

# Sistema Web de Gestion de Consultorios Medicos

Universidad Nacional de Salta



Facultad de Ciencias Exactas  
Seminario de Computación

Alumno: Ricardo Daniel Quiroga  
Director de Tesis: Ernesto Sanchez  
Proyecto Especifico  
Febrero de 2014

---

*Cuando crezcas, descubrirás que ya defendiste mentiras,  
te engañaste a ti mismo o sufriste por tonterías. Si eres  
un buen guerrero, no te culparás por ello, pero tampoco  
dejarás que tus errores se repitan.*

*Pablo Neruda*

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Objectivos Generales . . . . .	1
1.2. Resumen . . . . .	1
<b>2. Elección de la Tecnología</b>	<b>3</b>
2.1. ¿Por que Web y no Desktop? . . . . .	3
2.2. Apache . . . . .	4
2.3. mod_wsgi . . . . .	4
2.3.1. WSGI . . . . .	4
2.4. PosgreSQL . . . . .	4
2.5. Python . . . . .	5
2.5.1. La Filosofia detras de Python . . . . .	5
2.5.2. Baterias Incluidas . . . . .	6
2.5.3. Implementaciones . . . . .	6
2.6. Django . . . . .	7
2.6.1. MVC . . . . .	7
2.6.2. Django y el MVT . . . . .	7
2.6.3. El Modelo . . . . .	8
2.6.4. La Vista . . . . .	9
2.6.5. La Plantilla . . . . .	10
2.6.6. La Configuracion de Rutas . . . . .	12
<b>3. Antecedentes</b>	<b>13</b>
3.1. Historia Clinica . . . . .	13
3.2. Problemas del Sistema Actual . . . . .	16
3.2.1. Problemas de Almacenamiento . . . . .	16
3.2.2. Problemas de Busqueda y Localizacion . . . . .	16
3.2.3. Deterioro . . . . .	16
3.2.4. Otros Problemas . . . . .	16
3.2.5. La Solucion Planteada . . . . .	16
3.3. Gestión de Turnos . . . . .	17
<b>4. El Proyecto</b>	<b>19</b>
4.1. Motivación . . . . .	19
4.2. Descripción del Proyecto . . . . .	19
4.3. Arquitectura de la Aplicacion . . . . .	19
4.3.1. Modulo Usuarios . . . . .	19
4.3.2. Modulo Gestión de Turnos . . . . .	19
4.3.3. Modulo Historia Clínica . . . . .	19
4.4. Elección de la metodología de Programación . . . . .	19

---

<b>5. Instalacion y Configuracion</b>	<b>21</b>
5.1. Requerimientos . . . . .	21
5.1.1. Requerimientos de Hardware . . . . .	21
5.1.2. Requerimientos de Software . . . . .	21
5.2. Apache . . . . .	22
5.2.1. Instalacion en Limpio . . . . .	22
5.2.2. Instalacion mediante WAMP, LAMP, MAMP, WAPP . . . . .	22
5.2.3. Configuracion . . . . .	22
5.2.4. Instalacion de PosgreSQL . . . . .	23
5.2.5. Creacion de la Base de Datos . . . . .	23
5.3. Instalacion de Python . . . . .	24
5.3.1. Probando Python . . . . .	24
5.4. Instalar Django . . . . .	25
5.5. Instalando el Resto de Las Dependencias . . . . .	25
5.5.1. psycopg2 . . . . .	25
5.5.2. ReportLab . . . . .	25
5.5.3. easy-thumbnails . . . . .	25
5.5.4. django_extensions . . . . .	26
5.5.5. django_cron . . . . .	26
5.5.6. Descargar e Instalacion de mod_wsgi . . . . .	26
5.5.7. Cargando el Modulo en Apache . . . . .	26
5.5.8. Configuracion del Proyecto . . . . .	26

# Índice de figuras

2.1.	Diagrama del Patron MVC Modelo Vista Controlador . . . . .	8
2.2.	Generaciones de Herramientas de Desarrollo Web . . . . .	8
2.3.	El patron Modelo Vista Template de Django . . . . .	9
3.1.	Almacenamiento Fisico de Archivos . . . . .	13
3.2.	Modelo Historia Clinica Ministerio de Salud de La Nacion Pag 1 . . . . .	14
3.3.	Modelo Historia Clinica Ministerio de Salud de La Nacion Pag 2 . . . . .	15
5.1.	Ejecutando Python en la Terminal . . . . .	24



# **Capítulo 1**

## **Introducción**

El presente trabajo de tesis es para obtener el título intermedio de Computador Universitario perteneciente a la carrera Licenciatura en Análisis de Sistemas (plan 97).

El tema elegido es el desarrollo de un sistema de gestión para consultorios médicos, en el se intento reflejar todos los conocimientos que adquirí durante el cursado de la carrera. En cuanto a las razones para la elección del tema son entre otras, el buscar desarrollar un sistema que por sus dimensiones se proponga como un reto, ya que hasta la fecha solo tenía experiencia en cuanto al desarrollo de pequeñas aplicaciones. El área de aplicación quedó definida, por la simple razón que tenía contacto con profesionales del área de la Salud.

En cuanto a tecnología quise implementarlo usando algo distinto del tradicional PHP y MySQL para Web, opté por probar una tecnología que no conocía Python, Django y PostgreSQL, la cual tenía bastante buenas opiniones por parte de terceros y por suerte no decepcionó sobre todo Django, que cambió mi forma de pensar a la hora de encarar un proyecto web.

### **1.1. Objectivos Generales**

El objetivo del proyecto de tesis fue el de diseñar y desarrollar un Sistema Centralizado para el área Salud, específicamente aplicándose al área de consultorios médicos, permitir un mejor seguimiento y control de la evolución de los pacientes mediante la informatización de los diferentes exámenes y consultas que se le realizan al paciente permitiendo la unificación de su historia clínica. Además de también gestionar la asignación de turnos a los pacientes.

Lo que se pretende es brindar un sistema modular y eficiente que permita su fácil aplicación y además de brindar la posibilidad de modificación tanto para adecuación para casos específicos como extensión de sus funcionalidades.

### **1.2. Resumen**

El Sistema Web de Gestión de Consultorios Médicos ; desde ahora el Sistema, está pensado para satisfacer las necesidades de Consultorios Médicos o cualquier otra actividad en donde sea necesario almacenar información demográfica de Pacientes , Historias Clínicas, Prescripciones así también como la Asignación de Turnos.

## *Capítulo 1*

---

Al Ser un Sistema Centralizado, se puede acceder al el desde cualquier navegador web actual que cuente con conexión a Internet, lo que permite entre otras cosas:

Disminuir los tiempos de esperas por parte de pacientes a la hora de solicitar ser atendidos, solo necesita solicitar un turno vía web el sistema automáticamente le asignará una fecha y hora acorde a sus requerimientos. Permite a los médicos manejar más fácilmente su agenda para atención de pacientes.

Mejorar el Seguimiento de los Pacientes por parte de los médicos , centralizando toda su información ya que con ello el médico puede monitorear la evolución de su paciente donde sea que se encuentre ya que solo necesitará una PC con conexión a internet.

## Capítulo 2

# Elección de la Tecnología

### 2.1. ¿Por que Web y no Desktop?

Una aplicacion Desktop (tambien llamada de Escritorio) es aquella que requiere ser instalada en el Ordenador (PC) del Usuario, y que es ejecutada directamente por el sistema operativo, ya sea Microsoft Windows, GNU/Linux, Mac OS, etc.

Algunos Ejemplos de Estas Aplicaciones:

- Winamp
- Adobe Photoshop
- iTunes
- Microsoft Oficce (Word, Excel, Power Point. etc)

Aunque suelen ser mas robustas y estables que las aplicaciones Web Presentan varios inconvenientes tales como:

- Su acceso solo se limita al ordenador donde fue instalada.
- La Aplicacion es dependiente del Sistema Operativo que utilice el ordenador, aunque existen programas Multiplataforma no aseguran una compatibilidad completa.
- Requieren una instalacion personalizada
- En caso de Actualizaciones requieren que estas se hagan de forma manual en cada ordenador donde se instalo la Aplicacion.
- Suelen Tener requerimientos especiales de Software y librerias para poder funcionar.

Una aplicacion Web, es aquella que solo requiere ser instalada en un Servidor, su ejecucion requiereicamente Disponer de un ordenador con conexión a internet y un navegador en contraparte de las Desktop que requiere que se instale en cada ordenador donde se pretende usar.

Por lo cual brinda una serie de ventajas tales como:

- Portabilidad, se ejecuta desde cualquier ordenador que posea coneccion a internet sin depender de Software adicional, Plataforma y/o Sistema Operativo.
- La informacion que se maneja es multiusuario por lo que son especialmente utiles para desarrollar aplicaciones multiusuarios basadas en compartir informacion.
- Consumen muy pocos recursos, por lo que el usuario no necesita tener un ordenador con grandes prestaciones para trabajar con ellas.

- Son fáciles de Actualizar y mantener.
- Se pueden utilizar en miles de equipos sin limitación y restricción alguna.
- Su funcionalidad es independiente del Sistema Operativo Instalado en el ordenador del usuario.
- No hay problemas de incompatibilidad de Versión de software ya que los usuarios trabajan con la misma versión.

En resumen el Sistema por sus características podría haberse implementado como un sistema Desktop pero se hubiesen perdido las características deseadas del mismo.

## 2.2. Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPD 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y energético pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y civilizasen.<sup>el</sup> paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, a patchy server (un servidor "parcheado") suena igual que Apache Server.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

## 2.3. mod\_wsgi

mod\_wsgi es un módulo de Apache que provee una interfaz WSGI para correr Web en Python sobre Apache. Y esto, es todo lo que necesitas para que tus archivos \*.py se ejecuten por medio de un navegador Web.

### 2.3.1. WSGI

WSGI es el acrónico de Web Server Gateway Interface que es una especificación para una simple y universal interfaz entre una aplicación web (en nuestro caso una aplicación escrita en Django) y un servidor Web para el lenguaje de programación Python. Es un estándar de Python el cual se describe con detalle en la PEP 333<sup>1</sup>

## 2.4. PostgreSQL

PostgreSQL es un gestor de base de datos relacional que puede correr tanto bajo sistemas operativos Windows como en distribuciones Linux como Red Hat, Suse, Centos, etc.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de

---

<sup>1</sup><http://www.python.org/dev/peps/pep-0333/>

forma desinteresada, altruista, libre y/o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

El nombre hace referencia a los orígenes del proyecto como la base de datos "post-Ingres", y los autores originales también desarrollaron la base de datos Ingres.

El proyecto post-ingres pretendía resolver los problemas con el modelo de base de datos relacional que habían sido aclarados a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos", es decir, combinaciones de datos simples que conforman una única unidad. Actualmente estos son llamados objetos. Se esforzaron en introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la habilidad de definir tipos, pero también la habilidad de describir relaciones - las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario. En Postgres la base de datos «comprendía» las relaciones y podía obtener información de tablas relacionadas utilizando reglas. Postgres usó muchas ideas de Ingres pero no su código.

## 2.5. Python

Django esta escrito puramente en Python, por lo que Obiamente Necesitaremos Instalar Python. Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y 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,<sup>1</sup> que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

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.

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.

### 2.5.1. La Filosofía detrás de Python

Los usuarios de Python se refieren a menudo a la Filosofía Python que es bastante análoga a la filosofía de Unix. El código que sigue los principios de Python de legibilidad y transparencia se dice que es "pythonico". Contrariamente, el código opaco u ofuscado es bautizado como "no pythonico"(unpythonic.<sup>en</sup> inglés).

Estos principios fueron famosamente descritos por el desarrollador de Python Tim Peters en El Zen de Python, algunos de ellos son:

- Bello es mejor que feo.
- Explícito es mejor que implícito.
- Simple es mejor que complejo.

- Complejo es mejor que complicado.
- Plano es mejor que anidado.
- Disperso es mejor que denso.
- La legibilidad cuenta.
- Los casos especiales no son tan especiales como para quebrantar las reglas.
- Aunque lo práctico gana a la pureza.
- Los errores nunca deberían dejarse pasar silenciosamente.
- A menos que hayan sido silenciados explícitamente.
- Frente a la ambigüedad, rechaza la tentación de adivinar.
- Debería haber una y preferiblemente sólo una manera obvia de hacerlo.
- Aunque esa manera puede no ser obvia al principio a menos que usted sea holandés.<sup>15</sup>
- Ahora es mejor que nunca.
- Aunque nunca es a menudo mejor que ya mismo.
- Si la implementación es difícil de explicar, es una mala idea.
- Si la implementación es fácil de explicar, puede que sea una buena idea.
- Los espacios de nombres (namespaces) son una gran idea ¡Hagamos más de esas cosas!

### 2.5.2. Baterias Incluidas

Python tiene una gran biblioteca estándar, usada para una diversidad de tareas. Esto viene de la filosofía "pilas incluidas"("batteries included") en referencia a los módulos de Python. Los módulos de la biblioteca estándar pueden mejorarse por módulos personalizados escritos tanto en C como en Python. Debido a la gran variedad de herramientas incluidas en la biblioteca estándar, combinada con la habilidad de usar lenguajes de bajo nivel como C y C++, los cuales son capaces de interactuar con otras bibliotecas, **Python es un lenguaje que combina su clara sintaxis con el inmenso poder de lenguajes menos elegantes.**

### 2.5.3. Implementaciones

En la actualidad existen diversas implementaciones de Python

- **CPython** es la implementación original, disponible para varias plataformas en el sitio oficial de Python.
- **IronPython** es la implementación para .NET
- **Stackless Python** es la variante de CPython que trata de no usar el stack de C [www.stackless.com](http://www.stackless.com).
- **Jython** es la implementación hecha en Java
- **Pippy** es la implementación realizada para Palm [pippy.sourceforge.net](http://pippy.sourceforge.net)
- **PyPy** es una implementación de Python escrita en Python y optimizada mediante JIT [pypy.org](http://pypy.org)

## 2.6. Django

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el paradigma conocido como **Model Template View**. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la **World Company de Lawrence, Kansas**, y fue liberada al público bajo una licencia BSD en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt [http://es.wikipedia.org/wiki/Django\\_Reinhardt](http://es.wikipedia.org/wiki/Django_Reinhardt).

En junio del 2008 fue anunciado que la recién formada Django Software Foundation se haría cargo de Django en el futuro.

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.

### 2.6.1. MVC

Antes de Explicar como funciona Django empezare por una breve explicacion de el patrón (MVC) Modelo Vista Controlador el cual 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 diseño 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.

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 de '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' actuando como Middleware<sup>2</sup>.

**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.

### 2.6.2. Django y el MVT

Si hicieramos una clasificacion de Herramientas de desarrollo web, podríamos clasificar a Django como parte de la tercera generacion:

Sin embargo más alla de las clasificaciones que podrían existir, está el entender como funciona realmente, al entenderlo se puede llegar a dominarlo.

Dijimos que era un framework MTV (una modificación de MVC, nada que ver con un canal de música), esto se debe a que los desarrolladores no tuvieron la intención de seguir algún patron de desarrollo, sino hacer el framework lo más funcional posible.

---

<sup>2</sup>Middleware es el software que proporciona un enlace entre aplicaciones de software independientes. Middleware a veces se llama a la vía que conecta dos aplicaciones y pasa los datos entre ellas. Los Middleware permiten que los datos contenidos en una base de datos puedan ser accedidos a través de otra. Ahorra el tiempo a los programadores.

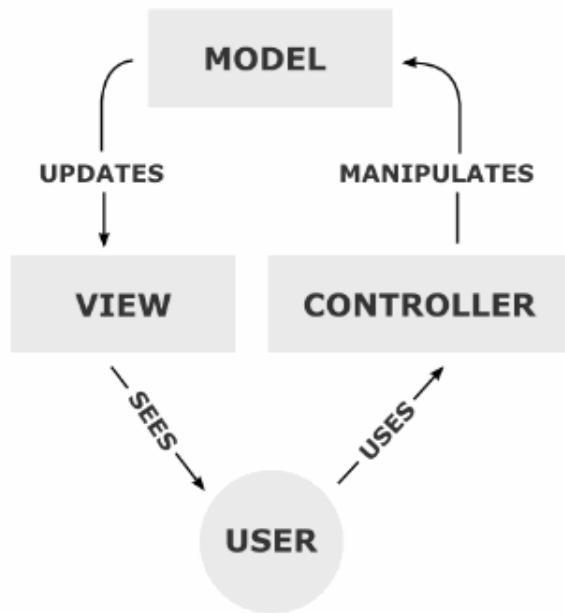


Figura 2.1: Diagrama del Patrón MVC Modelo Vista Controlador

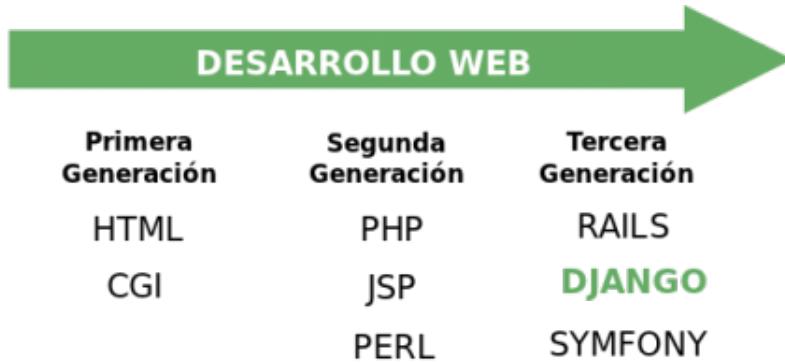


Figura 2.2: Generaciones de Herramientas de Desarrollo Web

- **El Modelo** en Django sigue siendo el modelo
- **La Vista** en Django se llama Plantilla (Template)
- **El controlador** en Django se llama Vista

Una imagen nos hará entender mejor esta relación:

### 2.6.3. El Modelo

El modelo define los datos almacenados, se encuentra en forma de clases de Python, las clases definidas son traducidas por Django y este genera las Tablas necesarias para el funcionamiento del modelo dentro de la base de datos, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros, posee métodos también. Todo esto permite indicar y controlar el comportamiento de los datos.

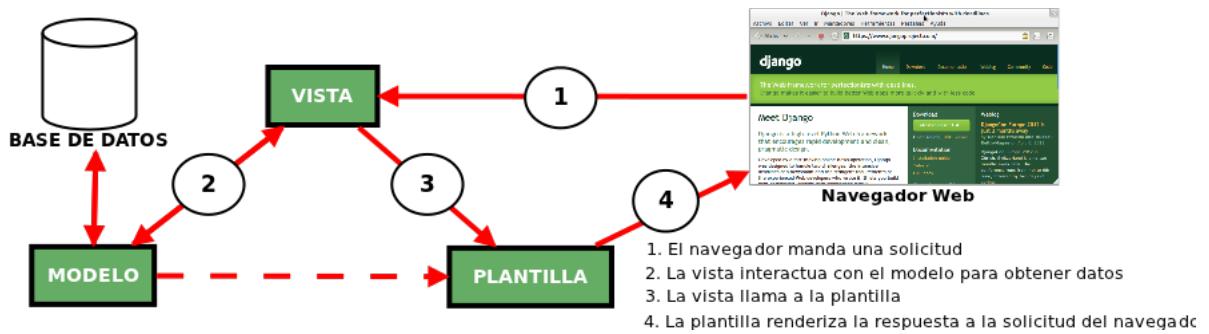


Figura 2.3: El patron Modelo Vista Template de Django

Aquí un extracto del código mostrando como se implementa uno de los tantos modelos con los que trabaja el Sistema

```

1
2 class Message(models.Model):
3     """
4         Clase Para Manejar mensajes entre usuarios
5     """
6     from_user = models.ForeignKey(User, related_name='from_user')
7     to_user = models.ForeignKey(User, related_name='to_user')
8     date = models.DateTimeField("Fecha y Hora", auto_now_add=True)
9     issue = models.CharField("Asunto", max_length=125, default='')
10    content = models.TextField("Cuerpo del Mensaje")
11    read = models.BooleanField("Leido", default=False)
12
13
14    class Meta:
15        db_table = "Messages"
16        verbose_name = "InboxMessage"
17        verbose_name_plural = "InboxMessages"

```

3

4

#### 2.6.4. La Vista

La vista se presenta en forma de funciones en Python, su propósito es determinar que datos serán visualizados, entre otras cosas más que iremos viendo conforme avanzamos con el curso. El ORM de Django permite escribir código Python en lugar de SQL para hacer las consultas que necesita la vista. La vista también se encarga de tareas conocidas como el envío de correo electrónico, la autenticación con servicios externos y la validación de datos a través de formularios. Lo mas importante a entender con respecto a la vista es que no tiene nada que ver con el estilo de presentación de los datos, sólo se encarga de los datos, la presentación es tarea de la plantilla.

Aquí muestro una vista sencilla que realiza una consulta base de datos que listara todos los usuarios que sean medicos.

<sup>3</sup>Como algunos lo notaran la variable `from_user` del modelo internamente es una relación 1:M dentro de la base de datos.

<sup>4</sup>La Clase interna Meta define atributos especiales como '`db_name`' que hace referencia a como se llamara la tabla dentro de la Bases de Datos.

```
1 def patient_show_medics_list(request):
2     """
3         Muestra el listado de Medicos
4     """
5     mi_template = get_template('Patients/GestionTurnos/medics-list.html')
6     dict = generate_base_keys(request)
7
8     if is_patient(request.user):
9         dict['medics'] = UserInformation.objects.filter( \
10             user__groups__name='Medico')
11         html_cont = mi_template.render(Context(dict))
12         return HttpResponseRedirect(html_cont)
13
14     else:
15         #si hay un usuario logueado intentanto acceder sera enviado a una
16         # pagina de error
17         path = request.META['PATH_INFO']
18         return HttpResponseRedirect("/restricted-access%s" %path)
```

Aunque es un ejemplo sencillo podemos apreciar el potencial de Django, como vemos no vemos ningun codigo SQL, pues bien dicho codigo SQL se ejecuta internamente nos aleja del problema de las restricciones de la Base de Datos ya sea que usemos PosgreSQL (como en este sistema), MySQL, SQLServer o SQLite nosotros solo escribiremos codigo Python, El framework se encargara de traducir esa instrucion al motor de bases de datos correspondiente que estemos usando.

```
dict['medics'] = UserInformation.objects.filter( \
    user__groups__name='Medico')
```

Traducido a SQL terminariamos con algo tan horrible como esto:

```
SELECT * FROM UserInformation as Info
INNER JOIN User ON Info.username = User.username
INNER JOIN GroupsByUsers ON User.username = GroupsByUsers.username
...
```

## 2.6.5. La Plantilla

La plantilla es básicamente una página HTML con algunas etiquetas extras propias de Django, en si no solamente crea contenido en HTML (también XML, CSS, Javascript, CSV, etc).

La plantilla recibe los datos de la vista y luego los organiza para la presentación al navegador web. Las etiquetas que Django usa para las plantillas permiten que sea flexible para los diseñadores del frontend, pueden Extenderse a partir de otras plantillas incluso tiene estructuras de datos como if, por por si es necesaria una presentación lógica de los datos, estas estructuras son limitadas para evitar un desorden poniendo cualquier tipo de código Python.

Esto permite que la lógica del sistema siga permaneciendo en la vista. Aqui la vista para Iniciar Session:

```

{%
    extends 'base.html'
}

{%
    block style %}
<link type="text/css" rel="stylesheet" media="all"
      href="/media/css/fancy-forms.css" />
<link type="text/css" rel="stylesheet" media="all"
      href="/media/css/gradient-buttons.css" />
<link type="text/css" rel="stylesheet" media="all"
      href="/media/css/messages.css" />
{%
    endblock %}

{%
    block contenido %}
<br /><br /><br />
    {%
        if not_login %}
            <div class="fancy-form-white" style="width: 350px;
                margin: 0 auto;">
                <h3 class="title">Iniciar Session</h3><br />
                <form action=". " method="POST">
                    <table style="margin: 0 auto; width: 330px;">
                        <tr>
                            <td><label for="username">Usuario:</label></td>
                            <td><input type="text" name="username" value=""
                                tabindex="1" id="username"></td>
                            <td rowspan="2">
                                <input type="submit" value="Login" tabindex="3"
                                    class="grad-button-blue" style="height: 50px;">
                            </td>
                        </tr>
                        <tr>
                            <td><label for="password">Contraseña:</label></td>
                            <td><input type="password" name="password" value=""
                                tabindex="2" id="password"></td>
                        </tr>
                    </table>
                </form>
                <br />
            </div>
    {%
        if login_error %}
            <br />
            <br />
            <div class="alert">Alerta: Error Usuario y/o Contraseña Incorrectos</div>
    {%
        endif %}

    {%
        else %}
            <div class="alert">Alerta: Usted ya ha iniciado session con el
                usuario <strong>{{ username }}</strong></div>
            <br />
            <a href="/logout">Cerrar Session</a>
    {%
        endif %}
{%
    endblock %}

```

## 2.6.6. La Configuración de Rutas

Django posee un mapeo de URLs que permite controlar el despliegue de las vistas, esta configuración es conocida como URLConf. El trabajo del URLConf es leer la URL que el usuario solicitó, encontrar la vista apropiada para la solicitud y pasar cualquier variable que la vista necesite para completar su trabajo. El URLConf está construido con expresiones regulares en Python y sigue la filosofía de Python: Explícito es mejor que implícito. Este URLConf permite que las rutas que maneje Django sean agradables y entendibles para el usuario.

Fragmento del archivo urls.py del Proyecto

```
(r'^$', base_views.index),
(r'^index/$', base_views.index),
(r'^login/$', base_views.login),
(r'^logout/$', base_views.logout),
(r'^change-password/$', base_views.change_password),
(r'^restricted-access/$', base_views.restricted_access),
(r'^restricted-access/(.+)$', base_views.restricted_access),
```

# Capítulo 3

## Antecedentes

### 3.1. Historia Clinica

Actualmente en la mayoria de los centros de salud la informacion que compone la Historias Clinicas de cada paciente es almacenan mediante documentos fisicos, en su mayoria totalmente elaborados a mano, en otros casos usando plantillas con campos preformateados (Como se puede Observar en las Figuras 3.1 y 3.2), lo cual genera varios problemas.

En cuanto al software existente aplicado a la manipulacion de historia clinica, este suele ser muy incompleto y abocado a la especialidad del medico, es por ello que si esta informatizado cada area suele manejar sistemas que terminan siendo incompatibles entre si.



Figura 3.1: Almacenamiento Fisico de Archivos

**Ministerio de Salud Presidencia de la Nación**

<b>Centro de atención</b>		<b>DNI</b>			
• Nombre	• Nro.	• Fecha de nacimiento	• Nacionalidad		
• Apellido	• Teléfono	• Lugar de Origen			
• Calle					
• Barrio					
• Partido					
• Localidad					
<b>Estado civil</b>		<b>Trabajo</b>		<b>Obra social</b>	
Soltero	Separado	Si	No	Si	No
Unión estable	Divorciado	Dependiente	Ocasional	¿Cuál?	
Casado		Cuenta propia	Tipo de ocupación	Nº	
<b>Educación</b>		<b>Condiciones de vida</b>		<b>Situación de vulnerabilidad</b>	
Analfabeto	Rancho o casilla	Agua corriente	Maltrato familiar		
Primaria incomp.	Vivienda móvil	Agua de pozo	Alcoholismo flia.		
Primaria comp.	Pieza de inquilinato	Baño interno o descarga	Adicciones flia.		
Secundario incomp.	Lugar no apto para vivienda	Baño público	Madre sola		
Secundario comp.	3 o más pers. por cuarto	Baño afuera	Situación de calle		
Terciario/univer.	Hijo 6 a 12 años sin escolaridad	Cloacas	N.B.I.		
	Baja capacidad subsistencia	Pozo ciego			
	Letrina	Gas envasado			
	Cocina / calefacción	Kerosene			
	Gas natural	Electricidad			
		Leña o carbón			
<b>Consumo de sustancias</b>					
Tabaquismo		Alcohol		Drogas	
¿Fuma? Desde (edad) cant.	¿Bebe?	¿Consumió alguna vez?	Actividad física	Si	No
Nunca Fumó fumador pasivo	¿Lo criticó alguien porque tomaba mucho?	Veces por sem.	¿Cuál?		
Dejó de fumar fecha	¿Sintió ganas de disminuir la bebida?	Si	Diuresis		
¿Cuántos minutos pasan desde que se levanta hasta que prende un cigarrillo?	¿Se sintió alguna vez culpable por tomar mucho?	No	¿Se inyectó alguna vez?	Catarsis	
¿Piensa dejar de fumar?	¿Toma algo a la mañana para sentirse mejor?	Si	Si	No	Sueño
En los próximos 6 meses		No			
En el próximo mes					
<b>Antecedentes personales</b>		<b>Antecedentes familiares</b>			
HTA	Menarca	IRS	madre	padre	herm.
DIABETES	G	P A C			
Enfermedad coronaria / IAM	Ritmo menstrual	FUM	HTA		
ACV	Fecha último PAP		Cardiopatías		
EPOC ASMA	MAC Si No		DIABETES		
Alergia	Dispareunia Si No		ACV		
Enfermedad reumática	Prob. Sexuales Si No		Ca. de colon		
Enf. Oncológico			Ca. de pulmón		
TBC	Eyaculación precoz Si No		Ca de mama		
VIH	Disfunción erectil Si No		Otra enf. Oncol.		
Chagas			Consumo drogas		
ITS			Abuso alcohol		
Psicopatológicos			Depresión		
Neurológicos			Otras		
Otros					
Internaciones / operaciones / accidentes	Si	No	Observaciones		
Indicar motivo y fecha					
¿ALERGIA a medicamentos?	Si	No	¿Cuál?		
Medicamentos (principio activo)	Presentación	Dosificación	Medicamentos (principio activo)	Presentación	Dosificación
1			5		
2			6		
3			7		
4			8		
			Observaciones		
			Matrimonio	Pareja no casada	
			Separación	Divorcio	
			Hombre Mujer	Hijos	
			Gemelos	Muerte	
			Relación estrecha	Relación conflictiva	
<b>Médico de cabecera</b>			<b>Fecha de apertura</b>		

Figura 3.2: Modelo Historia Clínica Ministerio de Salud de La Nacion Pag 1

	T.A.	F.C.	Peso	Talla	IMC.	Perímetro abdominal
<b>• Examen Físico</b>	Piel Ojos: ¿usa lentes? Agudeza visual Oídos Dentadura Pulmones Corazón Abdomen Genitales externos Mamas	Sin lesiones significativas no      si      último control OD      OI Otoscopía normal en buen estado buena entrada sin ruidos agregados ritmo regular, sin soplos blando, indoloro, sin visceromegalias normales sin nódulos	Hallazgos anormales			
<b>• Lista de problemas</b>	Problema Código Fecha inicio Fecha resolución	Problema Código Fecha inicio Fecha resolución	Problema Código Fecha inicio Fecha resolución	Problema Código Fecha inicio Fecha resolución	Problema Código Fecha inicio Fecha resolución	Problema Código Fecha inicio Fecha resolución
<b>Cuidados preventivos</b>	Intervención Consejo alimentación saludable Consejo actividad física Consejo salud sexual y reproductiva Consejo anti tabáquico Consejo prevención de accidentes Vacuna doble adultos Vacuna hepatitis b Vacuna sarampión - rubeóla Vacuna antigripal Vacuna antineumocócica Otra Examen odontológico Agudeza visual Tensión arterial Índice de masa corporal (peso/talla) Perímetro abdominal Colesterol total plasmático Glucemia Riesgo cardiovascular global Sangre oculta en materia fecal Mamografía Papanicolaou	Periodicidad / Población Variable, según riesgo Variable, según riesgo Variable, según riesgo Variable, según riesgo Variable, según riesgo 1 dosis cada 10 años 3 dosis (0-1-6 meses) en grupos de riesgo 1 dosis en personas No inmunizadas 1 dosis anual en grupos de riesgo según calendario 1 dosis en grupos de riesgo. Evaluar revacunación Variable según riesgo regional o campaña Anual, población general Anual, población general Variable, según el riesgo Anual población general para su cálculo: Anual población general. Medición con centímetro a la altura del ombligo Variable, según el riesgo Anual. Mayores de 45 años. Menores de 45 años IMC >29.9 y FRCV Variable según riesgo. Población general mayor de 39 años sin enfermedad cardiovascular previa Anual. Mayores de 50 años Anual mujeres mayores de 50 años a 69 años Mujeres a partir de los 25 años, 2 controles Consecutivos anuales normales, luego controles cada 3 años	Mes/año			
*Recomendaciones orientadas sujetas a modificaciones y actualización según guías de práctica de la autoridad sanitaria						

Figura 3.3: Modelo Historia Clínica Ministerio de Salud de La Nación Pag 2

## 3.2. Problemas del Sistema Actual

### 3.2.1. Problemas de Almacenamiento

A medida que crece el numero pacientes y la informacion que va anexando a cada Archivo se va necesitando mas espacio fisico para almacenar dicha informacion. Por ello con el paso del tiempo las instalaciones dedicadas a tal fin suelen verse colapsadas por los grandes volumenes de informacion que deben manejar.

### 3.2.2. Problemas de Busqueda y Localizacion

La Busqueda de expedientes se puede agilizar un poco utilizando una buena organizacion, el problema es que la mayoria de los edificios para tal fin, suelen estar saturados por grandes volumenes de archivos fisicos, Por lo que encontrar un archivo requerido suele ser una tarea costosa y lenta.

### 3.2.3. Deterioro

En lo que hace a la conservación propiamente dicha, el combate de los problemas habituales que se derivan de las condiciones climáticas, de la humedad, de las plagas, del deterioro natural del papel, especialmente por su fabricación con celulosa desde hace dos siglos, es una constante que, pese a sus avances, no ha encontrado soluciones definitivas. Por ende, una preocupación común a los archivos y bibliotecas, es encontrar remedios prácticos y asequibles para asegurar la preservación de sus acervos. De lo anterior se desprende la necesidad, en el nivel nacional, de procurar el establecimiento de políticas y normas sobre conservación en las instituciones públicas y privadas dedicadas a la protección del patrimonio.

### 3.2.4. Otros Poblemas

Otros problemas que acarrean los archivos fisicos son:

- Pérdida, alteración o daño de documentos importantes
- Gasto excesivo en fotocopias
- Altos costos de personal para administrar, suplir, mantener y recuperar el archivo fisico
- Costos asociados al transporte de documentos ya sea interna o externamente
- Costos asosiados al espacio fisico requerido para su almacenamiento
- Falta de condiciones adecuadas para el almacenamiento de documentos como: ventilación, humedad, temperatura
- Falta de respaldo adecuado en caso de catastrofe como incendio, inundación o terremoto

### 3.2.5. La Solucion Planteada

Por ello este era un escenario perfecto donde Es Necesario informatizar el Actual Sistema, lo cual solucionaria los 2 principales problemas del mismo que son el Excesivo espacio de almacenamiento y el lento trabajo de busqueda ademas de:

- Reducir costos de Personal Administrativo, ya que las busquedas y registro las hara el Sistema.

- Brindar la informacion de manera rapida en situaciones criticas que requieren un rapido accionar por parte del Medico.
- Disponibilidad En todo momento y cualquier lugar para consulta por parte de los Medicos ya que solo requerira disponer de un usuario y un ordenador con coneccion a Internet para poder consultar.

Es cierto que los sistemas informaticos sufren problemas de Almacenamiento, Busqueda (en el caso de grandes volumenes de informacion) y deterioro por el paso del tiempo Pero en este caso el primero se soluciona Agregando mas espacio de disco, cosa que hoy en dia es algo relativamente barato a razon de 1 peso = 1 Gb.<sup>1</sup>

El problema de las busqueda no afecta mucho con la velocidad de los equipos actuales se puede consultar bases de datos con millones de registro en unas pocas milesimas de segundos dicho tiempo resulta impersectible para la persona en la mayoria de las veses, en todo caso dependera de la implementacion y el motor de bases de datos mas que de las prestaciones del hardware.

En cuanto al deterioro, puede que con el tiempo los equipos de hardware tales como discos duros fallen en algun momento, pero esto es salvable siempre y cuando se realicen buenas practicas tales como implementar un sistemas de backup, tambien replicacion de datos en caso de que se necesite alta disponibilidad de la informacion que se almacena.

### **3.3. Gestión de Turnos**

En lo que respecta a asignacion de turnos, el sistema actual en la mayoria de los casos no ha tenido un mejor panorama en cuanto a implatancion de un software, aunque ya en esta area existen algunas aplicaciones que intentan solucionar el problema de manera mas o menos eficientes.

---

<sup>1</sup>Gb hace referencia a Gigabyte que es una medida utilizada en informatica la cual normalmente hace referencia a tamaños de almacenamiento.



# **Capítulo 4**

# **El Proyecto**

## **4.1. Motivación**

En la actualidad existen pocos sistemas Aplicados en el ambito de la gestion en el area de Medicina y los existentes suelen ser solo para areas especificas

## **4.2. Descripción del Proyecto**

## **4.3. Arquitectura de la Aplicacion**

Implementado en Python utilizando en Framework Django, utilizando el motor de bases de datos PosgreSQL, funciona con una interfaz web por lo que se accede al mismo mediante un Navegador Web, Internamente maneja 2 Modulos principales que son el Modulo de Gestión de Turnos y el Modulo de manejo de Historia Clinica ; al ser un sistema web implementa un tercer modulo de manera implicita que control de acceso mediante la definicion de Grupos Usuarios y sus correspondientes permisos.

## **4.4. Modulo Usuarios**

La gestion de usuarios es un proceso bastante comun en casi todos los sistemas, muchos desarrolladores terminan programando funcionalidades de autenticacion una y otra vez a lo largo de los años y casi siempre funcionando de la misma manera. Django se pensó para simplificar la vida no para complicarla, por eso al ser una tarea bastante común en casi todas las aplicaciones, viene incluido un completo sistema de autenticación que gestiona:

- Usuarios
- Grupos
- Permisos
- Sesiones de Usuarios y Cookies

Aunque en cuanto a lo que se refiere manejo de sesiones es un completo sistema solo maneja un pequeño conjunto de datos por lo que hubo que extender mediante la adición de un Modelo adicional para complementar la información de los usuarios.

#### 4.4.1. Modelos

Aqui un diagrama con todos los modelos que componen el modulo Usuarios.

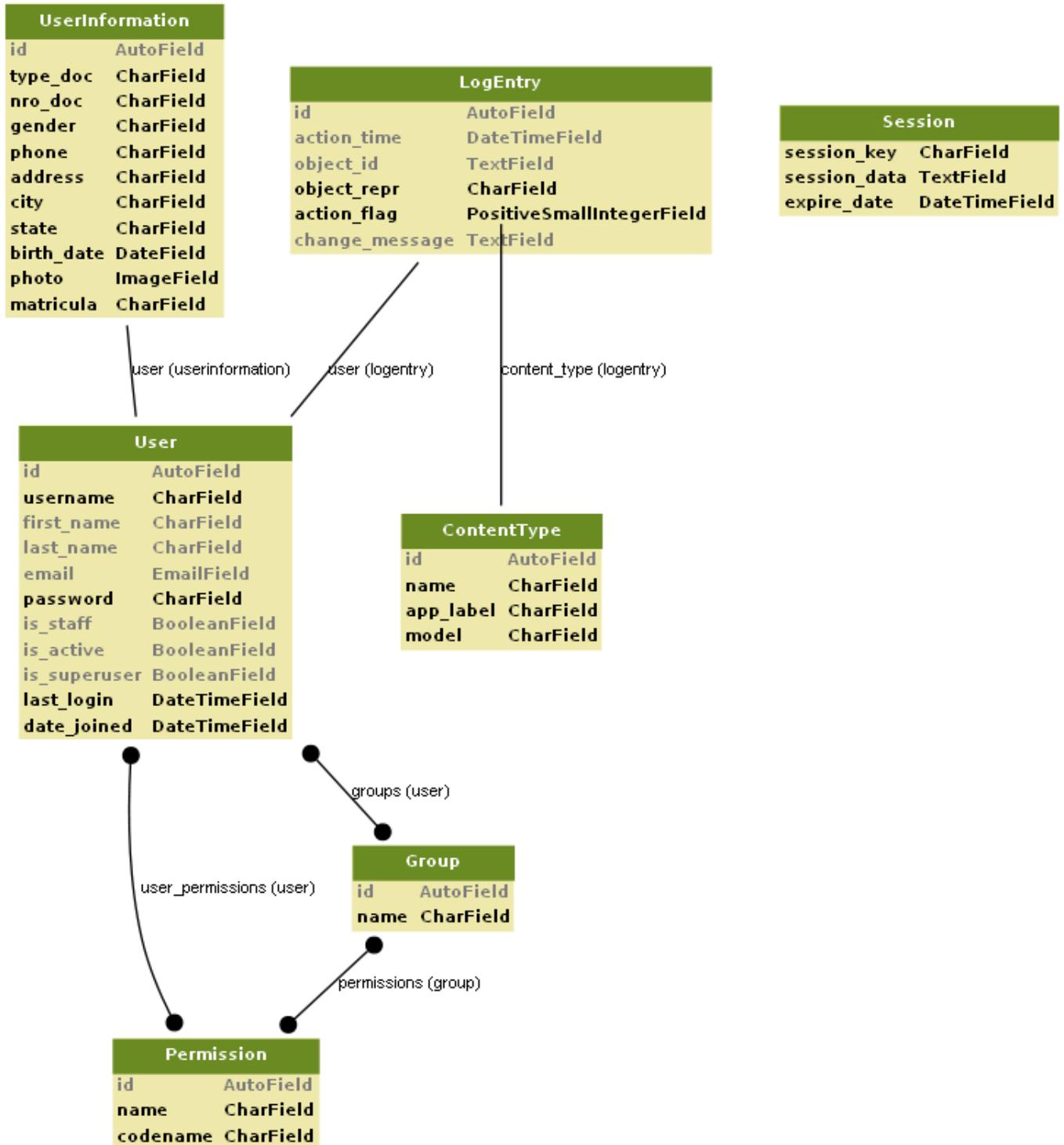


Figura 4.1: Modelo Historia Clinica Ministerio de Salud de La Nacion Pag 2

4.4.2. Modulo Gestión de Turnos

4.4.3. Modulo Historia Clínica

4.5. Elección de la metodología de Programación



## Capítulo 5

# Instalacion y Configuracion

En este Capitulo Aparte de guiarle para realizar una exitosa implementacion Local del Servidor de Produccion se hara referencia a cada una de las Herramientas y librerias Utilizadas.

### 5.1. Requerimientos

#### 5.1.1. Requerimientos de Hardware

Cualquier equipo que cumpla con las Caracteristicas para correr Windows 7 es suficiente en terminos de requerimientos minimos de Hardware siempre y cuando el numero de usuarios esperados no sea alto, despues el resto dependera de sus necesidades.

- Procesador x86, x64 de 1 Ghz o superior.
- Memoria Ram 1 GB o Superior

#### 5.1.2. Requerimientos de Software

- Apache 2.2
- PosgreSQL 9.2
- Python 2.7.x o Python 2.6.x
- Django 1.3.x o Superior
- PGAdmin
- psycopg2
- mod\_wsgi
- ReportLab
- easy-thumbnails
- django\_extensions
- django\_cron

## 5.2. Apache

Existen 2 caminos para instalar Apache La Primera Hacer una instalacion Limpia de Apache, la 2da es cuando no se quiere trastear con tanta configuracion por lo que opta por infraestructuras tipo WAMP, LAMP, WAPP, etc.

### 5.2.1. Instalacion en Limpio

Solo recomiendo este tipo de instalacion desde 0 para quienes ya poseen un conocimiento avanzado en cuanto a manejo de servidores.

Descargamos de [Apache.org](http://Apache.org) la ultima version disponible, puedes utilizar el siguiente vinculo: <http://www.apachehaus.com/cgi-bin/download.plx>.

Crea dos carpetas en la unidad C, la primera de nombre **Apache** y la segunda **servidor**. Descomprime el archivo descargado y ejecútalo, sigue los pasos de la instalación y de los datos que te piden solo escoge el destino de la instalación, que será la carpeta que creaste en **C:\Apache**, los otros datos déjalos de la forma predeterminada para configurarlos más tarde. El programa al instalarse crea un icono en el área de notificación que te permitirá: iniciar, detener y reiniciar Apache; tienes que tener en cuenta que cualquier cambio que hagas en el archivo de configuración no tendrá efecto hasta que reinicies el servidor.

### 5.2.2. Instalacion mediante WAMP, LAMP, MAMP, WAPP

Existen una infinidad de Paquetes precompilados y configurados, con Apache, PHP, PosgreSQL o MySQL y mas. Dichas infraestructuras suelen nombrarse como el acronomico de las herramientas que agrupan por ejemplo:

- WAMP Windows Apache MySQL PHP
- WAPP Windows Apache PosgreSQL PHP
- LAMP Linux Apache MySQL PHP
- MAMP Mac OS Apache MySQL PHP

Algunas distribuciones mas usadas disponibles Para Windows son WAMP Server <http://www.wampserver.com/> (WAMP), XAMPP <http://sourceforge.net/projects/xampp/> (WAMP + Perl), Bitnami <http://bitnami.com/stack/wapp> (WAPP) solo nos resta elegir cualquiera de ellas e instalarlas, aparte de la ruta de instalacion nos pediran el usuario y contraseña para acceder al motor de Base de Datos.

### 5.2.3. Configuracion

Toda la configuración para el funcionamiento de Apache se guarda en un archivo de texto nombrado: **httpd.conf** que se encuentra en la ruta **C:\Apache\conf** si realizamos una instalacion en limpio o **C:\wamp\bin\Apache\conf** si instalamos el paquete multiple preconfigurado no es necesario realizar este paso por lo que lo podremos saltar.

Al archivo **httpd.conf** lo podemos editar en cualquier editor de texto como Notepad.  
Buscamos la linea que dice

```
Listem LocalHost:80
```

y la Cambiamos por:

**Listem 80**

Ahora buscamos la instruccion:

```
DocumentRoot "C:\xxxxxxxx"
```

y la Cambiamos por:

```
DocumentRoot "C:\Servidor"
```

Recordar que al inicio de la instalacion creamos una Carpeta llamada Servidor en la unidad C. Por ultimo solo nos queda reiniciar el servidor Apache e introducir la siguiente direccion <http://127.0.0.1> si nos aparece una pagina **It's Work!** felicidades Apache esta Funcionando.

#### 5.2.4. Instalacion de PosgreSQL

La versión de PostgreSQL que he utilizado durante el desarrollo del sistema es la 9.2.x, quissas cuando leas esto haya salido una nueva version la cual no deberia generar inconvenientes ademas de que es posible que el proceso de instalación pueda variar.

El primer paso es descargar el instalador de PostgreSQL para Windows, lo puedes descargar desde el enlace siguiente <http://www.postgresql.org/download/windows>, nos bajara un instalador similar a **postgresql-9.2.3-rc1-windows.exe** lo ejecutamos como administrador.

Si tenemos activado el control de cuentas de usuario nos mostrará una advertencia con el texto ”¿Desea permitir que este programa realice cambios en el equipo?”, pulsaremos ”Sí”para continuar con la instalación de PostgreSQL.

Indicaremos la carpeta de instalación de PostgreSQL, donde se guardarán los ejecutables, librerías y ficheros de configuración de PostgreSQL en mi caso el directorio es **C: \PosgreSQL \9.2** , Indicaremos también la carpeta donde se guardarán los datos por defecto de PostgreSQL **C: \pgsql-data** .

Solo nos queda introducir la contraseña para el superusuario ”postgres”que será con el que iniciemos sesión para administrar la base de datos, despues podremos crear otros usuarios si es necesario. Ademas introduciremos el puerto de escucha para la conexión con el servidor PostgreSQL, por defecto el 5432.

Seleccionaremos la configuración regional y comenzara la instalacion, con esto PosgreSQL quedara instalado. Si tenemos algún cortafuegos (firewall) deberemos abrir el puerto 5432.

#### 5.2.5. Creacion de la Base de Datos

Junto con la Instalacion de PosgreSQL se instala el PGAdmin III que es una Heramienta GUI para administrar el motor de base de Datos. Iniciamos el Programa, desplegaremos ”Server Groups”, dentro desplegaremos ”Servidores” dentro de éste pulsaremos con el botón derecho del ratón sobre ”PostgreSQL 9.0 (localhost:5432)”, en el menú emergente seleccionaremos ”Conectar”.

Introduciremos la contraseña para el superusuario postgres (la contraseña introducida en la instalación).

Pulsaremos con el botón derecho del ratón sobre ”Bases de datos”, seleccionaremos ”Nueva Base de Datos”, en la pestaña ”Propiedades”introduciremos los siguientes datos:

- Nombre: nombre de la base de datos, en nuestro caso ”BDSem”.
- Propietario: seleccionaremos el usuario creado anteriormente ”postgres”.

- Codificado: seleccionaremos UTF8.
- Tablespace: seleccionaremos el tablespace creado anteriormente "pg\_default".
- Colación: seleccionaremos "Spanish, Argentina".
- Tipo carácter: seleccionaremos "Spanish, Argentina".

Pulsaremos ".K" para crear la base de datos, con esto ya tendremos nuestra base de datos aunque vacia, el resto como creacion de las Tablas correspondientes nesesarias para el proyecto lo haremos mas adelante mediante Django.

## 5.3. Instalacion de Python

Para este proyecto se utilizo CPython pero no la version Oficial url`http://www.python.org` sino la que distribuye Active State `http://www.activestate.com` llamada **Active Python** la cual provee caracteristicas adicionales a version oficial, podremos descargar la ultima version desde `http://www.activestate.com/activepython/downloads` aunque se recomienda instalar la version 2.7.x para evitar cualquier posible problema.

### 5.3.1. Probando Python

Para probar que la instalacion haya sido correcta abriremos la Terminal cmd.exe y escribiremos:

```
python
```

Si todo va bien nos debera aparecer algo similar a:

```
C:\ Administrador: Terminal - python
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Wyrven\Desktop>python
ActivePython 2.7.2.5 (ActiveState Software Inc.) based on
Python 2.7.2 (default, Jun 24 2011, 12:21:10) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> -
```

Figura 5.1: Ejecutando Python en la Terminal

En caso contrario deberias revisar que la ruta de Python este dentro de la variable PATH del sistema.

## 5.4. Instalar Django

Puedes bajarte Django desde el siguiente enlace <https://www.djangoproject.com/download/1.3.7/tarball/><sup>1</sup> te descargara un paquete llamado Django-1.3.7.tar.gz lo descomprimes en algun directorio luego abres la Terminal y te posicionas sobre el directorio donde descomprimiste y ejecutas:

```
$ python setup.py install
```

Sino mediante el instalador de Paquetes de Python de manera mas automatica escribes en la terminal

```
pip install django==1.3.7
```

Con esto ya tendremos instalado Django.

## 5.5. Instalando el Resto de Las Dependencias

Ademas de Django en el Proyecto se utilizaron otras Librerias de Python las cuales algunas vienen instaladas y Otras Requieren ser instaladas de manera similar a como instalamos Django.

### 5.5.1. psycopg2

psycopg2 es un adaptador de base de datos PostgreSQL para el lenguaje de programación Python. psycopg2 fue escrito con el objetivo de ser muy pequeño y rápido y estable.

psycopg2 es diferente del otro adaptador de base de datos, ya que fue diseñado para aplicaciones en gran medida de subprocessos múltiples que crean y destruyen un montón de cursor y hacen que un número notable de inserciones o actualizaciones concurrentes. psycopg2 también proporcionan operaciones asincrónicas completos y apoyo a las bibliotecas de co-rutinas.

Para instalar descargue el precompilado desde <http://www.stickpeople.com/projects/python/win-psycopg/> Ejecutelo con permisos de administrador, nos pedira que seleccionemos la version de python con que se instalar.

### 5.5.2. ReportLab

ReportLab es la ultra-robusto motor de código abierto a prueba de tiempo para la creación de documentos PDF y gráficos vectoriales personalizado. Escrito en Python, ReportLab es rápido, flexible y una plataforma cruzada.

Proporciona un completo conjunto de herramientas de programación para la creación de documentos y gráficos complejos. Ofrecemos una serie de componentes de forma gratuita y de código abierto, además de un paquete comercial con características adicionales.

Para Instalar descargue el instalado desde <http://www.reportlab.com/software/installation/> y proceda de manera similar a como hizo con la instalacion de psycopg2.

### 5.5.3. easy-thumbnails

Easy\_Thumbnails es Una potente aplicación thumbnailing <sup>2</sup>, pero fácil de implementacion para Django.

Para Instalar solo ejecute el siguiente comando en terminal, no Se necesita configurar nada en el proyecto el mismo esta previamente configurado.

```
pip install easy-thumbnails
```

<sup>1</sup>la version 1.3.7 no es la ultima version disponible a la hora de crear este informe estaba por la 1.6.2 ya que Django se actualiza constantemente.

<sup>2</sup>Cuando hablamos de thumbnails nos referimos a las diferentes miniaturas que son versiones en distintos tamaños de una imagen y son usadas para ayudar a su organización y reconocimiento.

#### 5.5.4. django\_extensions

Django\_Extensions es una colección de Extensiones (utilidades) Personalizadas de diferentes autores no relacionados con el Proyecto Django, para extender las capacidades del Framework.

Para Instalar solo ejecute el siguiente comando en terminal <sup>3</sup>

```
pip install django-extensions
```

#### 5.5.5. django\_cron

Django-cron permite ejecutar código de Django de manera recurrente para el seguimiento y ejecución de las tareas. En este caso no es Necesario Instalar Nada, viene junto con el Código Fuente del Proyecto. Igualmente si tiene curiosidad puede visitar la pagina del proyecto <https://github.com/Tivix/django-cron>

#### 5.5.6. Descargar e Instalacion de mod\_wsgi

Asumiendo que ya tienes instalado Python y Apache, solo debes descargar el paquete libapache2-mod-wsgi ,la ultima version de mod\_wsgi se puede descargar desde su pagina oficial <https://code.google.com/p/modwsgi/> descargaran un archivo similar a "mod\_wsgi-win32-ap22py27-3.3.so" la version que descarguen de mod\_wsgi depende como se ve, de la plataforma asi como de la version de python que correran en el servidor. luego por cuestiones de practicidad renombraremos el archivo de la siguiente manera:

```
mod_wsgi-win32-ap22py27-3.3.so -> mod_wsgi.so
```

Realizado dicho cambio copiamos el modulo dentro de la siguiente carpeta: APACHE\_FOLDER \modules \APACHE\_FOLDER vendria a ser el directorio donde tenemos la instalacion de WAMP en mi caso es: C:\Apache.

#### 5.5.7. Cargando el Modulo en Apache

Una vez que el módulo de Apache ha sido instalado en el directorio de módulos de su instalación de Apache, todavía es necesario configurar Apache para cargar el módulo en realidad.

Abrimos el archivo "httpd.conf" agregamos la siguiente linea en el mismo punto donde se cargan el resto de los modulos. <sup>4</sup>

```
LoadModule wsgi_module modules/mod_wsgi.so
```

Con todo esto echo solo tenemos que reiniciar el servidor Apache, en nuestro caso clic en el icono en la barra de notificaciones luego las opciones Apache->Service->Reiniciar Servicio.

#### 5.5.8. Configuracion del Proyecto

Bueno Ahora solo tenemos que crear un alias en Apache <sup>5</sup> para nuestra carpeta donde colocaremos en mi caso la carpeta destino sera:

```
C:\Servidor\SGCM
```

SGCM es la carpeta contenedora del proyecto, y el alias que usaremos sera:

```
/sgcm/
```

<sup>3</sup>Importante, no todas las funcionalidades estan soportadas en Windows, pero en cuanto al proyecto no hay problemas.

<sup>4</sup>El archivo httpd.conf esta en la siguiente ruta en el caso de mi instalacion: C:\Apache\conf\httpd.conf

<sup>5</sup>Para mayor informacion de como crear alias en Apache consulte <http://httpd.apache.org/docs/2.2/urlmapping.html>

tendremos que agregar las siguientes lineas al final del archivo httpd.conf de apache.

```
Alias /sgcm/ "C:/Servidor/SGCM/"
WSGIScriptAlias /sgcm "C:/Servidor/SGCM/handle.wsgi"

<Directory "C:/Servidor/SGCM">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride all
    Order allow,deny
    Allow from all
</Directory>
```

Hay un número de maneras en que una aplicación WSGI organizada por mod\_wsgi<sup>6</sup> puede montarse contra una URL específica. Estos métodos son similares a cómo se podría configurar las aplicaciones CGI tradicionales.

El principal enfoque implica declarar explícitamente en el archivo de configuración principal de Apache el punto de montaje URL y una referencia al archivo de comandos de aplicaciones WSGI. En este caso, el mapeo se fija, con cambios sólo ser capaz de ser hecho mediante la modificación de la configuración principal de Apache y reiniciar Apache.

Al utilizar mod\_cgi para alojar aplicaciones CGI, esto se haría mediante la directiva ScriptAlias. Para mod\_wsgi, la directiva en su lugar se llama WSGIScriptAlias.

```
WSGIScriptAlias /wsgi "C:/Servidor/SGCM/handle.wsgi"
```

Esta directiva sólo puede aparecer en los principales archivos de configuración de Apache. La directiva se puede utilizar en el ámbito del servidor, pero normalmente se coloca en el contenedor VirtualHost para un sitio en particular. No se puede utilizar en cualquiera de las directivas de contenedores ubicación, directorios o archivos, ni puede ser utilizada dentro de un archivo ".htaccess". El primer argumento de la directiva WSGIScriptAlias debe ser el punto de montaje URL para la aplicación WSGI. En este caso, la URL no debe contener una barra diagonal. La única excepción a esto es si la aplicación WSGI es para ser montado en la raíz del servidor web, en cuyo caso / sería utilizado.

El segundo argumento de la directiva WSGIScriptAlias debe ser una ruta absoluta para el archivo de comandos de aplicaciones WSGI. Es en este archivo que la muestra de código de la aplicación WSGI debe colocarse.

Tenga en cuenta que una ruta absoluta debe ser utilizado para el archivo de comandos de aplicaciones WSGI suministrado como segundo argumento. No es posible especificar una aplicación por sí sola Python nombre de módulo. Una ruta de acceso completa se utiliza para una serie de razones, la principal de las cuales por lo que todos los controles de acceso de Apache todavía pueden aplicarse para indicar que en realidad puede acceder a la aplicación WSGI.

Porque se aplicarán los controles de acceso de Apache, si la aplicación WSGI se encuentra fuera de los directorios que ya están configurados para ser accesible a Apache, habrá que decirle a Apache que los archivos dentro de ese directorio se pueden utilizar. Para ello se debe utilizar la directiva Directory.

Hasta aqui tenemos mod\_wsgi y nuestro directorio listo, ahora probaremos que todo va bien para ello dentro del directorio crearemos un archivo llamado "handle.wsgi" que tendrá el siguiente contenido:

```
1 # -*- coding: utf-8 -*-
2
3 import os, sys
4 import django.core.handlers.wsgi
5
6 sys.path.append('C:/Servidor/SGCM')
7 sys.path.append('C:/Servidor')
```

<sup>6</sup>Puede consultar <https://code.google.com/p/modwsgi/wiki/QuickInstallationGuide> si desea explorar otras opciones de configuración.

```
8  
9 os.environ['DJANGO_SETTINGS_MODULE'] = 'settings'  
10  
11 application = django.core.handlers.wsgi.WSGIHandler()
```

Con esto nuestro servidor de aplicacion ya deberia funcionar aunque como veran no se cargan los archivos estaticos como imagenes y hojas de estilo por lo que necesitamos agregarlo.

# Bibliografía

- [1] WIKIPEDIA *Servidor HTTP Apache* [http://es.wikipedia.org/wiki/Servidor\\_HTTP\\_Apache](http://es.wikipedia.org/wiki/Servidor_HTTP_Apache)
- [2] *Servidor Apache en Ubuntu* <http://kuyne.blogspot.com.ar/2013/03/servidor-apache-en-ubuntu-instalacion-y.html>
- [3] *Servidor Apache en Windows* <http://norfipc.com/internet/instalar-servidor-apache.html>
- [4] WIKIPEDIA *El Modelo Vista Controlador* [http://es.wikipedia.org/wiki/Modelo\\_Vista\\_Controlador](http://es.wikipedia.org/wiki/Modelo_Vista_Controlador)
- [5] ROGELIO LEÓN LÓPEZ, BÁRBARA GALLEGOS MACHADO Y JOSÉ DÍAZ NOVÁS  
*Formato recomendable para llenar la hoja de remisión médica de un paciente*  
[http://bvs.sld.cu/revistas/mgi/vol22\\_2\\_06/mgi10206.htm](http://bvs.sld.cu/revistas/mgi/vol22_2_06/mgi10206.htm)
- [6] SISTEMA CONSULTORIO WEB *Registro Consulta*  
<https://www.consultorioweb.com/intranet/doctor/pacienteConsultas.aspx>
- [7] PRACTICA FINAL OBLIGATORIA: INTERNADO ROTATORIO Y PASANTIA RURAL OBLIGATORIA *Modelo Historia Clinica*  
[http://www.med.unne.edu.ar/internado/his\\_cli.pdf](http://www.med.unne.edu.ar/internado/his_cli.pdf)
- [8] ANY FLOWERS *Modelo Historia Clinica*  
<http://www.slideshare.net/AnyFlowers/ejemplo-historia-clinica>
- [9] BICOM *Formato de Historia Clinica*  
[http://www.biocom.com/informatica\\_medica/historia\\_5\\_examen\\_fisico.html](http://www.biocom.com/informatica_medica/historia_5_examen_fisico.html)
- [10] INFOMED RED DE SALUD DE CUBA *Examen Fisico Regional*  
<http://www.sld.cu/galerias/pdf/sitios/pdguanabo/cap04.pdf>
- [11] ESMAS *Problemas Auditivos Comunes*  
<http://www.esmas.com/salud/enfermedades/notransmisibles/368755.html>
- [12] WIKIPEDIA *Perdida de Audicion*  
[http://es.wikipedia.org/wiki/P%C3%A9rdida\\_de\\_audiencia](http://es.wikipedia.org/wiki/P%C3%A9rdida_de_audiencia)
- [13] HERNANDO VARGAS VÁSQUEZ *Rinología*  
[http://sisbib.unmsm.edu.pe/BibVirtualdata/libros/Medicina/cirugia/Tomo\\_V/archivos%20PDF/7Rinolog](http://sisbib.unmsm.edu.pe/BibVirtualdata/libros/Medicina/cirugia/Tomo_V/archivos%20PDF/7Rinolog)
- [14] WIKIPEDIA *Examen Labios* [http://es.wikipedia.org/wiki/Examen\\_Labios](http://es.wikipedia.org/wiki/Examen_Labios)
- [15] CONSUMOTeca *Qué partes deben tener y datos incluir por ley las recetas médicas*  
<http://www.consumoteca.com/bienestar-y-salud/medicamentos/que-partes-deben-tener-y-datos-incluir-por-ley-las-recetas-medicas/>

- [16] WIKIPEDIA *Vias de Administracion de Farmacos*  
[http://es.wikipedia.org/wiki/V%C3%ADas\\_de\\_administraci%C3%B3n\\_de\\_f%C3%A1rmacos](http://es.wikipedia.org/wiki/V%C3%ADas_de_administraci%C3%B3n_de_f%C3%A1rmacos)
- [17] PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE - ESCUELA DE MEDICINA *Respiracion*  
<http://escuela.med.puc.cl/Publ/ManualSemiologia/190Respiracion.htm>
- [18] BIOCOP *Historia Clinica - Examen Fisico*  
[http://www.biocom.com/informatica\\_medica/historia\\_5\\_examen\\_fisico.html](http://www.biocom.com/informatica_medica/historia_5_examen_fisico.html)
- [19] *Examen Fisico del Sistema Osteomioarticular*  
<http://www.slideshare.net/wendy1971/examen-fisico-del-sistema-osteomioarticular>
- [20] MODWSGI *Guia de Configuracion*  
<https://code.google.com/p/modwsgi/wiki/QuickConfigurationGuide>
- [21] WSGI *Guia de Referencia WSGI - EN* <http://wsgi.readthedocs.org/en/latest/>
- [22] APACHE *URL Mapping* <http://httpd.apache.org/docs/2.2/urlmapping.html>
- [23] MINISTERIO DE SALUD *Formato de Historia Clinica*  
[http://msal.gov.ar/ENT/SRV/Materiales\\_Paciente/Herramientas\\_Utiles/Historia\\_Clinica/Historia\\_Clinica.aspx](http://msal.gov.ar/ENT/SRV/Materiales_Paciente/Herramientas_Utiles/Historia_Clinica/Historia_Clinica.aspx)