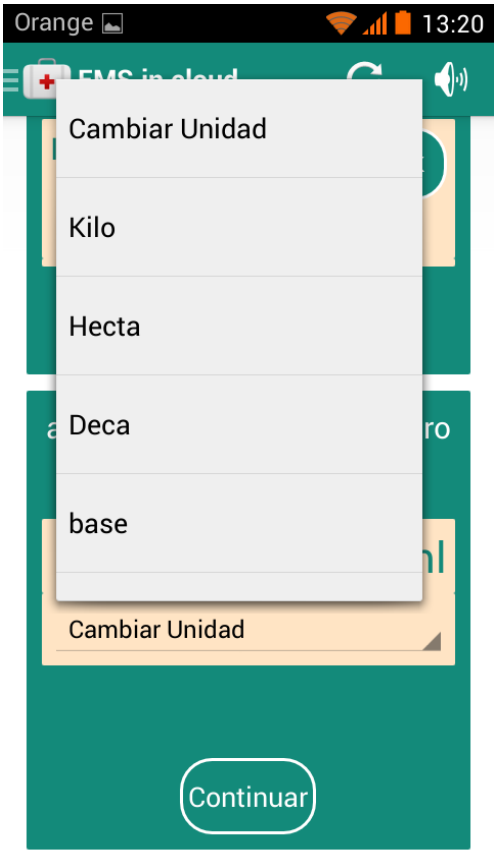


Ilustración 70: caja con el menú de cambio de unidad desplegado.

En las cajas de toma de decisiones cuyo contenido incluye un valor numérico se amplía su funcionalidad. El usuario puede, en lugar de contestar mediante los botones *Si* y *No*, introducir otro valor numérico para que la aplicación lo relacione con el original. El sistema detectará el número introducido en el texto y la relación con el resto del contenido. Por ejemplo, si tenemos una pregunta del tipo *"¿ … mayor que 5 … ?"* y el usuario introduce un 8, será equivalente a haber pulsado el botón *Sí*

Una vez se llega al final del protocolo, se muestra una pantalla con la recopilación de todos los pasos seguidos desde el inicio del protocolo. Se le da la opción al usuario de guardarlo como un archivo de texto en su dispositivo.

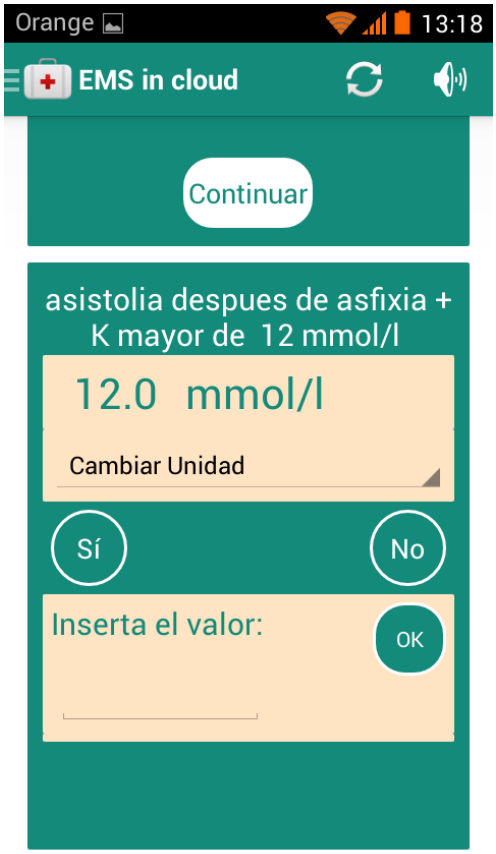


Ilustración 72: caja con toma de decisión, valor numérico y cambio de unidad.

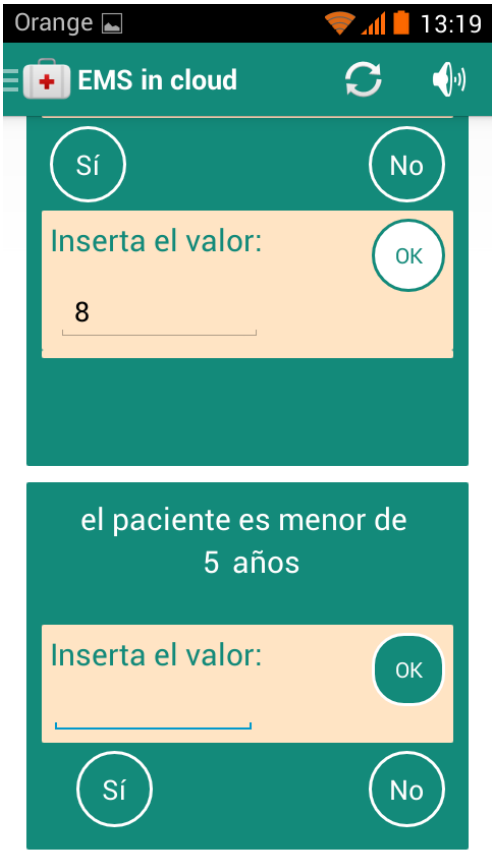


Ilustración 73: caja con toma de decisión, valor numérico.

4. Fármacos

Se accederá a esta vista desde el menú desplegable. En ella se mostrarán todos los medicamentos del usuario pudiendo pulsar sobre cada uno de ellos y ver su lista de detalle.

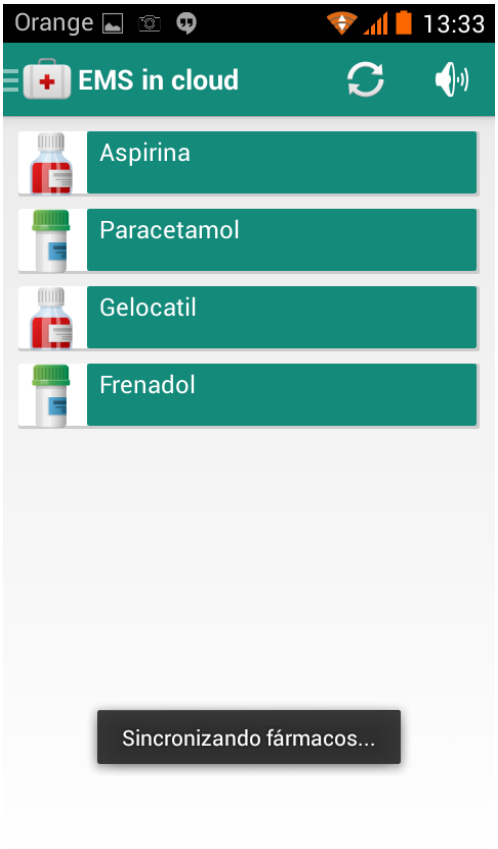


Ilustración 75: Vista de la lista de fármacos después de pulsar el botón Sincronizar.

Ilustración 74: Detalle de uno de los fármacos de la lista.

5. Notas

Es semejante a la vista de fármacos. En ella el usuario podrá consultar sus notas y ver en detalle la que seleccione

Bibliografía

[1] Abrahim Ikasud, “Getting & Managing Your First 1,000 Clients In Web Hosting”, 2009, pp 11-47. [Online]. Available: http://www.amazon.com/Getting-Managing-First-Clients-Hosting/dp/1448656761

[2] George Reese, “Cloud Application Architectures”, Building Applications and Infrastructure in the Cloud, 2009, pp 4-11.

[3] George Reese, “Cloud Application Architectures”, Building Applications and Infrastructure in the Cloud, 2009, pp 3-4.

[4] Thomas Erl, "Cloud Computing for Enterprise Architectures", 2011, pp 5-9.

[5] Scott Chacon, "Pro Git", 2009.

[6] Steve Pousty & Katie J. Miller, "Getting started with Openshift", 2014, pp 5-9.

[7] Janiel J. Barrett, "SSH, The Secure Shell: The Definitive Guide", 2005, pp 1-34

[8] Ben Frain, "Responsive Web Design with HTML5 and CSS3 (Community Experience Distilled)", 2012.

[9] Robert Daigneau, "Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services", 2011, pp 13-79.

[10] Jamie Munro, "20 Recipes for Programming PhoneGap: Cross-Platform Mobile Development for Android and iPhone", 2012, pp 1-29.

[11] Reto Meier, "Professional Android 2 Application Development", 2010, pp 1-30

[12] Chandra Kopparapu, "Load Balancing Servers, Firewalls, and Caches",2002, pp 83-108