Informe de Laboratorio 05

Tema: Django Admin

Nota

Estudiante	Escuela	Asignatura
Garcia Valdivia, Ronald Pablo ▪ rgarciava@unsa.edu.pe Hidalgo Chinchay, Paulo Andre ▪ phidalgo@unsa.edu.pe Huayhua Mayta, Iván Rodrigo ▪ ihuayhuam@unsa.edu.pe Jaita Chura, José Manuel ▪ jjaitac@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	PW2 Semestre: III Código: 1702122

Laboratorio	Tema	Duración
05	Django Admin	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 05 Junio 2023	Al 12 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/Lab5-Pweb2`

5. Tarea

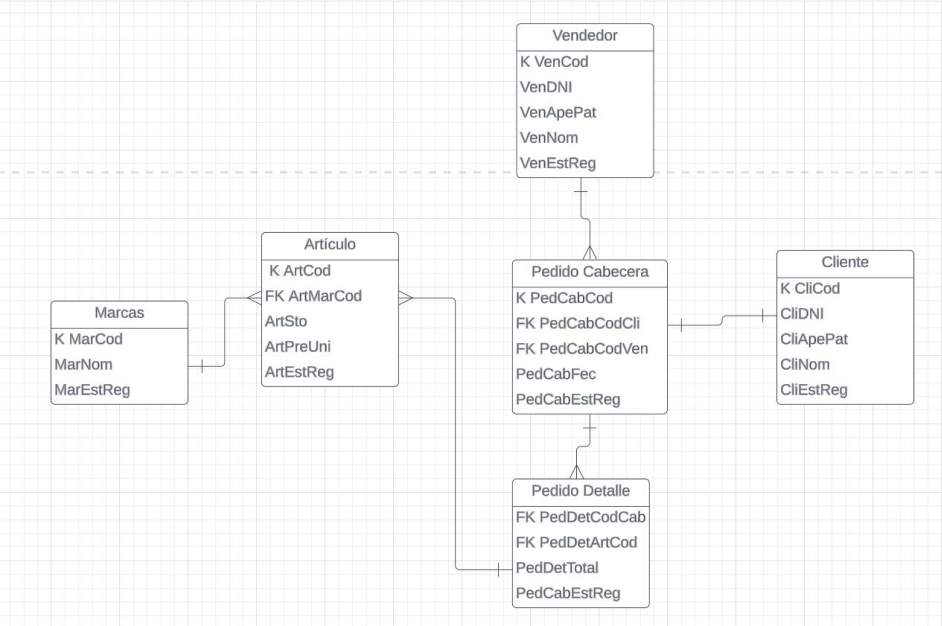
- Elabore un primer informe grupal de la aplicación que desarrollará durante este semestre.
- Utilicen todas las recomendaciones dadas en la aplicación library.
- Acuerdos :
  - Los grupos pueden estar conformado por 1 a 4 integrantes.
  - Sólo se presenta un informe grupal.
  - Sólo se revisa un repositorio. (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"



- 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)
    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):
```

```
25          return self.CliNom + ' ' + self.CliApePat
26
27      password = models.CharField(max_length=128, null=True)
28      username = models.CharField(max_length=150, unique=True, null=True)
29
30  #Marca
31  class Marca(models.Model):
32      MarNom = models.CharField(max_length=20)
33      MarEstReg = models.BooleanField(default=True)
34      #Mostrar nombre de las marcas
35      def __str__(self):
36          return self.MarNom
37
38  #Tipo Artculo
39  class TipoArticulo(models.Model):
40      TipArtNom = models.CharField(max_length=20)
41      TipArtEstReg = models.BooleanField(default=True)
42      #Mostrar nombre de los tipos de artculos
43      def __str__(self):
44          return self.TipArtNom
45
46  #Articulo
47  class Articulo(models.Model):
48      ArtMarCod = models.ForeignKey(Marca, on_delete=models.CASCADE, null=True)
49      ArtTipCod = models.ForeignKey(TipoArticulo, on_delete=models.CASCADE, null=True)
50      ArtNom = models.CharField(max_length=50, null=True)
51      ArtDes = models.TextField(max_length=1000, help_text='Ingresa la descripcin del
            artculo', null=True)
52      ArtSto = models.IntegerField()
53      ArtPreUni = models.FloatField()
54      ArtEstReg = models.BooleanField(default=True)
55      #Mostrar nombre del Artculo
56      def __str__(self):
57          return self.ArtNom
58
59  #Pedido Cabecera
60  class PedidoCabecera(models.Model):
61      PedCabCodCli = models.ForeignKey(Cliente, on_delete=models.CASCADE)
62      PedCabCodVen = models.ForeignKey(Vendedor, on_delete=models.CASCADE)
63      PedCabFec = models.DateField(auto_now=False, auto_now_add=False)
64      PedCabEstReg = models.BooleanField(default=True)
65      #Mostrar pedido cabecera
66      def __str__(self):
67          return str(self.PedCabCodCli)
68
69  #Pedido Detalle
70  class PedidoDetalle(models.Model):
71      PedDetCodCab = models.ForeignKey(PedidoCabecera, on_delete=models.CASCADE,
            related_name='detalles')
72      PedDetArtCod = models.ForeignKey(Articulo, on_delete=models.CASCADE)
73      PedDetCantidad = models.IntegerField(default=0)
74      PedDetPreUniArt = models.FloatField(default=0.0)
75      PedDetSubtotal = models.FloatField(default=0.0)
76      PedDetTot = models.FloatField(default=0.0)
77      PedDetEstReg = models.BooleanField(default=True)
78
```

```
    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)
```

- 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 2: admin.py

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

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

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

```
14
15      def dni(self, obj):
16          return obj.VenDNI
17      dni.short_description = 'DNI'
18
19  #Mostrar detalles en Cliente
20  class ClienteAdmin(admin.ModelAdmin):
21      list_display = ('nombre', 'apellido', 'dni')
22
23      def nombre(self, obj):
24          return obj.CliNom
25      nombre.short_description = 'Nombre'
26
27      def apellido(self, obj):
28          return obj.CliApePat
29      apellido.short_description = 'Apellido'
30
31      def dni(self, obj):
32          return obj.CliDNI
33      dni.short_description = 'DNI'
34
35  #Mostrar detalles en Artculo
36  class ArticuloAdmin(admin.ModelAdmin):
37      list_display = ('nombre_articulo', 'marca', 'stock', 'precio_unitario')
38
39      def nombre_articulo(self, obj):
40          return obj.ArtNom
41      nombre_articulo.short_description = 'Nombre'
42
43      def marca(self, obj):
44          return obj.ArtMarCod
45      marca.short_description = 'Marca'
46
47      def stock(self, obj):
48          return obj.ArtSto
49      stock.short_description = 'Cantidad en Stock'
50
51      def precio_unitario(self, obj):
52          return f'S/ {obj.ArtPreUni:,.2f}'
53      precio_unitario.short_description = 'Precio Unitario'
54
55
56  #Vistas y modelos
57  admin.site.register(Vendedor, VendedorAdmin)
58  admin.site.register(Cliente, ClienteAdmin)
59  admin.site.register(Marca)
60  admin.site.register(TipoArticulo)
61  admin.site.register(Articulo, ArticuloAdmin)
62  admin.site.register(PedidoCabecera)
63  admin.site.register(PedidoDetalle)
```

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

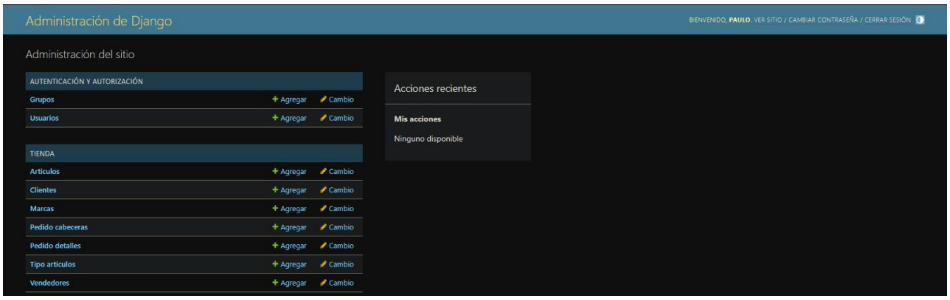


Figura 1: Administración Django.

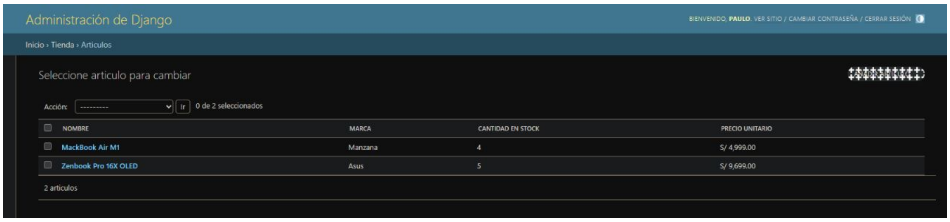


Figura 2: Administración Django Artículos.

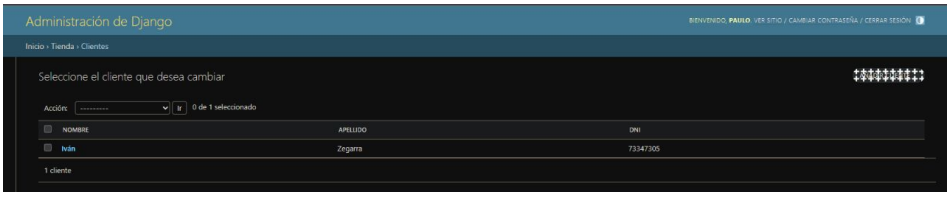


Figura 3: Administración Django Clientes.

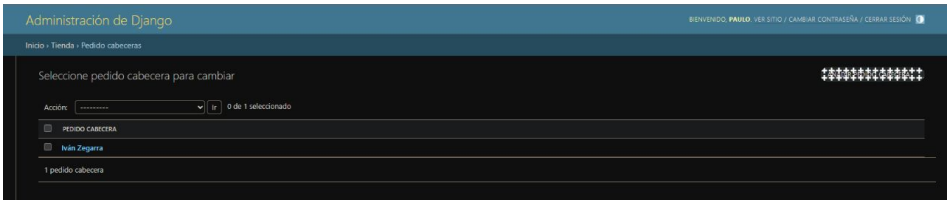


Figura 4: Administración Django Pedido cabeceras.

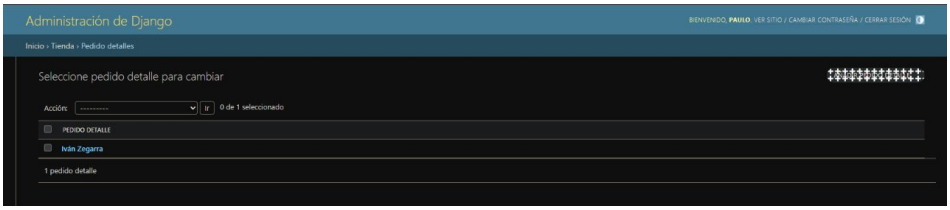


Figura 5: Administración Django Pedido detalles.

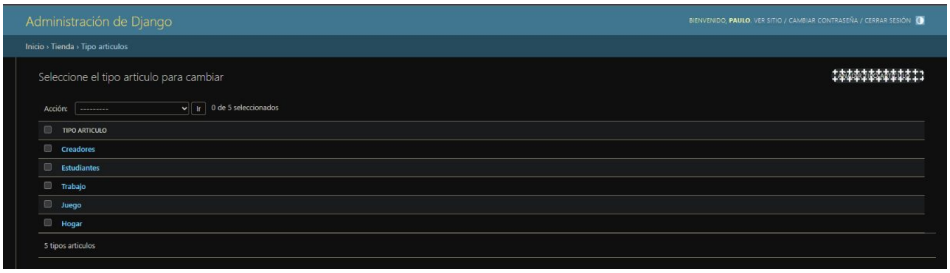


Figura 6: Administración Django Tipo artículos.

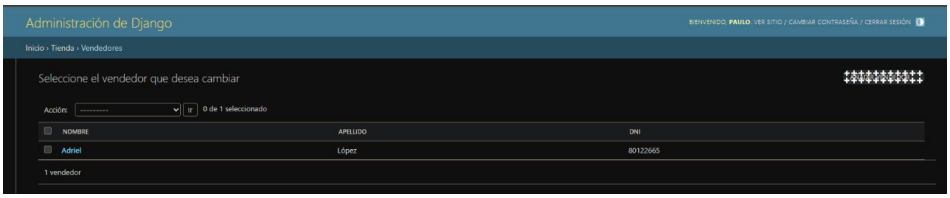
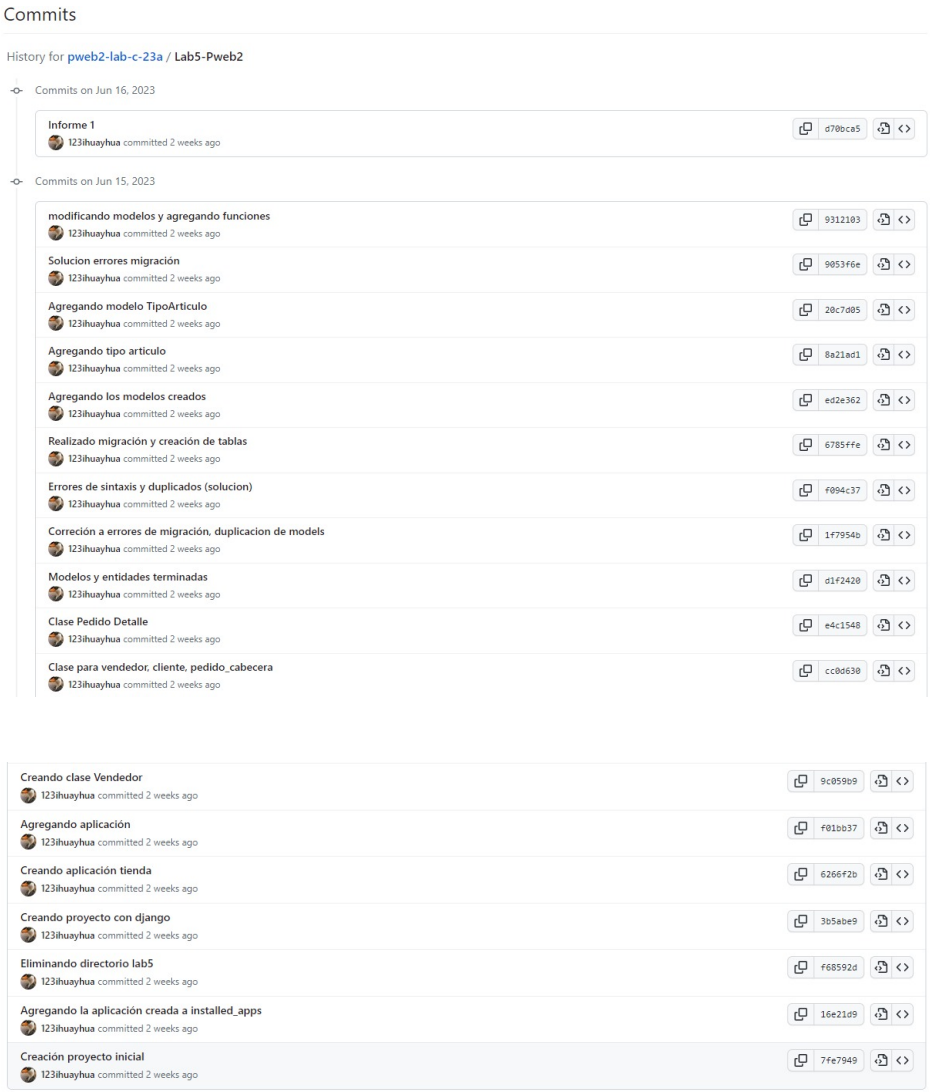


Figura 7: Administración Django Vendedores.

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

8. Commits

- Los commits más importantes fueron los siguientes:
  - Creación aplicación tienda: Es donde se crearán "views.py, models.py, forms.py, etc."
  - Corrección a errores de migración, duplicación de modelos: Se elminaron las diferentes clases modelos que se había repartido etre lo miembros del equipo para juntarlas en una sola de manera funcional
  - Modificación modelos y agregación de funciones: Se concretaron las funciones que nos permiten tener funciones en "Vendedor, Marcas, Artículo, Pedido Detalle, Pedido Cabecera y Cliente"
- Todos commits realizados fueron los siguientes:


Commits
History for pweb2-lab-c-23a / Lab5-Pweb2
Commits on Jun 16, 2023
Informe 1
123ihuayhua committed 2 weeks ago
d70bca5
Commits on Jun 15, 2023
modificando modelos y agregando funciones
9312103
Solucion errores migración
9053f6e
Agregando modelo TipoArticulo
20c7d05
Agregando tipo articulo
8a21ad1
Agregando los modelos creados
ed2e362
Realizado migración y creación de tablas
6785ffe
Errores de sintaxis y duplicados (solucion)
f094c37
Correción a errores de migración, duplicacion de models
1f7954b
Modelos y entidades terminadas
d1f2420
Clase Pedido Detalle
e4c1548
Clase para vendedor, cliente, pedido_cabecera
cc0d630

Creando clase Vendedor
9c059b9
Agregando aplicación
f01bb37
Creando aplicación tienda
6266f2b
Creando proyecto con django
3b5abe9
Eliminando directorio lab5
f68592d
Agregando la aplicación creada a installed_apps
16e21d9
Creación proyecto inicial
7fe7949

9. Estructura del directorio

```
.
|-- db.sqlite3
|-- Latex-InformeLab05-Pweb
|   |-- img
|   |   |-- C1.jpeg
|   |   |-- C2.jpeg
|   |   |-- Diagrama.jpeg
|   |   |-- logo_abet.png
|   |   |-- logo_episunsa.png
|   |   |-- logo_unsa.jpg
|   |   |-- pseudocodigo_insercion.png
|   |   |-- V11.jpeg
|   |   |-- V12.jpeg
|   |   |-- V13.jpeg
|   |   |-- V14.jpeg
|   |   |-- V15.jpeg
|   |   |-- V16.jpeg
|   |   `-- V17.jpeg
|   |-- Lab05.pdf
|   |-- Lab05.pdf:Zone.Identifier
|   |-- Laboratorio05.tex
|   `-- src
|       |-- admin.py
|       |-- Insertion01.java
|       `-- models.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
`-- tienda
    |-- admin.py
    |-- apps.py
    |-- __init__.py
    |-- migrations
    |   |-- 0001_initial.py
    |   |-- 0002_remove_articulo_artmarcod_articulo_artdes_and_more.py
    |   |-- 0003_remove_tipoarticulo_tipartcodmar.py
    |   |-- 0004_articulo_artmarcod.py
    |   |-- 0005_alter_pedidodetalle_peddetcodcab_and_more.py
    |   |-- 0006_pedidodetalle_peddetcantidad_and_more.py
    |   |-- 0007_remove_pedidodetalle_peddetprecio_and_more.py
    |   `-- __init__.py
```

```
|-- models.py
|-- __pycache__
|   |-- admin.cpython-310.pyc
|   |-- apps.cpython-310.pyc
|   |-- __init__.cpython-310.pyc
|   |-- models.cpython-310.pyc
|   `-- views.cpython-310.pyc
|-- 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/`