SISTEMA DE REGISTRO MEDICO PARA CLÍNICA DE MEDICINA INTEGRAL

Manual Técnico

Versión: 1.0

Fecha: 15/01/2018

Proyecto	Sistema de Registro Medico para Clínica de Medicina Integral		
Entregable	Manual Técnico		
Versión/Edición	1.0	**Fecha Versión**	16/01/2018
Aprobado por	Carlos Cativo	**Fecha Aprobación**	18/01/2018
		N.º Total de Páginas	8

Nombre y Apellidos	
Cativo Argueta Carlos Antonio	**CA14002**
Granados Orellana Mauricio Ernesto	**GO14002**
Granados Cruz German Steven	**GC14003**
Navarro Alfaro Melanie Carolina	**NA15004**
Oliva Avelar Irving Vladimir	**OA09020**
Orellana Zepeda Dennys Miguel Ángel	**OZ09001**

Contenido

INTRODUCCION....3
Objetivos el documento....4
Objetivo General:....4
Objetivos Específicos:....4
Generalidades del Sistema....4
Diseño de la base de datos. 4
Descripción de la base de datos. 4
Tabla 1. Paciente....4
Tabla 2. Expediente 4
Tabla 3. Antecedente....5
Tabla 4. Consulta Médica....5
Modelo conceptual y físico de la base de datos. 6
Modelo conceptual 6
Modelo Físico....6
Generación de consultas a la base de datos. 7
Descripción de las funciones esenciales del sistema. 8

INTRODUCCION.

El presente documento está orientado a describir cada una de las partes internas que posee el Sistema informático para la administración de la clínica de medicina integral, a fin de que sea de ayuda al momento de dar mantenimiento a la misma. En el documento se describen generalidades del sistema, diagramas de bases de datos, explicación de las funciones más importantes, librerías y complementos utilizados para mejorar el funcionamiento del sistema, validaciones, y otra información útil para comprender el funcionamiento interno.

Objetivos el documento

Objetivo General:

- Mostrar los componentes técnicos que componen el Sistema informático para la administración de la clínica de medicina integral.

Objetivos Específicos:

- Presentar diseño de la base de datos sobre la cual el sistema informático estará trabajando.
- Mostrar el contenido de los archivos y carpetas más importantes de la aplicación
- Describir las funciones más importantes del sistema a fin de comprender el código y poder obtener un correcto mantenimiento.

Generalidades del Sistema

Nombre del Sistema: Sistema informático para la administración de la clínica de medicina integral.

Versión del Sistema: 1.0

Lenguaje y Framework: Lenguaje Python, Framework Django:1.11.

Sistema gestor de base de datos: MySQL.

Diseño de la base de datos.

Descripción de la base de datos.

Tabla 1. Paciente.

Nombre Columna	Tipo de Dato	Null	Clave Primaria	Descripción
Id_paciente	Integer	No	Si	Clave primaria
Nombre_paciente	Varchar(40)	No	No	Nombre del Paciente
Apellido_paciente	Varchar(40)	No	No	Apellido del paciente
Fecha_nacimiento	Date	No	No	Fecha de nacimiento del paciente
Teléfono_fijo	Varchar(9)	No	No	Teléfono del paciente
direccion	Varchar(80)	No	no	Direccion donde reside el paciente

Tabla 2. Expediente

Nombre Columna	Tipo de Dato	Null	Clave Primaria	Descripción
Id_expediente	Integer	No	Si	Clave primaria
Fecha_apertura	Date	No	No	Fecha de apertura del expediente

Tabla 3. Antecedente

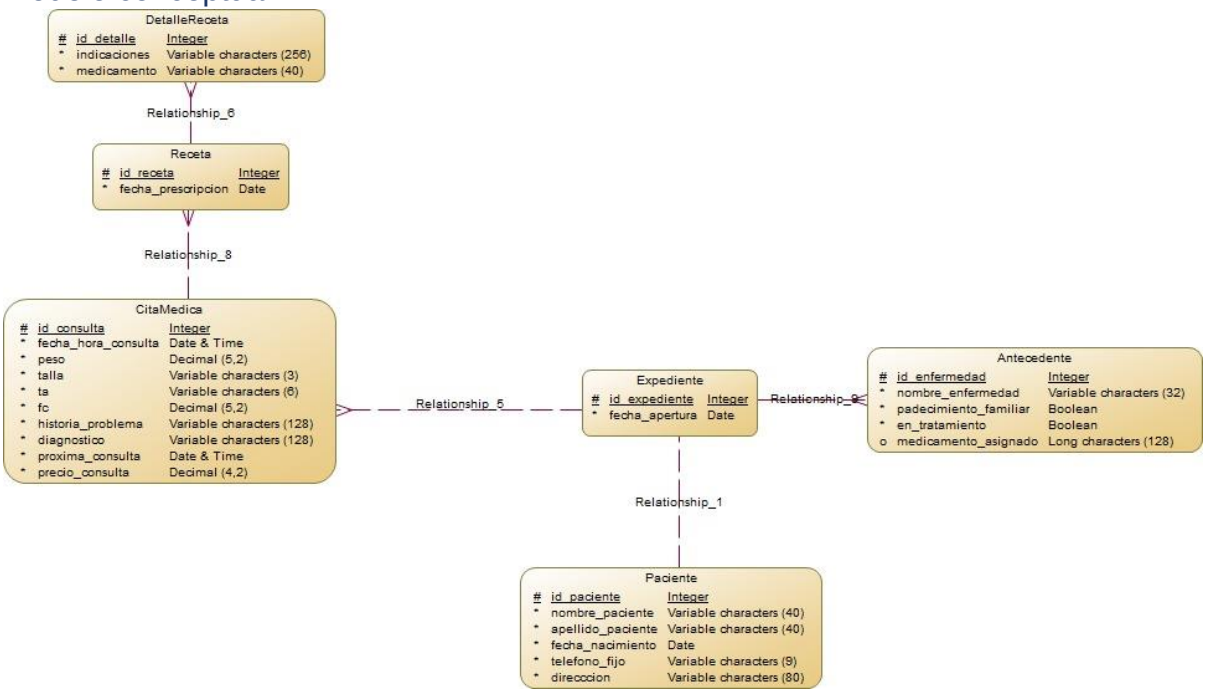
Nombre Columna	Tipo de Dato	Null	Clave Primaria	Descripción
Id_enfermedad	Integer	No	Si	Clave primaria
Nombre_enfermedad	Varchar(32)	No	No	Nombre de la enfermedad
Familiar_enfermedad	Varchar(32)	No	No	Familiar con la enfermedad
En_tratamiento	Varchar(32)	No	No	Tratamiento de la enfermedad
Medicamento_asignado	Varchar(50)	No	No	Medicamento asignado.

Tabla 4. Consulta Médica.

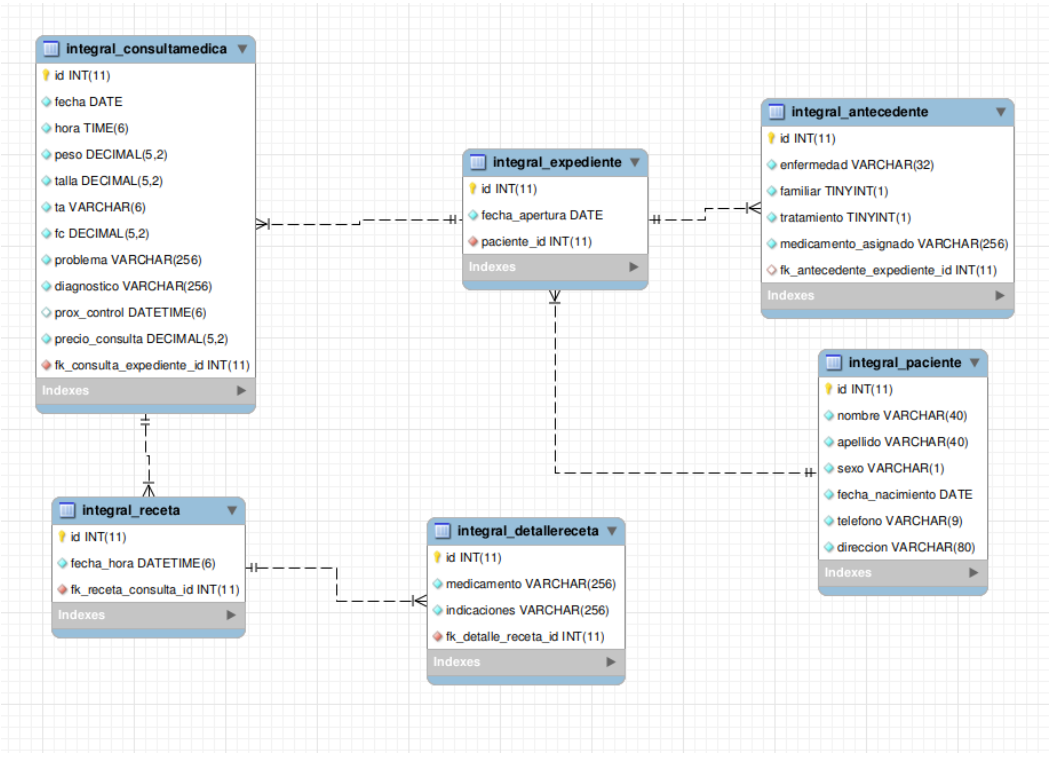
Nombre Columna	Tipo de Dato	Null	Clave Primaria	Descripción
Id_consulta	Integer	No	Si	Clave primaria
Fecha_consulta	Date	No	No	Fecha de la consulta
Hora_consulta	Time	No	No	Hora de la consulta
Peso	Decimal (5, 2)	No	No	Peso del paciente
Talla	Decimal (5, 2)	No	No	Talla del paciente
Ta	Varchar(4)	No	No	Tension arterial
Fc	Decimal (5, 2)	No	No	Frecuencia cardiaca
Problema	Varchar(128)	No	No	Problema motivo de consulta
Diagnostico	Varchar(128)	No	No	Diagnóstico de la consulta
Próxima_consulta	Date &Time	No	No	Proxima consulta
Precio_consulta	Decimal (4,2)	No	No	Precio de consulta

Modelo conceptual y físico de la base de datos.

Modelo conceptual



Modelo Físico.



Generación de consultas a la base de datos.

La facilidad que nos da el framework Django para manejar objetos, se aplica muy bien a las consultas, ya que es muy sencillo obtener todos los objetos que se necesitan y filtrarlos por medio de diversos parámetros, teniendo eso aplicando el patrón de diseño seguido (MVT -Model View Témplate), se envía el contexto obtenido del filtrado a un témplate donde será desplegado a través de ciclos de control.

```
@login_required
def Consulta_Medica(request,expedienteId):
        consulta = ConsultaMedica.objects.filter(fk_consulta_expediente=expedienteId)
        expediente= expedienteId
        return render(request, 'integral/Consulta_Expediente.html',{'consultaMedica':consulta,'expedienteId':expediente} )
```

Obteniendo todos los objetos de consulta y filtrándolos en base al id de un expediente para solo obtener las consultas pertinentes a ese expediente, luego teniendo el contexto necesario se envía al témplate *Consulta_Expediente.html* para ser presentado.

```
{% for consulta in consultaMedica%}
<tr>
<td>{{consulta.id}}</td>
<td>{{consulta.fecha}}</td>
<td><a class="button" href="/detalleConsulta/{{consulta.id}}/{{expedienteId}}">Detalle Consulta</a></td>
<td><a class="button" href="/nuevaReceta/{{consulta.id}}">Asignar Nueva Receta</a></td>
</tr>
{% endfor %}
```

De igual forma para los demás objetos como recetas y antecedentes las consultas se hacen de esa manera.

Otros ejemplos de consulta (Listado de Pacientes con ListView).

```
class PacienteList(LoginRequiredMixin, ListView):
        model = Paciente
```

Vista basada en clase en el views.py

Ahora pasando a la parte de paciente_list.html

```
{% for paciente in object_list %}
        <tr>
          <td>{{ paciente.nombre }}</td>
          <td>{{ paciente.fecha_nacimiento}}</td>
          <td>{{ paciente.telefono}}</td>
          <td>{{ paciente.direccion}}</td>
          <td><a class="button" href="{% url 'integral:detallePaciente' paciente.id %}">Ver</a></td>
          <td><a class="button" href="{% url 'integral:eliminarPaciente' paciente.id %}">Eliminar</a></td>
        </tr>
        {% endfor %}
```

Llenado de datos por medio de un for en el template.

Descripción de las funciones esenciales del sistema.

Función:	**nuevoPaciente**
Objetivos:	crear un nuevo paciente en el sistema de registro medico
Ubicación:	Src/integral/views.py
Parámetros:	Variable de tipo request que toma los valores de un formulario si el método es POST

```
@login_required
def nuevoPaciente(request):
        pacientes = Paciente.objects.all()
        paginator = Paginator(pacientes,15)
        page = request.GET.get('page')
        try:
                pacientes = paginator.page(page)
        except PageNotAnInteger:
                pacientes = paginator.page(1)
        except EmptyPage:
                pacientes = paginator.page(paginator.num_pages)
        if request.method == 'POST':
                form = nuevoPacienteForm(request.POST)
                if form.is_valid():
                        form.save()
                        ultimoPaciente = Paciente.objects.first()
                        Expediente.objects.create(paciente = ultimoPaciente)
                        messages.success(request, 'Paciente correctamente guardado!')
                        return redirect('integral:listarPaciente')
                else:
                        data = json.dumps([v for k , v in form.errors.items()]+[' ¡Error!'])
                        return HttpResponse(data, content_type='application/json')
        else:
                form = nuevoPacienteForm()
        return render(request, 'integral/nuevoPaciente.html', {'form':
form,'pacientes':pacientes})
```

Función:	**Expediente**
Objetivos:	crear un nuevo expediente al momento de crear un paciente nuevo y asignárselo
Ubicación:	Src/integral/views.py
Parámetros:	Variable de tipo request que toma los valores de un formulario si el método es POST

```
@login_required
def expediente(request):
        expedientes = Expediente.objects.all()
        paginator = Paginator(expedientes,15)
        page = request.GET.get('page')
        try:
                expedientes = paginator.page(page)
        except PageNotAnInteger:
                expedientes = paginator.page(1)
        except EmptyPage:
                expedientes = paginator.page(paginator.num_pages)
```

```
        if request.method == 'POST':
                form = nuevoExpedienteForm(request.POST)
                if form.is_valid():
                        form.save()
                        messages.success(request, 'Correctamente guardado!')
                        return HttpResponse("GUARDADO!")
                else:
                   data = json.dumps([v for k , v in form.errors.items()]+[' ¡Error!'])
                return HttpResponse(data, content_type='application/json')
        else:
                form = nuevoExpedienteForm()
        return render(request,
'integral/expediente/expediente.html',{'form':form,'expedientes':expedientes})
```

Función:	**NuevaConsulta**
Objetivos:	crear una nueva consulta medica para “x” expediente en el que se este trabajando
Ubicación:	Src/integral/views.py
Parámetros:	Variable de tipo request que toma los valores de un formulario si el método es POST

```
@login_required
def nuevaConsulta(request,expedienteId):
        expediente= expedienteId
        datos = Paciente.objects.get(id=expedienteId)
        if request.method == 'POST':
                ConsultaMedica.objects.create(
                        fecha=request.POST['fechaActual'],
                        hora=request.POST['horaActual'],
                        peso=request.POST['PesoPaciente'],
                        talla=request.POST['TallaPaciente'],
                        ta=request.POST['TensionArterial'],
                        fc=request.POST['FrecuenciaCardiaca'],
                        problema=request.POST['ProblemaEnfermedad'],
                        diagnostico=request.POST['DiagnosticoEnfermedad'],
                        fecha_prox_control=request.POST['fechaProxActual'],
                        hora_prox_control =request.POST['horaProxActual'],
                        precio_consulta=request.POST['PrecioConsulta'],

        fk_consulta_expediente=Expediente.objects.get(id=request.POST['expediente'])
                )
                messages.success(request, 'Correctamente guardado!')
                return HttpResponse('Datos Almacenados Correctamente')
        return render(request,'integral/nuevaConsulta.html',{'expedienteId':expediente,
'paciente':datos})
```

Función:	**NuevaReceta**
Objetivos:	crear una nueva receta médica para “x” consulta en el que se esté trabajando
Ubicación:	Src/integral/views.py
Parámetros:	Variable de tipo request que toma los valores de un formulario si el método es POST

```
@login_required
def nuevaReceta(request,consultaId):
        consulta = consultaId
        datos = ConsultaMedica.objects.get(id=consultaId)
        print(consultaId)
        if request.method == 'POST':
                Receta.objects.create(
                        fk_receta_consulta=ConsultaMedica.objects.get(id=consultaId),
                        fecha=request.POST['fechaActual'],
                        hora=request.POST['horaActual'],
                        medicamento=request.POST['MedicamentoPaciente'],
                        indicaciones=request.POST['IndicacionesPaciente'],
                )
                messages.success(request, 'Correctamente guardado!')
                return HttpResponse('Datos Almacenados Correctamente')
        return render(request,'integral/nuevaReceta.html',{'consultaId':consulta,
'consulta':datos})
```

Función:	**NuevoAntecedente**
Objetivos:	crear un nuevo antecedente para ser agregado al expediente de un paciente
Ubicación:	Src/integral/views.py
Parámetros:	Variable de tipo request que toma los valores de un formulario si el método es POST

```
@login_required
def nuevoAntecedente(request,expedienteId):
        expediente = expedienteId
        if request.method == 'POST':
                Antecedente.objects.create(

        fk_antecedente_expediente=Expediente.objects.get(id=expedienteId),
                        enfermedad=request.POST['enfermedad'],
                        familiar=request.POST['familiar'],
                        tratamiento=request.POST['tratamiento'],
                        medicamento_asignado=request.POST['medicamento_asignado'],
                )
                messages.success(request, 'Correctamente guardado!')
                return HttpResponse('Datos Almacenados Correctamente')
        return
render(request,'integral/nuevoAntecedente.html',{'expedienteId':expediente})
```