Informe de Laboratorio 06

Tema: Django Proyecto

Nota

Estudiante	Escuela	Asignatura
Hidalgo Chinchay, Paulo Andre phidalgo@unsa.edu.pe Huayhua Mayta, Iván Rodrigo ihuayhuam@unsa.edu.pe Jaita Chura, José Manuel jjaitac@unsa.edu.pe Garcia Valdivia, Ronald Pablo rgarciava@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 1702122

Laboratorio	Tema	Duración
06	Django Proyecto	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 16 Junio 2023	Al 30 Junio 2023

1. Competencias del curso

- General: C.c. Diseña responsablemente aplicaciones web, sus componentes o procesos para satisfacer necesidades dentro de restricciones realistas: económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad.
- Específica: C.m. Construye responsablemente soluciones con tecnología web siguiendo un proceso adecuado llevando a cabo las pruebas ajustada a los recursos disponibles del cliente.
- Específica: C.p. Aplica de forma flexible técnicas, métodos, principios, normas, estándares y herramientas del desarrollo web necesarias para la construcción de aplicaciones web e implementación de estos sistemas en una organización.

2. Resultado del estudiante

- RE. 2 La capacidad de aplicar diseño de ingeniería para producir soluciones a problemas y diseñar sistemas, componentes o procesos para satisfacer necesidades específicas dentro de consideraciones realistas en los aspectos de salud pública, seguridad y bienestar; factores globales, culturales, sociales, económicos y ambientales.

- RE. 8 La capacidad de crear, seleccionar y utilizar técnicas, habilidades, recursos y herramientas modernas de ingeniería y tecnologías de la información, incluyendo la predicción y el modelamiento, con una comprensión de las limitaciones.

3. Equipos, materiales y temas

- Sistema Operativo (GNU/Linux de preferencia).
- GNU Vim.
- Python 3.
- Git.
- Cuenta en GitHub con el correo institucional.
- Entorno virtual.
- Django 4.

4. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/123ihuayhua/pweb2-lab-c-23a.git`
- URL para el laboratorio 05 en el Repositorio GitHub.
- `https://github.com/123ihuayhua/pweb2-lab-c-23a/tree/main/Lab6-Pweb2`

5. Tarea

- Elabore un primer informe grupal de la aplicación que desarrollará durante este semestre.
- Utilicen todas las recomendaciones dadas en la aplicación library.
- Acuerdos :
  - Los grupos pueden estar conformado por 1 a 4 integrantes.
  - Sólo se presenta un informe grupal.
  - Sólo se revisa un repositorio. (El único que esté en el informe grupal).
  - Todos los integrantes del grupo tienen una copia del laboratorio e informe en su repositorio privado.
  - Todos los integrantes deben pertenecer al mismo grupo de laboratorio.
  - El docente preguntará en cualquier momento a un integrante sobre el proyecto, codigo fuente, avance.

6. Pregunta

- Por cada integrante del equipo, resalte un aprendizaje que adquirió al momento de estudiar Django. No se reprima de ser detallista. Coloque su nombre entre parentesis para saber que es su aporte.
- Django sigue el patrón de diseño MVC, lo que te permite organizar tu código de una manera estructurada y modular. (Hidalgo Chinchay, Paulo Andre)
- La ventaja de utilizar Django junto con el modelo entidad-relación es que puedes diseñar tu base de datos utilizando el modelo entidad-relación tradicional, y luego utilizar el ORM de Django para crear automáticamente la estructura de la base de datos y manipular los datos utilizando objetos de Python. El ORM se encarga de traducir las operaciones en objetos a consultas SQL que se ejecutan en la base de datos subyacente. (Huayhua Mayta, Iván Rodrigo)
- En relación al modelo entidad-relación (MER), es un modelo conceptual utilizado en el diseño de bases de datos para representar y describir las entidades, atributos y relaciones entre ellas. El MER es independiente de cualquier implementación específica de base de datos, como MySQL o PostgreSQL. (Jaita Chura, José Manuel)
- La facilidad con la que es posible crear una base de datos gracias a Django, gracias a esto al conocer el lenguaje Python es más fácil el administar la base de datos. (Garcia Valdivia, Ronald Pablo)

7. Entregables

- El informe debe tener un enlace al directorio específico del laboratorio en su repositorio GitHub privado donde esté todo el código fuente y otros que sean necesarios. Evitar la presencia de archivos: binarios, objetos, archivos temporales, cache, librerias, entornos virtuales. Si hay configuraciones particulares puede incluir archivos de especificación como: requirements.txt, o leeme.txt.
- No olvide que el profesor debe ser siempre colaborador a su repositorio (Usuario del profesor @rescobedoq).
- Para ser considerado con la calificación de máxima nota, el informe debe estar elaborado en LATEX
- Usted debe describir sólo los commits más importantes que marcaron hitos en su trabajo, adjutando capturas de pantalla, del commit, del código fuente, de sus ejecuciones y pruebas.
- En el informe siempre se debe explicar las imágenes (codigo fuente, capturas de pantalla, commits, ejecuciones, pruebas, etc.) con descripciones puntuales pero precisas.
- Partes de entrega:
  - Modelo de datos. (Diagrama Entidad-Relación)
    - En el siguiente diagrama se muestran 6 clases: "Vendedor, Marcas, Artículo, Pedido Detalle, Pedido Cabecera y Cliente"

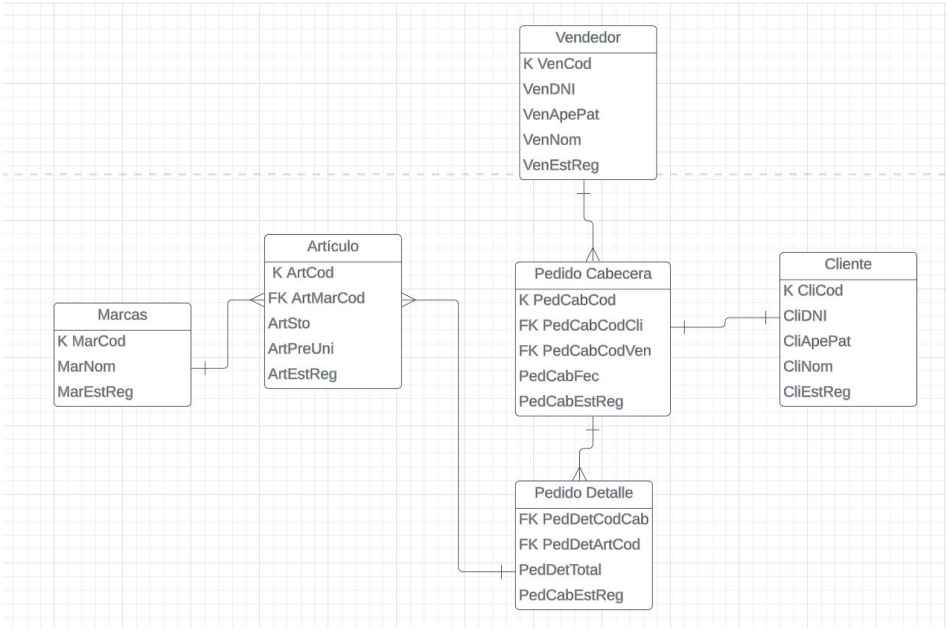


Figura 1: Diagrama

- Modelos Python.
  - La clase "Vendedor"define un modelo de Django que representa a un vendedor en el sistema o aplicación que estás desarrollando.
    Este modelo tiene varios campos que representan diferentes atributos de un vendedor, como su DNI (Documento Nacional de Identidad), apellido paterno, nombre y estado de registro.
  - El modelo Çliente"hereda de la clase AbstractUser, que es una clase de modelo proporcionada por Django para la autenticación de usuarios. Esto significa que el modelo Çliente.ºbtiene todos los campos y funcionalidades de AbstractUser, como nombre de usuario, contraseña, correo electrónico, etc.
    Además de los campos heredados, el modelo Çliente"también tiene campos personalizados para representar atributos específicos del cliente, como su DNI, apellido paterno, nombre y estado de registro.
  - Modelo marca: Este modelo permite almacenar información sobre diferentes marcas en tu sistema o aplicación.
  - El modelo "TipoArticulo"define una entidad de tipo de artículo con dos campos:
    TipArtNom: Un campo de tipo CharField con una longitud máxima de 20 caracteres, que representa el nombre del tipo de artículo. TipArtEstReg: Un campo de tipo BooleanField que representa el estado de registro del tipo de artículo, con un valor predeterminado de True. Este modelo te permite almacenar información sobre diferentes tipos de artículos en tu sistema o aplicación.
  - Clase .ªrticulo": Esta clase representa un artículo en el sistema o aplicación. Tiene los siguientes campos: ArtMarCod: Un campo de clave externa (ForeignKey) que referencia a la clase Marca y se utiliza para almacenar el código de marca del artículo. ArtTipCod: Un campo de clave externa (ForeignKey) que referencia a la clase TipoArticulo y se utiliza para almacenar el código de tipo de artículo. ArtNom: Un campo de tipo CharField con una longitud máxima de 50 caracteres, que representa el nombre del artículo. ArtDes: Un campo de tipo TextField que permite ingresar una descripción del

artículo con un máximo de 1000 caracteres. ArtSto: Un campo de tipo IntegerField que representa el stock (cantidad disponible) del artículo. ArtPreUni: Un campo de tipo FloatField que representa el precio unitario del artículo. ArtEstReg: Un campo de tipo BooleanField que representa el estado de registro del artículo. Tiene una función str() que devuelve el nombre del artículo como representación legible cuando se convierte en una cadena.

- Clase "PedidoCabecera":Esta clase representa la cabecera de un pedido en el sistema o aplicación. Tiene los siguientes campos: PedCabCodCli: Un campo de clave externa (ForeignKey) que referencia a la clase Cliente y se utiliza para almacenar el código de cliente relacionado con el pedido. PedCabCodVen: Un campo de clave externa (ForeignKey) que referencia a la clase Vendedor y se utiliza para almacenar el código de vendedor relacionado con el pedido. PedCabFec: Un campo de tipo DateField que representa la fecha del pedido. PedCabEstReg: Un campo de tipo BooleanField que representa el estado de registro de la cabecera del pedido. Tiene una función str() que devuelve el código de cliente como representación legible cuando se convierte en una cadena.
- Clase "PedidoDetalle": Esta clase representa los detalles de un pedido en el sistema o aplicación. Tiene los siguientes campos: PedDetCodCab: Un campo de clave externa (ForeignKey) que referencia a la clase PedidoCabecera y se utiliza para almacenar el código de cabecera del pedido al que pertenece este detalle. PedDetArtCod: Un campo de clave externa (ForeignKey) que referencia a la clase Articulo y se utiliza para almacenar el código de artículo relacionado con este detalle. PedDetCantidad: Un campo de tipo IntegerField que representa la cantidad de artículos en este detalle del pedido. PedDetPreUniArt: Un campo de tipo FloatField que representa el precio unitario del artículo en este detalle del pedido. PedDetSubtotal: Un campo de tipo FloatField que representa el subtotal calculado para este detalle (cantidad * precio unitario). PedDetTot: Un campo de tipo FloatField que representa el total calculado para este detalle (igual al subtotal en este caso). PedDetEstReg: Un campo de tipo BooleanField que representa el estado de registro del detalle del pedido. Tiene una función save() personal.

Listing 1: models.py

```python
from django.db import models
from django.contrib.auth.models import User, AbstractUser
# Create your models here.

# Vendedor
class Vendedor(models.Model):
    VenDNI = models.CharField(max_length=8, unique=True)
    VenApePat = models.CharField(max_length=20)
    VenNom = models.CharField(max_length=20)
    VenEstReg = models.BooleanField(default=True)
    #Mostrar nombre del vendedor
    def __str__(self):
        nombre = self.VenNom + ' ' +self.VenApePat
        return nombre

#Cliente
class Cliente(AbstractUser):
    CliDNI = models.CharField(max_length=8)
    CliApePat = models.CharField(max_length=20)
    CliNom = models.CharField(max_length=20)
    CliEstReg = models.BooleanField(default=True)

```

```
    # Mostrar nombre completo del cliente
    def __str__(self):
        return self.CliNom + ' ' + self.CliApePat

    password = models.CharField(max_length=128, null=True)
    username = models.CharField(max_length=150, unique=True, null=True)

#Marca
class Marca(models.Model):
    MarNom = models.CharField(max_length=20)
    MarImg = models.ImageField(upload_to='imagenes/imgs', null=True)
    MarEstReg = models.BooleanField(default=True)
    #Mostrar nombre de las marcas
    def __str__(self):
        return self.MarNom

#Tipo Artculo
class TipoArticulo(models.Model):
    TipArtNom = models.CharField(max_length=20)
    TipArtEstReg = models.BooleanField(default=True)
    #Mostrar nombre de los tipos de artculos
    def __str__(self):
        return self.TipArtNom

#Articulo
class Articulo(models.Model):
    ArtMarCod = models.ForeignKey(Marca, on_delete=models.CASCADE, null=True)
    ArtTipCod = models.ForeignKey(TipoArticulo, on_delete=models.CASCADE, null=True)
    ArtNom = models.CharField(max_length=50, null=True)
    ArtDes = models.TextField(max_length=1000, help_text='Ingresa la descripcin del
        artculo', null=True)
    ArtImg = models.ImageField(upload_to='imagenes/imgs', null=True)
    ArtSto = models.IntegerField()
    ArtPreUni = models.FloatField()
    ArtEstReg = models.BooleanField(default=True)
    #Mostrar nombre del Artculo
    def __str__(self):
        return self.ArtNom

#Pedido Cabecera
class PedidoCabecera(models.Model):
    PedCabCodCli = models.ForeignKey(Cliente, on_delete=models.CASCADE)
    PedCabCodVen = models.ForeignKey(Vendedor, on_delete=models.CASCADE)
    PedCabFec = models.DateField(auto_now=False, auto_now_add=False)
    PedCabEstReg = models.BooleanField(default=True)
    #Mostrar pedido cabecera
    def __str__(self):
        return str(self.PedCabCodCli)

#Pedido Detalle
class PedidoDetalle(models.Model):
    PedDetCodCab = models.ForeignKey(PedidoCabecera, on_delete=models.CASCADE,
        related_name='detalles')
    PedDetArtCod = models.ForeignKey(Articulo, on_delete=models.CASCADE)
    PedDetCantidad = models.IntegerField(default=1)
    PedDetPreUniArt = models.FloatField(default=0.0)
```

```
    PedDetSubtotal = models.FloatField(default=0.0)
    PedDetTot = models.FloatField(default=0.0)
    PedDetEstReg = models.BooleanField(default=True)

    def save(self, *args, **kwargs):
        # Obtener el precio del artculo seleccionado
        precio_articulo = self.PedDetArtCod.ArtPreUni

        # Actualizar el campo PedDetPreUniArt con el precio del artculo
        self.PedDetPreUniArt = precio_articulo

        # Calcular el subtotal y el total
        self.PedDetSubtotal = self.PedDetCantidad * self.PedDetPreUniArt
        self.PedDetTot = self.PedDetSubtotal

        super().save(*args, **kwargs)

    #Mostrar pedido detalle
    def __str__(self):
        return str(self.PedDetCodCab)
```

- A continuación se muestra forms.py:
- RegistroForm: Esta clase define un formulario personalizado para el registro de nuevos clientes. Está basado en el formulario UserCreationForm, que es provisto por Django para la creación de usuarios. Se configura para el modelo Cliente, que probablemente sea un modelo personalizado que extienda el modelo de usuario de Django. Los campos incluidos son 'username', 'password1', 'password2', 'CliDNI', 'CliApePat' y 'CliNom'. Además, se proporcionan etiquetas personalizadas para algunos campos.
- PedidoForm: Esta clase define un formulario para la creación de pedidos. Hereda de ModelForm, que es un formulario de Django generado automáticamente a partir de un modelo. Está configurado para el modelo PedidoCabecera. Los campos incluidos son 'PedCabCodCli', 'PedCabCodVen' y 'PedCabFec'. El campo 'PedCabCodCli' se excluye explícitamente utilizando el atributo exclude. Además, se proporcionan etiquetas personalizadas para los campos y se utiliza un widget para dar formato a la fecha.
- PedidoDetForm: Esta clase define un formulario para la creación de detalles de pedidos. También hereda de ModelForm y está configurado para el modelo PedidoDetalle. Los campos incluidos son 'PedDetArtCod' y 'PedDetCantidad'. Se proporcionan etiquetas personalizadas para los campos. Además, se define un método clean() en el formulario que realiza una validación personalizada. Verifica que la cantidad solicitada no supere el stock disponible del artículo seleccionado. Si la validación es exitosa, se actualiza el stock en la base de datos y se devuelve el conjunto de datos validados.
- La función update.stock.quantity es un receptor de señales (@receiver) que se ejecuta después de guardar un objeto de tipo PedidoDetalle. Obtiene el artículo relacionado con el detalle del pedido (PedDetArtCod) y guarda la instancia del artículo, lo que probablemente actualiza el stock en la base de datos.

Listing 2: forms.py

```
from django import forms
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.forms import ModelForm, formset_factory
from django.contrib.auth.forms import UserCreationForm
from .models import PedidoCabecera
```

```python
from .models import PedidoDetalle
from .models import Cliente

class RegistroForm(UserCreationForm):
    class Meta(UserCreationForm.Meta):
        model = Cliente
        fields = ('username', 'password1', 'password2', 'CliDNI', 'CliApePat', 'CliNom')
        labels = {
            'CliDNI': 'DNI',
            'CliApePat': 'Apellido Paterno',
            'CliNom': 'Nombre'
        }

class PedidoForm(ModelForm):
    class Meta:
        model = PedidoCabecera
        exclude = ['PedCabCodCli']
        fields = ['PedCabCodCli', 'PedCabCodVen', 'PedCabFec']
        labels = {
            'PedCabCodCli': 'Cliente',
            'PedCabCodVen': 'Vendedor',
            'PedCabFec': 'Fecha'
        }
        widgets = {
            'PedCabFec': forms.DateInput(attrs={'type': 'date', 'format': 'YY-MM-DD'})
        }


class PedidoDetForm(ModelForm):
    class Meta:
        model = PedidoDetalle
        fields = ['PedDetArtCod', 'PedDetCantidad']
        labels = {
            'PedDetArtCod': 'Artculo',
            'PedDetCantidad': 'Cantidad',
        }
        def clean(self):
            cleaned_data = super().clean()
            articulo = cleaned_data.get('PedDetArtCod')
            cantidad = cleaned_data.get('PedDetCantidad')
            if articulo and cantidad:
                if cantidad > articulo.stock:
                    raise forms.ValidationError(f'No hay suficiente stock de {articulo.nombre}')
                # Update the stock in the database
                articulo.stock -= cantidad
                articulo.save()
            return cleaned_data

        def save(self, *args, **kwargs):
            if self.pk is None:
                # object is being created
                self.actualizar_stock(self.PedDetArtCod, self.PedDetCantidad)
            else:
```

```
                # object is being updated
                old_detalle = PedidoDetalle.objects.get(pk=self.pk)
                if old_detalle.PedDetCantidad != self.PedDetCantidad:
                    difference = self.PedDetCantidad - old_detalle.PedDetCantidad
                    self.actualizar_stock(self.PedDetArtCod, difference)
            super().save(*args, **kwargs)


@receiver(post_save, sender=PedidoDetalle)
def update_stock_quantity(sender, instance, **kwargs):
    articulo = instance.PedDetArtCod
    # articulo.ArtSto = articulo.ArtSto - instance.PedDetCantidad
    articulo.save()

@receiver(post_save, sender=PedidoDetalle)
def actualizar_stock(sender, instance, **kwargs):
    articulo = instance.PedDetArtCod
    cantidad = instance.PedDetCantidad
    if cantidad > articulo.ArtSto:
        # error_message = 'La cantidad solicitada supera el stock disponible'
        # return render(request, 'actualizar_pedido.html', {'form': formatter,
            'error_message': error_message})
        raise forms.ValidationError(f'No hay suficiente stock de {articulo.nombre}')
    articulo.ArtSto = articulo.ArtSto - (cantidad-1)
    articulo.save()
```

- admin.py muestra cómo personalizar la interfaz de administración de Django para los modelos Vendedor, Cliente y Articulo. Aquí hay una descripción de cada clase y su funcionalidad:
- VendedorAdmin: Esta clase personaliza la interfaz de administración para el modelo Vendedor. Se define la variable list-display para especificar los campos que se mostrarán en la lista de registros de vendedores. Luego, se definen métodos para mostrar los valores de los campos personalizados en la lista. En este caso, se definen métodos como nombre, apellido y dni para mostrar los valores correspondientes de los campos del modelo Vendedor. También se proporcionan etiquetas personalizadas para los campos.
- ClienteAdmin: Similar a VendedorAdmin, esta clase personaliza la interfaz de administración para el modelo Cliente. Se define list-display y métodos para mostrar los valores de los campos personalizados en la lista de registros de clientes.
- ArticuloAdmin: Esta clase personaliza la interfaz de administración para el modelo Articulo. Se define list-display y métodos para mostrar los valores de los campos personalizados en la lista de registros de artículos. Además, se realiza un formato especial para el campo precio-unitario para mostrar el precio con un símbolo de moneda y formato decimal.
- Luego, se registran las clases personalizadas en el administrador de Django utilizando admin.site.register para que las personalizaciones se apliquen a los modelos correspondientes.

Listing 3: admin.py

```
from django.contrib import admin
from .models import *
#Mostrar detalles en Vendedor
class VendedorAdmin(admin.ModelAdmin):
    list_display = ('nombre', 'apellido', 'dni')
```

```python

    def nombre(self, obj):
        return obj.VenNom
    nombre.short_description = 'Nombre'

    def apellido(self, obj):
        return obj.VenApePat
    apellido.short_description = 'Apellido'

    def dni(self, obj):
        return obj.VenDNI
    dni.short_description = 'DNI'

#Mostrar detalles en Cliente
class ClienteAdmin(admin.ModelAdmin):
    list_display = ('nombre', 'apellido', 'dni')

    def nombre(self, obj):
        return obj.CliNom
    nombre.short_description = 'Nombre'

    def apellido(self, obj):
        return obj.CliApePat
    apellido.short_description = 'Apellido'

    def dni(self, obj):
        return obj.CliDNI
    dni.short_description = 'DNI'

#Mostrar detalles en Artculo
class ArticuloAdmin(admin.ModelAdmin):
    list_display = ('nombre_articulo', 'marca', 'stock', 'precio_unitario')

    def nombre_articulo(self, obj):
        return obj.ArtNom
    nombre_articulo.short_description = 'Nombre'

    def marca(self, obj):
        return obj.ArtMarCod
    marca.short_description = 'Marca'

    def stock(self, obj):
        return obj.ArtSto
    stock.short_description = 'Cantidad en Stock'

    def precio_unitario(self, obj):
        return f'S/ {obj.ArtPreUni:,.2f}'
    precio_unitario.short_description = 'Precio Unitario'


#Vistas y modelos
admin.site.register(Vendedor, VendedorAdmin)
admin.site.register(Cliente, ClienteAdmin)
admin.site.register(Marca)
admin.site.register(TipoArticulo)
admin.site.register(Articulo, ArticuloAdmin)
```

```
admin.site.register(PedidoCabecera)
admin.site.register(PedidoDetalle)
```

- A continuación se muestra el código de apps.py:

Listing 4: apps.py

```
from django.apps import AppConfig


class TiendaConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'tienda'
```

- En views.py podemos ver que existe una variedad de métodos que nos permiten: tener la vista principal, registrarnos con el usuario, ver pedidos, crear pedidos, detalles del pedido, actualizar el pedido, salir de la sesión, logearse y las diferentes marcas de los productos.

Listing 5: views.py

```
from datetime import date
import datetime
from django import forms
from django.http import HttpResponseForbidden
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth import login, logout, authenticate
from django.db import IntegrityError
from .forms import PedidoDetForm, PedidoForm, RegistroForm
from django.forms import formset_factory, ValidationError
from .models import PedidoCabecera, PedidoDetalle, Cliente, Marca, Articulo,
    Vendedor
from django.contrib.auth.decorators import login_required

# Create your views here.
#Vista principal
def Home(request):
    marcas = Marca.objects.all()
    arts = Articulo.objects.all()
    context = {'marcas': marcas, 'arts': arts}
    return render(request, 'home.html', context)

#Registrarse (usuario)
def Signup(request):
    if request.method == 'GET':
        return render(request, 'signup.html', {'form': RegistroForm})

    else:
        if request.POST['password1'] == request.POST['password2']:
            try:
                # Registrar usuario
                user =
                    Cliente.objects.create_user(username=request.POST['username'],
                    password=request.POST['password1'])
                # Guardar usuario
```

```
                user.CliDNI = request.POST['CliDNI']
                user.CliApePat = request.POST['CliApePat']
                user.CliNom = request.POST['CliNom']
                user.CliEstReg = True
                user.save()
                login(request, user)
                return redirect('index')
            except IntegrityError:
                return render(request, 'signup.html', {
                    'form': RegistroForm,
                    'error': 'El usuario ya existe'
                })

        return render(request, 'signup.html', {
            'form': RegistroForm,
            'error': 'Las contraseas no coinciden'
        })

@login_required
#Ver pedidos
def Index(request):
    pedido = PedidoCabecera.objects.filter(PedCabCodCli=request.user)
    return render(request, 'index.html', {'pedidos' : pedido})

        # initial=[{'PedCabCodCli': request.user.pk}]

#Crear pedido
PedidoFormSet = formset_factory(PedidoForm, extra=1)
@login_required
def CreatePedido(request):
    if request.method == 'GET':
        vendedores = Vendedor.objects.all()
        articulos = Articulo.objects.all()
        formset = PedidoFormSet(prefix='pedidos')
        form2 = PedidoDetForm(prefix='pedido_det')
        formset.forms[0].initial['PedCabCodCli'] = request.user.pk
        return render(request, 'pedido.html', {
            'formset': formset,
            'form2': form2,
            'vendedores': vendedores,
            'articulos': articulos
        })
    else:
        formset = PedidoFormSet(request.POST, prefix='pedidos')
        form2 = PedidoDetForm(request.POST, prefix='pedido_det')
        vendedores = Vendedor.objects.all()
        articulos = Articulo.objects.all()
        if formset.is_valid() and form2.is_valid():
            pedido_cab = None
            for form in formset:
                new_pedido = form.save(commit=False)
                new_pedido.PedCabCodCli = get_object_or_404(Cliente,
                    id=request.user.pk)
                if not pedido_cab:
                    pedido_cab = new_pedido
                    pedido_cab.save()
```

```
                new_pedido.PedDetCodCab = pedido_cab
                new_pedido.save()
            new_pedido2 = form2.save(commit=False)
            new_pedido2.PedCabCodCli = get_object_or_404(Cliente, id=request.user.pk)
            new_pedido2.PedDetCodCab = pedido_cab
            articulo = new_pedido2.PedDetArtCod
            cantidad = new_pedido2.PedDetCantidad
            if cantidad > articulo.ArtSto:
                return render(request, 'pedido.html', {
                    'formset': formset,
                    'form2': form2,
                    'vendedores': vendedores,
                    'articulos': articulos,
                    'error': f'No hay suficiente stock de {articulo.ArtNom}'
                })
            articulo.ArtSto = articulo.ArtSto - cantidad
            articulo.save()
            new_pedido2.save()
            return redirect('index')
        else:
            return render(request, 'pedido.html', {
                'formset': formset,
                'form2': form2,
                'vendedores': vendedores,
                'articulos': articulos,
                'error': 'Ingrese datos vlidos'
            })

#Detalle pedido
@login_required
def DetallePedido(request, pedidoID):
    pedido_cabecera = get_object_or_404(PedidoCabecera, pk=pedidoID,
        PedCabCodCli=request.user)
    detalles_pedido = PedidoDetalle.objects.filter(PedDetCodCab=pedido_cabecera)

    if detalles_pedido.exists():
        detalle = detalles_pedido.first()
    else:
        detalle = PedidoDetalle(PedDetCodCab=pedido_cabecera)

    if request.method == 'GET':
        form = PedidoDetForm(instance=detalle)
        return render(request, 'pedido_det.html', {'pedido_cabecera':
            pedido_cabecera, 'detalle': detalle, 'form': form})
    # else:
    #     form = PedidoDetForm(request.POST, instance=detalle)
    #     if form.is_valid():
    #         form.save()
    #         return redirect('index')
    #     return render(request, 'pedido_det.html', {'pedido_cabecera':
        pedido_cabecera, 'detalle': detalle, 'form': form, 'error': 'No se puede
        actualizar'})

#Actualizar pedido
@login_required
def Actualizar_pedido(request, pedidoID):
```

```
    pedido_cabecera = get_object_or_404(PedidoCabecera, pk=pedidoID,
        PedCabCodCli=request.user)
    detalles_pedido = PedidoDetalle.objects.filter(PedDetCodCab=pedido_cabecera)

    if detalles_pedido.exists():
        detalle = detalles_pedido.first()
    else:
        detalle = PedidoDetalle(PedDetCodCab=pedido_cabecera)

    if request.method == 'POST':
        form = PedidoDetForm(request.POST, instance=detalle)
        if form.is_valid():
            detalle = form.save(commit=False)
            articulo = detalle.PedDetArtCod
            old_cantidad = detalle.PedDetCantidad
            new_cantidad = form.cleaned_data['PedDetCantidad']
            difference = new_cantidad - old_cantidad

            if new_cantidad > articulo.ArtSto:
                error_message = 'La cantidad solicitada supera el stock disponible'
                return render(request, 'actualizar_pedido.html', {'form': form,
                    'error_message': error_message})
            else:
                articulo.ArtSto = articulo.ArtSto - difference
                articulo.save()
                detalle.save() # Guardar el objeto PedidoDetalle actualizado en la
                    base de datos
                return redirect('pedido_detalle', pedidoID=pedidoID)
    else:
        form = PedidoDetForm(instance=detalle)

    if pedido_cabecera.PedCabCodCli != request.user:
        return HttpResponseForbidden()

    return render(request, 'actualizar_pedido.html', {'form': form})
    # return render(request, 'actualizar_pedido.html', {'form': form})


#Salir de la sesin
def Signout(request):
    logout(request)
    return redirect('home')

#Logearse
def Signin(request):
    if request.method == 'GET':
        return render(request, 'signin.html', {
        'form' : AuthenticationForm
        })
    else:
        user = authenticate(
            request, username = request.POST['username'], password =
                request.POST['password']
        )
        if user is None:
            return render(request, 'signin.html', {
```

```
            'form' : AuthenticationForm,
            'error': 'Usuario o contrasea incorrecta'
            })
        else:
            login(request, user)
            return redirect('index')

#Marcas
def marca_detail(request, marca_id):
    marca = get_object_or_404(Marca, pk=marca_id)
    arts = Articulo.objects.filter(ArtMarCod=marca)
    templates = {
        'Acer': 'marca_acer.html',
        'Apple': 'marca_apple.html',
        'Asus': 'marca_asus.html',
        'Dell': 'marca_dell.html',
        'HP': 'marca_hp.html',
        'Lenovo': 'marca_lenovo.html',
        'MSI': 'marca_msi.html',
    }
    templates = templates.get(marca.MarNom, 'marca_default.html')
    context = {'marca': marca, 'arts': arts}
    return render(request, f'{templates}', context)

#Elimininar pedido
@login_required
def eliminar_pedido(request, pedidoID):
    pedido_cabecera = get_object_or_404(PedidoCabecera, pk=pedidoID,
        PedCabCodCli=request.user)
    detalles_pedido = PedidoDetalle.objects.filter(PedDetCodCab=pedido_cabecera)

    for detalle in detalles_pedido:
        articulo = detalle.PedDetArtCod
        articulo.ArtSto += detalle.PedDetCantidad
        articulo.save()

    pedido_cabecera.delete()

    return redirect('index')
```

- Ahora pasaremos con los templates que se usaron:
- Para comenzar está signup.html que sirve para un registro de formulario, en este se usa csrf-token que se utiliza en formularios de Django para protegerlos contra ataques de falsificación de solicitudes entre sitios (CSRF, por sus siglas en inglés). Usado para verificar además que al repetir la contraseña para la creación del usuario sea la misma que la primera.

Listing 6: signup.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<section class="vh-130" style="background-color: #eee;">
    <div class="container h-80">
        <div class="row d-flex justify-content-center align-items-center h-100">
          <div class="col-lg-12 col-xl-11">
```

```
          <div class="card text-black" style="border-radius: 25px;">
            <div class="card-body p-md-5">
              <div class="row justify-content-center">
                <div class="col-md-10 col-lg-6 col-xl-5 order-2 order-lg-1">

                  <p class="text-center h1 fw-bold mb-5 mx-1 mx-md-4 mt-4">Sign up</p>
                  {{error}}
                  <form class="mx-1 mx-md-4" action="/signup/" method="POST">
                      {% csrf_token %}
                      {{error}}
                      <!-- Nombre -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-user fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
                          <label class="form-label" id="id_CliNom">Name</label>
                          <input type="text" name="CliNom" id="id_CliNom" class="form-control" placeholder="Write your Name"/>
                      </div>
                    </div>
                    <!-- Apellido -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-user fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
                          <label class="form-label" id="id_CliApePat">Surname</label>
                          <input type="text" name="CliApePat" id="id_CliApePat" class="form-control" placeholder="Write your Surname"/>
                      </div>
                    </div>
                    <!-- dni -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-user fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
                          <label class="form-label" id="id_CliDNI">DNI</label>
                          <input type="number" maxlength="8" name="CliDNI" id="id_CliDNI" class="form-control" placeholder="Write your Surname"/>
                      </div>
                    </div>
                    <!-- Usuario -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-user fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
                          <label class="form-label" id="id_username">Username</label>
                          <input type="text" name="username" id="id_username" class="form-control" placeholder="Write your Surname"/>
                      </div>
                    </div>
                    <!-- Contrasea -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-lock fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
```

```
                        <label class="form-label" for="id_password1">Password</label>
                        <input type="password" name="password1" id="id_password1" class="form-control" placeholder="Write your password"/>
                    </div>
                  </div>
                  <!-- Contrasea2  -->
                  <div class="d-flex flex-row align-items-center mb-4">
                    <i class="fas fa-lock fa-lg me-3 fa-fw"></i>
                    <div class="form-outline flex-fill mb-0">
                        <label class="form-label" for="id_password2">Password</label>
                        <input type="password" name="password2" id="id_password2" class="form-control" placeholder="Repite your password"/>
                    </div>
                  </div>

                  <!-- Boton Register -->
                  <div class="d-flex justify-content-center mx-4 mb-3 mb-lg-4">
                    <button type="submit" class="btn btn-primary btn-lg">Register</button>
                  </div>
                </form>
              </div>
              <div class="col-md-10 col-lg-6 col-xl-7 d-flex align-items-center order-1 order-lg-2">
                <img src="{% static '/img/register.svg' %}"
                  class="img-fluid" alt="Sample image">
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
</section>
{% endblock %}
```

- Continuamos con signin.html que es un formulario para logearse, este lleva usuario y contraseña, en caso de no encontrarse el susuario se mostrará un mensaje.

Listing 7: signin.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
    <section class="vh-100">
        <div class="container py-5 h-100">
            <div class="row d-flex align-items-center justify-content-center h-100">
                <div class="col-md-8 col-lg-7 col-xl-6">
                    <img src="{% static '/img/login.svg' %}"
                    class="img-fluid" alt="Phone image">
                </div>
                <div class="col-md-7 col-lg-5 col-xl-5 offset-xl-1">
```

```
                    <form action="/signin/" method="POST">
                        {% csrf_token %}
                        <h1 class="text-center fw-bold mb-2 text-uppercase">Signin</h1>
                        {{error}} {% csrf_token %}
                        <!-- Username -->
                        <div class="form-outline mb-4">
                            <label for="username">Username</label>
                            <input type="text" name="username" id="username" class="form-control form-control-lg" placeholder="Write your username"/>
                        </div>
                        <!-- Contrasea -->
                        <div class="form-outline mb-4">
                            <label class="password" for="password">Password</label>
                            <input type="password" name="password" id="password" class="form-control form-control-lg" placeholder="Write your password"/>
                        </div>

                        <div class="d-flex justify-content-around align-items-center mb-4">
                            <!-- Checkbox -->
                            <div class="form-check">
                            <input class="form-check-input" type="checkbox" value="" id="form1Example3" checked />
                            <label class="form-check-label" for="form1Example3">Remember me </label>
                            </div>
                            <a href="#!">Forgot password?</a>
                        </div>
                        <!-- Boton -->
                        <button type="submit" class="btn btn-primary btn-lg btn-block">Sign in</button>
                    </form>
            </div>
        </div>
    </section>
{% endblock %}
```

- base.html, es la estructura que acompañará a todas nuestras páginas, en esta se encontrarán varios enlaces como a index.html, logout.html, signup.html, signin.html.

Listing 8: base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <title>Tienda</title>
</head>
<body>
  {% load static %}
```

```
    <nav class="navbar navbar-expand-lg bg-dark navbar-dark">
        <div class="container">
          <!-- Agrega el logotipo -->
          <a class="navbar-brand" href="/">
            <img src="{% static '/img/logo.png' %}" alt="Logo de la tienda"
                width="30%" height="auto">
          </a>
          <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
              data-bs-target="#navbarNav" aria-controls="navbarNav"
              aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ms-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/">Home</a>
                </li>
                {% if user.is_authenticated %}
                <li class="nav-item">
                    <a  href="/index" class="nav-link">Mis Pedidos</a>
                </li>
                <li class="nav-item">
                    <a href="/index/crear_pedido/" class="nav-link">Crear Pedido</a>
                </li>
                <li class="nav-item">
                    <a href="/logout" class="nav-link">Logout</a>
                </li>
                {% else %}
                <li class="nav-item">
                    <a href="/signup" class="nav-link">Signup</a>
                </li>
                <li class="nav-item">
                    <a href="/signin" class="nav-link">Signin</a>
                </li>
                {% endif %}
              </li>
            </ul>
          </div>
        </div>
      </nav>
    {% block content %}
    {% endblock %}
    {% include 'footer.html' %}
</body>
</html>
```

- marca-msi nos permitirá ver la descripción, nombre, precio unitario de nuestros productos, esta será utilizada por todas las páginas de los productos de las diferentes marcas.

Listing 9: marca$_{m}si.html$

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
{% include 'encabezado.html' %}
</section>
```

```
<main class="container py-5">
    <section class="card card-body">
        <h1 class="display-4 text-center mb-4 fw-bold">{{ marca.MarNom }}</h1>
    </section>
</main>
<div class="d-flex flex-wrap justify-content-center">
    {% for art in arts %}
    {% if art.ArtMarCod == marca %}
    <div class="card m-3" style="width: 250px;">
        <img src="{{ art.ArtImg.url }}" class="card-img-top img-fluid" alt="{{ art.ArtNom }}">
        <div class="card-body">
            <h2 class="card-title h5 fw-bold" >{{ art.ArtNom }}</h2>
            <p class="card-text fw-bold">S/.{{ art.ArtPreUni }}</p>
            <p class="card-text fw-bold">Stock: {{ art.ArtSto }}</p>
            <p class="card-text description" style="display: none;">{{ art.ArtDes }}</p>
        </div>
        <div class="card-footer d-flex justify-content-center bg-dark">
            <button class="btn btn-warning" onclick="toggleDescription(this)">Mostrar Descripcin</button>
        </div>
    </div>
    {% endif %}
    {% endfor %}
</div>
<script>
    function toggleDescription(button) {
        var cardBody = button.parentNode.parentNode.querySelector('.card-body');
        var description = cardBody.querySelector('.description');

        if (description.style.display === 'none') {
            description.style.display = 'block';
            button.textContent = 'Ocultar Descripcin';
        } else {
            description.style.display = 'none';
            button.textContent = 'Mostrar Descripcin';
        }
    }
</script>
{% endblock %}
```

- Se muestran las marcas de los productos: acer, apple, asus, dell, hp, lenovo
- El código que cada una tiene:

Listing 10: Código de las marcas

```
        {% load static %}
            {% include 'marca_msi.html' %}
```

- extends 'base.html': Esta línea indica que la plantilla actual (base.html) hereda el contenido de otra plantilla llamada 'base.html'. La plantilla base suele contener la estructura general de la página, como el encabezado, el menú de navegación y el pie de página, y se puede personalizar utilizando bloques.
  block content y endblock : Estas etiquetas delimitan un bloque de contenido llamado

'content'. El contenido dentro de estas etiquetas se reemplazará en la plantilla base donde se defina el bloque content. Permite insertar contenido específico en esa área de la plantilla base sin modificarla directamente.
Dentro del bloque content, se encuentra el contenido específico de la plantilla actual. En este caso, se muestra una tabla que lista los pedidos. Los pedidos se obtienen del contexto de la vista y se iteran utilizando la etiqueta for pedido in pedidos y endfor . Para cada pedido, se muestra la información del cliente, la fecha del pedido y un enlace para ver los detalles del pedido.
En resumen, este código extiende una plantilla base, define un bloque de contenido llamado 'content' y muestra una tabla con información de pedidos. El contenido de esta plantilla se insertará en el área correspondiente de la plantilla base al renderizar la página.

Listing 11: index.html

```
{% extends 'base.html' %}

{% block content %}
    <div class="container">
        <h1 class="text-center mt-3 mb-3 display-3 text-primary font-weight-bold">MIS PEDIDOS</h1>
        <div class="row justify-content-center">
            <div class="col-md-8">
                <div class="card">
                    <div class="card-body">
                        <table class="table table-bordered text-center">
                            <thead class="thead-dark table-dark">
                                <tr>
                                    <th class="col-4">CLIENTE</th>
                                    <th class="col-4">FECHA PEDIDO</th>
                                    <th class="col-4">DETALLES</th>
                                </tr>
                            </thead>
                            <tbody class="table-light">
                                {% for pedido in pedidos %}
                                    <tr>
                                        <td class="col-4">{{ pedido.PedCabCodCli }}</td>
                                        <td class="col-4">{{ pedido.PedCabFec }}</td>
                                        <td class="col-4">
                                            <a href="{% url 'pedido_detalle' pedido.id %}">
                                                <button type="button" class="btn btn-info btn-rounded" >
                                                    VER
                                                </button>
                                            </a>
                                        </td>
                                    </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>
```

```
    </div>
{% endblock %}
```

- home.html
- Dentro del bloque content, se encuentra el contenido específico de la plantilla actual. En este caso, el código verifica si el usuario está autenticado (if user.is-authenticated) y muestra un mensaje de bienvenida si es así. Luego, se muestra un carrusel de imágenes utilizando el componente Carousel de Bootstrap. Las imágenes se cargan desde rutas específicas y se muestran en forma de carrusel.
  A continuación, se muestra una lista de marcas. Se itera sobre la variable marcas y se muestra cada marca en una tarjeta. Cada tarjeta contiene un enlace a la página de detalles de la marca (url 'marca-detail' marca-id=marc.pk) y muestra la imagen de la marca.
  El código también incluye un fragmento JavaScript al final para cargar el script de Bootstrap necesario para el funcionamiento del carrusel.
  En resumen, este código extiende una plantilla base, define un bloque de contenido llamado 'content' y muestra un mensaje de bienvenida, un carrusel de imágenes y una lista de marcas. El contenido de esta plantilla se insertará en el área correspondiente de la plantilla base al renderizar la página.

Listing 12: home.html

```
{% extends 'base.html' %}
{% block content %}
{% load static %}
{% if user.is_authenticated %}
    <h2 class="text-center mb-4">Bienvenido {{ user.username }}</h2>
{% endif %}
<!-- Carrusel de imgenes -->
<div id="carouselExampleIndicators" class="carousel slide" data-bs-ride="carousel">
    <ol class="carousel-indicators">
        <li data-bs-target="#carouselExampleIndicators" data-bs-slide-to="0"
            class="active"></li>
        <li data-bs-target="#carouselExampleIndicators" data-bs-slide-to="1"></li>
        <li data-bs-target="#carouselExampleIndicators" data-bs-slide-to="2"></li>
    </ol>
    <div class="carousel-inner">
        <div class="carousel-item active">
            <img src="/public/imagenes/imgs/carrusel1.svg" class="d-block w-100"
                alt="Imagen 1">
        </div>
        <div class="carousel-item">
            <img src="/public/imagenes/imgs/carrusel2.svg" class="d-block w-100"
                alt="Imagen 2">
        </div>
        <div class="carousel-item">
            <img src="/public/imagenes/imgs/carrusel3.svg" class="d-block w-100"
                alt="Imagen 3">
        </div>
    </div>
    <a class="carousel-control-prev" href="#carouselExampleIndicators"
        role="button" data-bs-slide="prev">
        <span class="carousel-control-prev-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Previous</span>
    </a>
```

```
    <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-bs-slide="next">
        <span class="carousel-control-next-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Next</span>
    </a>
</div>
<br>
<h3 class="mt-5 mb-5 text-center">Marcas</h3>
<!-- Marcas -->
<div class="row row-cols-1 row-cols-md-4 g-4">
    {% for marc in marcas %}
    <div class="col mb-4">
        <div class="card shadow-sm">
            <a href="{% url 'marca_detail' marca_id=marc.pk %}">
                <div class="rounded-circle overflow-hidden mx-auto d-flex justify-content-center align-items-center" style="width: 150px; height: 150px;">
                <img src="{{ marc.MarImg.url }}" class="card-img-top" alt="">
                </div>
            </a>
        </div>
    </div>
    {% if forloop.counter|divisibleby:4 and not forloop.last %}
        </div><div class="row row-cols-1 row-cols-md-3 g-4">
    {% endif %}
    {% endfor %}
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
{% endblock %}
```

- En encabezado.html se carga la etiqueta load static al comienzo de la plantilla. A continuación, se utiliza la función static para referenciar archivos estáticos en los atributos src de las etiquetas img. Estas etiquetas img muestran imágenes de marcas, como Acer, Apple, Asus, Dell, HP, Lenovo y MSI, en el menú de navegación de la barra superior.
  Al utilizar la función static , se genera la URL correcta para el archivo estático, tomando en cuenta la configuración de archivos estáticos del proyecto Django. Esto asegura que las imágenes se carguen correctamente desde la carpeta de archivos estáticos definida en el proyecto.
  En resumen, la etiqueta load static permite cargar archivos estáticos, y la función static se utiliza para referenciar archivos estáticos, como las imágenes de las marcas, en la plantilla.

Listing 13: encabezado.html

```
{% load static %}
<section>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">
            <a class="navbar-brand" href="/"></a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
```

```
            </button>
            <div class="collapse navbar-collapse d-flex justify-content-around" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'marca_detail' marca_id=1 %}">
                            <img src="{% static 'img/acer.svg' %}" class="rounded img-fluid" alt="acer" style="width: 60px; height: 40px;">
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'marca_detail' marca_id=2 %}">
                            <img src="{% static 'img/apple.svg' %}" class="rounded img-fluid" alt="apple" style="width: 60px; height: 40px;">
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'marca_detail' marca_id=3 %}">
                            <img src="{% static 'img/asus.svg' %}" class="rounded img-fluid" alt="asus" style="width: 60px; height: 40px;">
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'marca_detail' marca_id=4 %}">
                            <img src="{% static 'img/dell.svg' %}" class="rounded img-fluid" alt="dell" style="width: 60px; height: 40px;">
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'marca_detail' marca_id=5 %}">
                            <img src="{% static 'img/hp.svg' %}" class="rounded img-fluid" alt="hp" style="width: 60px; height: 40px;">
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'marca_detail' marca_id=6 %}">
                            <img src="{% static 'img/lenovo.svg' %}" class="rounded img-fluid" alt="lenovo" style="width: 60px; height: 40px;">
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'marca_detail' marca_id=7 %}">
                            <img src="{% static 'img/msi.svg' %}" class="rounded
```

```
                                img-fluid" alt="msi" style="width: 60px; height:
                                40px;">
                        </a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
```

- El código representa una sección de pie de página (footer) en una plantilla Django. Aquí se explica qué hace cada parte:
  load static : Esta línea carga los archivos estáticos en la plantilla, lo cual permite acceder a ellos en la sección de CSS, JavaScript o imágenes.
  El código HTML representa el contenido del pie de página. Aquí se encuentran varios elementos como información sobre la empresa, enlaces a las marcas, detalles de contacto y el logotipo de la empresa.
  Los enlaces a las marcas se generan utilizando etiquetas ¡a¿con la sintaxis url 'marca-detail' marca-id=1 . Estos enlaces hacen uso de la función url de Django y se espera que exista una vista llamada 'marca-detail' que reciba un parámetro marca-id.
  La imagen del logotipo se muestra utilizando la etiqueta ¡img¿y el atributo src con la sintaxis static '/img/logo.png' . Aquí se espera que exista una imagen llamada 'logo.png' en la carpeta img dentro de la carpeta de archivos estáticos.
  En resumen, este código representa la estructura y el contenido de un pie de página en una plantilla Django, incluyendo información de la empresa, enlaces a marcas, detalles de contacto y un logotipo.

Listing 14: footer.html

```
{% load static %}
<footer class="text-center text-lg-start text-white" style="background-color:
    black">
    <!-- Grid container -->
    <div class="container ">
        <!-- Section: Links -->
        <section class="">
            <!--Grid row-->
            <div class="row">
                <!-- Grid column -->
                <div class="col-md-3 col-lg-3 col-xl-3 mx-auto mt-5">
                    <h6 class="text-uppercase mb-4 font-weight-bold">
                       LaPC
                    </h6>
                    <p>
                        En nuestra empresa, nos dedicamos a ofrecer soluciones
                            tecnolgicas de alta calidad para satisfacer las
                            necesidades de nuestros clientes. Nos enfocamos en la
                            venta de laptops de ltima generacin, brindando una amplia
                            variedad de marcas y modelos para que nuestros clientes
                            puedan encontrar la opcin que mejor se adapte a sus
                            necesidades.
                    </p>
                </div>
                <!-- Grid column -->

                <hr class="w-100 clearfix d-md-none" />
```

```

            <!-- Grid column -->
            <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mt-5">
                <h6 class="text-uppercase mb-4 font-weight-bold">Marcas</h6>
                <li><a href="{% url 'marca_detail' marca_id=1 %}" class="text-white" style="text-decoration:none">Acer</a></li>
                <li><a href="{% url 'marca_detail' marca_id=2 %}" class="text-white" style="text-decoration:none">Apple</a></li>
                <li><a href="{% url 'marca_detail' marca_id=3 %}" class="text-white" style="text-decoration:none">Asus</a></li>
                <li><a href="{% url 'marca_detail' marca_id=4 %}" class="text-white" style="text-decoration:none">Dell</a></li>
                <li><a href="{% url 'marca_detail' marca_id=5 %}" class="text-white" style="text-decoration:none">HP</a></li>
                <li><a href="{% url 'marca_detail' marca_id=6 %}" class="text-white" style="text-decoration:none">Lenovo</a></li>
                <li><a href="{% url 'marca_detail' marca_id=7 %}" class="text-white" style="text-decoration:none">MSI</a></li>
            </div>
            <!-- Grid column -->

            <hr class="w-100 clearfix d-md-none" />

            <!-- Grid column -->
            <hr class="w-100 clearfix d-md-none" />

            <!-- Grid column -->
            <div class="col-md-4 col-lg-3 col-xl-3 mx-auto mt-5">
                <h6 class="text-uppercase mb-4 font-weight-bold">Contactos</h6>
                <p><i class="fas fa-home mr-3"></i> Av.Alianza 430, ASA, Arequipa</p>
                <p><i class="fas fa-envelope mr-3"></i> LaPC@gmail.com</p>
                <p><i class="fas fa-phone mr-3"></i> +51 931316402</p>
                <p><i class="fas fa-print mr-3"></i> +51 965789531</p>
            </div>
            <div class="col-md-3 col-lg-2 col-xl-2 mx-auto mt-5">
                <img src="{% static '/img/logo.png' %}" alt="">
            </div>
        </div>
    </section>

  </div>


  <!-- Copyright -->
  <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.2)">
        2023 Copyright:
    <a class="text-white" href="/">lapc.com</a>
  </div>

</footer>
```

- pedido-det.html es capaz de llarmar a vendedores, llamando su id, nombre, epellido paterno

Listing 15: pedido-det.html

```
{% extends 'base.html' %}

{% block content %}
<h1 class="text-center mt-5 mb-4">Detalles del Pedido</h1>
{{error}}
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <table class="table table-bordered table-hover">
                <thead class="thead-dark">
                    <tr>
                        <th colspan="2" class="bg-secondary text-white">Pedido
                            Cabecera</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td> Cdigo  Pedido:</td>
                        <td>{{ pedido_cabecera.id }}</td>
                    </tr>
                    <tr>
                        <td>Fecha:</td>
                        <td>{{ pedido_cabecera.PedCabFec }}</td>
                    </tr>
                    <tr>
                        <td>Cliente:</td>
                        <td>{{ pedido_cabecera.PedCabCodCli }}</td>
                    </tr>
                    <!-- Mostrar ms  informacin  de PedidoCabecera si es necesario -->
                </tbody>
            </table>
            <table class="table table-bordered table-hover">
                <thead class="thead-dark">
                    <tr>
                        <th colspan="2" class="bg-secondary text-white">Pedido
                            Detalle</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td> Artculo :</td>
                        <td>{{ detalle.PedDetArtCod }}</td>
                    </tr>
                    <tr>
                        <td>Cantidad:</td>
                        <td>{{ detalle.PedDetCantidad }}</td>
                    </tr>
                    <tr>
                        <td>Precio Unitario:</td>
                        <td>S/. {{detalle.PedDetPreUniArt}}</td>
                    </tr>
                    <tr>
                        <td>Subtotal:</td>
                        <td>S/. {{detalle.PedDetSubtotal}}</td>
                    </tr>-
```

```
                    <!-- Mostrar ms  informacin  de PedidoDetalle si es necesario -->
                </tbody>
            </table>
            <button onclick="mostrarFormulario()" type="button" class="btn
                 btn-primary">Actualizar</button>
        </div>
    </div>
</div>
<script>
    function mostrarFormulario() {
        window.open("{% url 'actualizar_pedido' pedido_cabecera.id %}", "_self");
    }
</script>
{% endblock %}
```

- Actualización del pedido.

Listing 16: actualizar$_p$*edido.html*

```
{% extends 'base.html' %}

{% block content %}
    <h1 class="text-center mt-5 mb-4">Actualizar Pedido</h1>
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-8">
                <form method="POST" target="_self">
                    {% csrf_token %}
                    {{ form.as_p }}
                    <button type="submit" class="btn btn-primary">Guardar</button>
                </form>
            </div>
        </div>
    </div>
{% endblock %}
```

- el código representa el contenido específico de la plantilla. Aquí se encuentra un formulario de registro de pedido con diferentes campos como vendedor, fecha, artículo y cantidad. También hay un botón de registro.
  La parte final del código incluye un script JavaScript que se ejecuta cuando se envía el formulario. Este script utiliza AJAX para enviar los datos del formulario al servidor de forma asíncrona. Luego, maneja la respuesta del servidor y realiza acciones dependiendo de la respuesta. Por ejemplo, si el pedido se registra con éxito, muestra un mensaje de confirmación y redirige al usuario a la página principal.

Listing 17: pedido.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<section class="vh-130" style="background-color: #eee;">
    <div class="container h-80">
        <div class="row d-flex justify-content-center align-items-center h-100">
          <div class="col-lg-12 col-xl-11">
            <div class="card text-black" style="border-radius: 25px;">
```

```
            <div class="card-body p-md-5">
              <div class="row justify-content-center">
                <div class="col-md-10 col-lg-6 col-xl-5 order-2 order-lg-1">

                  <p class="text-center h1 fw-bold mb-5 mx-1 mx-md-4 mt-4">Register Order</p>
                  {% if error %}
                  <div class="alert alert-danger" role="alert">
                     {{ error }}
                  </div>
                  {% endif %}
                  <form action="{% url 'crear_pedido' %}" method="POST" class="mx-auto">
                     {% csrf_token %}
                     {{ formset.management_form }}

                     <!-- Vendedor -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-user fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
                          <label for="id_pedidos-0-PedCabCodVen" class="form-label">Vendedor</label>
                          <select class="form-control" name="pedidos-0-PedCabCodVen" id="id_pedidos-0-PedCabCodVen">
                             {% for vendedor in vendedores %}
                                <option value="{{ vendedor.id }}">{{ vendedor.VenNom}} {{vendedor.VenApePat}}</option>
                             {% endfor %}
                          </select>
                          <!-- <input type="text" name="CliNom" id="id_CliNom" class="form-control" placeholder="Write your Name"/> -->
                      </div>
                    </div>
                    <!-- Fecha -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-user fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
                          <label class="form-label" id="id_CliApePat">Fecha</label>
                          <input type="date" name="pedidos-0-PedCabFec" format="YY-MM-DD" id="id_pedidos-0-PedCabFec" class="form-control" placeholder="Write your Surname"/>
                      </div>
                    </div>
                    <!-- Articulo -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-user fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
                          <label class="form-label" id="id_CliDNI">Artculo</label>
                          <!-- <input type="number" maxlength="8" name="CliDNI" id="id_CliDNI" class="form-control" placeholder="Write your Surname"/> -->
                          <select class="form-control"
```

```
                            name="pedido_det-PedDetArtCod" required
                            id="id_pedido_det-PedDetArtCod">
                           <option value selected>-----------</option>
                           {% for articulo in articulos %}
                               <option value="{{ articulo.id }}">{{
                                   articulo.ArtNom }}</option>
                           {% endfor %}
                        </select>
                      </div>
                    </div>
                    <!-- Cantidad -->
                    <div class="d-flex flex-row align-items-center mb-4">
                      <i class="fas fa-user fa-lg me-3 fa-fw"></i>
                      <div class="form-outline flex-fill mb-0">
                        <label class="form-label"
                            for="id_pedido_det-PedDetCantidad">Cantidad</label>
                        <input type="number" class="form-control"
                            name="pedido_det-PedDetCantidad" value="1"
                            required="" id="id_pedido_det-PedDetCantidad">
                      </div>
                    </div>

                    <!-- Boton Register -->
                    <div class="d-flex justify-content-center mx-4 mb-3 mb-lg-4">
                      <button type="submit" class="btn btn-primary
                          btn-lg">Register</button>
                    </div>
                  </form>
                </div>
                <div class="col-md-10 col-lg-6 col-xl-7 d-flex align-items-center
                    order-1 order-lg-2">
                  <img src="{% static '/img/order.svg' %}"
                    class="img-fluid" alt="Sample image">
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
</section>
    <script>
        $(document).ready(function() {
            $('form').submit(function(event) {
                event.preventDefault();
                var form = $(this);
                var formData = form.serialize();
                $.ajax({
                    url: form.attr('action'),
                    type: form.attr('method'),
                    data: formData,
                    success: function(response) {
                        var order = JSON.parse(response);
                        var confirmation = confirm('Su pedido ha sido registrado con
                            xito. Desea revisar su pedido?');
                        if (confirmation) {
```

```
                        window.location.href = '/index/'
                    } else {
                        window.location.href = '/';
                    }
                },
                error: function(xhr, status, error) {
                    console.log(xhr.responseText);
                }
            });
        });
    });
  </script>
{% endblock %}
```

- Implementación del Django Administrador. (CRUD para todas las tablas)
  - A continuación se muestran imágenes de la creación exitosa de la Base de datos.

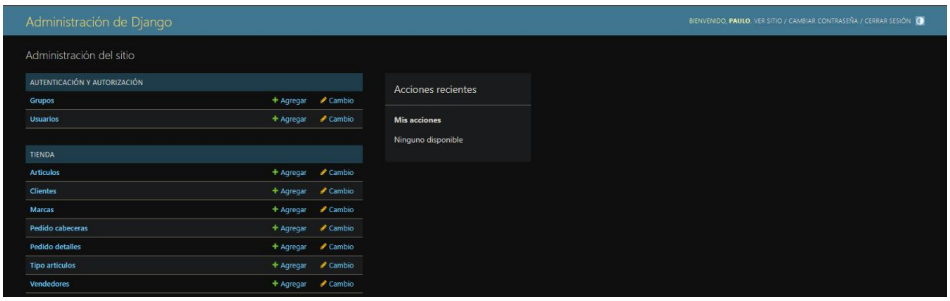


Figura 2: Administración Django.

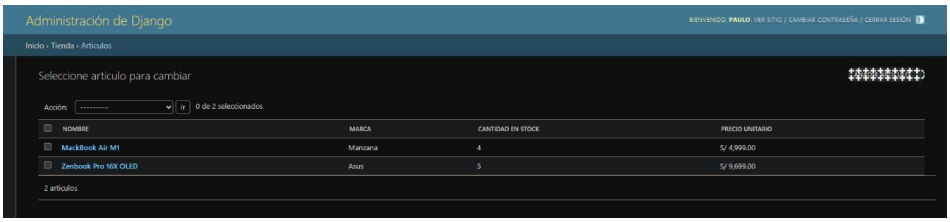


Figura 3: Administración Django Artículos.

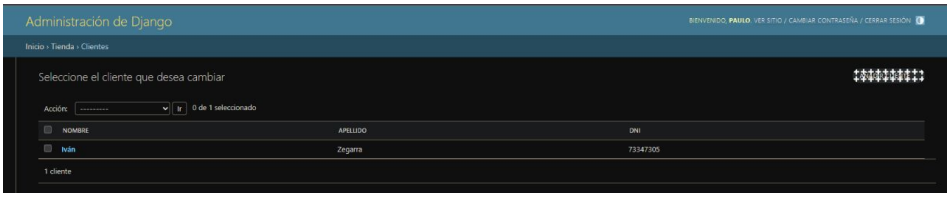


Figura 4: Administración Django Clientes.

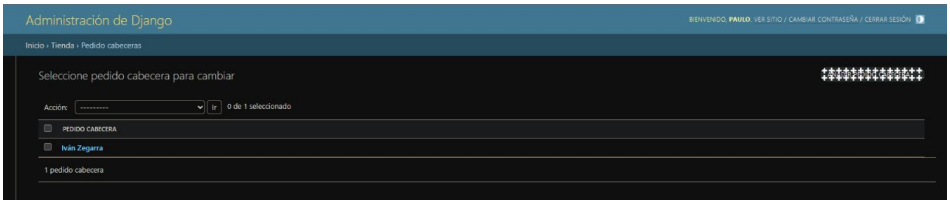


Figura 5: Administración Django Pedido cabeceras.

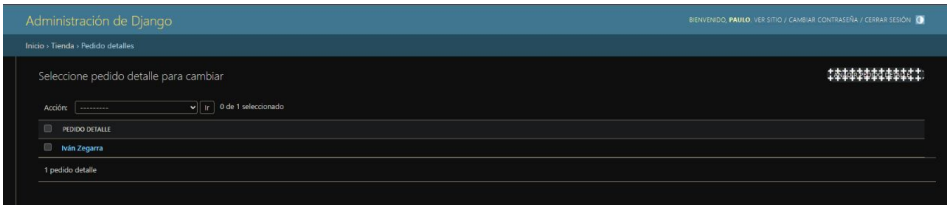


Figura 6: Administración Django Pedido detalles.

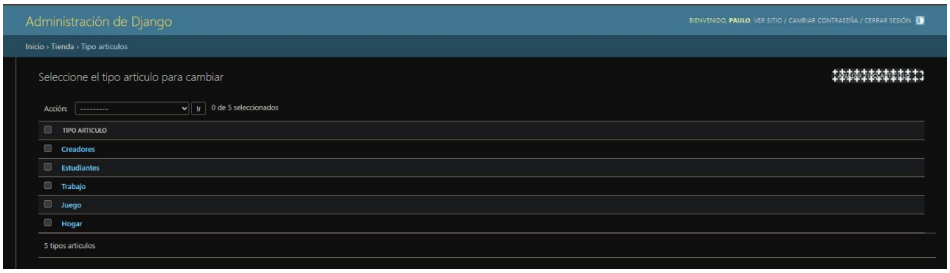


Figura 7: Administración Django Tipo artículos.

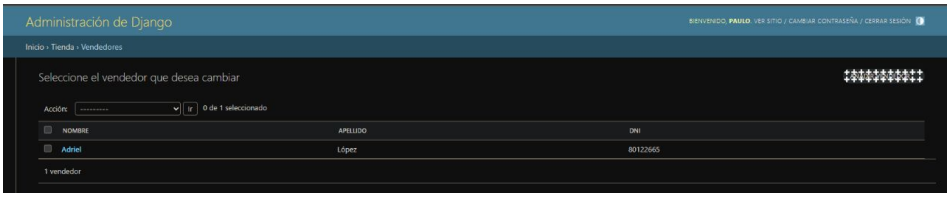


Figura 8: Administración Django Vendedores.

Figura 9: Vista principal - Home 1

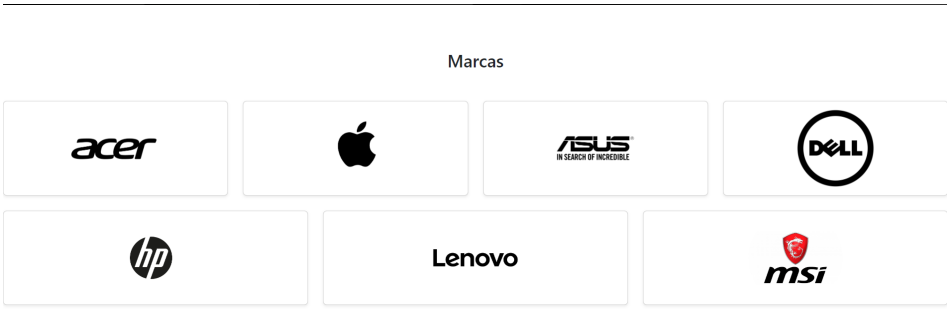


Figura 10: Vista principal - Home 2

Figura 11: Footer

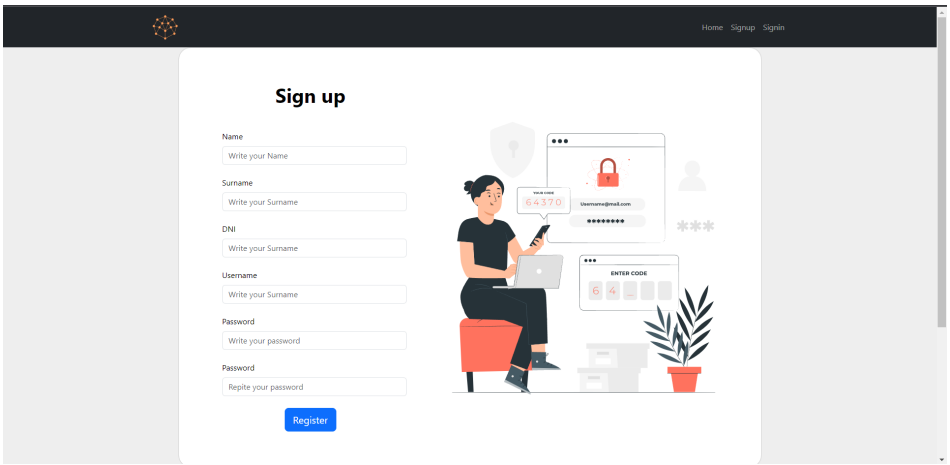


Figura 12: Formulario de Registro de Usuario

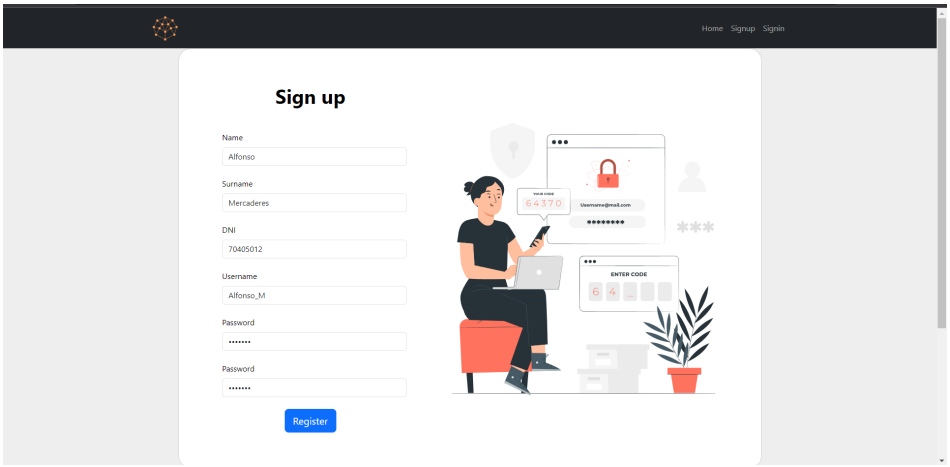


Figura 13: Registro de Usuario Ejemplo

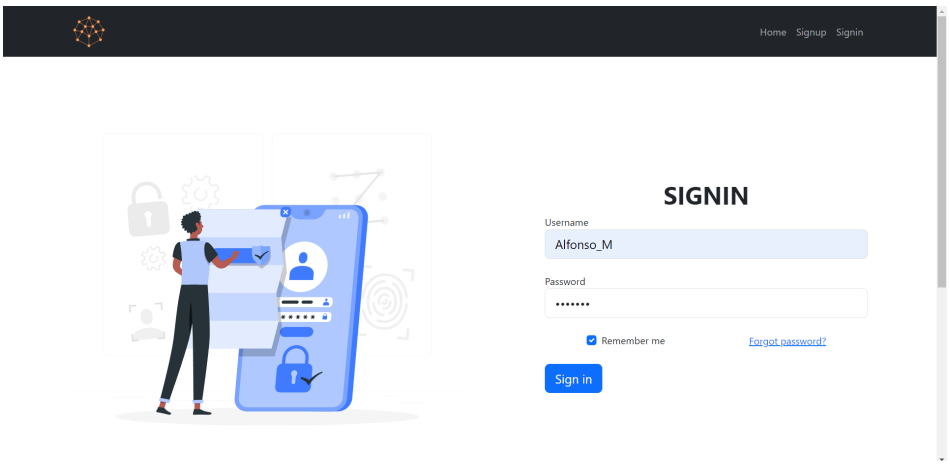


Figura 14: Login

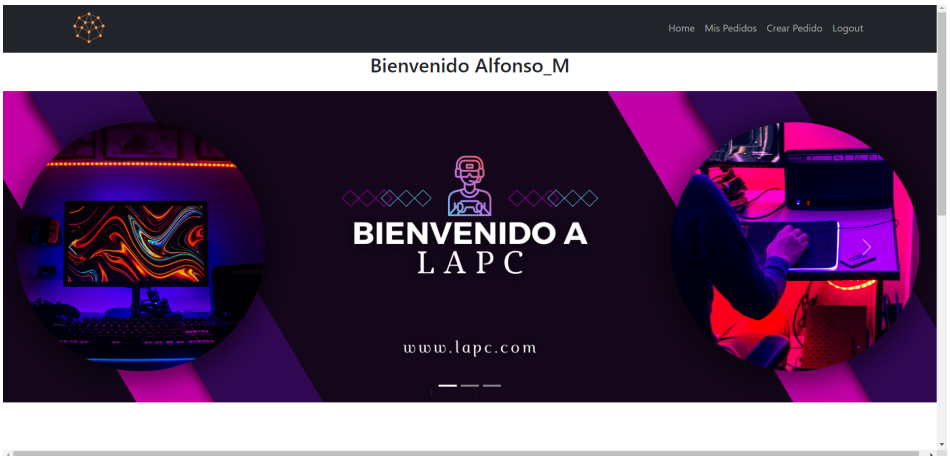


Figura 15: Pagína Bienvenida al Usuario

Figura 16: Mis pedidos

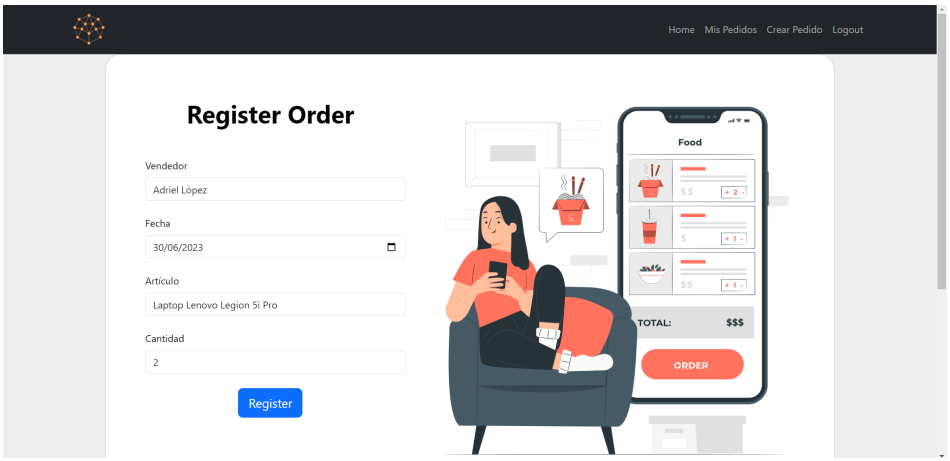


Figura 17: Crear pedido

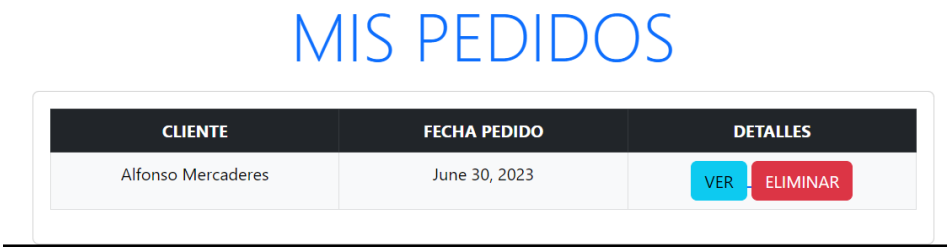


Figura 18: Vista Pedidos Realizados

Detalles del Pedido

Pedido Cabecera	
Código Pedido:	21
Fecha:	June 30, 2023
Cliente:	Alfonso Mercaderes

-

Pedido Detalle	
Artículo:	Laptop Lenovo Legion 5i Pro
Cantidad:	2
Precio Unitario:	S/. 7999.0
Subtotal:	S/. 15998.0

Actualizar

Figura 19: Detalle Pedido

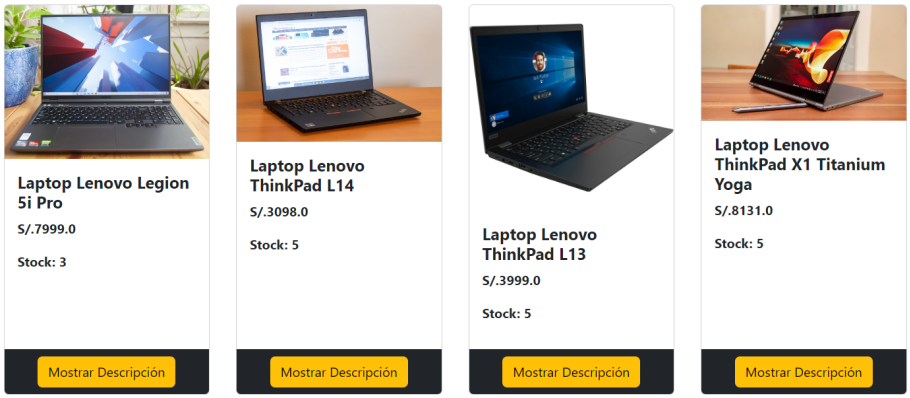


Figura 20: Verificación Stock - Antes de pedido

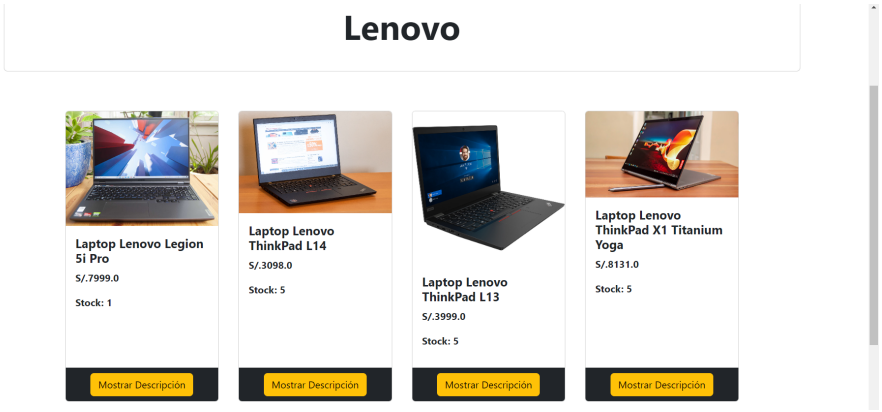


Figura 21: Verificación Stock - Después de pedido

-

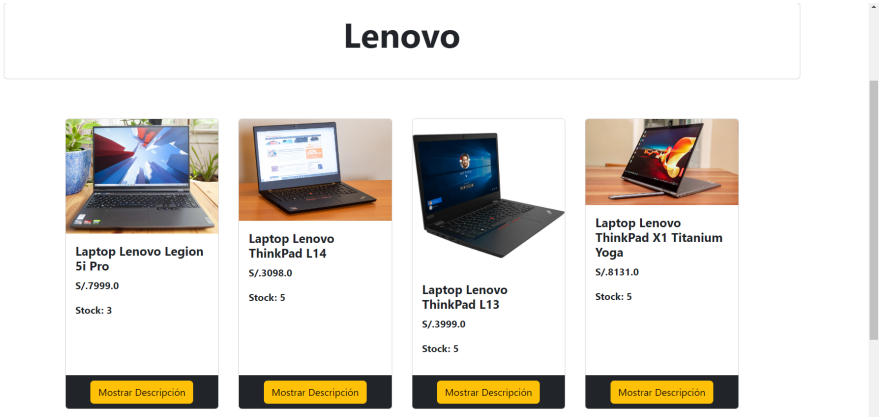


Figura 22: Verificación Stock - Después Eliminar pedido

-

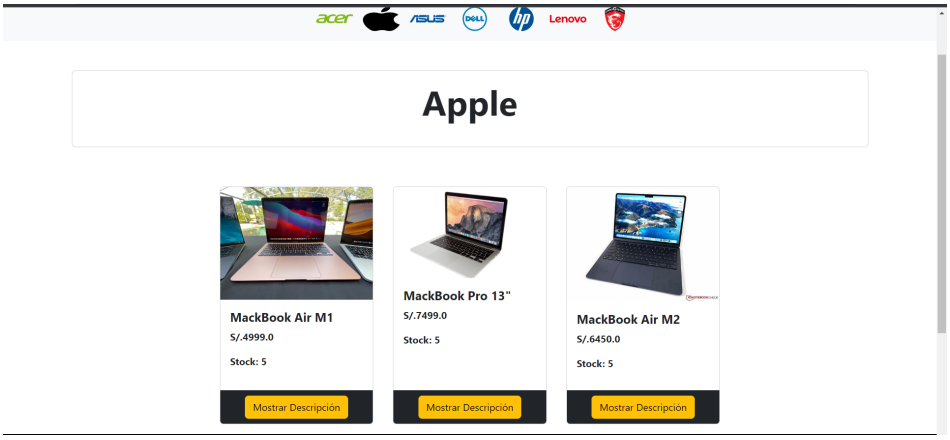


Figura 23: Visualización Artículos Apple - Tarjetas

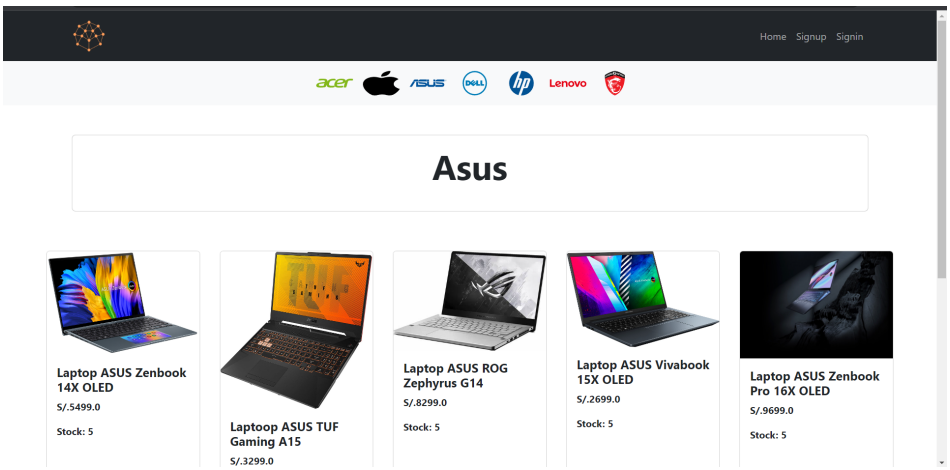
- 

Figura 24: Visualización Artículos Asus - Tarjetas

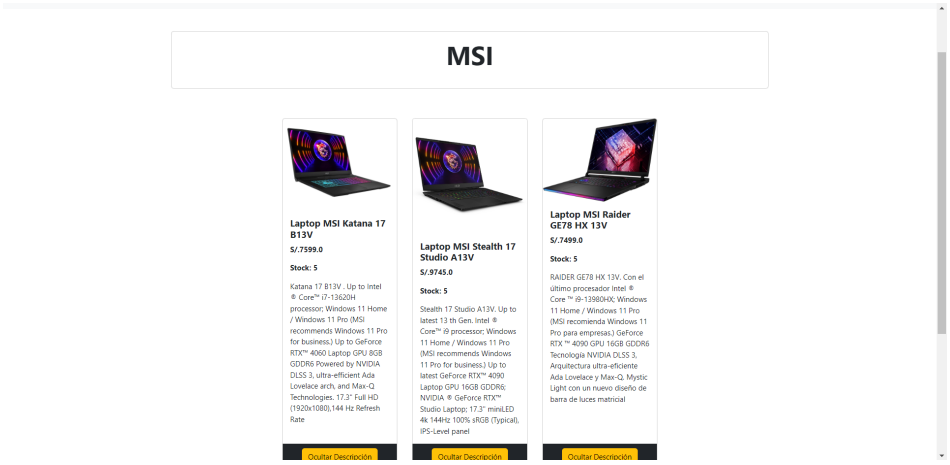
- 

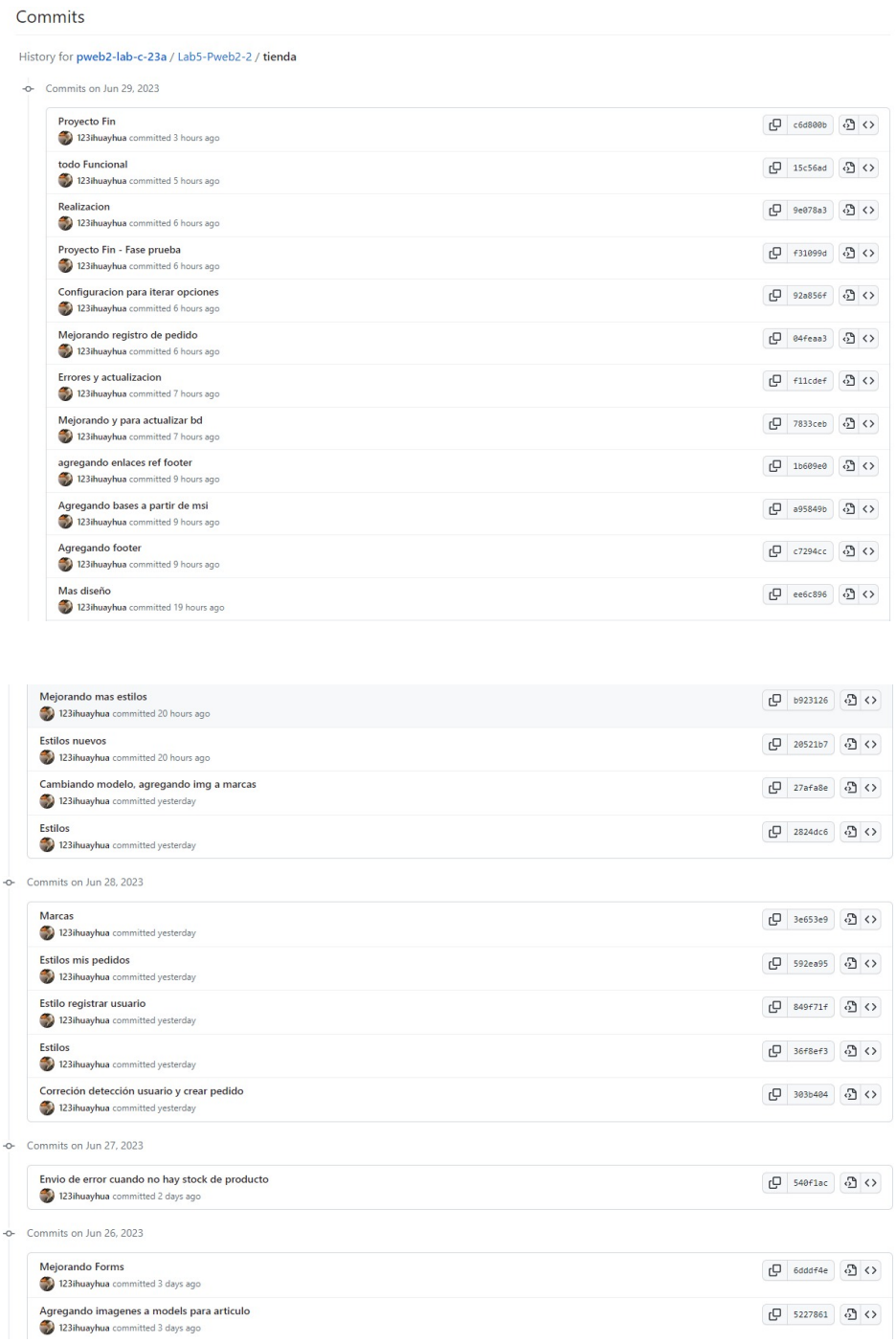
Figura 25: Visualización Artículos MSI - Tarjetas Desplegadas

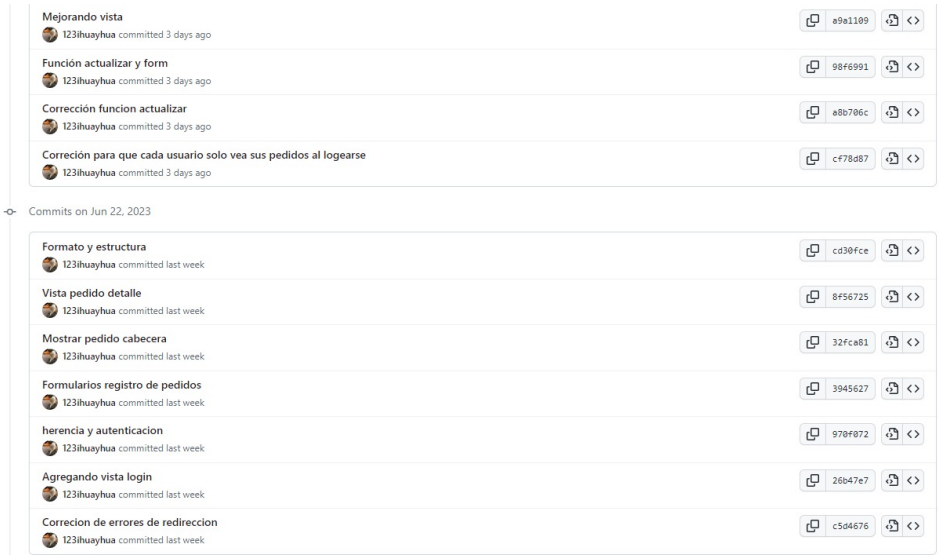
-
- Considerar: atributos adecuados, relaciones necesarias, visualización de datos adecuadas, restricciones en el modelo importantes.

8. Commits

- Correción detección usuario y crear pedido: En esta parte se logró la mejora del usuario, para así poder modficar sus adtos dentro de un formulario.
- Mejorando registro de pedido: Se logró que una orden de pedidos no pueda sobrepasar el stock de un producto.

- Proyecto Fin: Se completó la funcionalidad del código, todo se ejecuta con la perfección, sin errores.
- Los commits son los siguientes:




Mejorando vista
123ihuayhua committed 3 days ago
a9a1109
Función actualizar y form
123ihuayhua committed 3 days ago
98f6991
Corrección funcion actualizar
123ihuayhua committed 3 days ago
a8b706c
Correción para que cada usuario solo vea sus pedidos al logearse
123ihuayhua committed 3 days ago
cf78d87
Commits on Jun 22, 2023
Formato y estructura
123ihuayhua committed last week
cd30fce
Vista pedido detalle
123ihuayhua committed last week
8f56725
Mostrar pedido cabecera
123ihuayhua committed last week
32fca81
Formularios registro de pedidos
123ihuayhua committed last week
3945627
herencia y autenticacion
123ihuayhua committed last week
970f072
Agregando vista login
123ihuayhua committed last week
26b47e7
Correcion de errores de redireccion
123ihuayhua committed last week
c5d4676

9. Estructura del Laboratorio

```
db.sqlite3
Latex_InformeLab06_Pweb
   img
      apple.png
      asus.png
      C1.jpeg
      C2.jpeg
      det1.png
      det2.png
      det3.png
      Diagrama.jpeg
      footer.png
      hom1.png
      Home.png
      lap2.png
      lap3.png
      lap4.png
      logo_abet.png
      logo_episunsa.png
      logo_unsa.jpg
      marca.png
      msi.png
      N1.jpeg
      N2.jpeg
      N3.jpeg
      orden1.png
      pedidos1.png
      pedidos.png
      signin.png
      signup1.png
      signup2.png
      V11.jpeg
      V12.jpeg
      V13.jpeg
      V14.jpeg
      V15.jpeg
      V16.jpeg
      V17.jpeg
   Latex_InformeLab06_Pweb.pdf
   Latex_InformeLab06_Pweb.tex
   src
       actualizar_pedido.html
       admin.py
       apps.py
       base.html
       encabezado.html
       footer.html
       forms.py
       home.html
       index.html
```

```
        marca_acer.html
        marca_apple.html
        marca_asus.html
        marca_dell.html
        marca_hp.html
        marca_lenovo.html
        marca_msi.html
        models.py
        pedido-det.html
        pedido.html
        signin.html
        signup.html
        views.py
localstore
   asgi.py
   __init__.py
   __pycache__
      __init__.cpython-310.pyc
      settings.cpython-310.pyc
      urls.cpython-310.pyc
      wsgi.cpython-310.pyc
   settings.py
   urls.py
   wsgi.py
manage.py
media
   imagenes
       imgs
           acer_4MLZE6r.png
           acer.png
           acer_U4hti2q.png
           apple_DbtnB42.png
           apple.jpeg
           apple.png
           apple_ro0SnVu.png
           asus_d42I5yc.png
           asus.png
           asus_qN5gi3G.png
           carrusel1.svg
           carrusel2.svg
           carrusel3.svg
           dell_ID9dR59.png
           dell_NPG46q3.png
           dell.png
           hp_d7yjRCD.png
           hp_gfoGiQ3.png
           hp.png
           Laptoop_ASUS_TUF_Gaming_A15.jpeg
           Laptoop_ASUS_TUF_Gaming_A15_Vgzi50p.jpeg
           Laptop_Acer_Enduro_N3_dnjhoud.jpg
           Laptop_Acer_Enduro_N3.jpeg
           Laptop_Acer_Enduro_N3.jpg
```

```
Laptop_Acer_Enduro_Urban_N3.jpeg
Laptop_Acer_Enduro_Urban_N3_K8mzXmD.jpeg
Laptop_ASUS_ROG_Zephyrus_G14_DuOzM45.jpeg
Laptop_ASUS_ROG_Zephyrus_G14.jpeg
Laptop_ASUS_Vivabook_15X_OLED_f4o9ELg.jpeg
Laptop_ASUS_Vivabook_15X_OLED.jpeg
Laptop_ASUS_Zenbook_14X_OLED.jpeg
Laptop_ASUS_Zenbook_14X_OLED_sCKOhxw.jpeg
Laptop_ASUS_Zenbook_Pro_16X_OLED.jpg
Laptop_ASUS_Zenbook_Pro_16X_OLED_mJizIqd.jpg
Laptop_Dell_Inspiron_14_G1iX4ck.jpg
Laptop_Dell_Inspiron_14.jpg
Laptop_Dell_Inspiron_15.jpeg
Laptop_Dell_Inspiron_15_v2ikqVz.jpeg
Laptop_Dell_Latitude_3190_8YvKruk.jpg
Laptop_Dell_Latitude_3190.jpg
Laptop_Gamer_Acer_17-3_fXiPSlH.jpg
Laptop_Gamer_Acer_17-3.jpg
Laptop_Gamer_Acer_AN515-57-5323_15-6_fcXOkVi.jpeg
Laptop_Gamer_Acer_AN515-57-5323_15-6.jpeg
Laptop_Gamer_Acer_Nitro_5_15-6.jpeg
Laptop_Gamer_Acer_Nitro_5_15-6_SG03wGO.jpeg
Laptop_HP_14.jpeg
Laptop_HP_14_z8aF7Ms.jpeg
Laptop_HP_Pavilion_15_ah27WLx.jpeg
Laptop_HP_Pavilion_15.jpeg
Laptop_HP_Victus_16.jpeg
Laptop_HP_Victus_16_mDdM8wf.jpeg
Laptop_Lenovo_Legion_5i_Pro.jpg
Laptop_Lenovo_Legion_5i_Pro_tFkrRyG.jpg
Laptop_Lenovo_ThinkPad_L13.jpeg
Laptop_Lenovo_ThinkPad_L13_nYQxn2s.jpeg
Laptop_Lenovo_ThinkPad_L14_20D6B4o.jpg
Laptop_Lenovo_ThinkPad_L14.jpg
Laptop_Lenovo_ThinkPad_X1_Titanium_Yoga_gPbjFBV.jpeg
Laptop_Lenovo_ThinkPad_X1_Titanium_Yoga.jpeg
Laptop_MSI_Katana_17_B13V_pHV8UUc.png
Laptop_MSI_Katana_17_B13V.png
Laptop_MSI_Raider_GE78_HX_13V_EFv97Lj.png
Laptop_MSI_Raider_GE78_HX_13V.png
Laptop_MSI_Stealth_17_Studio_A13V_7MdHpeV.jpeg
Laptop_MSI_Stealth_17_Studio_A13V.jpeg
lenovo_8UB7dlv.png
lenovo.png
lenovo_zWe4l8d.png
logo.png
MackBook_Air_M1_2Zj7C3v.jpeg
MackBook_Air_M1.jpeg
MackBook_Air_M2.jpeg
MackBook_Air_M2_s8VDzG6.jpeg
MackBook_Pro_13.jpeg
MackBook_Pro_13_ZyofE0q.jpeg
```

```
            msi_0OitKl2.png
            msi_n4PeYby.png
            msi.png
 package.json
 package-lock.json
 requirements.txt
tienda
     admin.py
     apps.py
     forms.py
     __init__.py
     migrations
        0001_initial.py
        0002_alter_cliente_clidni.py
        __init__.py
        __pycache__
            0001_initial.cpython-310.pyc
            0002_alter_cliente_clidni.cpython-310.pyc
            __init__.cpython-310.pyc
     models.py
     __pycache__
        admin.cpython-310.pyc
        apps.cpython-310.pyc
        forms.cpython-310.pyc
        __init__.cpython-310.pyc
        models.cpython-310.pyc
        views.cpython-310.pyc
     static
        img
            acer.svg
            apple.svg
            asus.svg
            carrusel1.svg
            carrusel2.svg
            carrusel3.svg
            dell.svg
            hp.svg
            lenovo.svg
            login.svg
            logo.png
            msi.svg
            order.svg
            register.svg
     templates
        actualizar_pedido.html
        base.html
        encabezado.html
        footer.html
        home.html
        index.html
        marca_acer.html
        marca_apple.html
```

```
    marca_asus.html
    marca_dell.html
    marca_hp.html
    marca_lenovo.html
    marca_msi.html
    pedido_det.html
    pedido.html
    signin.html
    signup.html
tests.py
views.py
```

10. Rúbricas

10.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

10.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		20	

11. Referencias

- https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Tutorial$_{l}ocal_{l}ibrary_{w}ebsite$
- `https://github.com/rescobedoq/pw2/tree/main/labs/lab05`

  William S. Vincent. (2022). Django for Beginners: Build websites with Python. Django 4.0. leanpub.com. [URL]

  `https://docs.djangoproject.com/en/4.1/ref/models/fields/`

  `https://docs.djangoproject.com/en/4.0/topics/db/examples/many_to_many/`

  `https://docs.djangoproject.com/en/4.0/topics/db/examples/many_to_one/`

  `https://blog.hackajob.co/djangos-new-database-constraints/`

  `https://stackoverflow.com/questions/3330435/is-there-an-sqlite-equivalent-to-mysqls-describe-table`

  `https://docs.djangoproject.com/en/4.1/ref/validators/#how-validators-are-run`

  `https://docs.djangoproject.com/en/4.1/ref/models/instances/`

  `https://www.youtube.com/watch?v=rHux0gMZ3Eg`

  `https://www.youtube.com/watch?v=OTmQOjsl0eg`

  `https://tex.stackexchange.com/questions/34580/escape-character-in-latex`

  `https://www.sqlitetutorial.net/sqlite-show-tables/`

  `https://www.wplogout.com/export-database-diagrams-erd-from-django/`