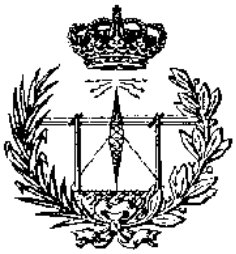

**UNIVERSITAT OBERTA DE CATALUNYA**

**GRADO DE TECNOLOGÍAS DE LA TELECOMUNICACIÓN**

## TRABAJO FINAL DE GRADO

### INGENIERÍA DE SISTEMAS DE TELECOMUNICACIÓN

# APLICACIÓN DE GESTIÓN DE BASES DE DATOS PARA LA COLECCIÓN DE MUESTRAS DE HEMATOPATOLOGÍA

---


AUTOR: Agustín Llamas Ballestero

TUTOR: Aleix López Antón

PROFESOR: Carlos Monzo Sánchez


7 de Junio de 2017

# Ficha del Trabajo Final

| | |
|---|---|
| Título del trabajo | Aplicación de gestión de bases de datos para la Colección de muestras de Hematopatología |
| Nombre del autor | Agustín Llamas Ballestero |
| Nombre del tutor | Aleix López Antón |
| Nombre del profesor | Carlos Monzo Sánchez |
| Fecha de entrega | 06/2017 |
| Área del Trabajo Final | Desarrollo de aplicaciones electrónicas |
| Titulación | Grado de Tecnologías de la Telecomunicación |

Resumen del trabajo (máximo 200 palabras)

El servicio de Hematopatología del Hospital Clínic de Barcelona representado por; la Dra. Marta Aymerich desempeñando las funciones de Consultor Senior dentro de la Unidad de Hematopatología en el Servicio de Anatomía Patológica y cuyo responsable es el Prof. Elías Campo siendo Jefe de Sección de la Unidad de Hematopatología del Servicio de Anatomía Patológica y cuyas responsabilidades son las de Director de Investigación del Hospital Clínic de Barcelona; requieren una aplicación informática muy intuitiva y de fácil manejo para poder gestionar los distintos datos obtenidos provenientes de varias fuentes.

La aplicación deberá ser una interfaz web que gestione una base de datos con la información relativa a los pacientes. Sus funciones básicas serán: crear una entrada para un nuevo paciente, buscar pacientes por diferentes campos, editar campos de un paciente y eliminar un paciente. Las funciones avanzadas estarán relacionadas con la importación de datos de forma automática. Dicha importación se podrá realizar desde diferentes fuentes de datos.

Abstract (in English, 200 words or less)

The Hematopathology service of the Hospital Clínic of Barcelona represented by; Dr. Marta Aymerich performing the duties of Senior Consultant within the Hematopathology Unit in the Pathology Service and whose head is Prof. Elías Campo being Head of Section of the Hematopathology Unit of the Pathology Service and whose responsibilities are The Research Director of Hospital Clínic of Barcelona; Require a very intuitive and easy to use computer application to manage the different data obtained from various sources.

The application should be a web interface that manages a database with patient information. Its basic functions will be: to create an entrance for a new patient, to search for patients by different fields, to edit fields of a patient and to eliminate a patient. Advanced features will be related to importing data automatically. This import can be made from different data sources.

Palabras clave (entre 4 y 8)

Hospital, Pacientes, Aplicación, Base de Datos.

# Resumen


## *Síntesis*

El servicio de Hematopatología del Hospital Clínic de Barcelona representado por; la Dra. Marta Aymerich desempeñando las funciones de Consultor Senior dentro de la Unidad de Hematopatología en el Servicio de Anatomía Patológica y cuyo responsable es el Prof. Elías Campo siendo Jefe de Sección de la Unidad de Hematopatología del Servicio de Anatomía Patológica y cuyas responsabilidades son las de Director de Investigación del Hospital Clínic de Barcelona; requieren una aplicación informática muy intuitiva y de fácil manejo para poder gestionar los distintos datos obtenidos provenientes de varias fuentes.

La aplicación deberá ser una interfaz web que gestione una base de datos con la información relativa a los pacientes. Sus funciones básicas serán: crear una entrada para un nuevo paciente, buscar pacientes por diferentes campos, editar campos de un paciente y eliminar un paciente. Las funciones avanzadas estarán relacionadas con la importación de datos de forma automática. Dicha importación se podrá realizar desde diferentes fuentes de datos.

## *Objetivos principales*

Diseñar una aplicación web para el desarrollo de un sistema integral de gestión de bases de datos para la Colección de Hematopatología, de fácil implantación, rápida y cómoda para el usuario; con escaso coste de mantenimiento.

## *Beneficios*

Facilitar la codificación y el almacenamiento de las muestras y la información asociada.

Permitir relacionar tablas de diferentes procedencias y formatos con información de muestras y pacientes.

Proporcionar un motor de búsqueda que sea fácilmente configurable.

Desarrollar un sistema de registro de intercambio de muestras con los investigadores.

Ofrecer las máximas garantías de seguridad exigibles por la LOPD para la protección de datos personales de los pacientes.

## Entregables

Los diferentes elementos entregables de este Trabajo Final de Grado son: memoria, presentación y una máquina virtual que contenga tanto la aplicación desarrollada como la base de datos.

La memoria recogerá todos los datos de la evolución, desarrollo e implementación de la aplicación.

La presentación mostrará la labor realizada en el desarrollo del Trabajo Final de Grado para tener una idea global del funcionamiento de la aplicación.

La máquina virtual tendrá la aplicación y la base de datos con el objeto de poder exportar la arquitectura de forma fácil y sencilla a cualquier servidor.

## Motivación

Conocidas las necesidades en la gestión de datos de forma automatizada de la Unidad de Hematopatología del Servicio de Anatomía Patológica del Hospital Clínic de Barcelona, se decide realizar una aplicación para automatizar ciertas tareas y permitir una gestión de los datos relativos a un paciente de forma más sencilla para intentar conseguir que los profesionales sanitarios tengan que dedicar menos tiempo al procesamiento de la información y así de esta manera puedan dedicar sus esfuerzos a sus tareas de investigación.

# AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi tutor Aleix todo el tiempo dedicado y todas las correcciones realizadas, así como su constante apoyo sobre todo en los momentos de mayor carga de trabajo.

Me gustaría agradecer todo su apoyo a mi familia, muy especialmente a mis padres, Manuel y María Ángeles; y a mis tres hermanos: Isaac, Manuel y Ángel, gracias a todos ellos he podido llegar hasta aquí.

Un agradecimiento muy especial para mi novia Silvia que me ha ayudado más de lo que estaba a su alcance, principalmente en los malos momentos. Su constante apoyo ha conseguido que pueda salvar cada uno de los retos que han ido surgiendo.

Quisiera hacer una mención especial y agradecer enormemente su apoyo a mis abuelos: Agustín e Isidora, Ángel y Cándida; de quienes aprendí el espíritu de sacrificio necesario para seguir adelante en la vida, incluso en los peores momentos.

No puedo olvidarme de todos mis tíos, tías, primos y primas; quienes siempre han estado presentes y dispuestos a prestarme su ayuda, sobre todo cuando he tenido alguna necesidad.

También quisiera agradecer su gran apoyo y su generosidad a mis amigos que siempre se han mantenido cerca en la distancia. Todos ellos me han ayudado enormemente para llegar hasta esta meta.

# ÍNDICE

# ÍNDICE DE FIGURAS

# Glosario de acrónimos

AJAX: *Asynchronous JavaScript And XML*.

API: *Application Programming Interface*.

APS: Atención Primaria de la Salud.

ASP: *Active Server Pages*.

BIRT: *Business Intelligence and Reporting Tools*.

BSD: *Berkeley Source Distribution*.

CHITS: *Community Health Information Tracking System*.

CIDRZ: *Center for Infectious Disease Research in Zambia*.

CPRS: *Computerized Patient Record System*.

CPT: *The Current Procedural Terminology*.

CPU: *Central Processing Unit*.

DHCP: *Decentralized Hospital Computer Program*.

DCOM: *Distributed Component Object Model*.

EHR: *Electronic Health Record*.

EMR: *Electronic Medical Record*.

EME: Expediente Médico Electrónico.

ETL: *Extraction Transformation and Loading*.

FFEHR: *Free Feathers Electronic Health Record*.

FMF: *FreeMedForms*.

GNU: GNU No es Unix.

GPL: *General Public License*.

GUI: *Graphical User Interface*.

HL7: *Health Level 7*.

HMA: *Health Message Archiver*.

HMS: *Health Message Server*.

HOSxP: *Hospital experience*.

HQL: *Hibernate Query Language*.

HTML: *HyperText Markup Language*.

HTTP: *Hyper-Text Transfer Protocol*.

ICD: *International Statistical Classification of Diseases and Related Health Problems*.

ICD-10-PCS: *International Statistical Classification of Diseases and Related Health Problems Procedure Coding System.*

IDRC: *International Development Research Center.*

IPD: *In Patient Department*.

Java EE: *Java Enterprise Edition*.

JPA: *Java Persistence API.*

LAMP: Linux, Apache, MySQL, Perl, PHP o Python.

LAN: *Local Area Network.*

LDAP: *Lightweight Directory Access Protocol.*

LGPL: *Lesser General Public License*.

LOINC: *Logical Observation Identifiers Names and Codes*.

Mac OS: *Macintosh Operative System.*

MDO: *Medical Domain Objects*.

MIME: *Multipurpouse Internet Mail Extensions*.

MIT: *Massachusetts Institute of Technology*.

MS SQL: *Microsoft SQL Server.*

MSDE: *Microsoft SQL Server Desktop Environment*.

MSXML: *Microsoft XML Core Services*.

MUMPS: *Massachusetts General Hospital Utility Multi-Programming System.*

NCBI: *National Center for Biotechnology Information.*

OMS: Organización Mundial de la Salud.

OPD: *Out Patient Department*.

OSCAR: *Open Source Clinical Application and Resource.*

PDF: *Portable Document Format.*

PC: *Personal Computer.*

PHP: *Preprocessed Hypertext Pages*.

PIH: *Partners In Health.*

POO: Programación Orientada a Objetos.

RAM: *Random Access Memory*.

REMITT: *Record Electronic Medical Information Translation and Transmission.*

RPC: *Remote Procedure Call*.

SAX: *Simple API for XML*.

SGML: *Standard Generalized Markup Language*.

SNOMED: *Systematized Nomenclature of Medicine*.

SQL: *Structured Query Language*.

SSL: *Secure Socket Layer*.

SOAP: *Subjective, Objective, Assessment, Plan diagnostics, Plan therapeutics*.

TAR: Terapia Anti-Retroviral.

TCP/IP: *Transmission Control Protocol/Internet Protocol.*

TORCH: *Trusted Open source Records for Care and Health.*

TRAX: *TRansformation API for XML.*

UDDI: *Universal Description, Discovery and Integration.*

VHA: *Veterans Health Administration.*

VHE: *HealtheVet*.

VIH: Virus de la Inmunodeficiencia Humana.

VistA: *Veterans Health Information Systems and Technology Architecture*.

W3C: *World Wide Web Consortium*.

WAN: *Wide Area Network.*

WSDL: *Web Services Description Language*.

WWW: *World Wide Web*.

XML: *eXtensible Markup Language*.

XML-RPC: *eXtensible Markup Language Remote Procedure Call*.

XMTP: *XML MIME Transformation Protocol.*

XSLT: *EXtensible Stylesheet Language Transformations*.

ZEPRS: *Zambia Electronic Perinatal Record System*.

# 1. Introducción

Las nuevas tecnologías son de gran relevancia para el desarrollo de todo tipo de aplicaciones y en el caso de las destinadas a la medicina aportan un valor extraordinario, ya que no solo permiten una gestión más rápida de distintos trámites, sino que además hacen que los profesionales de la medicina puedan dedicar más tiempo a tratar enfermedades y no a la gestión de la información. En distintos departamentos sanitarios, donde los recursos económicos pueden escasear, el presupuesto destinado a sanidad en muchos casos no es suficiente y por ello una adecuada gestión de los recursos monetarios podrá salvar un gran número de vidas (Fraser *et al.*, 2006; Noor *et al.*, 2004; Stansfield, 2005; Tomasi *et al.*, 2004). Del binomio carencia económica versus epidemias surge la idea e implementación de los *Electronic Medical Record* (EMR) ya que solventan los problemas relacionados con la gestión de información de una forma rápida y eficaz a la par que su presupuesto de instalación no es demasiado elevado (Diero *et al.*, 2006; Mamlin & Biondich, 2005). Entre sus ventajas está el ahorro de trabajo y rapidez en el acceso a la información relativa a un enfermo. Frente a los inconvenientes que supone la instalación de un sistema de gestión de historiales médicos con software propietario (Biondich *et al.*, 2003; Fraser *et al.*, 2005), con los sistemas de software libre se evita el pago de licencias de software que encarecen la instalación, pero en ningún caso reduce las prestaciones ofrecidas (Fraser *et al.*, 2004; Mamlin & Biondich, 2005).

El software de código abierto da un soporte eficaz a la investigación que se desarrolla en estos países subdesarrollado permitiendo la evaluación de nuevas modalidades de diagnóstico, medidas para la prevención de enfermedades, pandemias, estudios epidemiológicos y análisis estadístico de la salud pública por regiones (Häyrinen *et al.*, 2008; Murray *et al.*, 2004; Siika *et al.*, 2005).

En este proyecto final de carrera se revisan un conjunto de aplicaciones que permiten una adecuada gestión de EHR, cuya característica común es que son de código abierto. Se hará un breve análisis de aquellos software libre más destacados en el mercado por sus prestaciones (cantidad de pacientes gestionados,

número de lugares en los que se utiliza y heterogeneidad de las clínicas, mayor nivel de desarrollo del software, funcionalidades ofrecidas) o por sus características técnicas (tecnología del núcleo, lenguaje de programación en el que se desarrolla la interfaz de la aplicación, lenguaje de interacción con la base de datos).

A partir del estudio minucioso de las aplicaciones más importantes de software libre sobre la gestión de EHR y teniendo en cuenta las necesidades del servicio de Hematopatología del Hospital Clínic de Barcelona, se realiza un diseño de la aplicación necesaria junto con el plan tecnológico asociado que permita llevar a cabo el software con la mejor eficiencia posible.

La memoria del proyecto se organiza de la siguiente manera:

En la sección 2 se revisan un conjunto de aplicaciones que permiten la gestión de EHR, cuya característica común es que son de código abierto. Se realizará un breve análisis de las principales aplicaciones de software libre existentes en el mercado teniendo en cuenta los servicios que ofrecen (número de pacientes gestionados, número de lugares donde se utilizan y la heterogeneidad de las clínicas, nivel de desarrollo de la aplicación, funcionalidades ofrecidas) o sus características técnicas (tecnología del núcleo, lenguaje de programación en el que se implementa la interfaz de usuario, lenguaje de interacción con la base de datos).

En la sección 3 y 4; se describirán las secciones de diseño e implementación de un programa para la gestión de la información relativa a los pacientes organizada en una base de datos. Se destaca la necesidad de realizar tanto un diseño como una implementación para que se puedan añadir fácilmente diferentes módulos que permitan tanto funcionalidades generales como específicas. La aplicación desarrollada se basará en diferentes en diferentes características analizadas en las aplicaciones de software libre analizadas en la sección 2.

En la sección 5 se tratarán las conclusiones del Trabajo Final de Grado y las posibles futuras mejoras a la aplicación desarrollada, bien sean tecnológicas, funcionales o líneas futuras.

# 2. Análisis de los programas más importantes de código abierto

## *2.1 CHITS*

CHITS (*Community Health Information Tracking System*) es un prestigioso software de código libre utilizado para la gestión de historiales médicos. Ha sido diseñado principalmente para los centros de salud locales en las islas Filipinas. Su desarrollo se ha llevado a cabo por el Dr. Herman Tolentino (Unidad de Informática Médica de Filipinas) y gestionado en el Centro Nacional de Telesalud situado en Manila.

Esta aplicación genera informes semanales y mensuales en un formato estándar. Es capaz de almacenar la información del paciente en formato digital y guardarla para realizar una recuperación rápida de dicha información.

Al ser un software de código abierto, CHITS ha sido capaz de abrir varias oportunidades para el desarrollo de los sistemas de salud en el país de la siguiente manera:

- Ha hecho posible que las instituciones gubernamentales capaciten a sus trabajadores del campo de la medicina en los registros electrónicos de salud a bajo costo; gracias a evitar el pago por licencias de software, lo que ha permitido que los trabajadores tengan la oportunidad de aprender los principios de la gestión de información de salud sin temor a romper los acuerdos de propiedad creados por licencias de software.
- Ha preparado el escenario para el desarrollo de aplicaciones basadas en estándares que pueden operar sintáctica y semánticamente entre ellos. Tal norma de desarrollo está orientada a fomentar una mayor confianza entre los desarrolladores para innovar sin miedo y así fomentar la creación de software aislado que por medio de los estándares serán fácilmente integrables en CHITS.

- Ha dado la posibilidad de recoger datos rutinariamente y de aprovecharlos para estadísticas matemáticas con el fin de abrir un área interesante de estudio sobre la predicción de los brotes de ciertas enfermedades y garantizar el suministro de medicamentos suficientes. Tanto la epidemiología y la gestión de los recursos se beneficiarán de esta aplicación.

Por último, cabe destacar que como la mayoría de las poblaciones humanas en todo el mundo residen en zonas rurales que necesitan sistemas de gestión de historiales médicos eficaces, los sistemas de información eficientes son fundamentales para garantizar la prestación de servicios de salud de calidad a estas zonas remotas a pesar de su aparente aislamiento de la civilización.

## Datos técnicos

### Requisitos mínimos

Servidor: "*Personal Computer*" (PC) compatible con IBM, 512 MB de memoria RAM (*Random Access Memory*), 40GB de disco duro, tarjeta de red Ethernet y sistema operativo Linux.

Estaciones de trabajo: PC compatible con IBM, 128 MB de memoria RAM, 20GB de disco duro, Ethernet o Wi-Fi y navegador Web.

Hub de red de 8 puertos / switch o router *wireless*.

### Instalación del servidor

Lo primero será instalar el sistema operativo en el servidor. Se recomienda Linux (Debian, Mandrake, o Fedora) aunque también se puede utilizar Windows. Después de la instalación del sistema operativo, también se deberá instalar y configurar un determinado software en el servidor para que la aplicación CHITS funcione correctamente. Si está utilizando Linux como sistema operativo puede usar los gestores de paquetes de la distribución para instalar todos los componentes necesarios.

**Instalación de una estación de trabajo**

Instalar el sistema operativo elegido para el puesto de trabajo. Windows o Linux (Debian, Mandrake, Fedora) pueden ser utilizados.

CHITS es un sistema basado en tecnología Web, por lo que una vez que el software del servidor se ha instalado, en las estaciones de trabajo sólo se requiere un navegador Web. Se recomienda el uso de Mozilla Firefox ya que ha sido la plataforma en la que se han realizado un mayor número de pruebas con la interfaz de usuario.

**Instalación de la red**

CHITS puede ser configurado para funcionar con conexión a través de un cable o configuración inalámbrica. Para utilizar una conexión inalámbrica, se puede utilizar un router inalámbrico en lugar del switch de red o un hub. Después se deberá configurar el enrutador inalámbrico o el hub de red para proporcionar direcciones IP privadas (chits.ph/web, 2017).

## *2.2 ClearHealth*

ClearHealth es una aplicación de código abierto cuyos registros médicos electrónicos son sistemas bajo la GNU (GNU No es Unix), que se ha perfilado como una posible opción de código abierto para la utilización de historiales médicos electrónicos. En la actualidad está desplegado en aproximadamente 600 lugares en todo el mundo incluyendo las versiones con apoyo comercial, aunque se puede decir que hay un gran número de instalaciones realizadas en diferentes hospitales sin ánimo de lucro.

La historia de ClearHealth tiene sus inicios a partir de los desarrolladores principales de varios sistemas de software de código abierto utilizados para temas relacionados con la medicina incluyendo las aplicaciones OpenEMR y FreeMed.

Fred Trotter y David Uhlman estuvieron muy involucrados en la creación y la revisión del sistema de facturación FreeB que fue el primer programa de código abierto que aplicó un conjunto de normas estándar para la facturación médica

electrónica. Tener un sistema en los Estados Unidos (EE.UU.) de facturación médica disponible bajo una licencia GPL (*General Public License*), dio un impulso para muchos otros sistemas de código abierto como FreeMed, OpenEMR, ClearHealth y MirrorMed. ClearHealth a partir de su versión 2.0 añade la implementación para el manejo de los registros médicos electrónicos y una capacidad para el manejo de las bases de datos mediante SQL (*Structured Query Language*).

Este programa está escrito en el lenguaje PHP y es posible ejecutarlo en la mayoría de las configuraciones de los servidores ya sean; Windows, Linux o Mac OS X (*Macintosh Operative System*). ClearHealth al igual que Apache y MySQL cumple con las normas de la mayoría de sistemas basados en tecnología Web de código abierto

Entre las distintas soluciones de código abierto para la industria de la salud de la California HealthCare Foundation, la aplicación ClearHealth se puede identificar como una solución viable basada en la idea de utilizar código abierto para la creación de distinto software relativo al campo de la medicina.

ClearHealth es un sistema totalmente integral que abarca las siguientes características:

- Programación:
  - Programación completa de todos los eventos destacados.
  - Reglas de nombramiento.
  - Buscar primero.
  - Calendario optimizado.
  - Programas complejos.
  - Habitaciones y recursos.
  - Multi-Dispositivo de apoyo.

- Registro:
  - Inscripción de pacientes destacados.
  - Automatizado de elegibilidad.
  - Programas de participación.

- Remisión de seguimiento.
- Estadísticas.
- Detección de duplicados.

- EHR:
  - Alertas de estado de salud.
  - Pacientes.
  - Encuentros.
  - Alergias.
  - Historia social.
  - Lista de problemas.
  - Autogestión de objetivos.
  - Historial médico.
  - Resumen clínico.
  - Los resultados del laboratorio en tiempo real.
  - Laboratorio de electrónica.
  - Fotos de pacientes.
  - Generación de código de barras.
  - Soporte a la decisión.
  - *Subjective, Objective, Assessment, Plan diagnostics, Plan therapeutics* (SOAP).
  - Imágenes.
  - Tarjetas de identificación.
  - Portal del paciente.
  - Formularios PDF (*Portable Document Format*).
  - Informes PDF.
  - Etiquetas.
  - Configuraciones especiales.
  - Listas de trabajo.
  - El intercambio de datos.

- Movilidad:
  - Dispositivos móviles.
  - iPhone.
  - Blackberry.

- Facturación:
  - Electrónica.
  - Presentación electrónica.
  - Estado en tiempo real.
  - Reclamaciones en papel.
  - Especialidad formas PDF.
  - Cuentas por cobrar.
  - Presentación de informes.

- Presentación de informes:
  - Informes definibles por el usuario.
  - Plantillas PDF.
  - Excel.
  - Exportación de datos.
  - Soporte SQL.

- Módulos de Especialidad:
  - Ginecología y obstetricia.
  - Quiropráctica.
  - Urología.
  - Oncología.
  - Salud en el hogar.
  - Salud mental.

- Integración:
  - HL7 (*Health Level 7*).

- Servicios Web.
- XML (*eXtensible Markup Language*).

- Tecnología:
  - Basado en la Web.
  - Facilidad de despliegue.
  - Acceso remoto.
  - Configuraciones ASP (*Active Server Pages*) (en.wikipedia.org/wiki/ClearHealth, 2017).

## 2.3 elementalClinic

La aplicación elementalClinic está basada en tecnología Web para la gestión de pacientes de salud mental. Este programa realiza las siguientes tareas: programación, evaluación, toma nota de los progresos, los objetivos, el tratamiento de grupo, y la facturación médica electrónica.

Este software es de código abierto tiene como objetivo ayudar a las clínicas de salud mental a tener un EHR que le permitirá beneficiarse de la multitud de ventajas que presenta la gestión de los informes médicos de forma electrónica.

**Características**

- Presentación de forma electrónica de reclamaciones.
- Completamente personalizable a las necesidades de una clínica determinada.
- Seguimiento de la medicación con receta.
- Códigos de facturación personalizable.
- Programación.
- Plantillas para formularios y notas de progreso de los pacientes.
- Facilidades para la clonación de notas de progreso.
- Configuración de las evaluaciones.
- Terapia de grupo.

- Presentación de informes de hospitalización.
- Ver la evolución del paciente con los objetivos y planes de tratamiento.
- Historia del diagnóstico completo.
- Nivel de atención de un paciente determinado.
- Seguridad en el intercambio de datos.
- Soporte administrativo.

**Elementos personalizables**

- Observar de forma completa los informes.
- Crear y modificar la facturación y los códigos de diagnóstico.
- Cambiar toda la información que se encuentra en las secciones desplegables.
- Establecer las funciones y los permisos para el acceso y edición de información.

**Seguridad**

- Alojamiento disponible en los servidores seguros de la compañía que son alimentados por energía eléctrica obtenida mediante métodos no contaminantes.
- Instalación personalizada.
- Soporte rápido y fácil (a través de Internet, teléfono o en persona).
- Inicio de sesión segura.
- Dirección web única para su clínica.

**Servicios**

Aplicado con éxito, el software de los registros médicos electrónicos mejora la calidad de la atención sanitaria. Sin embargo, los costes iniciales y la lenta aplicación del software propietario se convierten en los principales problemas para la difusión de los registros médicos electrónicos. La seguridad, el desarrollo de la comunidad que implementa aplicaciones para registros médicos

electrónicos y la concesión de licencias libres de EHR son las principales características de elementalClinic.

**Ética**

Al ofrecer elementalClinic como software libre, se espera hacer del mundo un lugar mejor.

**Viabilidad**

Al hacer elementalClinic de código abierto, se anima a la contribución de gente de cualquier lugar del mundo interesada en colaborar en el proyecto.

**Datos técnicos**

La aplicación elementalClinic está realizada en lenguaje de programación Perl utilizando la base de datos PostgreSQL que tiene un excelente rendimiento para dicho lenguaje de programación. Todo ello bajo la filosofía de código abierto (directory.fsf.org/project/elemental_clinc, 2017).

## 2.4 FreeMedForms

FreeMedForms (FMF) es un software multi-plataforma (disponible en MacOS, Linux, FreeBSD, Windows), multilingüe, multiusuario, de código abierto ya que ha sido liberado bajo la licencia del nuevo BSD (*Berkeley Source Distribution*) y programado básicamente en C++. Esta aplicación está en un estado de desarrollo activo llevado a cabo por médicos y se destina principalmente a los profesionales de la salud. Actualmente, el software está en desarrollo, por lo que sólo está disponible para propósitos de prueba. El objetivo principal de FreeMedForms es la realización de una eficiente gestión de los EHR tanto en hospitales como para los grupos de investigación médica por medio de un conjunto de archivos XML. Sus registros serán totalmente personalizables mediante el uso de *plugins*. La interoperabilidad y la internacionalización son los objetivos que pretende conseguir la aplicación.

Por otro lado, algunas partes del software están ya en funcionamiento y son utilizables en la práctica, como por ejemplo; la aplicación FreeDiams (antes DrugsInteractions). Este *plugin* sirve para la gestión de recetas de la aplicación principal FreeMedForms y está integrado en un programa independiente. Además, es multi-plataforma (MacOS, Linux, FreeBSD, Windows) y de código abierto liberado bajo la solicitud de nueva licencia de BSD. También, igual que la aplicación principal, es desarrollado por médicos y se destina para el uso de éstos mismos profesionales. Su única utilización es para prescribir medicamentos. FreeDiams puede estar vinculado a cualquier software, gracias a sus parámetros de línea de comandos.

La aplicación FreeMedForms se destina a ser utilizado:

- En la práctica médica general.
- En las clínicas y hospitales.
- En los grupos de investigación médica.

El programa puede ser rápidamente mejorado gracias a la fácil instalación de los numerosos complementos presentados en forma de *plugins*. El código está disponible para cualquier persona interesada.

La idea principal del proyecto es que cualquier médico pueda crear sus propios "Formularios" para sus pacientes sin ninguna configuración tediosa.

Las principales características de FreeMedForms se describen a continuación:

- Diseñado por los médicos.
- Multiplataforma: misma aplicación para Linux, Mac y Windows.
- Fácil instalación guiada.
- Código abierto.
- Interfaz multilingüe (Francés, Inglés, Alemán) con la posibilidad de cambio dinámico de idioma sin perder información.
- Creación automática de la base de datos.
- Presentación de estadísticas para cada elemento (en desarrollo).

**Funcionalidades implementadas en FreeMedForms**

Esta aplicación está siendo desarrollada y la versión disponible permite realizar las siguientes opciones:

- Abrir y añadir registros de pacientes.
- Probar las posibilidades que ofrecen la inserción de registros.
- Probar las diferentes opciones gráficas.
- Probar la opción que permite el cambio dinámico de lenguaje.

Las siguientes opciones están siendo probadas para su implementación:

- Interpretación de la línea de comandos (que permite la ubicación de los recursos utilizados por FreeMedForms).
- Comprobación automática de actualizaciones disponibles a través de Internet.
- Impresión de los formularios en blanco.
- Copia de seguridad de los datos.

**Principales objetivos de FreeMedForms**

Este software multi-plataforma, libre y de código abierto está destinado a profesionales del campo de la medicina. Actualmente está en fase de desarrollo y sólo está disponible para las pruebas. Al finalizar el proyecto, FreeMedForms presentará una versión completa que permitirá una correcta gestión de historiales médicos.

El objetivo principal de este proyecto es proveer una herramienta de software libre, multilingüe y multi-usuario para los profesionales de la medicina. La aplicación tiene una interfaz intuitiva y muy modular. Sus características son totalmente personalizables por medio de los nombrados anteriormente *plugins*. El programa permite la gestión de los informes médicos.

**Instalación de FreeMedForms**

En cada sistema operativo la instalación de la aplicación se ha intentado hacer lo más fácil posible. Los pasos generales son: descargarlo, descomprimirlo e instalarlo. FreeMedForms tiene un asistente de instalación que le ayudará en todas las fases de configuración (freemedforms.com/en/start, 2017).

## *2.5 FreeMED*

FreeMED es una aplicación para la gestión de informes médicos electrónicos de código abierto, basado en Linux, Apache, MySQL y PHP.

**Historia**

El proyecto para el desarrollo de este EHR se inició oficialmente en 1999 por Jeffrey Buchbinder en Estados Unidos. Desde entonces, se ha convertido en un esfuerzo internacional, con miles de descargas y varias traducciones.

Esta aplicación es un descendiente directo del software AMOS que era un programa implementado en Pascal y fue creado en 1983 antes del uso generalizado de las bases de datos relacionales y de la programación orientada a objetos. En estos momentos el programa se aproxima a su novena revisión. Al abordar esta revisión se reconoce que ha habido cambios en la medicina y en la prestación de asistencia médica y por ello en esta actualización se pretende adaptar la aplicación a las nuevas necesidades requeridas.

FreeMED es uno de los más antiguos y sofisticados EHR de código abierto de todos los productos disponibles en este ámbito. Este EHR está siendo aplicado con éxito en un gran número de clínicas en todo el mundo, incluyendo los hospitales públicos y clínicas privadas tanto grandes como pequeñas.

**Implementación y datos técnicos**

FreeMED está principalmente escrito en PHP haciendo uso intensivo de SQL, lo que favorece el motor de base de datos MySQL. También utiliza *bash*, Perl y pequeños módulos escritos en otros lenguajes. Su interfaz está

principalmente basado en la Web, pero las interfaces de servicios Web, tales como XML-RPC (*eXtensible Markup Language Remote Procedure Call*), también están disponibles.

Este software ha sido diseñado para el sistema operativo Linux. Si se desea utilizar para cualquier otro sistema operativo como Windows o Mac OS X, entonces se deberá utilizar una máquina virtual y allí proceder a la instalación de la aplicación FreeMED para *VMware*. La instalación directa del software en el sistema operativo Windows no es estable por lo que no se recomienda.

Una vez realizada la instalación, al acceder a la aplicación se puede ver la página principal del sistema que contiene alguna información de utilidad. En la esquina inferior derecha de la pantalla se tiene el nombre del usuario (administrador). Hay cuatro campos generales en el menú principal. En el centro de la barra de encabezado se pueden observar los siguientes apartados: Sistema, principal, el usuario y los pacientes. Junto a las pestañas anteriormente descritas está el botón de Ayuda. El menú Ayuda relacionado con el contexto proporciona información sobre la página en uso. Cada una de estas pestañas contiene elementos de submenú con distintas opciones.

**Módulos EHR**

La aplicación FreeMED representa los datos médicos como un grupo de "módulos", que consiste en un modelo de base de datos y las interfaces de usuario. Cada uno de los distintos componentes está virtualmente conectado entre sí por medio de los campos relacionales de la tabla de referencia a otros módulos y bases de datos del paciente. Esto permite a la aplicación principal la posibilidad de añadir y eliminar funcionalidades en la base de datos central instalando o quitando módulos sin tener que reprogramar su interfaz.

**Sistema de Facturación Externa (REMITT)**

FreeMED utiliza un programa de facturación externo llamado REMITT (*Record Electronic Medical Information Translation and Transmission*). Se comunica con REMITT XML a través de una conexión de RPC (*Remote*

*Procedure Call*) autenticados. Esta conexión, una vez establecida, permite la transmisión de los datos de facturación médica como un bloque monolítico de XML. Esta información se transforma en un meta-modelo a través de XSLT (*EXtensible Stylesheet Language Transformations*) y finalmente es cambiada a su formato final y se transmite a su destino final. Esta metodología permite que múltiples formatos de salida se generen a partir de la misma base de datos (freemedsoftware.org, 2017).

## 2.6 GNUmed

GNUmed es un programa de código abierto diseñado principalmente para médicos y que puede ser instalado en sistemas tipo Unix (BSD, Linux, y UNIX), Microsoft Windows, Mac OS X y otras plataformas. La comunidad GNUmed está desarrollando un paquete de software médico que será de código abierto, libre, seguro, respetuoso con la intimidad del paciente, basado en estándares abiertos, flexible y totalmente equipado. Para su uso en red (arquitectura cliente-servidor) sus características serán: fácil de usar, multiplataforma y multilingüe.

La aplicación que estamos describiendo se basa en herramientas de código abierto como PostgreSQL y está escrito principalmente en Python. Se apoya en una interfaz de usuario gráfica (GUI) basada en wxPython.

**Historia**

La primera versión del programa fue creada por Horst Herb. Cuando cesó Herb en el desarrollo activo, el trabajo fue dirigido por Karsten Hilbert quien asumió las funciones de jefe de proyecto. Karsten Hilbert no estaba solo en sus esfuerzos. Varios desarrolladores de todo el mundo se unieron al equipo y ayudaron en un momento u otro: Syan Tan, Ian Haywood, Hilmar Berger, Sebastian Hilbert, Carlos Moro, Michael Bonert, Rcihard Terry, Tony Lembke y muchos más. Mientras que algunos se concentraron en la programación del código, muchos más contribuyeron enormemente mediante la creación de una excelente documentación (Jim Busser).

El nombre fue elegido inicialmente para dar crédito al proyecto GNU y por la conexión con la profesión médica. El logotipo representa a un ñu como una referencia al proyecto GNU acompañado de una pitón como una referencia al lenguaje de programación en el cual se desarrolla la aplicación (Phyton), así como a la profesión médica.

**Características**

Este software se utiliza principalmente para la gestión de registros médicos electrónicos (historiales médicos), aunque soporta una variedad de características, muchas se ejecutan como *plugins* para extender su funcionalidad básica.

A continuación, se muestran algunos de los usos de esta aplicación:

**Administración**

Utilización de diferentes aplicaciones asociadas a GNUmed.

- KOrganizer.
  - Ver las citas o registros en KOrganizer.
  - Dar de alta un paciente en KOrganizer.

- Terminiko.
  - Transferencia de un paciente de GNUmed a Terminiko.
  - Dar de alta un paciente en Terminiko.

- Otras aplicaciones permiten:
  - Dar de alta un paciente en GNUmed.
  - Crear una entrada de un paciente GNUmed.

**Administrar listas de espera.**

- Añadir estado actual del paciente según la zona de la lista de espera.
- Filtrado de pacientes en lista de espera según la residencia para una visualización de: la zona, urgencia, tiempo de espera, paciente, etc.

**Manejo de personal.**

- Gestión de los miembros del personal.

**Historial Médico**

**Control de alergias en los pacientes.**

- Ver a simple vista la ausencia o existencia de alergias.
- Ver y actualizar cuando el estado de una alergia fue modificada.
- Confirmar rápidamente / actualizar / revisar el estado de las alergias.
- Comprobar las alergias a medicamentos específicos.

**Gestión de documentos**

- Agregar nuevos documentos.
    - Seleccionar los documentos del sistema de archivos.
    - Obtener datos de escáneres y cámaras de fotos.
- Ver los documentos en diferentes tipos de modos.
- Clasificar y firmar los documentos.
- Creación de documentos para su utilización en aplicaciones diferentes de GNUmed.
- Generación de documentos de impresión, fax y correo.
- Acceso a los informes originales guardados en el exterior a través de enlaces permanentes.

**Formularios y cartas**

- Plantillas para escribir cartas y formularios basados en OpenOffice desde dentro de la aplicación GNUmed.
- Transferencia de datos en formularios y cartas con la ayuda de marcadores de posición interactivos.
- Los formularios y las cartas se almacenan con la información del paciente, como cualquier otro documento.

**Bandeja de entrada**

- Ver la bandeja de entrada.
- Ir desde los mensajes de entrada a los documentos sin revisar de un paciente.

**Características nuevas de administración**

- Administrar manualmente los resultados de las pruebas.
- Clasificar y firmar los resultados.
- Gestión de las pruebas de un paciente.

**Notas**

- Observar cronológicamente pacientes según sus problemas de salud.
- Exportación a un archivo de texto.

**Lista ordenada de problemas de salud**

- Gestionar los problemas de salud.
- Gestión de consultas.
- Administrar las notas de progreso.

**Manejo de la información del paciente**

**Añadir pacientes a la aplicación**

- Agregar manualmente un nuevo paciente.

**El ingreso de pacientes**

- Ingreso de nuevos pacientes con informes en otro programa.
- Ingreso de pacientes nuevos a través de la importación de tarjetas médicas.

**Combinación de los pacientes**

- Combinar dos pacientes.

**Manejo de pacientes**

- Búsqueda de pacientes por:
  - Fragmentos de nombres (nombres anteriores, apodos...).
  - Cualquier identificación externa e interna (permiso de conducir, plan de tratamiento...).
  - Fecha de nacimiento.
- Modificar datos demográficos del paciente.
  - Las identidades múltiples o de ayuda (nombres y alias), direcciones, números de teléfono, etc.
- El acceso a un paciente por más de una persona a la vez.
- Deshabilitar el registro de pacientes.

**Otras funcionalidades**

**Personalización**

- Configuración.
  - Basado en perfiles por usuario, por lugar de trabajo, bases de datos, etc.
- Scripts personalizados.
  - Ejecutar un script después de la activación de un paciente.

**Interfaces**

- Enlaces externos a diferentes aplicaciones.
- Control remoto GNUmed de una aplicación.
  - Activar un *plugin* particular para un determinado paciente.

**Multi-lenguaje**

- Una sola base de datos puede presentar:
  - Los menús
  - Elementos de la pantalla.
  - Listados en varios idiomas para diferentes usuarios.

**Estadísticas e Informes**

- Elaborar informes estadísticos sobre la información en la base de datos.
- Activar a los pacientes desde el informe.
- Visualizar los resultados.

**La funcionalidad de base de datos**

- Instalación de dispositivos de cifrado.
- Acceso desde Linux, Windows y MacOSX.
- Copia de seguridad automática (wiki.gnumed.de/bin/view/Gnumed, 2017).

## 2.7 FFEHR

FFEHR (*Free Feathers Electronic Health Record*) es una aplicación desarrollada utilizando el marco de la programación de Mozilla. Este programa se puede ejecutar de forma independiente o dentro del navegador Firefox. Dos de las principales características del software son: su seguridad en el intercambio de datos en red y su aplicación en entornos multiplataforma. Este programa se puede ejecutar como una extensión de Firefox u otros navegadores compatibles. También puede funcionar como una aplicación local independiente a través de Xulrunner.

El objetivo inicial del proyecto es el diseño de una interfaz de usuario común que sea eficaz, eficiente y ampliamente aceptable para los profesionales de la medicina en Filipinas y en el futuro, en todo el mundo. Este programa tiene licencia bajo la GNU GPL3 y por lo tanto está disponible para descarga gratuita y su uso, incluyendo su código fuente.

**Objetivos del diseño de FFEHR**

- Para desarrollar una extensión para Firefox que sirve como un historial médico electrónico; el objetivo principal es generar informes estándar, tal como resúmenes clínicos.
- Debe ser capaz de guardar archivos con formato XML.

- Los archivos XML se deben validar con el esquema HL7 (esquema de copia local).
- Para simplificar la implementación, el formato específico XML se suministrará a los desarrolladores (estos son implementaciones simples del esquema de HL7).
- Formularios: se intentarán evitar los problemas relacionados con SOAP.
- Añadir nuevo paciente.
- Añadir un nuevo SOAP para el paciente.

En resumen, se pretende que la aplicación sea como un bloc de notas con capacidad para presentar formularios estructurados, guardar los datos con formato XML y generar informes (de los archivos XML) en un formato estructurado.

**Instalación con XULRunner**

XULRunner es una herramienta que ayuda enormemente a los usuarios en la instalación de FFEHR. Si se desea que la aplicación pueda ejecutarse como una extensión del navegador basado en Mozilla (Firefox), se deberán descargar los paquetes necesarios para la extensión junto con el navegador.

**Instalación en Linux**

- Descargar la última versión del archivo tar comprimido con gzip.
- Extraer los archivos y ejecutar la aplicación FFEHR.

**Instalación en Windows**

- Descargar la última versión archivo comprimido.
- Extraer los archivos y ejecutar la aplicación FFEHR (trac.afterfivetech.com/ffehr, 2017).

## 2.8 HealthForge

Esta aplicación ha sido diseñada para su despliegue en distintos complejos hospitalarios de diversos tamaños para así apoyar en la medida de lo posible las necesidades que tiene la comunidad de sanitaria. Su principal misión es proporcionar una solución flexible para la gestión de historiales clínicos en los consultorios médicos y conectar a la comunidad sanitaria de la mejor manera posible.

Toda la aplicación ha sido desarrollada en código abierto utilizando las herramientas más robustas. Puede ser instalado en cualquier servidor de Microsoft con MS SQL (Microsoft SQL Server). Una vez que se instalan, el médico y sus pacientes pueden mantenerse en contacto mediante un gran conjunto de herramientas telemáticas que hacen posible el seguimiento médico de un enfermo.

La aplicación básicamente contiene lo siguiente:

- Historiales médicos electrónicos (EHR).
- Sistema de gestión práctica.
- Herramientas paciente-médico.
- Herramientas para la gestión de contenidos.
- Herramientas para el acceso al usuario.

El software es una plataforma totalmente interoperable para conectarse a la comunidad sanitaria a la cual pertenece. Esto permite a los pacientes estar en contacto con otras organizaciones de tipo sanitarias que conectan sus servicios en una red más amplia.

Dentro de la comunidad sanitaria esta aplicación es totalmente funcional y muy robusta utilizándola para la gestión de registros médicos electrónicos (EHR) y además ofrece una práctica gestión de los datos. Estas características dan al programa la capacidad de expandirse con facilidad.

El desarrollo de toda la aplicación se realiza bajo la filosofía de código abierto y utilizando el concepto de modularidad lo cual ayuda a la integración del programa en todo tipo de sistemas utilizados por los hospitales ya que su adaptación consistirá en la instalación del módulo correspondiente y en el caso de

no existir, siempre cabe la posibilidad de realizar la implementación de dicho módulo gracias al código libre.

Cada día están apareciendo nuevos módulos creados que pueden ser utilizados bajándolos e instalándolos puesto que están exentos de licencias, lo cual es una característica importante que ayuda a su difusión.

El software puede ser fácilmente modificado para satisfacer todas las necesidades del hospital en el que se implemente. Por ejemplo, se pueden personalizar distintas características como: cambiar diseños de página, agregar usuarios y personalizar la apariencia. Todo está disponible dentro de la solución presentada en este software libre.

También se puede decir que la aplicación es una solución segura para los hospitales y centros de salud. El programa es totalmente compatible y se integra fácilmente dentro de un entorno de red con LDAP (*Lightweight Directory Access Protocol*) (healthforge.codeplex.com, 2017).

## 2.9 Hospital OS

Hospital OS es un proyecto de investigación y desarrollo de un software de gestión sanitaria para apoyar a los hospitales pequeños. Es financiado por el Fondo de Investigación de Tailandia.

Los esfuerzos del proyecto se centran en facilitar el trabajo a los hospitales de las zonas remotas donde la tecnología parece tener dificultades para llegar. Para ello se ha diseñado y desarrollado un sistema de información para la gestión hospitalaria llamado "Hospital OS". Este software es un programa de código abierto, dirigido a proporcionar servicios médicos eficientes y gestión clínica. A pesar de la falta de presupuesto y el lento avance tecnológico en las comunidades rurales, la tarea de los desarrolladores es crear un sistema de información médica eficaz, con el fin de fomentar un desarrollo sostenible para todas las comunidades en Tailandia.

La aplicación es un sistema para la gestión de la información del hospital que se centra en la organización de sus tareas y recursos. Se trata de un modelo

cliente-servidor en el que el servidor funciona como una unidad central que almacena toda la información y los clientes son unidades software que acceden al servidor remotamente para consultar, agregar o modificar diferentes datos.

La aplicación servidora para su funcionamiento utiliza el sistema operativo Linux y PostgreSQL como base de datos. Linux y PostgreSQL son software de código abierto disponible para descargar en Internet. El software utilizado por el cliente se desarrolla mediante el uso del lenguaje de programación Java y puede ejecutarse bajo diferentes versiones de Windows, Linux y otros sistemas operativos que ofrecen la posibilidad de instalar la máquina virtual Java.

Las características de la aplicación Hospital OS se pueden agrupar en los siguientes campos:

- Interfaz de usuario.
- Registro.
- Diagnóstico.
- Pedidos.
- Nombramiento.
- Historiales médicos.
- Farmacia.
- Facturación.
- Laboratorio.
- Rayos-X.
- Urgencias y traumatología.

El software desarrollado ayuda a mantener la base de datos digital y los recortes en el tiempo de procesamiento ya que al inicio del proyecto se observó que los pacientes tienen que esperar un largo tiempo para obtener servicios médicos en los hospitales. Por lo tanto, surgió la idea de un sistema que puede ayudar a mejorar los servicios clínicos.

El programa Hospital OS ha sido hasta ahora utilizado en 80 hospitales y 60 centros de salud del gobierno en toda Tailandia.

La aplicación ayuda enormemente a los hospitales a hacer la transición de los sistemas basados en papel a los sistemas electrónicos, ayudándoles a acelerar los procesos que son rutinarios. Se estima que el programa Hospital OS puede reducir el tiempo de espera para pacientes en ambulatorios en un 20 por ciento.

El programa viene con módulos de apoyo para el departamento de atención ambulatoria (OPD) y el departamento de pacientes internos (IPD).

Los módulos incluyen: la gestión de flujo de trabajo, el registro, diagnóstico, ordenado, el historial y su orden de servicio de pacientes, farmacia, facturación, laboratorio y los informes radiológicos, etc.

Desde el proceso de registro para la identificación del paciente o el número utilizado en el hospital, el personal sanitario puede acceder a los registros rápidamente a través del sistema y enviar los pacientes al médico especialista de inmediato.

Los médicos también pueden agregar muy fácilmente nueva información médica sobre un paciente del sistema. Además, pueden enviar recetas por vía electrónica a los farmacéuticos a través de la aplicación. En lugar de leer la escritura de un médico, el farmacéutico puede leer la receta médica directamente desde el ordenador y esto ayuda a reducir los errores.

Por otra parte, una vez que un mayor número de hospitales públicos y clínicas instalen el software Hospital OS, sus sistemas pueden estar conectados.

La Oficina Provincial de Salud Pública de Tailandia puede utilizar la información sanitaria para realizar una correcta planificación. La aplicación también puede estar vinculada a Google Maps y mantener a los funcionarios de salud pública al corriente de la situación sanitaria general y así poder hacer una gestión de las epidemias más rápido. Toda la información clínica se encuentra disponible en forma de imágenes, texto y mapas.

Cuando estén conectados los hospitales, los funcionarios del campo de la medicina podrán comprobar el estado de los pacientes de cada área y buscar las personas infectadas con una enfermedad. También es posible concentrarse en la zona que parece haber un brote epidemiológico (hospital-os.com/en, 2017).

## 2.10 HOSxP

HOSxP es un sistema para la gestión de la información en los hospitales, incluidos los historiales médicos electrónicos (HME). Actualmente se encuentra en uso en más de 70 hospitales de toda Tailandia. El programa tiene como objetivo facilitar el flujo de trabajo en los centros de salud, sanatorios y en los hospitales centrales.

Antes de convertirse en HOSxP, el software se llama KSK-HDBMS. Se buscó un nombre más amigable, por lo que el equipo de desarrollo ha optado por llamarlo HOSxP, que viene de; Hospital y experiencia. Su denominación también refleja la interfaz gráfica de usuario del software, que imita el tema de Windows XP. Se distribuye bajo Licencia Pública General GNU (GPL), HOSxP es de software libre.

**Historia**

El desarrollo de la aplicación comenzó en el año 1999. Surgió de un proyecto en solitario de Chaiyaporn Suratemekul, farmacéutico de profesión. Los desarrolladores principales del software son el personal de Bangkok Medical Software Co., Ltd., una empresa gestionada por Chaiyaporn. El desarrollo de la infraestructura de la aplicación, incluyendo repositorio de código fuente, está alojado en sourceforge.net.

**Datos técnicos**

El programa utiliza una arquitectura cliente-servidor. Para el servidor de la base de datos, se pueden realizar consultas con diferentes aplicaciones como; MySQL, Microsoft SQL Server, Interbase / Firebird y PostgreSQL.

- Cliente-servidor.
    - El software del servidor puede ejecutarse en Linux o Microsoft Windows.
    - El software del cliente sólo se puede ejecutar en Microsoft Windows.
- Tecnología multi-nivel (Borland DataSnap).

- Distributed Component Object Model (DCOM).

Borland Delphi y Kylix que es su homólogo en Linux, son los entornos de desarrollo integrado elegidos para el proyecto.

Un usuario puede escribir scripts en el lenguaje de programación C++ para automatizar tareas en HOSxP (hosxp.net, 2017).

## 2.11 Indivo

Indivo es un sistema que sirve para el registro de los historiales médicos con la posibilidad de controlarlos personalmente. Esta aplicación permite a un individuo poseer y administrar una completa y segura, copia digital de su historial clínico donde puede ver el estado de su salud. Indivo es libre y de código abierto. Es utilizado activamente en diversos lugares, en particular en el Hospital de Boston y en el Consorcio de Dossia.

Indivo está diseñado para extenderse y ser personalizado por lo que los usuarios pueden conectar su registro a aplicaciones de terceros que mejoran la gestión y el análisis de su información de salud. Indivo acelera el desarrollo de estas aplicaciones de terceros, proporcionando un conjunto básico de características comunes:

- El almacenamiento seguro, la categorización y la agregación de los datos de salud.
- Inicio de sesión único y las normas de datos basados en la delegación de acceso.
- Tecnología simple, basada en Web haciendo uso de la *Application Programming Interface* (API).
- Notificación de usuario unificada.

**Investigación**

El sistema Indivo es esencialmente un proyecto de futuro para la gestión de los historiales médicos, en el que los pacientes conceden permiso para el manejo

de sus datos clínicos a las instituciones, médicos, investigadores y otros usuarios de la información médica. Indivo es una distribución basada en tecnología Web y disponible bajo una licencia de código abierto.

**Historia**

El proyecto Indivo tiene sus raíces en el proyecto "*Guardian Angel*", una colaboración entre la Universidad de Harvard y el MIT (*Massachusetts Institute of Technology*), que prevé una red en Internet con un gestor automatizado para organizar los datos de salud y las decisiones. El modelo de Indivo ha inspirado una serie de esfuerzos comerciales a lo largo de su evolución. Más recientemente, se ha promovido un cambio hacia un modelo de plataforma, de modo que otra aplicación puede conectarse para extender su funcionalidad básica (indivohealth.org, 2017).

## 2.12 Medical

Medical es una aplicación de software libre que ofrece tanto la funcionalidad de un Expediente Médico Electrónico (EME) como un sistema de Información Hospitalaria y de Salud para la aplicación OpenERP.

El objetivo principal es proveer un Sistema de Información de Salud y Hospitalario universal de código abierto, en el que doctores e instituciones de todo el mundo, especialmente en los países en vías de desarrollo se beneficien de una aplicación escalable, centralizada y libre que mejore la calidad de vida de sus habitantes.

**Características**

- Focalizado en la medicina familiar y APS (Atención Primaria de la Salud).
- Interés en condiciones socio-económicas (estilos de vida, ámbito familiar, educación...).

- Enfermedades y procedimientos médicos standard como ICD-10 (*International Statistical Classification of Diseases and Related Health Problems*).
- Marcadores genéticos y riesgos hereditarios: más de 4200 genes relacionados con enfermedades.
- Epidemiología y otros registros estadísticos.
- Registro electrónico. Sin necesidad de papel.
- Administración del paciente (creación, evaluación/consultas, historia...).
- Administración del doctor.
- Administración de laboratorio.
- Medicamentos.
- Gestión de stock y de cadena de abastecimiento.
- Administración financiera y gestión hospitalaria.
- Diseñado con los estándares de la industria en mente.
- Software libre: licencia GPL v3.

El modelo de datos está diseñado de tal forma que centraliza la información para que no ocurran duplicaciones.

Por otro lado, Medical optimiza la colaboración y comunicación entre profesionales de la salud. Por ejemplo, un doctor puede solicitar un análisis de laboratorio específico a un paciente, esto es procesado por el patólogo quien lo diagnostica y vuelca el resultado en el sistema. El doctor ahora tiene toda la información relacionada en el registro del paciente.

Otra característica importante del programa es que permite adjuntar documentos (radiografías, resultados de biopsias...) al registro del paciente.

**Standards**

La información generada y recopilada en Medical usa los estándares de la industria médica. Se utiliza ICD-10 para patologías / diagnósticos y ICD-10-PCS (*International Statistical Classification of Diseases and Related Health Problems*

*Procedure Coding System*) para los procedimientos médicos. Por ejemplo, en lugar de registrar un nombre arbitrario, el doctor podrá seleccionar entre alrededor de 14000 variables exclusivas de enfermedades. Esto es muy importante para estudios epidemiológicos y estadísticos, además de la interoperabilidad entre instituciones de otros países.

De esta forma, el historial del paciente estará en un formato que los centros médicos de todo el mundo estarían en condiciones de procesar. Efectivo, sin papeles y una manera rápida de practicar la medicina.

La aplicación utiliza la información genética del "*National Center for Biotechnology Information"* (NCBI) y "*Genecards*" para vincular desórdenes genéticos con enfermedades. Esta información es importante para evaluar riesgos del paciente de contraer una enfermedad específica y de probablemente transmitirla a futuras generaciones.

Finalmente, hay que añadir que el programa también registra el historial familiar del paciente para hacer un mejor seguimiento de las enfermedades que puedan desarrollarse en el futuro.

**Seguridad**

Sigue el control de acceso y modelo de seguridad de OpenERP 5.0.

**Localización**

El idioma principal es el inglés, pero puede ser fácilmente traducido a cualquier otro idioma utilizando el módulo de traducción de OpenERP 5.0. La versión en español coexiste con la de inglés (medical.sourceforge.net, 2017).

## *2.13 OpenEMR*

Este software fue desarrollado, como todas las aplicaciones destinadas a los registros médicos, para agilizar el acceso y manejo de historiales clínicos con el mínimo gasto económico. Su desarrollo en un entorno Web y su transparencia

permiten a múltiples usuarios introducir datos de forma simultánea desde diferentes ubicaciones geográficas.

Actualmente todas las evoluciones son revisadas meticulosamente por la comunidad de OpenEMR que se dedica a guardar su estado como solución de software libre. Dicha comunidad está compuesta principalmente por desarrolladores software y médicos. Todos ellos trabajan con el objetivo de conseguir que OpenEMR sea mejor aplicación que las alternativas presentadas por el software propietario.

**Especificaciones técnicas**

En cuanto a la arquitectura técnica de OpenEMR se puede decir que es un "LAMP" (Linux, Apache, MySQL, Perl, PHP o Python). Es un tipo de aplicación de software basado en Web que utiliza un servidor Web como Apache, MySQL como base de datos y PHP como lenguaje de programación. Como con la mayoría de arquitecturas tipo "LAMP", OpenEMR puede ejecutarse en Linux, Unix, BSD, Mac OS X, y las arquitecturas de Microsoft. Los datos del paciente en OpenEMR se pueden transmitir a través de Internet utilizando la tecnología de red de área amplia (WAN) o en la Intranet a través de la red de área local (LAN). OpenEMR se basa en el uso de los protocolos *Transmission Control Protocol/Internet Protocol* (TCP/IP) para el intercambio de datos a través de la red. TCP/IP es un conjunto de protocolos de comunicación utilizados normalmente en Internet y en distintas redes privadas Intranet. La capa de aplicación incluye la tecnología HTTP, que apoya la aplicación de la *World Wide Web* (WWW) y que en OpenEMR se ve representado por la ventana del navegador. TCP asegura la transmisión de datos precisos y fiables en toda la red. La forma normal para asegurar OpenEMR es habilitar *Secure Socket Layer* (SSL) en el servidor Web. En caso de producirse el acceso a través de Internet, el certificado SSL se puede instalar en el navegador Web. De esta manera el cliente se autentica ante el servidor evitando intrusiones.

OpenEMR puede exportar algunos de sus datos a otros sistemas, tales como nombres, direcciones, números de teléfono, información del seguro, etc; que

no se pueden transferir directamente, sino que requieren un filtro en el sistema de recepción para saber lo que significan los diferentes campos. La forma más fácil de transmitir la información es una salida en un formato estándar, tal como HL7. Otros datos se guardan en la base de datos sólo como una serie de casillas de verificación sí/no. Esta información sería muy difícil de interpretar sin la información de páginas Web asociadas. Esto es típico en los formularios de una visita médica donde los profesionales médicos generalmente no son mecanógrafos y prefieren los sistemas de apuntar y hacer clic. Estos se guardan en la base de datos de información booleana (sí/no, verdadero/falso) que será más difícil a la hora de transmitir ya que habrá que darle sentido a la información allí contenida. El modo de funcionamiento es el siguiente: el servidor Web recibe la solicitud del explorador, que utilizando el intérprete PHP con el fin de evaluar la solicitud, actúa sobre los paquetes entrantes para finalmente pasar las consultas al motor de base de datos según sea necesario. Luego, el motor de la base de datos realiza las consultas en el servidor de base de datos (MySQL). Después de ejecutar la consulta el resultado de la misma se devuelve al servidor Web a través del motor de PHP, que forma una página http con el formato apropiado. El servidor Web devuelve esta página http al navegador solicitante. El servidor Web puede ser cualquier servidor Web compatible con PHP, aunque normalmente se suele utilizar Apache para la aplicación OpenEMR. El servidor de base de datos, teóricamente, puede ser cualquier base de datos SQL. MySQL es la base de datos que utiliza OpenEMR. MS SQL Server u Oracle aún no han sido probados como bases de datos para la aplicación.

Otra característica de OpenEMR es que, dependiendo de la configuración realizada por el administrador, la aplicación puede funcionar como distribuida (cliente/servidor) o como un sistema informático centralizado. En el caso de tener un modelo distribuido y teniendo en cuenta que dicho funcionamiento consiste en un conjunto de equipos autónomos conectados a través de un *middleware* de red y distribución, permitiendo a los equipos coordinar sus actividades y compartir los recursos del sistema, todos los componentes (servidores de base de datos, servidores de facturación, etc.) de OpenEMR se pueden distribuir de forma

transparente a través de una red de ordenadores heterogéneos y aún se percibe como un centro de cómputo. Esta arquitectura distribuida aumenta la flexibilidad en la configuración de OpenEMR en un hospital, donde los principales cambios se realizan en una máquina y se replican en todos los equipos de la misma red. Además, hay bajos costes de transmisión de datos con mayor capacidad de procesamiento de la información. Sin embargo, en comparación con la computación centralizada, los costes de los usuarios finales de equipos y costes de gestión de red se incrementan en un entorno de computación distribuida, por lo que es ideal en los hospitales grandes.

En pequeñas clínicas, hospitales privados pequeños y consultorios, donde no se necesita la gran potencia de procesamiento ofrecida por los sistemas de computación distribuida, OpenEMR es lo suficientemente flexible como para permitir la computación centralizada con el fin de reducir costes. En un entorno informático centralizado todos los datos críticos y los programas se almacenan en un ordenador principal, el ordenador central. El entorno informático centralizado tiene bajos costes de los equipos de usuario final al mismo tiempo que se reducen los costes de gestión de red, aunque el rendimiento está estrechamente vinculado con el hardware y el software que suelen tener un gran tiempo de uso. (open-emr.org, 2017).

## *2.14 Open Healthcare*

El grupo encargado del desarrollo del proyecto Open Healthcare es una organización dedicada a la promoción y la distribución de una aplicación de código abierto utilizada para la gestión de historiales médicos. Utiliza la tecnología XML para su implementación y se ha desarrollado con el fin de ayudar a automatizar y mejorar la práctica de la medicina. Además, al estar desarrollado en código fuente abierto, podrá ser desarrollado y personalizado según sus preferencias creando así sus propias plantillas.

Al igual que otros programas de código abierto, esta aplicación para los historiales médicos, permite que todo el mundo pueda usarlo y modificarlo de forma gratuita.

El grupo que desarrolla este proyecto quiere crear una comunidad de personas que compartan el objetivo de mejorar la atención médica. Esta comunidad será capaz de utilizar libremente la aplicación Open Healthcare. Esto maximiza el acceso de todos a la tecnología sanitaria Open Group.

La comercialización de la asistencia sanitaria y el software a menudo han conducido al desarrollo de los "secretos empresariales". Esto ha ahogado la cooperación y el progreso de ambas industrias. Los miembros de este proyecto Open Healthcare consideran que colectivamente se puede trabajar en conjunto para crear un registro de salud que mejorará considerablemente la práctica de la medicina. Además, la filosofía del proyecto es que todos deben beneficiarse y tener acceso a esta nueva tecnología.

**Datos técnicos**

El objetivo es crear un instrumento universal, de larga duración, para la búsqueda de historiales médicos electrónicos. La aplicación está desarrollada en XML nativo por lo que es independiente del sistema operativo y la aplicación de software. Está en proceso el desarrollo de un sistema para habilitar el procesamiento de XML basado en XML.

Se han desarrollado técnicas para interactuar con sistemas tradicionales usados en las bases de datos como SQL que se utilizaron en la implementación inicial. Actualmente se está en proceso de transformar este sistema en un sistema plenamente XML nativo y se han desarrollado técnicas para editar archivos XML como si fueran tablas de SQL.

La aplicación *servlet* utiliza la técnica de XMTP (*XML MIME Transformation Protocol*) de transformación de la solicitud de MIME (*Multipurpouse Internet Mail Extensions*) origen en una representación XML, aunque es SAX (*Simple API for XML*) realmente, que son introducidas en una cadena de procesamiento de SAX. La interfaz de TRAX (*TRansformation API for*

*XML*) facilita el trabajo utilizando los motores de la transformación Xalan como filtros SAX.

El proceso de creación de este gestor de historiales médicos basado en XML como una secuencia de paquetes individuales, permite agrupar los estudios de investigación médica, los resultados de análisis médicos y los informes de errores médicos. Se cree firmemente que este enfoque proporcionará grandes beneficios al sistema de salud, aunque claramente hay que realizar más investigación y trabajo. Este es un comienzo y parece que el desarrollo del software va en la dirección correcta (openhealth.org, 2017).

## *2.15 OpenMRS*

OpenMRS se diseñó como un sistema genérico para la gestión de historiales médicos que pueda ayudar en el cuidado de los pacientes, la recopilación de observaciones, visitas al especialista, notas aclaratorias sobre un determinado paciente y otros datos necesarios para realizar diagnósticos. La finalidad de la aplicación es conseguir facilitar el tratamiento de la información sobre los pacientes a los facultativos. Así se podrán realizar diagnósticos invirtiendo menos tiempo disminuyendo el tiempo de espera entre pacientes.

Ha demostrado gran eficacia en la gestión de datos médicos de enfermos de VIH y tuberculosis, principalmente en Kenia, aunque posteriormente se implementó en Ruanda y Sudáfrica. Una vez comprobado su buen funcionamiento, se instaló en otros países como Malawi, Mozambique, Lesotho, Tanzania, Uganda y Haití (Allen *et al.*, 2006). En la actualidad OpenMRS ofrece servicio en numerosos hospitales y lugares de gestión médica, destacando su gran funcionalidad y su bajo coste de instalación. Un ejemplo claro de su uso es la versión de OpenMRS creada por la Organización Mundial de la Salud (OMS) llamada OpenMRS Express, que adapta el software a los requerimientos exigidos por una gran institución gestora.

En OpenMRS se definen todos los conceptos básicos utilizados en el sistema. Usando combinaciones de preguntas y respuestas, se pueden definir las

observaciones (datos importantes a tener en cuenta), así como los formularios que recogen múltiples observaciones. Los primeros sistemas fueron construidos a partir de formularios en papel convirtiéndolos en formato electrónico partiendo de la catalogación de todos los formularios y la organización de estos de forma jerárquica.

OpenMRS es una aplicación desarrollada en Java, cuya utilidad es la gestión de los EMR mediante una interfaz Web. Su planteamiento inicial consiste en un sencillo modelo de datos que mediante una API obtenía un correcto funcionamiento. Luego se construyó una aplicación basada en Web que utiliza la API y así consigue realizar el trabajo deseado (Mamlin *et al.*, 2006). La API OpenMRS funciona como una "caja negra" ocultando la complejidad del modelo de datos y garantizando que las aplicaciones y los módulos tengan un correcto funcionamiento.

Aunque en sus inicios el grupo de desarrolladores era pequeño y la implementación lenta, los organizadores del proyecto de desarrollo de OpenMRS han aprovechado las conferencias semanales, listas de correo, Wikis, un sistema de código de versiones y el software de seguimiento de proyectos, para gestionar la colaboración de la gente interesada en trabajar en el proyecto de forma altruista (Allen *et al.*, 2007).

**Especificaciones técnicas**

La aplicación ha sido implementada con una arquitectura definida en distintos niveles. Estas capas están claramente separadas como se puede observar en la figura 1.

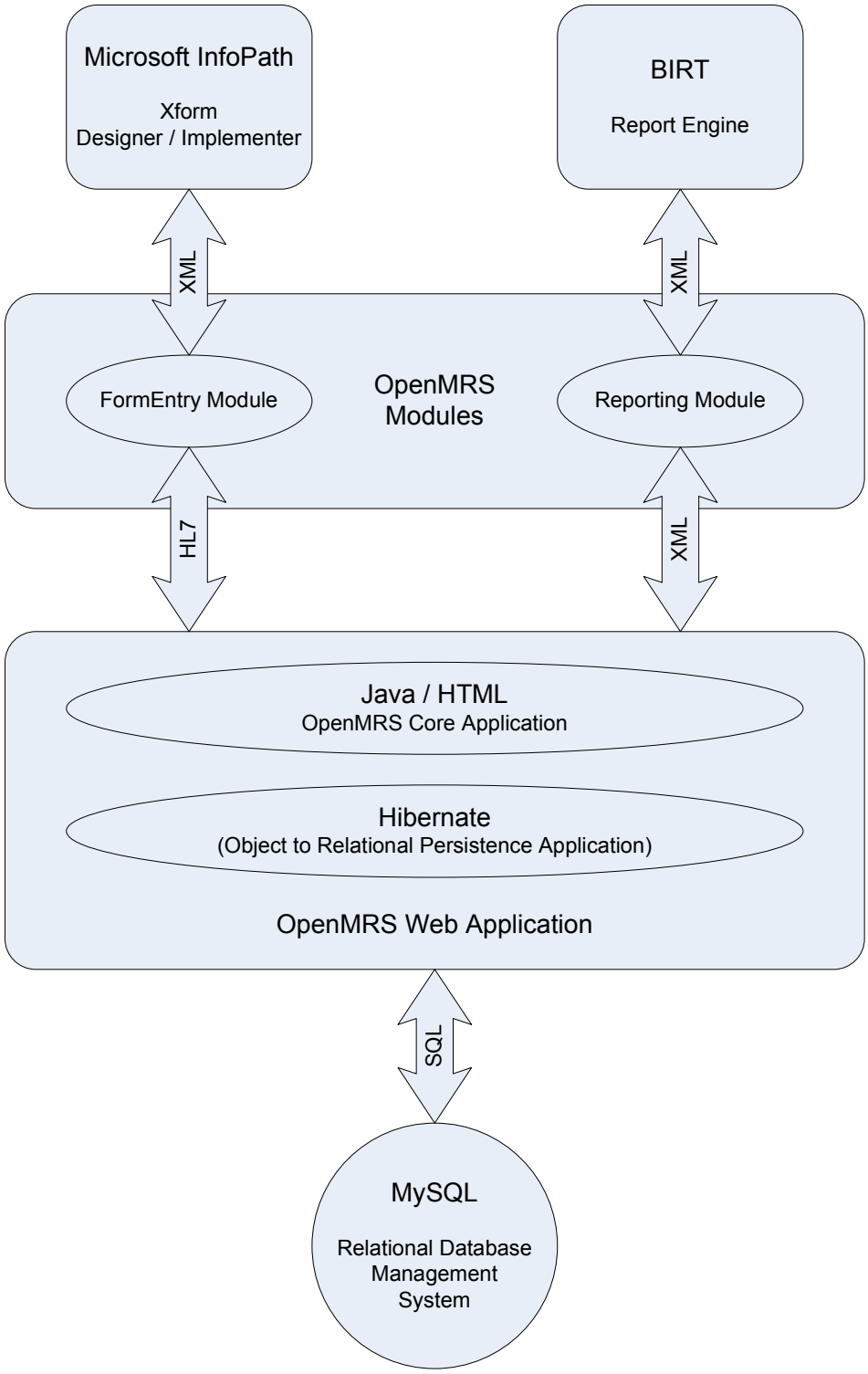


**Figura 1. Arquitectura de OpenMRS.**

A continuación, se explicarán todas y cada una de las capas que componen la arquitectura de la aplicación OpenMRS.

- **MySQL** es una base de datos muy rápida en la lectura, pero puede provocar problemas en entornos de alta concurrencia. En aplicaciones Web hay baja concurrencia en la modificación de los datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL perfecto para este tipo de aplicaciones OpenMRS (mysql.com, 2017).
- **SQL** es un lenguaje de acceso a bases de datos relacionales. Una de sus características es permitir efectuar consultas con el fin de recuperar, de una forma simple, información de interés de una base de datos, así como también hacer cambios sobre ella. Es un lenguaje informático declarativo de "alto nivel" o "de no procedimiento" de cuarta generación, que gracias a sus características permite una alta productividad en codificación y la orientación a objetos. Una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros haciéndolo más eficiente (sql.org, 2017).
- **Hibernate** es una herramienta de mapeo objeto-relacional utilizada para la plataforma Java (también disponible para .NET con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos de tipo XML o anotaciones en los *beans* de las entidades que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL (*Lesser General Public License*). Básicamente esta herramienta busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria del ordenador (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr su finalidad permite al desarrollador detallar completamente cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate permite a la aplicación manipular la información de la base de datos operando sobre objetos, con todas las características de la

Programación Orientada a Objetos (POO). Hibernate convertirá la información entre los tipos de datos utilizados por Java y los definidos por SQL. Esta herramienta genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución que compensa el trabajo realizado. Es una herramienta muy flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible en un momento dado. Además, la herramienta Hibernate ofrece también un lenguaje de consulta de datos llamado *Hibernate Query Language* (HQL), al mismo tiempo que una API para construir las consultas. En el caso de usar la herramienta descrita para Java puede ser utilizada en aplicaciones Java independientes o en aplicaciones *Java Enterprise Edition* (Java EE), mediante el componente *Hibernate Annotations* que implementa el estándar *Java Persistence API* (JPA), que es parte de esta plataforma (hibernate.org, 2017).

- **Java** es un lenguaje de programación orientado a objetos. Mucha de su sintaxis es similar a la de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria, facilitando el trabajo al desarrollador. El software programado en java está compilado en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código máquina para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible, pero por lo general menos utilizada. (java.com/es, 2017).

- **HTML** (*HyperText Markup Language*) es el lenguaje de marcado predominante, principalmente usado para el desarrollo de páginas Web. Es utilizado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). Este lenguaje también puede describir la apariencia de un documento y puede incluir un script (por ejemplo, Javascript), el cual puede afectar al comportamiento de navegadores Web y otros intérpretes de código HTML (w3.org/html, 2017).
- **XML** es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es una simplificación y adaptación del *Standard Generalized Markup Language* (SGML) que permite definir la gramática de lenguajes específicos. Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades que pueden surgir. XML no sirve simplemente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, etc. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y hacen aumentar las posibilidades de su utilización. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil (w3.org/XML, 2017).
- **HL7** es un sistema creado para el intercambio de información en entornos relacionados con la medicina, que proporciona un entorno integrado con el fin de gestionar los datos médicos de una manera más sencilla. Para llevar a cabo la implementación del estándar se diseñó el servidor de mensajes HL7 (*Health Message Server* (HMS)), como una interfaz que se instala en cada institución médica, y el HL7 *Message Archiver* (HMA) como interfaz para la base de datos central. Estas dos

interfaces se comunican entre sí mediante el uso de mensajes HL7. Dada su eficiencia en el intercambio de datos se ha propuesto como estándar de comunicación para transacciones de información médica (hl7.org, 2017).

- **Microsoft InfoPath** es una aplicación usada para desarrollar formularios de entrada de datos basados en XML. Inicialmente se le dio el nombre "Xdocs". La principal característica de Infopath es la posibilidad ofrecida de crear y ver documentos XML con soporte para XML schemata. Infopath puede conectarse a sistemas externos usando servicios Web XML a través de *Microsoft XML Core Services* (MSXML) y el SOAP *Toolkit*, *back-end* y sistemas *middle-tier* pudiendo configurar la comunicación usando estándares de servicios Web como SOAP, UDDI (*Universal Description, Discovery and Integration*) y *Web Services Description Language* (WSDL). En esta herramienta, los usuarios completan los formularios en sus ordenadores mientras se trabaje sin conexión usando el cliente Infopath. Puede verificar los campos del formulario para su validación. Además, los usuarios pueden anexar una firma digital para mayor seguridad (office.microsoft.com/en-us/infopath, 2017).
- **BIRT** (*Business Intelligence and Reporting Tools*) es un proyecto de código abierto que proporciona información y las capacidades de inteligencia de negocios para el cliente y las aplicaciones Web, especialmente las basadas en Java y Java EE. Está diseñado para colaborar con la inteligencia de negocios en los procesos de las organizaciones. Específicamente se trata de herramientas que asisten el análisis y la presentación de los datos. Aunque algunas herramientas de Inteligencia de Negocios incluyen la funcionalidad ETL (*Extraction Transformation and Loading*), no son consideradas generalmente como herramientas de Inteligencia de Negocios. El objetivo principal de este conjunto de programas es hacer frente a una amplia gama de

necesidades de información dentro de una aplicación (eclipse.org/birt/phoenix, 2017).

Como se puede ver en la figura 1 hay una capa en la que se observa la posibilidad de implementar distintos módulos en la aplicación. Esta característica es muy importante ya que gracias a dicha modularidad el programa puede cambiar sus funciones notablemente sin apenas necesidad de realizar grandes cambios en el software. Casi todas las instalaciones de la aplicación OpenMRS utilizan el módulo FormEntry que permiten la captura de datos y la entrada al sistema. La versión actual del módulo FormEntry utiliza la herramienta InfoPath, que es una aplicación contenida dentro del paquete software Microsoft Office Professional. Aunque no es gratis, InfoPath se utiliza porque emplea XML y sigue un modelo que permite distinguir fácilmente los datos propios del formulario de la información introducida por el usuario. También InfoPath facilita la cumplimentación de formularios, proporcionando un entorno parecido a un procesador de textos, lo que ayuda notablemente a cambiar campos del formulario a las personas que no saben demasiado de informática. Aunque hay en marcha un proyecto para la creación de una herramienta alternativa a InfoPath de código abierto, por el momento la funcionalidad ofrecida es menor. Para que la aplicación OpenMRS pueda funcionar correctamente en un modelo cliente-servidor, en aplicaciones distribuidas y con otros sistemas sin problemas de interoperabilidad, los datos y formularios están representados en XML y en XForms (XML Formularios), respectivamente. Los datos se comunicarán entre el módulo FormEntry y la capa de aplicación mediante el uso de mensajes del sistema HL7. La información se almacenará en la base de datos de acuerdo a un modelo de relación abierta y puede ser tanto manipulada como consultada utilizando el lenguaje SQL (Seebregts *et al.*, 2009).

La principal característica del programa OpenMRS reside en su modelo de datos robusto y flexible al mismo tiempo. Desde su diseño inicial la arquitectura técnica de la aplicación se ha mantenido destacando en ella su modularidad. La implementación de una capa API facilita a los desarrolladores el trabajo, ya que no

necesitan conocer íntegramente el modelo de datos, únicamente necesitan conocer los objetos Java a utilizar. Esta capa API es la columna vertebral de la aplicación. En ella hay métodos para todas las funciones principales. Las principales clases que pueden ser utilizadas dentro de la API son: *PatientService*, *ConceptService*, *EncounterService* y *ObsService*. El contexto es una clase estática utilizada para permitir que la aplicación pueda guardar datos en la memoria. El contexto de los servicios se divide en dos categorías: los métodos de los Servicios y para los usuarios. Los servicios se mantienen en la clase *ServiceContext*. Por otro lado, el *UserContext* contiene métodos para actuar sobre los usuarios: la autenticación y autorización, iniciar sesión, cerrar sesión, etc. Para cada nuevo usuario que se incorpora al sistema se instancia una nueva clase *UserContext* y la actual clase *UserContext* se almacena en el subproceso actual. Cada vez que se accede al sistema se define dentro de una "unidad de trabajo". Esta unidad está bordeada por las llamadas a `Context.openSession()` y `Context.closeSession()` que en una aplicación Web se realizan en OpenmrsFilter, por lo que la mayoría de los desarrolladores no tienen que preocuparse de hacer esas llamadas. Todo esto facilita la rápida implementación de diferentes módulos para OpenMRS lo que permite un desarrollo de la aplicación a alta velocidad. Como se dijo anteriormente, si hay que destacar algo dentro de las especificaciones técnicas, la API sería lo principal a resaltar (openmrs.org, 2017).

## 2.16 OSCAR McMaster

OSCAR McMaster es una aplicación para la gestión de historiales médicos electrónicos (EMR) basada en tecnología web. El sistema fue desarrollado inicialmente para los hospitales universitarios de atención primaria, pero ahora se ha convertido en una aplicación global utilizado también como sistema de facturación que usan muchos consultorios médicos y clínicas privadas principalmente en Canadá, aunque también en otras partes del mundo. El nombre se deriva de donde fue creado y un acrónimo, OSCAR significa "*Open Source*

*Clinical Application and Resource*" y McMaster se refiere a la Universidad de McMaster, donde se desarrolló.

**Características**

La aplicación funciona básicamente como un servidor Web, el cual debe ser instalado en una máquina con Linux. Se recomienda usar Ubuntu (una de las tantas distribuciones Linux) que es realmente fácil de instalar y usar, ya que existe mucha información en la red sobre esta distribución; puede descargarse gratuitamente. Es importante instalar en idioma inglés el sistema operativo del servidor para evitar posteriores problemas. Una vez instalado el servidor, se accede al programa como si fuese una página Web por medio de un navegador en una máquina con cualquier sistema operativo.

Normalmente, para acceder a OSCAR simplemente deberá poner su nombre de usuario y contraseña. Esta medida, conocida como la autenticación, garantiza que sólo usted puede acceder a su cuenta. Por ello, nunca debe revelar su *password* a nadie. Su contraseña se almacena en un formato codificado de tal manera que incluso el administrador del sistema no puede saber cuál es. Si usted ha olvidado su nombre de usuario y/o contraseña, el administrador puede restablecer la contraseña inicial.

Dependiendo de la privacidad de la información de su cuenta la configuración, puede ser una clave de acceso de segundo nivel. Esto es típicamente un código que el administrador de la aplicación ha proporcionado al usuario y que normalmente se requiere cuando se está conectando desde el exterior de la oficina o red local (por ejemplo, desde casa o en el hospital). Esta es una característica de seguridad adicional. Es poco probable que un hacker adivine tanto la contraseña como el código de acceso de segundo nivel y pueda realizar operaciones no deseadas con los datos de los pacientes.

Si introduce la contraseña incorrecta tres veces, el equipo que está utilizando quedará excluido de utilizar el programa durante un período de tiempo. Si esto sucede, usted puede esperar un tiempo determinado para poder acceder a la aplicación o decirle a su administrador que proceda a desbloquear el equipo (él

puede hacerlo de forma remota). Esta es otra característica de seguridad que sirve para evitar que los hackers puedan tener más de tres oportunidades en el intento de adivinar una contraseña.

Cuando haya terminado de trabajar con la aplicación, salga con el botón situado en la esquina superior derecha de la pantalla principal. Al cerrar la sesión se asegura que ninguna persona no autorizada pueda acceder a su cuenta (en.wikipedia.org/wiki/OSCAR_McMaster, 2017).

## 2.17 PatientOS

PatientOS es un sistema de información sanitaria de código abierto, lanzado bajo licencia GPL 3.0. Este software empresarial está diseñado para facilitar el trabajo en el sector de la salud, como, por ejemplo; a los médicos, enfermeros, farmacéuticos, técnicos de laboratorio y otro tipo de empleados sanitarios.

Esta herramienta es multiplataforma, pues está desarrollada en un lenguaje interpretado (Java) y trabaja empleando diferentes sistemas para la gestión de bases de datos; PostgreSQL (software libre) y Oracle (software propietario).

**Características**

**Comunes a todos los módulos**

- Cada menú y barra de herramientas puede ser personalizado por el usuario, según su función o grupo utilizando el editor de formularios.
- Cada pieza visual del texto puede ser traducido a otro idioma.
- JavaScript permite interacción con el usuario.

**Registro**

- Los formularios de registro en ambulatorios y hospitales pueden ser creados usando el editor de formularios.
- También es posible: aplicar campos obligatorios, crear reglas personalizadas de datos de entrada, poner de relieve los campos

recomendados, realizar formatos de verificación y elegir la opción para auto-rellenar los campos.

- Exportación e importación de los pacientes.

**Programación de tareas**

- Se pueden programar las actividades para: días, semanas o meses.
- Los tipos de tareas con cualquier duración o de varios bloques de duración escalonada también pueden ser programados.
- Los recursos con horarios personalizados pueden ser creados usando el editor de recursos.
- Los horarios pueden ser bloqueados cuando sea necesario.
- Los usuarios pueden personalizar la vista de los recursos.

**Presentación de los resultados**

- Se pueden ver los resultados de una prueba en un formato estándar.

**Formas clínicas y administrativas**

- Número ilimitado de formularios personalizados mediante el editor de formularios clínicos.
- Se pueden definir campos propios.

**Medicamentos**

- Control de la alergia.
- Control de las dosis.
- Control del efecto adverso de un fármaco.

**Informes**

- Crear informes independientes utilizando la aplicación iReport.
- Generar informes del lado del servidor SQL Server para ver los clientes.
- Generación de informes basado en XML.

**Sistema**

- El editor de usuario permite: administrar los usuarios y las múltiples funciones asociadas a cada usuario.
- Crear listas personalizadas para trabajos por lotes.
- Importar o exportar usuarios a XML para la migración de sistema (patientos.org, 2017).

## 2.18 SmartCare

La aplicación SmartCare es un sistema para la gestión de los historiales médicos que ha sido desarrollado y desplegado por el Ministerio de Salud de Zambia, en colaboración con los Centros para el Control y la Prevención de Enfermedades y otros muchos socios de desarrollo.

**Características**

- Un sistema completo de historiales médicos electrónicos para asegurar la continuidad de la atención médica.
- Un sistema de información de gestión clínica a nivel de gestión y administración.
- Un componente clave en un sistema público sanitario.

**Funcionalidades**

- Cada habitante de Zambia recibe una atención médica confidencial donde y cuando lo necesita.
- Cada hombre, mujer y niño tiene un historial médico electrónico.
- Cada punto de servicio clínico puede acceder y actualizar este registro.
- Cada médico entiende el valor de este registro y se compromete a mantenerlo.
- Cada centro médico: municipal, provincial y nacional; envía los datos a la Sede del Ministerio de Salud donde se monitorizan y evalúan sistemáticamente los informes. Se utiliza esta información para asignar

óptimamente los recursos humanos y medios técnicos, para lograr una mejora continua y sistemática en los servicios de salud.

**Datos técnicos**

- Sistema de bases de datos distribuidas: teniendo en cuenta las limitaciones de recursos en los países en desarrollo como Zambia, donde la electricidad todavía no está disponible en algunas partes del país, el acceso a Internet en todo el territorio llevará muchos más años. La gestión de los datos se lleva a cabo en cada centro en un diseño de distribución, a diferencia de los diseños de la mayoría de los sistemas centralizados. Internet no es esencial, sólo es un beneficio añadido.
- Smart Card: el cliente de la aplicación se puede utilizar con unidades de memoria flash para así tener una solución de menor conectividad y de alta tecnología que funciona en la actualidad. La información sobre el estado de salud de un individuo se almacena en un mismo archivo comprimido que facilita su portabilidad.
- Pantalla táctil: la captura de datos puede ser la parte más difícil del diseño de una aplicación para historiales médicos. SmartCare extiende una idea de éxito, la pantalla táctil para la entrada de datos. El software funciona bien con esta forma de entrada de datos ya que permite al médico ver y grabar los datos del paciente. Esta herramienta, en combinación con la información específica del enfermo, puede proporcionar apoyo a la decisión del diagnóstico.
- GIS de visualización de datos: datos almacenados en las instalaciones de salud pueden ser visualizados en mapas GIS. Esto incluye los datos de pacientes vivos, así como los datos estáticos de encuestas de salud.
- SmartCare es una aplicación escrita en C#. La base de datos se implementó en Microsoft MSDE (*Microsoft SQL Server Desktop Environment*) y se ha migrado a SQL Server 2005 Express. La aplicación se instala en ordenadores con sistema operativo Windows

XP que será el único software con licencia que se utilizará para la implementación del sistema.

**Estado de implementación en Zambia**

- El Ministerio de Salud de Zambia ha desplegado la aplicación SmartCare en 200 instalaciones (clínicas; municipales, provinciales y nacionales). El número de pacientes los cuales poseen un historial médico electrónico asciende a más de 200.000.

**Los planes futuros en Zambia**

- Fortalecimiento de los sistemas médicos mediante el desarrollo de SmartCare.
- Finalización de los módulos de servicios para ambulatorios destinados a: la planificación familiar y la hospitalización de los niños.
- Sistemas de integración con sistemas de registro de salud en Zambia.
- Integración de las acciones de gestión de los medicamentos (smartcare.org.zm, 2017).

## *2.19 Tolven Healthcare*

La solución propuesta por la aplicación Tolven garantiza al personal sanitario la posesión de toda la más reciente información que ayudará en la toma de decisiones a los profesionales de la medicina. Un sistema de historiales electrónico como éste ofrece al consumidor una manera intuitiva de navegar por la información del paciente basada en aplicaciones tipo Web con el objetivo de crear, visualizar, almacenar y compartir información sanitaria. Con el fin de comunicarse con sus proveedores de servicios y tener el acceso necesario a la información sanitaria ubicada en servidores situados lejos de su área local (LAN) se utiliza Internet, aunque simplemente sea para realizar tareas rutinarias, como volver a comprobar una receta.

Los historiales electrónicos no son simplemente un registro estático, sino que permiten al personal sanitario de forma selectiva incluir los datos que un

médico ha recogido en una primera valoración de un paciente, lo que permite que el personal sanitario pueda fácilmente y con seguridad tener una visión parcial o completa de la información médica del paciente y que está a disposición de otros médicos que puedan atender al enfermo. Una de las principales ventajas del uso de historiales electrónicos es la rapidez que tiene el personal sanitario para conocer las dolencias de un paciente porque toda la información necesaria está disponible en línea y pueden mirarla sin tener necesidad de buscar en enormes formularios de papel. Los médicos de urgencias son unos de los grandes beneficiados por la utilización de los historiales electrónicos puesto que así pueden evitar riesgos innecesarios y retrasos en el tratamiento de los pacientes que necesitan atención inmediata.

El uso de la aplicación Tolven de código abierto, se extiende más allá de los Estados Unidos en regiones como Europa y la costa del Pacífico de Asia.

La rápida utilización del software es un reflejo directo de la calidad y amplitud de los productos y servicios que se han introducido en los últimos tres años. La plataforma del programa y el marco de aplicación proporcionan un conjunto completo de servicios técnicos que se puede utilizar para desplegar de forma sencilla herramientas basadas en los estándares de salud.

En definitiva, las aplicaciones Tolven son soluciones basadas en Web a las que se puede acceder utilizando las últimas versiones de distintos navegadores. El administrador del sistema proporcionará a los usuarios una dirección Web para que pueda acceder a las distintas aplicaciones de forma segura. El programa también proporciona un entorno de demostración para que pueda ser utilizado con fines didácticos (tolven.org, 2017).

## *2.20 TORCH*

La aplicación TORCH (*Trusted Open source Records for Care and Health*) está desarrollada en lenguaje Python con lo que puede ser ejecutada en distintas plataformas con la única salvedad de tener instalado el intérprete correspondiente.

Una de las principales características de la aplicación consiste en el uso de las plantillas con el fin de completar la información de un paciente de manera más rápida. El uso de un conjunto de dichas plantillas bien diseñadas hace que el profesional de la salud pueda generar con rapidez y precisión un historial del paciente mientras le pregunta sus dolencias.

**Funcionamiento**

Antes de crear plantillas propias, es aconsejable conocer el funcionamiento completo del programa y de las distintas plantillas generadas por defecto que vienen instaladas con el programa. Para comenzar a conocer el manejo del programa el primer paso será mirar la selección de registros de pacientes que aparecen en la pantalla principal de TORCH. Luego se puede hacer un listado de todos los registros de los pacientes o la búsqueda de un enfermo si tiene varios ya ha introducidos. Para abrir en una ventana el historial de un paciente, basta con hacer clic en el enlace del apellido.

Las plantillas principales son creadas a partir de otras plantillas ya definidas. Una plantilla es un objeto que contiene uno de los conjuntos de datos siguientes:

- Texto: sólo aparece en la plantilla generada. Se utiliza para los títulos o descripciones.
- Los botones de radio: se utiliza para seleccionar una opción de un grupo.
- Casillas de verificación: se utiliza para seleccionar uno o más elementos de un grupo.
- Cuadro de entrada de texto: se utiliza para introducir texto o números que se muestran en la nota.
- Nota de texto: se utiliza para añadir texto a la nota que no se muestra en la plantilla.
- Llamada a un script: una llamada a cualquier tipo de script que devuelve los resultados a la plantilla.

- Las plantillas importantes no pueden anidarse en otras grandes plantillas.

Una plantilla puede ser tan simple como un "*if*" de selección o tan complicado como una selección de varios elementos de una lista de muchos artículos.

Si crea una referencia circular, haciendo referencia a una plantilla que apunta a la plantilla actual, el sistema constantemente tratará de poner los datos en los dos campos y resultará una plantilla inservible. En caso de realizar tal referencia circular se puede reparar editando la plantilla.

Todas las plantillas se crean y editan con el generador de plantillas FreePM. La pantalla del generador de plantillas tiene tres marcos. El marco superior contiene el menú. El marco derecho del cuadro de la nueva plantilla es donde los datos se introducen o donde se editan las plantillas existentes. El marco izquierdo puede contener una lista de todas las plantillas o una lista de sólo las principales plantillas.

**Conexión al servidor**

Lo primero que debe hacer es acceder a su servidor utilizando el identificador de usuario. Para ello, utilizando en un navegador la dirección http://NombreDelServidorTORCH:9080/manage accederemos a la aplicación. Al realizar esto se debe obtener un cuadro de inicio de sesión HTTP (*Hyper-Text Transfer Protocol*). A partir de aquí se debe introducir su identificador de usuario y su contraseña.

El usuario "*admin*" se crea durante la instalación donde deberá configurar la contraseña de inicio. En caso de no recordar dicha clave, se puede cambiar para el usuario que se quiera haciendo clic en el nombre de usuario. El usuario "*admin*" se trata básicamente de un usuario de emergencia y no puede ser propietario de ningún objeto.

El usuario "*torch_admin*" viene con una contraseña muy simple y básica "abc123". Cada copia de la aplicación TORCH tiene este mismo usuario y

contraseña. Por lo tanto, se debe cambiar la contraseña de este usuario de inmediato. Se debe hacer clic en el nombre de usuario y rellenar los dos campos de la contraseña. Luego se hace clic en el botón "Cambiar". Después se cierra el navegador y luego iniciar la sesión utilizando el identificador de usuario y contraseña nueva para el usuario "*torch_admin*". NO se debe eliminar este usuario. Este usuario posee todos los objetos del programa TORCH. Se puede estropear todo el trabajo de la instalación si lo elimina (launchpad.net/trusted, 2017).

## 2.21 VistA

*Veterans Health Information Systems and Technology Architecture* (VistA) tiene como objetivo prioritario la creación de una aplicación de historiales médicos electrónicos. Esta aplicación fue desarrollada para el sistema médico de Estados Unidos conocido como *Veterans Health Administration* (VHA). En 2003 el VHA era el mayor sistema médico en Estados Unidos y prestaba atención a más de 4 millones de posibles pacientes, empleaba 180.000 médicos distribuidos en 163 hospitales y más de 800 clínicas sin tener en cuenta las 135 residencias de ancianos. Hoy en día la mitad de los hospitales de Estados Unidos utilizan VistA.

Debido a la gran cantidad de datos necesarios para el buen funcionamiento del sistema sanitario VHA, desde 1985 se comenzó a informatizar diferentes departamentos con el fin de realizar una mejor gestión de los datos. Inicialmente se llamó *Decentralized Hospital Computer Program* (DHCP) y ha tenido tanto éxito que en 1995 fue galardonado con el premio *Computerworld Smithsonian* para el mejor uso de Tecnologías de la Información aplicadas a la medicina.

La aplicación VistA es compatible tanto para ambulatorios como para grandes centros de hospitalización e incluye varias mejoras importantes en el sistema DHCP original. El cambio más significativo es una interfaz gráfica de usuario para los hospitales conocido como *Computerized Patient Record System* (CPRS), que fue lanzado en 1997. Además, VistA incluye la entrada de pedidos computarizados, código de barra de la administración de medicamentos, la

prescripción electrónica y las directrices clínicas. El ahorro de trabajo es considerable puesto que se ha comprobado que mejora la eficiencia del trabajo médico en un 6 % por año.

CPRS proporciona una interfaz cliente-servidor que permite a los proveedores de atención médica revisar y actualizar el registro médico de un paciente. Esto incluye la capacidad de realizar pedidos, incluidos los medicamentos, los procedimientos especiales, rayos X, las intervenciones de enfermería, las dietas y pruebas de laboratorio. CPRS proporciona flexibilidad en una amplia variedad de configuraciones de forma que es orientado a eventos, tiene una interfaz del estilo de Windows y por tanto se presenta a un amplio espectro de trabajadores del campo de la salud.

El programa VistA es una aplicación de software libre que consta de una colección de cerca de 100 módulos de software integrados. VistA fue desarrollado utilizando el lenguaje M o MUMPS (*Massachusetts General Hospital Utility Multi-Programming System*) para el uso de las bases de datos. Actualmente la mayoría de los sistemas de VistA se ejecutan en la versión patentada de InterSystems, aunque se ha desarrollado una versión libre utilizando el motor de fuente abierta MUMPS para la interacción con la base de datos. La versión libre está orientada principalmente para ordenadores Linux y Unix. Existe un módulo libre de código abierto de M / Gateway llamado MGWSI que ha sido desarrollado para actuar como una pasarela entre GT.M (versión libre para la gestión de la base de datos) y Caché (versión propietaria para la gestión de la base de datos) utilizando herramientas de programación como PHP, ASP, .NET, o Java, con el fin de crear una interfaz basada en Web.

La asociación VHA tiene un proyecto piloto en curso, conocido como *HealtheVet* (VHE) que prevé la próxima generación de VistA, con la modernización de las capacidades de la base de datos y las interfaces.

El desarrollo es continuo, puesto que de forma ilimitada se realizan las actualizaciones necesarias (500-600 parches por año) y se ofrecen como software de dominio público. Esto fue hecho por el gobierno de los Estados Unidos en un esfuerzo por hacer que VistA esté disponible por un bajo coste para los hospitales

no gubernamentales y otras entidades de atención médica. Universidades, como la de California y la de Texas, así como organizaciones sin ánimo de lucro como WorldVistA, han trabajado para ampliar y mejorar la aplicación VistA.

**Especificaciones técnicas**

La aplicación VistA de imágenes es un sistema coordinado para comunicarse con las imágenes de radiología y otros tipos de imágenes como electrocardiogramas, muestras patológicas y documentos escaneados que son añadidos al sistema de historiales médicos de VistA. Este tipo de integración de la información en un historial médico es fundamental para la utilización eficiente. VistA de imágenes se ha hecho de código libre y está disponible en el dominio público para uso tanto privado como en hospitales públicos. Se puede utilizar de forma independiente o integrada en el sistema electrónico de registros médicos de VistA.

El módulo VistAWeb es una aplicación Web utilizada para revisar la información del paciente que se encuentra en el sistema de gestión de historiales médicos Vista. En gran medida, VistAWeb es el módulo encargado de reflejar los informes médicos. Hay tres formas de acceder a VistAWeb, aunque la principal está disponible mediante la adición al menú Herramientas y puede ser seleccionada como el método predeterminado de recuperación de datos. VistAWeb es compatible con HL7, estándar utilizado en la mayoría de software de salud. El diseño de VistAWeb utiliza principios de arquitectura de n niveles, donde VistAWeb representa el nivel de presentación. El proceso de capa de negocio está representado por varios componentes que VistAWeb utiliza para acceder al nivel de datos. Los componentes de procesos de negocio incluyen elementos tales como páginas de código subyacente (una función de programación del lenguaje Microsoft.NET), *Medical Domain Objects* (MDO) y una colección de otros componentes reutilizables. El nivel de datos cuenta con múltiples fuentes de datos, tales como VistA, XML y numerosas bases de datos relacionales con SQL (por ejemplo, Oracle, Microsoft SQL Server y Microsoft Access). Aunque algunas páginas de código subyacente no interactuar con una base de datos con SQL y

XML, el resto de los datos de nivel de las interacciones tienen lugar en componentes reutilizables, tales como MDO. La Figura 2 muestra, en un alto nivel, la interacción de los diferentes niveles de aplicación (va.gov/vista_monograph, 2017).

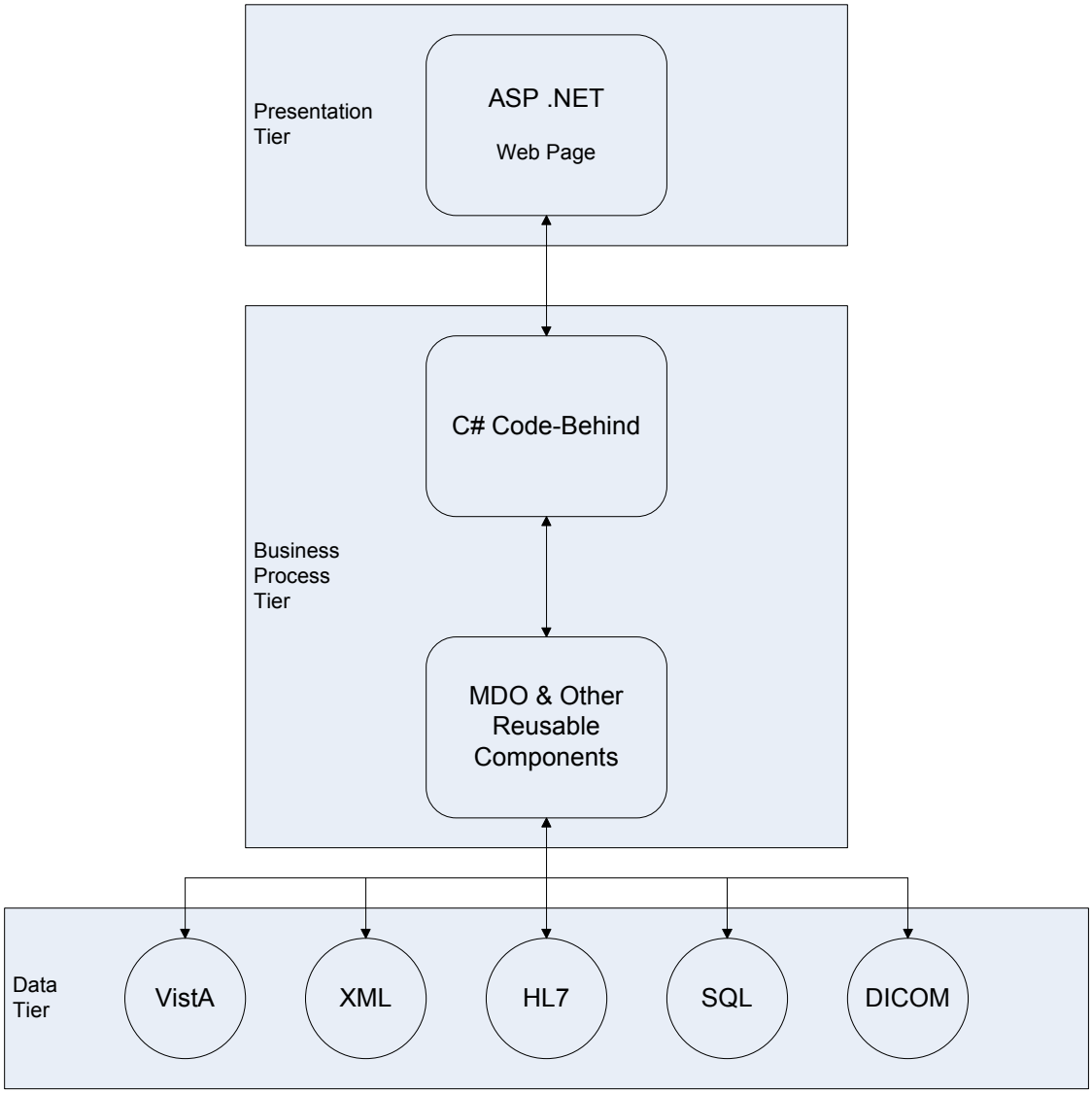


**Figura 2. Interacción entre las capas de VistAWeb.**

## 2.22 ZEPRS

A principios de 2001 un equipo de médicos de la Universidad de Alabama en Birmingham concibió la idea para la implementación de una aplicación de historiales electrónicos que se utilizará inicialmente en Zambia.

Los integrantes del proyecto solicitaron propuestas para el diseño técnico y la aplicación del sistema a varias empresas privadas del sector de la información antes de adjudicar un contrato. Una vez decidida la empresa encargada de parte de la implementación, se acordaron las bases del proyecto cuyo rasgo más significativo era el desarrollo del mismo sin ánimo de lucro. Otros socios del proyecto han sido; el "*Center for Infectious Disease Research in Zambia*" (CIDRZ) y el distrito de salud de Lusaka.

**Objetivos**

El gestor de historiales médicos ZEPRS (*Zambia Electronic Perinatal Record System*), fue diseñado para mejorar los resultados maternos y perinatales por:

- Mejorar la atención perinatal para las mujeres y la atención posnatal para los recién nacidos por:
  - Promover las prácticas de cuidado.
  - Identificar y documentar los posibles problemas médicos para que los tratamientos eficaces puedan ser administrados.
  - Mejorar la comunicación entre los proveedores y las referencias.
  - Mejorar la supervisión y la evaluación de los resultados.
- La mejora de la eficiencia, integridad, exactitud de la documentación y presentación de informes.

**Los componentes principales**

ZEPRS alcanza estos objetivos mediante las siguientes aportaciones:

- Un sistema de registros electrónicos de pacientes con una base de datos que contiene los historiales clínicos y que son compartidas entre distintas instalaciones.
- Un sistema que guía a los médicos a través del estándar de atención de Zambia.
- Un sistema electrónico utilizado por primera vez por los médicos durante el transcurso de la atención al paciente.
- Un sistema de remisión electrónica para mejorar la eficiencia y la eficacia de las referencias.

**Datos técnicos**

ZEPRS ha sido desarrollado utilizando el lenguaje de programación Java. Otras tecnologías utilizadas son: AJAX (*Asynchronous JavaScript And XML*), Quartz, MySQL y otros. La aplicación tiene su propio sistema de gestión de contenido llamado DynaSite, lo que hace más fácil añadir formularios, diferentes campos y alguna característica más sin necesidad de programación. También se ha desarrollado una versión instalable de forma local del software con la base de datos integrada. (ictedge.org/zeprs, 2017).

# 3. Diseño

Se desea implementar una aplicación para la gestión de datos clínicos que será una interfaz web que gestione una base de datos con la información relativa a los pacientes. Sus funciones básicas serán: crear una entrada para un nuevo paciente, buscar pacientes por diferentes campos, editar campos de un paciente y eliminar un paciente. Las funciones avanzadas estarán relacionadas con la importación de datos de forma automática. Dicha importación se podrá realizar desde diferentes fuentes de datos.

## *3.1 Modelo-Vista-Controlador*

El diseño de la aplicación se realizará utilizando la arquitectura de modelo-vista-controlador (MVC). Este modelo es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado, define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

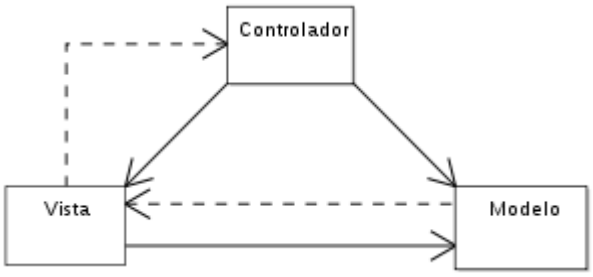


**Figura 3. Modelo-Vista-Controlador.**

De manera genérica, los componentes de MVC se podrían definir como sigue:

El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tanto consultas, como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto, se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

La Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

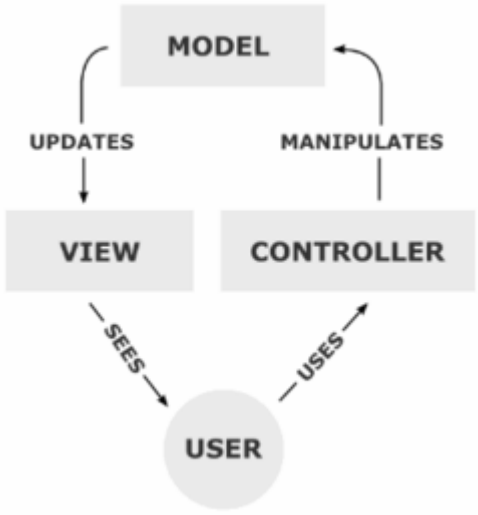


**Figura 4. Interacción del usuario con M-V-C.**

Interacción de los componentes

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control que se sigue generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. Este uso del patrón Observador no es posible en las aplicaciones Web puesto que las clases de la vista están desconectadas del modelo y del controlador. En general el controlador no pasa objetos de dominio (el modelo) a la vista, aunque puede dar la orden a la vista para que se actualice.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

MVC y bases de datos

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos que debe utilizar la aplicación; en líneas generales del MVC dicha gestión corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre la Vista y su correspondiente Controlador de eventos y acceso a datos. MVC no discrimina entre capa de negocio y capa de presentación, pero si separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

Uso en aplicaciones Web

Aunque originalmente MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adaptado como arquitectura para diseñar e implementar aplicaciones web en los principales lenguajes de programación. Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón. Estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre cliente y servidor.

Los primeros frameworks MVC para desarrollo web planteaban un enfoque de cliente ligero en el que casi todas las funciones, tanto de la vista, el modelo y el controlador recaían en el servidor. En este enfoque, el cliente manda la petición de cualquier hiperenlace o formulario al controlador y después recibe de la vista una página completa y actualizada (u otro documento); tanto el modelo como el controlador (y buena parte de la vista) están completamente alojados en el servidor.

## 3.2 Plan tecnológico

El plan tecnológico a seguir en la implementación de la aplicación será: Django, Python y mongoDB. A continuación, se explican las características de cada uno de ellos.

### Django

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–vista–controlador.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

Visión general y características

Al igual que Ruby on Rails, otro popular framework de código abierto, Django se usó en producción durante un tiempo antes de que se liberara al público.

Los orígenes de Django en la administración de páginas de noticias son evidentes en su diseño, ya que proporciona una serie de características que facilitan el desarrollo rápido de páginas orientadas a contenidos. Por ejemplo, en lugar de requerir que los desarrolladores escriban controladores y vistas para las áreas de administración de la página, Django proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página hecha con Django y que puede administrar varias páginas hechas con Django a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno, y proporciona una interfaz para administrar los usuarios y los grupos de usuarios (incluyendo una asignación detallada de permisos).

La distribución principal de Django también aglutina aplicaciones que proporcionan un sistema de comentarios, herramientas para sindicar contenido vía RSS y/o Atom, "páginas planas" que permiten gestionar páginas de contenido sin necesidad de escribir controladores o vistas para esas páginas, y un sistema de redirección de URLs.

Otras características de Django son:

- Un mapeador objeto-relacional.
- Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con Django.
- Una API de base de datos robusta.
- Un sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un despachador de URLs basado en expresiones regulares.
- Un sistema "middleware" para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.
- Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones).

Django también es una plataforma habitual que brinda múltiples herramientas.

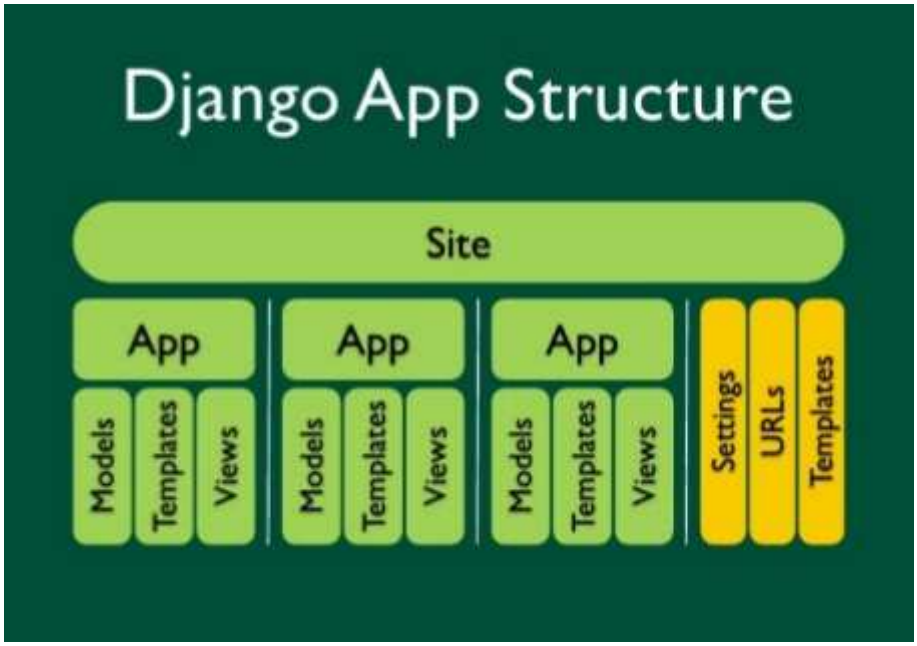


**Figura 5. Estructura de una aplicación Django.**

Arquitectura

Aunque Django está fuertemente inspirado en la filosofía de desarrollo Modelo Vista Controlador, sus desarrolladores declaran públicamente que no se sienten especialmente atados a observar estrictamente ningún paradigma particular, y en cambio prefieren hacer "lo que les parece correcto". Como resultado, por ejemplo, lo que se llamaría "controlador" en un "verdadero" framework MVC se llama en Django "vista", y lo que se llamaría "vista" se llama "plantilla".

Gracias al poder de las capas mediator y foundation, Django permite que los desarrolladores se dediquen a construir los objetos Entity y la lógica de presentación y control para ellos.

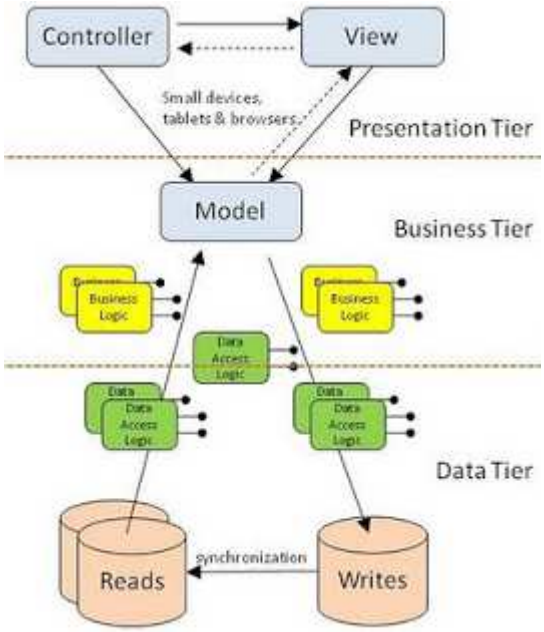


**Figura 6. Arquitectura de una aplicación Django.**

Presentación

Aquí se maneja la interacción entre el usuario y el computador. En Django, ésta tarea la realizan el template engine y el template loader que toman la información y la presentan al usuario (vía HTML, por ejemplo). El sistema de configuración de URLs es también parte de la capa de presentación.

Control

En esta capa reside el programa o la lógica de aplicación en sí. En Django son representados por las views y manipulators. La capa de presentación depende de ésta y a su vez ésta lo hace de la capa de dominio.

Mediator

Es el encargado de manejar la interacción entre el subsistema Entity y foundation. Aquí se realiza el mapeo objeto-relacional a cargo del motor de Django.

Entity

El subsistema entity maneja los objetos de negocio. El mapeo objeto-relacional de Django permite escribir objetos de tipo entity de una forma fácil y estándar.

Foundation

La principal tarea del subsistema foundation es la de manejar a bajo nivel el trabajo con la base de datos. Se provee soporte a nivel de foundation para varias bases de datos y otras están en etapa de prueba.

# Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Características y paradigmas

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

## MongoDB

MongoDB (de la palabra en inglés "*humongous*" que significa enorme) es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

El desarrollo de MongoDB empezó en octubre de 2007 por la compañía de software 10gen. Ahora MongoDB es una base de datos lista para su uso en producción y con muchas características. Esta base de datos se utiliza mucho en la industria, contando con implantaciones en grandes empresas.

El código binario está disponible para los sistemas operativos Windows, Linux, OS X y Solaris.

### Características principales

Lo siguiente es una breve descripción de las características principales de MongoDB:

### Consultas Ad hoc

MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares. Las consultas pueden devolver un campo específico del

documento, pero también puede ser una función JavaScript definida por el usuario.

Indexación

Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios. El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales.

Replicación

MongoDB soporta el tipo de replicación primario-secundario. Cada grupo de primario y sus secundarios se denomina replica set. El primario puede ejecutar comandos de lectura y escritura. Los secundarios replican los datos del primario y sólo se pueden usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras. Los secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

Balanceo de carga

MongoDB se puede escalar de forma horizontal usando el concepto de "shard". El desarrollador elige una clave de sharding, la cual determina cómo serán distribuidos los datos de una colección. Los datos son divididos en rangos (basado en la clave de sharding) y distribuidos a través de múltiples shard. Cada shard puede ser una réplica set. MongoDB tiene la capacidad de ejecutarse en múltiples servidores, balanceando la carga y/o replicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware. La configuración automática es fácil de implementar bajo MongoDB y se pueden agregar nuevos servidores a MongoDB con el sistema de base de datos funcionando.

Almacenamiento de archivos

MongoDB puede ser utilizado como un sistema de archivos, tomando la ventaja de la capacidad que tiene MongoDB para el balanceo de carga y la

replicación de datos utilizando múltiples servidores para el almacenamiento de archivos. Esta función se llama GridFS15 y es más bien una implementación en los drivers, no en el servidor, por lo que está incluida en los drivers oficiales que la compañía de MongoDB desarrolla. Estos drivers exponen funciones y métodos para la manipulación de archivos y contenido a los desarrolladores. En un sistema con múltiples servidores, los archivos pueden ser distribuidos y replicados entre los mismos y de una forma transparente, de esta forma se crea un sistema eficiente que maneja fallos y balanceo de carga.

Agregación

MongoDB proporciona un framework de agregación que permite realizar operaciones similares a las que se obtienen con el comando SQL "GROUP BY". El framework de agregación está construido como un pipeline en el que los datos van pasando a través de diferentes etapas en los cuales estos datos son modificados, agregados, filtrados y formateados hasta obtener el resultado deseado. Todo este procesado es capaz de utilizar índices si existieran y se produce en memoria. Asimismo, MongoDB proporciona una función MapReduce que puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación.

Ejecución de JavaScript del lado del servidor

MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

Fragmentación (Sharding)

Si estas desarrollando un servicio que se va haciendo popular o los niveles de acceso a base de datos son cada vez más altos, empezarás a notar que tu base de datos está siendo martillada por el exceso de tráfico y tu servidor esté sufriendo por los altos niveles de procesamiento continuo y te podrías ver en la necesidad de actualizar tu infraestructura para soportar la demanda.

Aquí entra en juego la fragmentación, es el modo en el que hacemos nuestra base de datos escalable. En lugar de tener una colección en una base de datos, la pondríamos en varias bases de datos distribuidas, de modo que, a la hora de consultar los datos de dicha colección, los recuperemos como si de una única base de datos se tratase. Mongo se encargará de averiguar de manera transparente en que base de datos se encuentran los datos.

Los fragmentos estarán formados por replica set, de modo que, si creamos tres fragmentos, cada uno de los cuales tiene una réplica set con tres servidores, estaríamos hablando de un total de nueve servidores.

Si hacemos consultas, estas se realizarán de manera distribuida a través de un módulo enrutador llamado "MongoS" que mantendrá un pequeño pull de conexiones a los distintos hosts. Para conocer en que fragmento debe consultar para recuperar datos de una colección ordenada, se utilizan rangos y shard_key, de modo que se trocea la colección en rangos y les asigna un id (shard_key), que puede ser una parte del propio documento, y se distribuye en los fragmentos (replica set). De modo que cuando se consulte la colección debemos proporcionar el "shard_key".

Principales problemas

No implementa las propiedades ACID

El no implementar las propiedades ACID genera que la base de datos no asegure la durabilidad, la integridad, la consistencia y el aislamiento requeridos obligatoriamente en las transacciones. Es posible que en futuras versiones esto se solucione.

Sobre la base de este punto se detallan los cuatro siguientes:

Problemas de consistencia

Las lecturas estrictamente consistentes ven versiones obsoletas de documentos, también pueden devolver datos incorrectos de lecturas que nunca deberían haber ocurrido.

*Bloqueo a nivel de documento*

MongoDB bloquea la base de datos a nivel de documento ante cada operación de escritura. Sólo se podrán hacer operaciones de escritura concurrentes entre distintos documentos.

Las escrituras no son durables ni verificables

MongoDB retorna cuando todavía no se ha escrito la información en el espacio de almacenamiento permanente, lo que puede ocasionar pérdidas de información. En MongoDB 2.2 se cambia el valor por defecto para escribir en al menos una réplica, pero esto sigue sin satisfacer la durabilidad ni la verificabilidad.

Escalabilidad

En el caso de necesitar incrementar el tamaño de la base de datos mongoDB debido a un incremento del número de pacientes, se realizará dividiendo la información en shard cada uno de ellos formando por un replica set para tener asegurarnos que no se pierdan los datos.

El esquema de la implementación será de la siguiente manera:

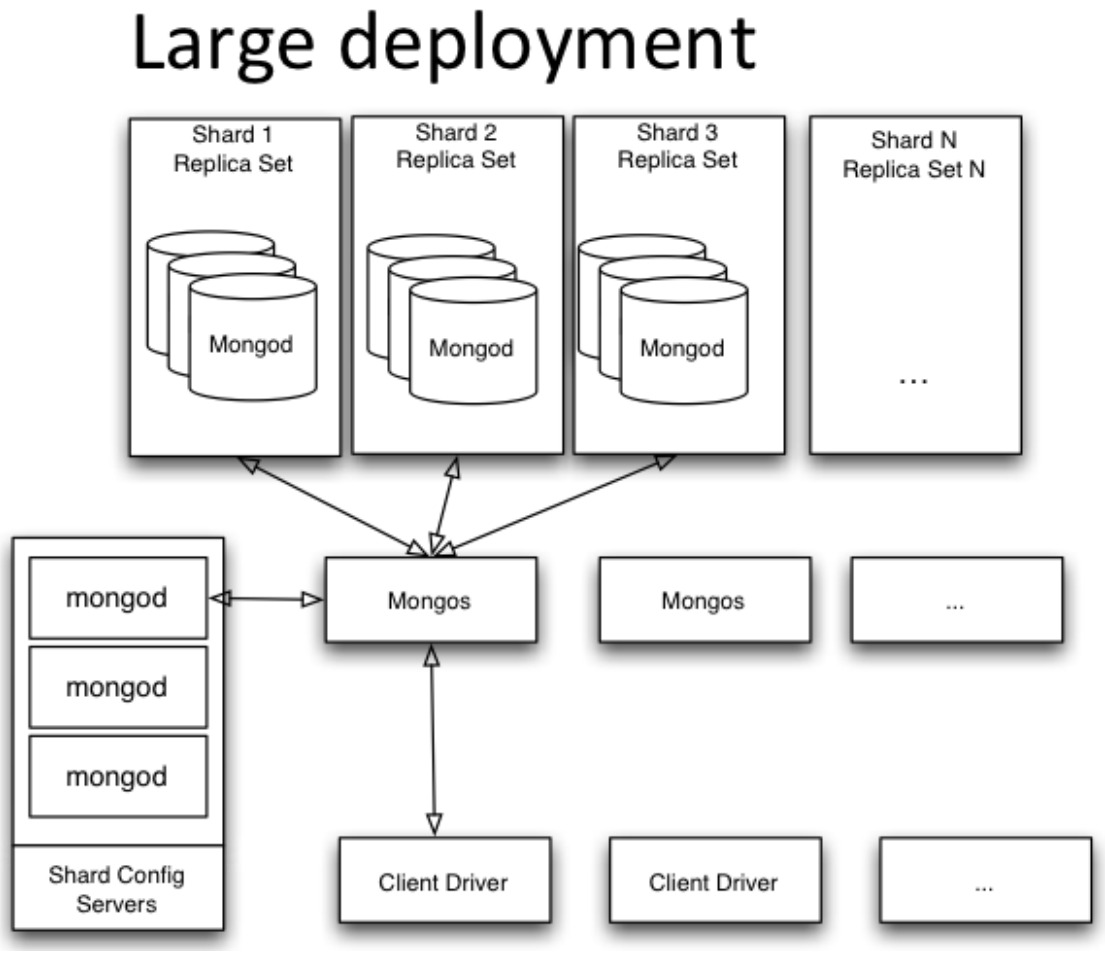


**Figura 7. Esquema de implementación MongoDB.**

# 4. Implementación

NOTA: Todos los datos relativos a pacientes utilizados en las simulaciones no son reales.

El menú principal de la aplicación consta de los siguientes apartados:

- Nuevo paciente.
- Buscar paciente.
- Ver todos los pacientes.

A continuación, se presenta un mapa de las funcionalidades que tiene implementadas la aplicación:

- Nuevo paciente.
    - Visualizar
        - Editar
        - Borrar
    - Editar
        - Visualizar
        - Editar
        - Borrar
    - Eliminar
- Buscar paciente.
- Ver todos los pacientes.

Ahora se van a explicar con ejemplos las funcionalidades más importantes de la aplicación. En primer lugar, se observa el menú principal del programa:


**Figura 8. Página principal de la aplicación.**

A continuación, se muestran ejemplos de las funcionalidades consideradas más importantes de la aplicación para poder conocer mejor su funcionamiento.

Nuevo paciente

Esta funcionalidad nos permitirá agregar un nuevo paciente a la base de datos con los campos siguientes:

- Nombre.
- Apellido 1.
- Apellido 2.
- Nhc.
- Icgc.
- Notes.

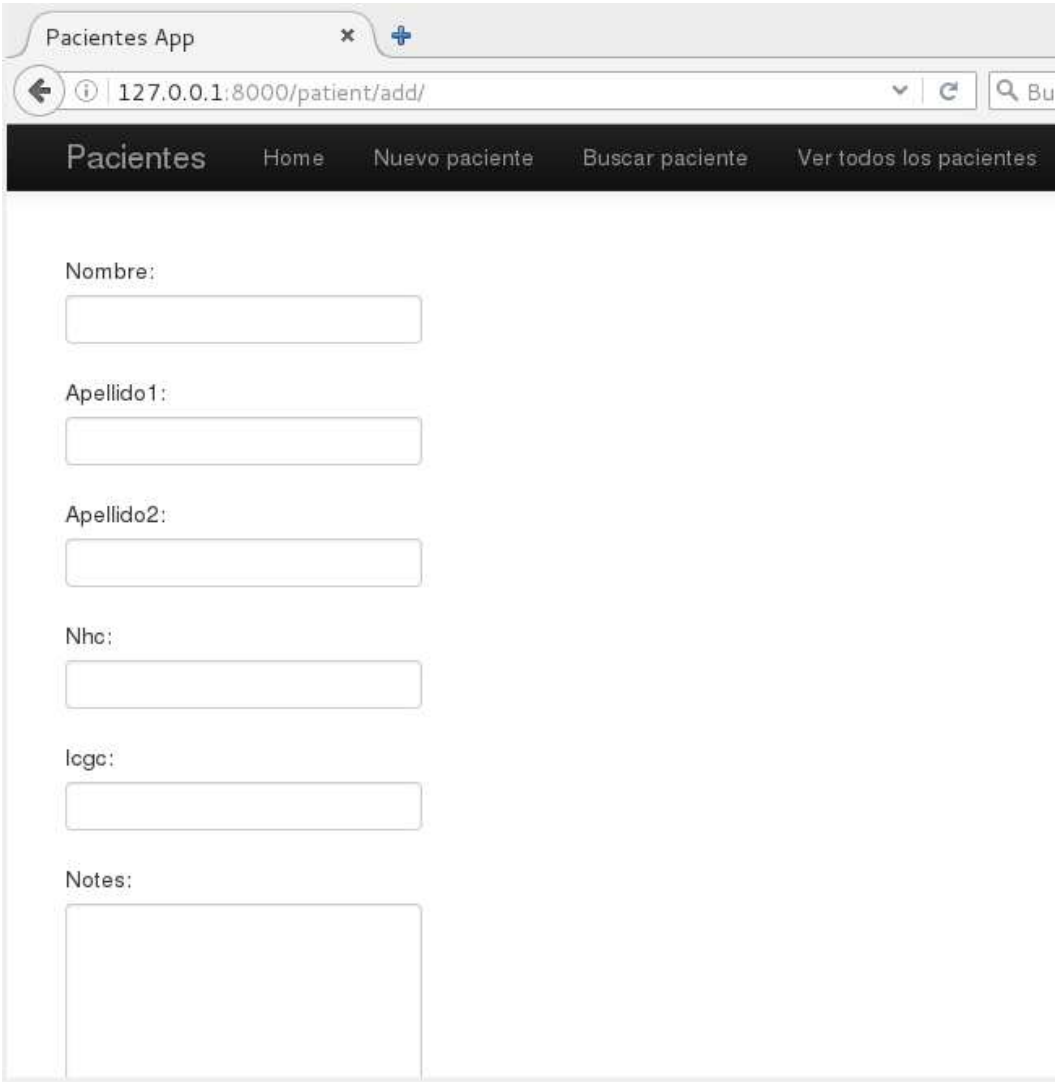


**Figura 9. Página para añadir un nuevo paciente.**

Una vez insertado en la base de datos un nuevo paciente, se tiene la posibilidad de: visualizar, editar o eliminar dicho paciente. A continuación, se muestran las imágenes que muestran dichas funcionalidades:

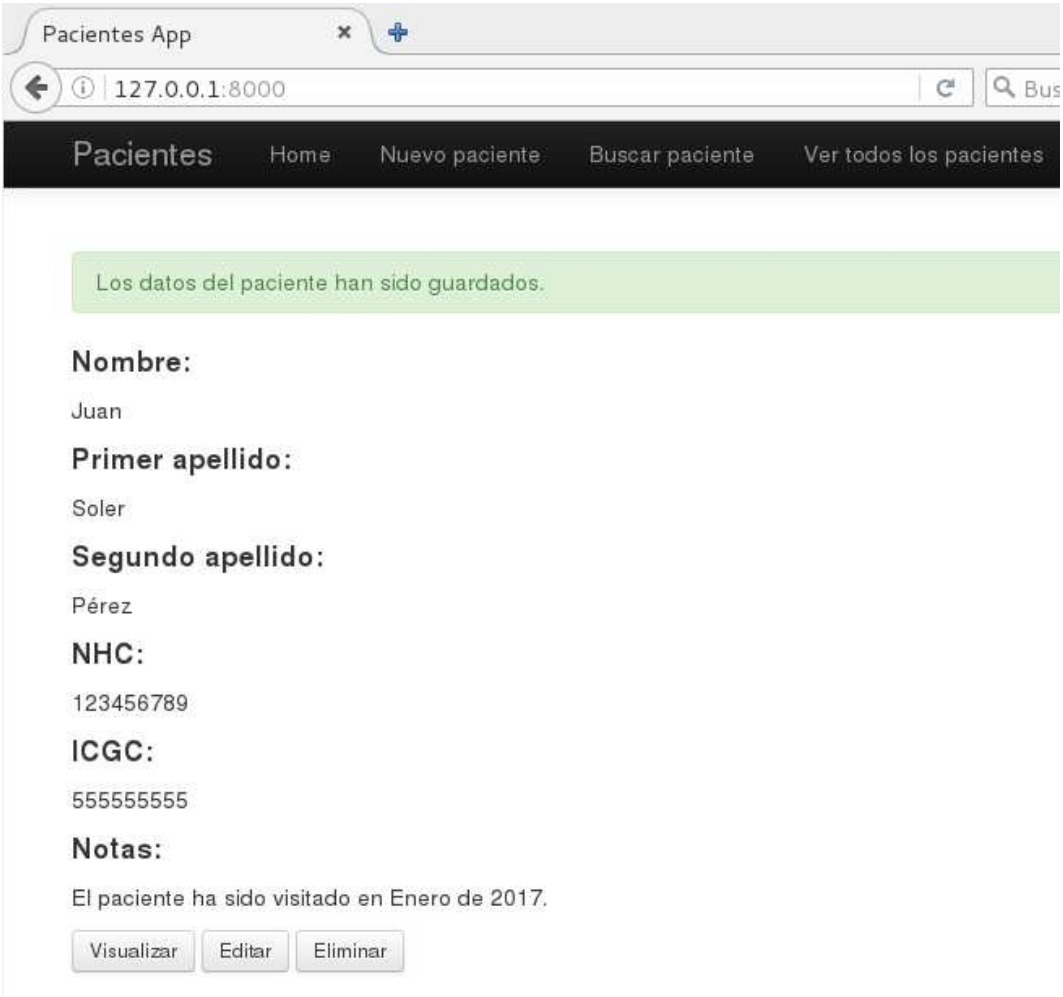


**Figura 10. Página de nuevo paciente añadido.**

Visualizar

Dentro de la funcionalidad de visualizar se encuentran las opciones de editar y borrar que se muestran a continuación:


**Figura 11. Página de visualizar paciente.**

La funcionalidad de editar paciente, permite cambiar cualquiera de los campos relativos a un paciente, tal y como se puede ver a continuación:

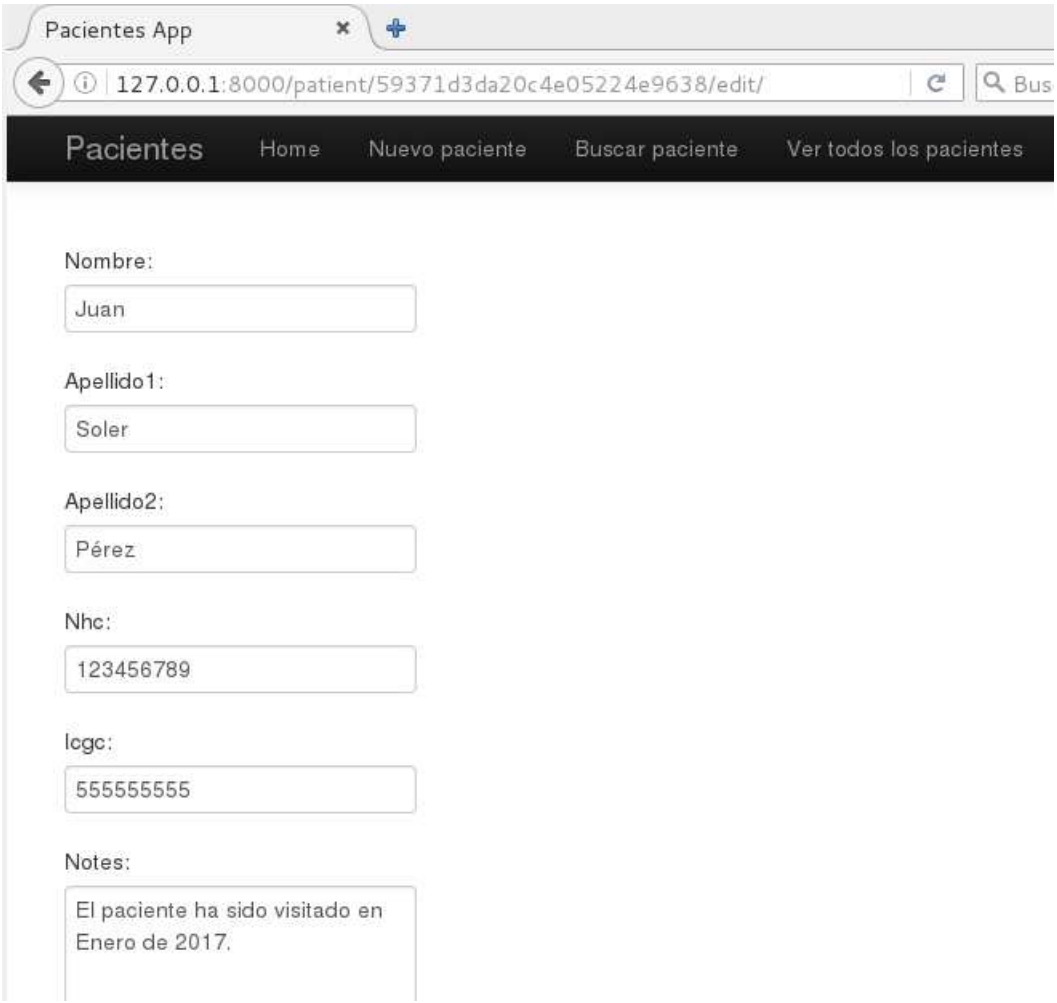


**Figura 12. Página de editar paciente.**

Una vez editado un paciente y guardado se puede ver que la información es actualizada y la aplicación permite de nuevo visualizar, editar o eliminar el paciente modificado tal y como se ve en la imagen que se muestra a continuación:


**Figura 13. Página de confirmación de edición de un paciente.**

Como se puede ver en la siguiente imagen, en la base de datos MongoDB se observa que se realizan correctamente tanto las nuevas inserciones de pacientes como las modificaciones.

```
> db.patient.find()
{ "_cls" : "Patient", "_id" : ObjectId("59371d3da20c4e05224e9638"), "_types" : [
 "Patient" ], "apellido1" : "Soler", "apellido2" : "Pérez", "date_modified" : ISO
Date("2017-06-06T23:23:09.794Z"), "icgc" : "555555555", "is_published" : true, "n
hc" : "123456789", "nombre" : "Juan", "notes" : "El paciente ha sido visitado en
Enero de 2017.\r\nEl paciente ha sido visitado en Marzo de 2017." }
```

**Figura 14. Paciente insertado en MongoDB.**

Eliminar

Una vez mostradas las funcionalidades visualizar y editar queda únicamente la función de eliminar un paciente que lo que realiza es un borrado de dicho paciente de la base de datos, como se puede apreciar en las siguientes imágenes:

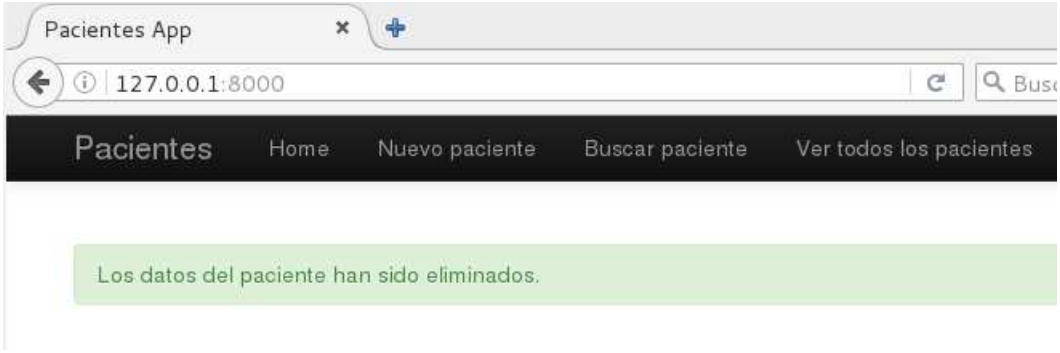


**Figura 15. Página de confirmación de paciente eliminado.**

Se observa que el borrado en MongoDB se ha realizado correctamente:

```
> db.patient.find()
{ "_cls" : "Patient", "_id" : ObjectId("59371d3da20c4e05224e9638"), "_types" : [
 "Patient" ], "apellido1" : "Soler", "apellido2" : "Pérez", "date_modified" : ISO
Date("2017-06-06T23:23:09.794Z"), "icgc" : "555555555", "is_published" : true, "n
hc" : "123456789", "nombre" : "Juan", "notes" : "El paciente ha sido visitado en
Enero de 2017.\r\nEl paciente ha sido visitado en Marzo de 2017." }
> db.patient.find()
>
```

**Figura 16. Paciente eliminado en MongoDB.**

Buscar paciente

Esta funcionalidad nos permitirá buscar un nuevo paciente en la base de datos por cualquiera de los campos siguientes:

- Nombre.
- Apellido 1.
- Apellido 2.
- Nhc.
- Icgc.
- Notes.

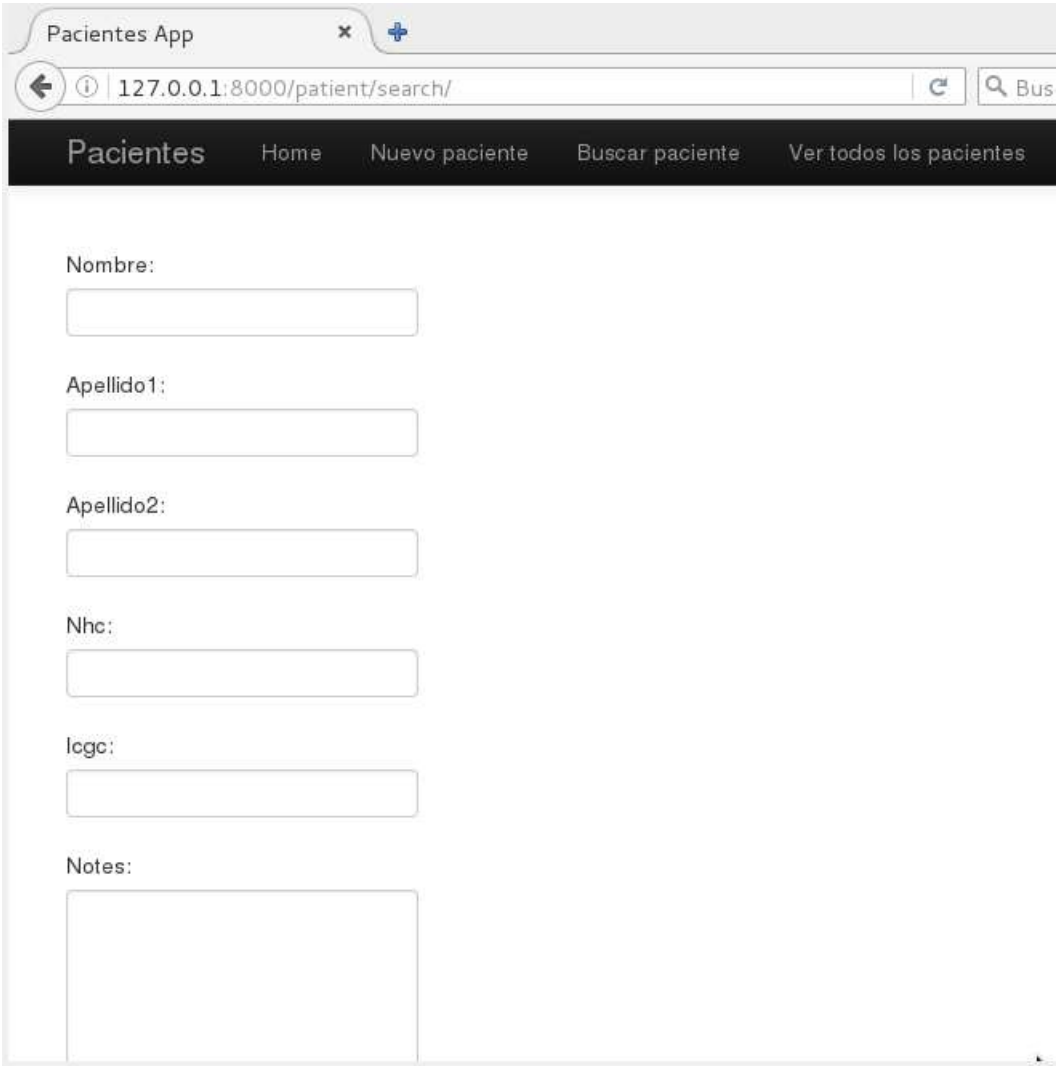


**Figura 17. Página de búsqueda de paciente por diferentes campos.**

Ver todos los pacientes

Esta funcionalidad nos permitirá mostrar todos los pacientes insertados en la base de datos. (Se han insertado dos pacientes a modo de ejemplo).

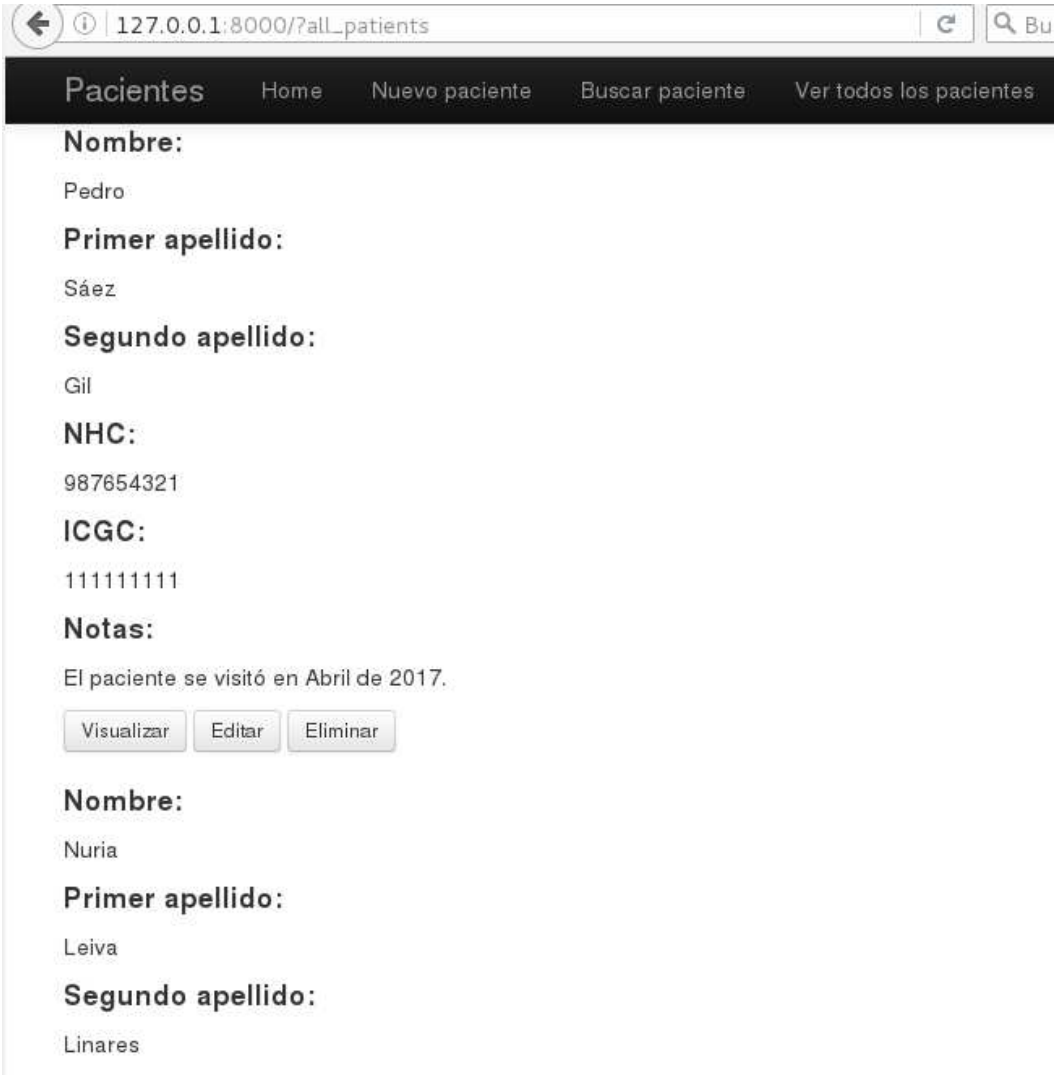


**Figura 18. Página de visualización de todos los pacientes.**

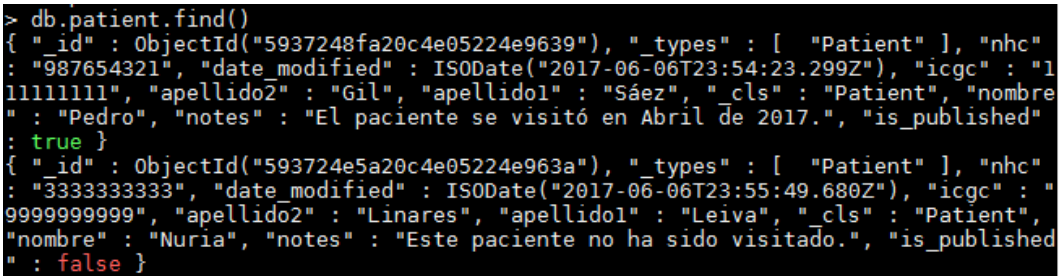


**Figura 19. Pacientes insertados en MongoDB.**

# 5. Conclusiones y posibles futuras mejoras

Para dar soporte a los EHR se adoptan sistemas integrados, que deben caracterizarse por ser sistemas sólidos, modulares, configurables, versátiles, seguros e integrales. En este proyecto final de carrera se revisan un conjunto de aplicaciones que permiten una adecuada gestión de EHR, cuya característica común es que son de código abierto.

Se ha realizado un análisis de los principales programas de software libre a nivel mundial. La elección de analizar dichas aplicaciones de gestión sanitaria se debe a su importancia en cuanto a la cantidad de pacientes gestionados, número de lugares en los que se utiliza la aplicación, heterogeneidad de las clínicas en las que se utilizan, mayor nivel de desarrollo del software, cantidad de funcionalidades ofrecidas a los profesionales de la medicina, tecnología del núcleo, lenguaje de programación en el que se desarrolla la interfaz de la aplicación, lenguaje de interacción con la base de datos, etc. Desgraciadamente la información disponible para hacer la revisión de dichas aplicaciones no es siempre accesible y sólo ha sido posible hacer un análisis de ciertos aspectos.

Finalmente con el minucioso análisis llevado a cabo de las distintas aplicaciones de software libre para la gestión de historiales médicos, se ha desarrollado una aplicación de acuerdo a las necesidades del servicio de Hematopatología del Hospital Clínic de Barcelona.

Por otra parte, la elección del plan tecnológico se ha realizado en base a las tecnologías utilizadas por las diferentes aplicaciones de software libre para la gestión de pacientes y por otra parte teniendo en cuenta las tecnologías actuales más eficientes para las necesidades del proyecto desarrollado.

Por último, en cuanto a líneas futuras, se puede comentar la posibilidad de cifrar el almacenamiento de la máquina virtual para ofrecer mayor seguridad a los datos. También se puede estudiar la posibilidad de conexión fuera de la máquina virtual con un montaje sobre protocolo seguro https en un servidor web como puede ser apache.

# Anexo I

## *Instalación y configuración de la plataforma*

### Descripción general

Estas instrucciones son para instalar manualmente la plataforma. También se indicarán los pasos para instalar todo el software necesario para el correcto funcionamiento de la aplicación. A continuación, se describen los pasos a seguir para la completa instalación de la infraestructura.

### Pasos a seguir

1. Instalación de VirtualBox y creación de una nueva máquina virtual.
2. Instalación de Linux Debian.
3. Instalación de MongoDB.
4. Instalación de Django.
5. Arrancar la aplicación.

**Requisitos mínimos**

En un servidor no productivo con el objetivo de realizar pequeñas pruebas se recomienda: procesador de 1 GHz, 512 MB de memoria RAM, 40 GB de disco duro. Por el contrario, para su uso en producción, se recomienda al menos: uno o dos procesadores de 1,5 GHz, 2 GB de memoria RAM, y 150 GB de espacio en disco con las correspondientes instalaciones y configuraciones para realizar copias de seguridad.

**Instalación de VirtualBox y creación de una nueva máquina virtual**

1. Descargar el instalador de la última versión de VirtualBox y ejecutarlo.
2. Crear una nueva máquina tipo Linux x64 con los requisitos mínimos anteriormente descritos.

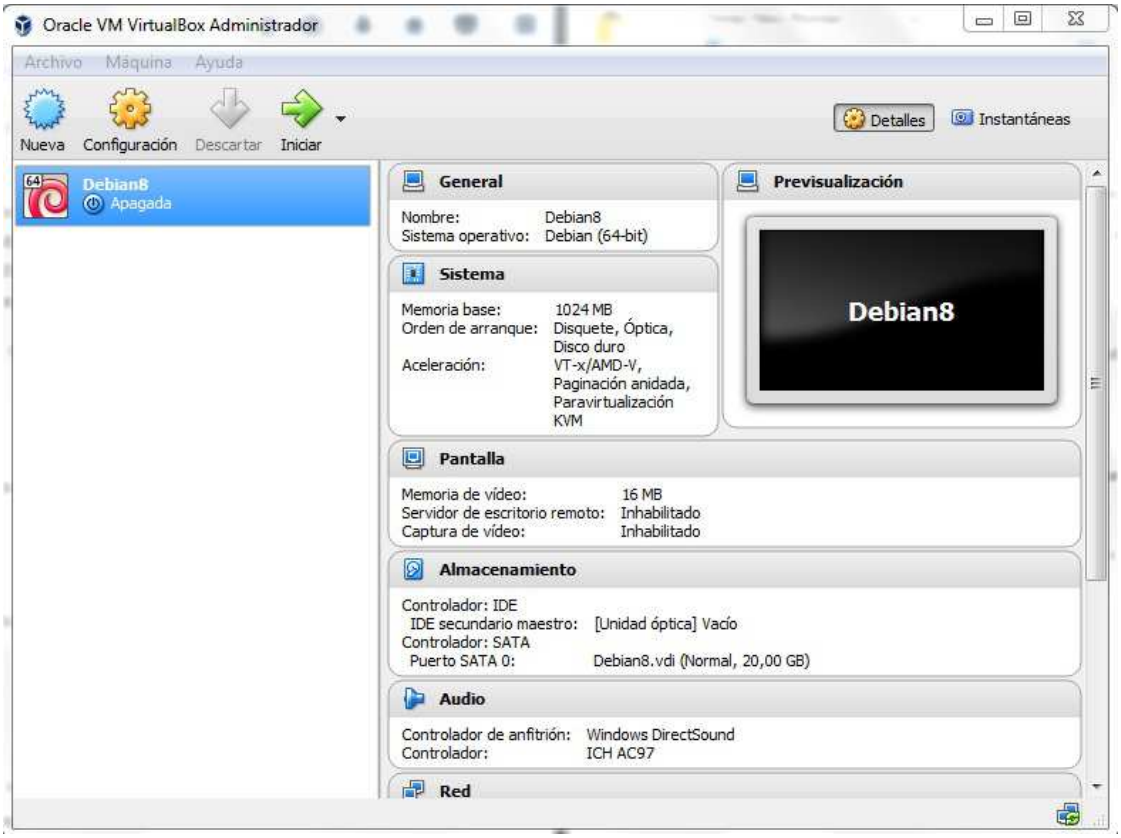


**Figura 20. Máquina Debian en VirtualBox.**

### Instalación de Linux Debian

1. Descargar la última versión estable de Linux Debian.
2. Configurar la máquina virtual para arrancar desde CD-ROM con la ISO Linux.
3. Seguir los pasos del instalador de Linux.

   NOTA:

   Para hacer login en la máquina entregada utilizar las siguientes credenciales:

   Usuario: allabal Password: a4s5d6we

   Usuario: root Password: mu9ny8

### Instalación de MongoDB

1. Abrir un terminal dentro de la máquina Debian y como root ejecutar:

   apt-get install mongodb
2. Comprobar que el servicio está funcionando.

```
root@shaka:~# service mongodb status
● mongodb.service - An object/document-oriented database
   Loaded: loaded (/lib/systemd/system/mongodb.service; enabled)
   Active: active (running) since mié 2017-06-07 19:31:36 CEST; 26min ago
     Docs: man:mongod(1)
 Main PID: 387 (mongod)
   CGroup: /system.slice/mongodb.service
           └─387 /usr/bin/mongod --config /etc/mongodb.conf

jun 07 19:31:36 shaka systemd[1]: Started An object/document-oriented database.
jun 07 19:31:37 shaka mongod[387]: all output going to: /var/log/mongodb/mon...og
Hint: Some lines were ellipsized, use -l to show in full.
root@shaka:~# mongo --version
MongoDB shell version: 2.4.10
root@shaka:~#
```

**Figura 21. Comprobación MongoDB.**

### Instalación de Django

1. Abrir un terminal dentro de la máquina Debian y como root ejecutar:

   apt-get install python-django

2. Comprobar la versión de Django instalada.

```
root@shaka:~# django-admin version
1.7.11
```

**Figura 22. Comprobación versión Django**

### Arrancar la aplicación

1. Para arrancar la aplicación desarrollada, abrir un terminal dentro de la máquina Debian y con usuario allabal ir al directorio:

   /home/allabal/TFG

   para posteriormente ejecutar:

   python manage.py runserver

   En la imagen se observa la puesta en marcha del programa:

```
allabal@shaka:~$ cd TFG
allabal@shaka:~/TFG$ pwd
/home/allabal/TFG
allabal@shaka:~/TFG$ ls
manage.py  Patients  Patients_App
allabal@shaka:~/TFG$ python manage.py runserver
Performing system checks...

System check identified some issues:

WARNINGS:
?: (1_6.W001) Some project unittests may not execute as expected.
        HINT: Django 1.6 introduced a new default test runner. It looks like this
 project was generated using Django 1.5 or earlier. You should ensure your tests
are all running & behaving as expected. See https://docs.djangoproject.com/en/dev
/releases/1.6/#new-test-runner for more information.

System check identified 1 issue (0 silenced).
June 07, 2017 - 20:08:41
Django version 1.7.11, using settings 'Patients.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

**Figura 23. Puesta en marcha de la aplicación.**

2. Una vez arrancada la aplicación, para probarla, desde un navegador se deberá poner la dirección:

   http://127.0.0.1:8000

   Obteniendo como resultado:


**Figura 24. Aplicación en funcionamiento.**

# 6. Bibliografía

Alchin, M. (2013). Pro Django, *Editorial Professional Apress.*

Allen, C., Jazayeri, D., Miranda, J., Biondich, P.G., Mamlin, B.W., Wolfe, B.A., Seebregts, C., Lesh, N., Tierney, W.M., Fraser, H.S. (2007). Experience in implementing the OpenMRS medical record system to support HIV treatment in Rwanda. *Studies in health technology and informatics 129* (Pt 1): 382–386.

Allen, C., Manyika, P., Jazayeri, D., Rich, M., Lesh, N., Fraser, H. (2006). Rapid deployment of electronic medical records for ARV rollout in rural Rwanda. *AMIA Annual Symposium proceedings*: 840.

Biondich, P.G., Anand, V., Downs, S.M., McDonald, C.J. (2003). Using Adaptive Turnaround Documents to Electronically Acquire Structured Data in Clinical Settings. *AMIA Annual Symposium proceedings*: 86–90.

BIRT Web page (2017). Retrieved June 2, 2017 from http://www.eclipse.org/birt/

ClearHealth Web page (2017). Retrieved May 12, 2017 from http://en.wikipedia.org/wiki/ClearHealth

Community Health Information Tracking System Web page (2017). Retrieved May 12, 2017 from http://www.chits.ph/web

Diero, L., Rotich, J.K., Bii, J., Mamlin, B.W., Einterz, R.M., Kalamai, I.Z., Tierney, W.M. (2006). A computer-based medical record system and personal digital assistants to assess and follow patients with respiratory tract infections visiting a rural Kenyan health centre, BMC Med. Inform. Decis. Mak. 6: 21.

Elemental Clinic Web page (2017). Retrieved May 12, 2017 from http://directory.fsf.org/project/elemental_clinc

Fraser, H.S., Biondich, P., Moodley, D., Choi, S., Mamlin, B.W., Szolovits, P. (2005). Implementing electronic medical record systems in developing countries, *Inform Prim Care*. 13(2): 83-95.

Fraser, H.S., Blaya, J., Choi, S.S., Bonilla, C., Jazayeri, D. (2006). Evaluating the impact and costs of deploying an electronic medical record system to support TB treatment in Peru. *AMIA Annual Symposium proceedings*: 264–268.

Fraser, H.S., Jazayeri, D., Nevil, P., Karacaoglu, Y., Farmer, P.E., Lyon, E., Fawzi, M.K., Leandre, F., Choi, S.S., Mukherjee, J.S. (2004). An information system and medical record to support HIV treatment in rural Haiti. *BMJ (Clinical research ed.)* 329 (7475): 1142–1146.

FreeMED Web page (2017). Retrieved May 12, 2017 from http://freemedsoftware.org

FreeMedForms Web page (2017). Retrieved May 12, 2017 from http://www.freemedforms.com/en/start

GNUmed Web page (2017). Retrieved May 12, 2017 from http://wiki.gnumed.de/bin/view/Gnumed

Häyrinen, K., Saranto, K., Nykänen, P. (2008). Definition, structure, content, use and impacts of electronic health records: a review of the research literature. *Int. J. Med. Inform.* 77(5): 291-304.

HealthForge Web page (2017). Retrieved May 12, 2017 from http://healthforge.codeplex.com

Hibernate Web page (2017). Retrieved June 2, 2017 from http://www.hibernate.org

Hinojosa A.P. (2016). Python: paso a paso, *Editorial RA-MA*.

HL7 Web page (2017). Retrieved June 2, 2017 from http://www.hl7.org

Hospital OS Web page (2017). Retrieved May 12, 2017 from http://www.hospital-os.com

HOSxP Web page (2017). Retrieved May 12, 2017 from http://hosxp.net

HTML Web page (2017). Retrieved June 2, 2017 from http://www.w3.org/html

Indivo Web page (2017). Retrieved May 12, 2017 from http://indivohealth.org

Java Web page (2017). Retrieved June 2, 2017 from http://www.java.com/es

Kabir, M.J. (2003). La biblia del Servidor Apache 2, *Editorial Anaya multimedia.*

Littlejohns, P., Wyatt, J.C., Garvican, L. (2003). Evaluating computerised health information systems: hard lessons still to be learnt, BMJ 326 (7394): 860–863.

Mamlin, B.W. & Biondich, P.G. (2005). AMPATH Medical Record System (AMRS): collaborating toward an EMR for developing countries. *AMIA Annual Symposium proceedings*: 490–494.

Mamlin, B.W., Biondich, P.G., Wolfe, B.A., Fraser, H., Jazayeri, D., Allen, C., Miranda, J., Tierney, W.M. (2006). Cooking up an open source EMR for

developing countries: OpenMRS - a recipe for successful collaboration. *AMIA Annual Symposium proceedings*: 529–533.

Medical Web page (2017). Retrieved May 12, 2017 from http://medical.sourceforge.net

Microsoft Infopath Web page (2017). Retrieved June 14, 2017 from http://office.microsoft.com/en-us/infopath

Murray, C.J.L., Lopez, A.D., Wibulpolprasert, S. (2004). Monitoring global health: Time for new solutions. *BMJ 329*: 1096–1100.

MySQL Web page (2017). Retrieved June 2, 2017 from http://www.mysql.com

Noor, A.M., Gikandi, P.W., Hay, S.I., Muga, R.O., Snow, R.W. (2004). Creating spatially defined databases for equitable health service planning in low-income countries: The example of Kenya. *Acta Trop 91*: 239–251.

OpenEMR Web page (2017). Retrieved May 4, 2017 from http://www.oemr.org

Open Healthcare Web page (2017). Retrieved May 12, 2017 http://www.openhealth.org

OpenMRS Web page (2017). Retrieved May 14, 2017 from http://www.openmrs.org

OpenVista Web page (2017). Retrieved May 15, 2017 from http://worldvista.sourceforge.net/openvista/index.html

OSCAR McMaster Web page (2017). Retrieved May 12, 2017 from http://en.wikipedia.org/wiki/OSCAR_McMaster

Partners In Health Web page (2017). Retrieved May 14, 2017 from http://www.pih.org

Sainz, B., de la Torre, I., Bermejo, P., García, E., Díaz, F.J., Díez, J.F., López, M., de Castro, Carlos. (2010). Evolución, beneficios y obstáculos en la implantación del Historial Clínico Electrónico en el sistema sanitario, *Revista esalud.com* 6(22).

Sainz, B., Llamas, A. (2012). Overview of the most important open source software. Analysis of the benefits of OpenMRS, OpenEMR and VistA in Telemedicine and E-Health Services, Politicies and Applications: Advancements and Developments. *Editorial IGI Global.*

Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MongoDB, *Editorial UOC (Universitat Oberta de Catalunya).*

Seebregts, C.J., Mamlin, B.W., Biondich, P.G., Fraser, H.S., Wolfe, B.A., Jazayeri, D., Allen, C., Miranda, J., Baker, E., Musinguzi, N., Kayiwa, D., Fourie, C., Lesh, N., Kanter, A., Yiannoutsos, C.T., Bailey, C. (2009). The OpenMRS Implementers Network. *International journal of medical informatics* 78 (11): 711–720.

Siika, A.M., Rotich, J.K., Simiyu, C.J., Kigotho, E.M., Smith, F.E., Sidle, J.E., Wools-Kaloustian, K., Kimaiyo, S.N., Nyandiko, W.M., Hannan, T.J., Tierney,W.M. (2005). An electronic medical record system for ambulatory care of HIV-infected patients in Kenya. Int. J. Med. Inform. 74 (5): 345–355.

SQL Web page (2017). Retrieved May 14, 2017 from http://www.sql.org

Stansfield S. (2005). Structuring information and incentives to improve health. *Bull World Health Organ* 83: 562.

Tolven Healthcare Web page (2017). Retrieved May 12, 2017 from http://www.tolven.org

Tomasi, E., Facchini, L.A., Maia, M. F. (2004). Health information technology in primary health care in developing countries: a literature review. Bulletin of the World Health Organization. 82(11): 867–74.

Trusted Opensource Records for Care and Health Web page (2017). Retrieved May 12, 2017 from https://launchpad.net/trusted

XML Web page (2017). Retrieved May 14, 2017 from http://www.w3.org/XML