Crear una nueva aplicación en Awakening

Tabla de contenido

[Crear Modelo para guardar los datos. 1](#_Toc157604258)

[Formulario para el modelo: 2](#_Toc157604259)

[Agregando página HTML para un template: 2](#_Toc157604260)

[Agregando Views 3](#_Toc157604261)

[Visualización de Registros 3](#_Toc157604262)

[Ingreso de registro Nuevo 4](#_Toc157604263)

[Actualizar registro ya creado 5](#_Toc157604264)

[Eliminar un registro ya creado 5](#_Toc157604265)

[Configurando Rutas para URL 6](#_Toc157604266)

[Agregar el acceso a menú General: 7](#_Toc157604267)

## Crear Modelo para guardar los datos.

Escribir el modelo en el archivo models.py puede ser en el siguiente formato:

class Rhmotiv(models.Model):  
 codmot = models.IntegerField(primary\_key=True)  
 # agecod = models.ForeignKey(Rhagencia, on\_delete=models.CASCADE)  
 nommot = models.CharField(max\_length=100, help\_text='Nombres de Empleados')  
  
 def \_\_str\_\_(self):  
 return '{}:{}'.format(self.codmot, self.nommot)  
  
 def save(self):  
 super(Rhmotiv, self).save()  
  
 class Meta:  
 verbose\_name\_plural = 'Rhmotiv'  
 unique\_together = ('codmot', 'nommot')

al finalizar de escribir el modelo ejecutar los siguientes comandos desde la consola de la aplicación para sincronizar los modelos con la base de datos:

**py.exe .\manage.py makemigrations**

**py.exe .\manage.py migrate**

## Formulario para el modelo:

Para agregar el formulario para poder ingresar un registro nuevo o también para actualizar se debe de editar el archivo **core/forms.py**, copiar y pegar el código y editar el modelo, nombre de campos y las etiquetas para los campos.

class Rhmotiv(forms.ModelForm):  
 class Meta:  
 model=Rhmotiv # acá se agrega el modelo creado anteriormente  
 fields=['codmot','nommot'] # acá se agregan los campos a ingresar  
 labels={  
 'codmot':'Codigo de Motivo',  
 'nommot':'Descripcion de Motivo'} # acá se agregan el campo y la etiqueta asociada  
 widget={'Cod.':forms.TextInput(), 'Descripcion':forms.TextInput()}

#Aca se agrega el tipo de widget que se utilizará para mostrar en la página en html  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args,\*\*kwargs)  
 for field in iter(self.fields):  
 self.fields[field].widget.attrs.update({  
 'class': 'form-control'  
 })

\*Recuerde agregar el modelo en la parte de import del archivo.

from django import forms  
from core.models import Rhempleado, Rhmotiv

## Agregando página HTML para un template:

Agregar el siguiente código para generar la vista del formulario, este servirá solo para mostrar la información, colocarlo en la carpeta **core/templates/core/rhmotiv\_list.html**.

{% extends 'base/base.html' %}  
{% block contenido %}  
 <div class="panel panel-primary">  
 <div class="panel-heading">  
 <a href="{% url 'core:rhmotiv\_new' %}" class="btn btn-info"><span class="fa fa-plus-circle"></span>  
 Nueva</a>  
 </div>  
 <div class="panel-body">  
 <table width="100%" class="table table-striped table-bordered table-hover" id="dataTables-example">  
 <thead>  
 <tr>  
 <th>Codigo Motivo</th>  
 <th>Descripcion Motivo</th>  
 <th>Acciones</th>  
 </tr>  
 </thead>  
 <tbody> {% for item in obj %}  
 <tr>  
 <td>{{ item.codmot }}</td>  
 <td>{{ item.nommot }}</td>  
 <td>  
 <a href="{% url 'core:rhmotiv\_edit' item.codmot %}" class="btn btn-primary btn-circle"><span class="fa fa-edit"></span></a>  
 </td>  
 </tr> {% endfor %} </tbody>  
 </table>  
 </div>  
 </div>  
{% endblock contenido %}

\*Verifique el nombre de la llave primaria, en este caso se llama codmot

## Agregando Views

Para agregar un nuevo view se tiene que agregar nuevo código para el formulario y modelo anterior, esto quiere decir que para poder completar un modelo completo se necesitan de 4 vistas las cuales son:

1. Visualización de registros (listar registros).
2. Ingreso de registro nuevo.
3. Actualización de registro ya almacenado.
4. Eliminación de registro ya almacenado.

Para lo cual existen 4 tipos de vista genéricos que se deben agregar en el archivo **core/views.py** que se muestran a continuación.

\*Agregar las importaciones de los modelos en el archivo **core/views.py**

from django.urls import reverse\_lazy  
from django.views import generic  
  
from django.contrib.messages.views import SuccessMessageMixin  
from django.contrib.auth.mixins import LoginRequiredMixin  
  
from core.models import Rhagencia, Rhempleado, Rhmotiv  
from core.forms import Empleadoform, Rhmotiv\_Form  
from generales.views import SinPrivilegios

## Visualización de Registros

Para visualizar registros se usa el siguiente código, verificar el texto marcado en amarillo y agregarlo en **core/views.py**

class Motivosview(LoginRequiredMixin, generic.ListView):  
 model = Rhmotiv  
 template\_name = 'core/rhmotiv\_list.html'  
 context\_object\_name = "obj"  
 login\_url = 'generales:login'

y Guardar la vista rhmotiv\_list.html en la ubicación core/templates/core/rhmotiv\_list.html con el siguiente código:

{% extends 'base/base.html' %}  
{% block contenido %}  
 <div class="panel panel-primary">  
 <div class="panel-heading">  
 <a href="{% url 'core:rhmotiv\_new' %}" class="btn btn-info"><span class="fa fa-plus-circle"></span>  
 Nueva</a>  
 </div>  
 <div class="panel-body">  
 <table width="100%" class="table table-striped table-bordered table-hover" id="dataTables-example">  
 <thead>  
 <tr>  
 <th>Codigo Motivo</th>  
 <th>Descripcion Motivo</th>  
 <th>Acciones</th>  
 </tr>  
 </thead>  
 <tbody> {% for item in obj %}  
 <tr>  
 <td>{{ item.codmot }}</td>  
 <td>{{ item.nommot }}</td>  
 <td>  
 <a href="{% url 'core:rhmotiv\_edit' item.codmot %}" class="btn btn-primary btn-circle"><span class="fa fa-edit"></span></a>  
 <a href="{% url 'core:rhmotiv\_del' item.codmot %}" class="btn btn-danger btn-circle"><span class="fa fa-trash"></span></a>  
 </td>  
 </tr> {% endfor %} </tbody>  
 </table>  
 </div>  
 </div>  
{% endblock contenido %}

## Ingreso de registro Nuevo

Para Ingreso de un nuevo registro se utiliza el siguiente código, verificar el código en amarillo y agregarlo en **core/views.py**.

class MotivoNew(SuccessMessageMixin, LoginRequiredMixin, SinPrivilegios, generic.CreateView):  
 permission\_required = "core.add\_rhmotiv"  
 model = Rhmotiv  
 template\_name = "core/rhmotiv\_form.html"  
 context\_object\_name = 'obj'  
 form\_class = Rhmotiv\_Form  
 success\_url = reverse\_lazy("core:rhmotiv\_list")  
 success\_message = "Motivo Creado Satisfactoriamente"

y guardar la vista con nombre rhmotiv\_form.html en la ubicación **core/templates/core/** con el código a continuación.

{% extends 'base/base.html' %}  
  
{% block contenido %}  
 <form method="POST">  
 <div class="panel panel-default">  
 <div class="panel-heading">  
 Editar/Crear Motivos  
 </div>  
 <div class="panel-body">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit" class="btn btn-success"><span class="fa fa-save"></span> Guardar</button>  
 <a href="{% url 'core:rhmotiv\_list' %}" class="btn btn-danger"><span class="fa fa-undo"></span> Cancelar</a>  
 </div>  
 </div>  
 </form>  
  
{% endblock contenido %}

\*Hay que tener en cuenta que este mismo formulario se va a utilizar para **actualizar** los datos del modelo.

## Actualizar registro ya creado

Para actualizer un registro ya creado se utiliza el siguiente código, hay que aclarar que se utiliza el mismo código en form class y template name ya que django automatiza la forma en que se reutiliza el formulario. (Verificar el código en amarillo) y agregarlo en **core/views.py**.

class MotivoEdit(SuccessMessageMixin, LoginRequiredMixin, SinPrivilegios, generic.UpdateView):  
 permission\_required = "core.add\_rhmotiv"  
 model = Rhmotiv  
 template\_name = 'core/rhmotiv\_form.html'  
 context\_object\_name = 'obj'  
 form\_class = Rhmotiv\_Form  
 success\_url = reverse\_lazy("core:rhmotiv\_list")  
 success\_message = "Motivo Actualizado Satisfactoriamente"

\*El código de la vista es exactamente el mismo que al ingresar un nuevo registro, rhmotiv\_form.html.

## Eliminar un registro ya creado

Para eliminar un registro hay que agregar el código en el archivo core/views.py

class MotivoDel(SuccessMessageMixin, LoginRequiredMixin, SinPrivilegios, generic.DeleteView):  
 permission\_required = "core.add\_rhmotiv"  
 model=Rhmotiv  
 template\_name="core/rhmotiv\_del.html"  
 context\_object\_name = 'obj'  
 success\_url = reverse\_lazy("core:rhmotiv\_list")

y guardar la vista con el siguiente nombre rhmotiv\_del.html en la ubicación core**/templates/core/** y agregar el siguiente código.

{% extends 'base/base.html' %}  
{% block contenido %}  
<form method="POST">  
 <div class="panel panel-default">  
 <div class="panel-heading">  
 Eliminar  
 </div>  
 <!-- /.panel-heading -->  
 <div class="panel-body">  
 {% csrf\_token %}  
 <div class="alert alert-success">  
 ¿Eliminar {{ obj.nommot }} ?  
 </div>  
 <button type="submit" class="btn btn-danger"><span class="fa fa-save"></span> Si, Eliminar</button>  
 <a href="{% url 'core:rhmotiv\_list' %}" class="btn btn-success"><span class="fa fa-undo"></span>Cancelar</a>  
 </div>  
 </div>  
 </form>  
{% endblock contenido %}

Además, el archivo de rhmotiv\_list agregarle los botones de editar y eliminar.

Agregando el siguiente código.

{% extends 'base/base.html' %}  
{% block contenido %}  
 <div class="panel panel-primary">  
 <div class="panel-heading">  
 <a href="{% url 'core:rhmotiv\_new' %}" class="btn btn-info"><span class="fa fa-plus-circle"></span>  
 Nueva</a>  
 </div>  
 <div class="panel-body">  
 <table width="100%" class="table table-striped table-bordered table-hover" id="dataTables-example">  
 <thead>  
 <tr>  
 <th>Codigo Motivo</th>  
 <th>Descripcion Motivo</th>  
 <th>Acciones</th>  
 </tr>  
 </thead>  
 <tbody> {% for item in obj %}  
 <tr>  
 <td>{{ item.codmot }}</td>  
 <td>{{ item.nommot }}</td>  
 <td>  
 <a href="{% url 'core:rhmotiv\_edit' item.codmot %}" class="btn btn-primary btn-circle"><span class="fa fa-edit"></span></a>  
 <a href="{% url 'core:rhmotiv\_del' item.codmot %}" class="btn btn-danger btn-circle"><span class="fa fa-trash"></span></a>  
 </td>  
 </tr> {% endfor %} </tbody>  
 </table>  
 </div>  
 </div>  
{% endblock contenido %}

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Configurando Rutas para URL

Para poder configurar las rutas se debe editar el archivo core/urls.py, y agregar dentro de path las nuevas rutas que conforman la manipulación del modelo, de las 4 vistas anteriores, “Vista, crear, actualizar y eliminar”. Hay que tomar en cuenta que para editar o eliminar debe tener <int:pk> siendo int:pk la indicación para que django coloque el id del registro a actualizar o eliminar.

from django.urls import path  
  
from core.views import AgenciaView, EmpleadosView, EmpleadosNew, EmpleadosEdit, MotivoNew, Motivosview, MotivoEdit, MotivoDel  
  
urlpatterns = [path('core', AgenciaView.as\_view(), name='agencia\_list'),  
  
 path('empleado', EmpleadosView.as\_view(), name='empleado\_list'),  
 path('empleado/new', EmpleadosNew.as\_view(), name='empleado\_new'),  
 path('empleado/edit/<int:pk>', EmpleadosEdit.as\_view(), name='empleado\_edit'),  
  
 # \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MANTENIMIENTO DE MOTIVOS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 path('motivo', Motivosview.as\_view(), name='rhmotiv\_list'),  
 path('motivo/new', MotivoNew.as\_view(), name='rhmotiv\_new'),  
 path('motivo/edit/<int:pk>', MotivoEdit.as\_view(), name='rhmotiv\_edit'),  
 path('motivo/delete/<int:pk>', MotivoDel.as\_view(), name='rhmotiv\_del'),  
  
 ]

Las rutas identificadas con name, es el nombre que se debe de colocar en los url, anteponiendo core:%nombre%, como por ejemplo: <a href="{% url 'core:rhmotiv\_list' %}">Mantenimiento de Motivos</a>

## Agregar el acceso a menú General:

Para poder Agregar un nuevo Elemento al menú general, el menú como se muestra en la imagen:

Interfaz de usuario gráfica, Texto, Aplicación, Word, Correo electrónico

Descripción generada automáticamente

Abrir el archivo en la ubicación del proyecto llamado: base.html

templates/base/base.html

Buscar la línea que contiene y agregar el nuevo menú debajo de los href puestos con anterioridad.

<div class="navbar-default sidebar" role="navigation">  
 <div class="sidebar-nav navbar-collapse">  
 <ul class="nav" id="side-menu">  
 <li>  
 <a href="#"><i class="fa fa-cogs fa-fw"></i> Nombre de modulo <span class="fa arrow"></span></a>  
 <ul class="nav nav-second-level">  
 <li>  
  
 <a href="{% url 'core:agencia\_list' %}">Agencias</a>  
 <a href="{% url 'core:empleado\_list' %}">Empleados</a>  
 </li>

\*\*\*\* Agregar la nueva opcion debajo de este texto \*\*\*\*  
 </ul>  
 </li>  
 </ul>  
 </div>  
 </div>  
</nav>

Agregar los menús necesarios en el código en donde esta \*\*\*\* Agregar la nueva opción debajo de este texto \*\*\*\*

Resultado Final en el archivo templates/base/base.html

<div class="navbar-default sidebar" role="navigation">  
 <div class="sidebar-nav navbar-collapse">  
 <ul class="nav" id="side-menu">  
 <li>  
 <a href="#"><i class="fa fa-cogs fa-fw"></i> Nombre de modulo <span class="fa arrow"></span></a>  
 <ul class="nav nav-second-level">  
 <li>  
 <a href="{% url 'core:agencia\_list' %}">Agencias</a>  
 <a href="{% url 'core:empleado\_list' %}">Empleados</a>  
 </li>  
 <li>  
 <a>Recursos Humanos <span class="fa arrow"></span></a>  
 <ul class="nav nav-stacked">  
 <li>  
 <a> Parametrizacion <span class="fa arrow"></span> </a>  
 <ul class="nav nav-second-level">  
 <li>  
 <a href="{% url 'core:rhmotiv\_list' %}">Mantenimiento de Motivos</a>  
 <a href="{% url 'core:empleado\_list' %}">Mantenimiento de Profesiones</a>  
 <a href="{% url 'core:empleado\_list' %}">Mantenimiento de Academincos</a>  
 </li>  
 </ul>  
 </li>  
 </ul>  
 </li>  
 </ul>  
 </li>  
 </ul>  
 </div>  
 <!-- /.sidebar-collapse -->  
</div>

## Problemas con el modelo

En algunos casos donde se modifica la clase en el archivo models.py y se intenta migrar o sincronizar con la base de datos, django presenta problemas al hacer la migración, para solucionar ese problema se necesita realizar los siguientes pasos.

Por ejemplo, se cambia el campo de integer a autofield en models.py

Texto

Descripción generada automáticamente

Entonces se ejecuta el comando para migrar a la base de datos: makemigrations y migrate

Texto

Descripción generada automáticamente

Dando el error anterior, entonces para corregir la migración seguir los siguientes pasos.