Proyecto Ecomerce

Instalación de entorno virtual

Se debe instalar inicialmente virtualenv

luego nos paramos en la consola en la carpeta donde vamos a crear el entorno virtual y desde linux ejecutamos el comando

python3 -m venv nombreDelEntornoVirtual

luego para activarlo ejecutamos el comando

source nombreDelEntornoVirtual/bin/activate

y con esto ya queda activo para instalar paquetes del proyecto

para desactivarlo se escribe solo deactivate

para no tener que estar instalando de nuevo los programas utilizados, se crea un archivo llamado requirements.txt con el siguiente comando

pip freeze > 'requirements.txt'

automáticamente en este archivo se crea un listado de los programas utilizados en el proyecto

Para el proyecto usaremos posgresSqul por lo que lo descargamos de su web oficial

Una vez configurada la base de datos, empezamos con la creación de dos aplicaciones, Usuarios y Tienda

En la tienda creamos el primero modelo, pero primero debemos instalar la librería autoslug que nos va a permitir realizar búsquedas con un buscador dentro de la aplicación

Para su instalación ejecutamos el comando pip install django-autoslug

Y lo importamos dentro del modelo con

from autoslug import AutoSlugField

CREACIÓN CARRITO DE COMPRAS

Para crear el carro de compras, se crea una aplicación nueva llamada carrito y dentro de esta creamos un archivo llamado carro.py dentro de este creamos una clase nueva llamada Carro

*class* Carro:

*def* \_\_init\_\_(*self*,*request*):

*self*.request = *request*

*self*.session = *request*.session

        carro = *self*.session.get("carro")

        if not carro:

            carro = *self*.session["carro"]={}

        else:

*self*.carro = carro

Creamos el constructor y en el creamos una condición, si el usuario no a agregado objetos al carrito, se crea un diccionario vacío pero si ya hay objetos se muestran los objetos que se almacenarán en la variable carro

Ahora creamos una función para guardar los productos, para ello

*def* agregar(*self*,*Productos*,*slug*):

        if (str(*Productos*.id) not in *self*.carro.keys()):

*self*.carro[*Productos*.id]={

                "producto\_id": *Productos*.id,

                "slug": *slug*,

                "nombre": *Productos*.nombre,

                "precio": *Productos*.precio,

                "cantidad": 1,

                "imagen":*Productos*.imagen.url,

            }

        else:

            for key,value in *self*.carro.items():

                if key == str(*Productos*.id):

                    value["cantidad"] =  value["cantidad"] + 1

                    break

*self*.guardar\_carro()

En donde le estamos poniendo dos condiciones, la primera es en caso de que el producto no esté en el carrito, va agregarlo con el diccionario descrito, pero si ya está, se toma el valor del diccionario anterior y en especifico al valor de cantidad, le vamos a incrementar 1

Luego creamos una función nueva que se llama guardar carro

*def* guardar\_carro(*self*):

*self*.session["carro"] = *self*.carro

*self*.session.modified = True

esto para guardar los cambios realizados en el carro

Ahora creamos la función de eliminar

*def* eliminar(*self*,*Productos*):

*Productos*.id = str(*Productos*.id)

        if *Productos*.id in *self*.carro:

            del *self*.carro[*Productos*.id]

*self*.guardar\_carro()

Para restar productos tomamos el bloque de código usado para incrementar, pero ahora le cambiamos el + por el –

*def* restar\_producto(*self*,*Productos*):

        for key,value in *self*.carro.items():

            if key == str(*Productos*.id):

                value["cantidad"] =  value["cantidad"] - 1

                break

Pero si llegamos a uno, no debemos seguirle restando, por lo que aquí creamos un condicional

                if value["cantidad"] < 1:

*self*.eliminar(*Productos*)

                break

*self*.guardar\_carro()

Y por último para limpiar el carro

*def* limpiar\_carro(*self*,*Productos*):

        carro = *self*.session["carro"]={}

*self*.session.modified = True

*self*.guardar\_carro()

En donde le estamos diciendo que cree un diccionario vacío, luego que especifiquemos que la session fue modificada y por último guardarmos cambios

Ahora vamos al archivo vistas de la misma app

Y agregamos una vista por cada función que realizamos en el carrito, así

*def* agregar\_producto(*request*, *productos\_id*):

    carro = Carro(*request*)

    producto = Productos.objects.get(*id* = *productos\_id*)

    carro.agregar(*producto* = producto)

    return redirect ("index")

*def* eliminar\_producto(*request*, *productos\_id*):

    carro = Carro(*request*)

    producto = Productos.objects.get(*id* = *productos\_id*)

    carro.eliminar(*producto* = producto)

    return redirect ("index")

*def* restar\_producto(*request*, *productos\_id*):

    carro = Carro(*request*)

    producto = Productos.objects.get(*id* = *productos\_id*)

    carro.restar\_producto(*producto* = producto)

    return redirect ("index")

*def* limpiar\_carro(*request*, *productos\_id*):

    carro = Carro(*request*)

    carro.limpiar\_carro()

    return redirect ("index")

En donde estamos inicialmente trayendo el carro y también obteniendo el producto de nuestro modelo y se lo estamos pasando como parámetro a cada una de las funciones creadas

Ahora para manejar el precio de los productos y que este valor será una variable global en toda nuestra aplicación, vamos a crear una variable de context processor, para ello dentro de la app carrito creamos un archivo py con el mismo nombre y dentro de ella creamos una función con las siguientes características

from .carro import Carro

*def* importe\_total\_carro(*request*):

    carro = Carro(*request*)

    total = 0

    if *request*.user.is\_authenticated:

        for key, value in *request*.session["carro"].items():

            total = total + (value["precio"] \* value["cantidad"])

    return{"importe\_total\_carro":total}

se crea la variable total y luego hacemos una condicional para validar si el usuario está logueado, luego traemos de la session los ítems y le decimos que al total que inicia en 0 le sume el precio por la cantidad de los productos y luego que retorne la variable total

Ahora en el archivo settings vamos a templates y agregamos la ruta completa de la función así

'Apliaciones.carrito.context\_processor.importe\_total\_carro',