**CREACION DEL CONTENIDO DE UN MODELO:**

1)Primero que nada, creamos un html llamado order\_form. Dentro de esto cargamos los static, incluimos el encabezado y todo lo relacionado con la decoración de nuestra pagina.

Luego cargamos el siguiente código dentro del html:

<form *method*="post" *action*="">

    </form>

2) Ahora, definimos nuestro html desde las view con la siguiente informacion:

*def* createOrder(*request*):

    context = {}

    return render(request,'order\_form.html',context)

3) Cargamos esa url creada en estadisticas.html, en la parte de “créate\_order” como pagina

4) desde el form.py, cargamos la siguiente información sobre las librerías que vamos a usar:

from django.forms import ModelForm

from .models import order

y creamos una clase que hace referencia a la de order:

*class* OrderForm(*ModelForm*):

*class* meta:

        model= order

        fields ='\_\_all\_\_'

luego nos vamos a las views.py e importamos la libreria que creamos en form.py:

from .forms import OrderForm

y agregamos la siguiente información con respecto a la función creada con el de nombre de ‘createOrder’:

*def* createOrder(*request*):

    form =OrderForm()

    context = {'form':form}

    return render(request,'order\_form.html',context)

y para visualizar esto, nos vamos al order\_form.html y cargamos lo siguiente:

    <form *method*="post" *action*="">

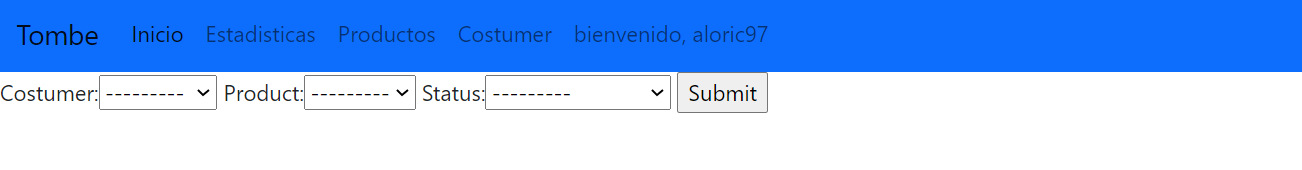
        {% csrf\_token %}

        {{form}}

<input *type*="submit" *name*="Submit">

    </form>

Se vera de la siguiente forma:



Y para que se puedan guardar correctamente los datos en nuestra base de datos, modificamos nuestra función de order y utilizamos el request post.

Una vez que insertamos los datos, verificamos con el método is\_valid que todo este correcto y luego guardamos la información en la base de datos con el método save() y nos redireccionamos hacia la pagina de inicio:

*def* createOrder(*request*):

    form =OrderForm()

    if request.method == 'POST':

        form = OrderForm(request.POST)

        if form.is\_valid():

            form.save()

return redirect('/')

    context = {'form':form}

    return render(request,'order\_form.html',context)

**ACTUALIZACION DEL CONTENIDO DE UN MODELO:**

1) En este caso creamos una función ‘updateOrder’ que tenga la misma vista de order\_form.py

*def* updateOrderForm(*request*):

    form = OrderForm()

    context ={'form':form}

    return render(request,'order\_form.html',context)

pero dentro de la url del proyecto se define de la siguiente forma:

path('update\_order/<str:pk>/', views.updateOrder, *name*='update\_order')

esto nos sirve para que nuestra url sea dinamica y se maneje por el id de la order que queramos actualizar.

Ahora mismo rellenamos la función de updateOrder con lo siguiente:

*def* updateOrderForm(*request*,*pk*):

    orders = order.objects.get(*id*=pk)

    form = OrderForm(*instance*=order)

    if request.method == 'POST':

        form = OrderForm(request.POST, *instance*=orders)

        if form.is\_valid():

            form.save()

            return redirect('/')

    context ={'form':form}

    return render(request,'order\_form.html',context)

lo diferente con la creacion, es que ahora se usa el parametron del id y ademas se instancia la clase con el id, usando la palabra reservada ‘instance’

**ELIMINAR EL CONTENIDO DE UN MODELO**

Funcion de la views :

*def* deleteOrder(*request*, *pk*):

    orders = order.objects.get(*id*=pk)

    context ={'item': orders}

    if request.method == 'POST':

        orders.delete()

        return redirect('estadisticas')

    return render(request,'delete.html',context)

la carga del url:

path('delete/<str:pk>/', views.deleteOrder, *name*='delete\_order')

el html del delete.html:

    <p>Seguro que quieres eliminar la orden? orden: <b>{{item.product}}</b> del estado: <b>{{item.status}}</b></p>

    <form *action*="{% url 'delete\_order' item.id %}" *method*="POST">

        {% csrf\_token %}

        <a *href*="{% url 'estadisticas' %}">Cancelar</a>

        <input *type*="submit" *name*="Confirmar">

    </form>