

# Informe de Laboratorio 05

## Tema: Django

Nota

Estudiante	Escuela	Asignatura
Hancoccallo Achircana, Frank Tito Campos Rutbel Carlos Choque Dongo Gonzalo	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: I Código: 20231001

Laboratorio	Tema	Duración
05	Django	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 13 Junio 2023	Al 18 Junio 2023

### 1. Tarea

- Elabore un primer informe grupal de la aplicación que desarrollara durante este semestre. Utilicen todas las recomendaciones dadas en la aplicación library.
- Utilizar Git para evidenciar su trabajo.
- Enviar trabajo al profesor en un repositorio GitHub Privado, dándole permisos como colaborador.

### 2. Equipos, materiales y temas utilizados

- Sistema Operativo Ubuntu GNU Linux 20 lunar 64 bits Kernel 6.2.
- VIM 9.0.
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Django
- Python3

### 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/RutbelCarlosTC/PW2-proyecto-final.git>
- URL para el laboratorio 05 en el Repositorio GitHub.
- <https://github.com/RutbelCarlosTC/PW2-proyecto-final/tree/main/ministock>

### 4. Explicacion de la aplicacion

- Es una aplicación de software para una tienda de productos de minimarket diseñado específicamente para ayudar en la gestión y operación de la tienda.
- Proporciona una solución tecnológica para simplificar y automatizar diversas tareas, lo que facilita la administración eficiente del negocio.



Figura 1: Minimarket

#### 4.1. Funcionalidades y características

- Nuestra aplicacion cuenta con las siguientes funcionalidades y características:
  1. Gestión de inventario: La aplicación puede realizar un seguimiento del inventario de productos, lo que permite registrar las existencias disponibles, controlar las fechas de caducidad y realizar pedidos automáticos cuando sea necesario. Esto ayuda a evitar la escasez de productos y reduce el riesgo de pérdidas por productos vencidos.
  2. Punto de venta: La aplicación puede actuar como un sistema de punto de venta (POS, por sus siglas en inglés), permitiendo realizar transacciones de venta de manera rápida y precisa. Puede calcular el total de la compra, imprimir recibos.

- 2. Administración de clientes: La aplicación puede almacenar y gestionar información sobre los clientes, como nombres y datos de contacto. Esto facilita la creación de perfiles de clientes y la implementación de programas de lealtad o descuentos personalizados.
- 3. Generación de reportes: La aplicación puede generar reportes detallados sobre las ventas, el inventario, las tendencias de compra, los ingresos y otros datos relevantes.

## 5. Django parte 1

### 5.1. Creación de modelos

Listing 1: Creando models.py

```
$ vim lab05/inventario/models.py
```

- Los modelos en Django representan las estructuras de datos que se utilizan para interactuar con la base de datos.
- Nuestro código define cuatro clases de modelos: Producto, Categoría, Venta y Reporte. Cada clase representa una tabla en la base de datos y define los campos y relaciones entre ellas.
- 1. La clase Producto representa un producto en la tienda. Tiene los siguientes campos:
  - nombre: un campo de texto que almacena el nombre del producto.
  - precioVenta: un campo de tipo decimal que almacena el precio de venta del producto.
  - precioCompra: un campo de tipo decimal que almacena el precio de compra del producto.
  - cantidad: un campo de tipo entero positivo que almacena la cantidad disponible del producto.
  - categoria: una relación de clave externa (ForeignKey) con la clase Categoría, lo que indica que un producto pertenece a una categoría específica.
- Además tiene dos métodos definidos:
  - a) get\_absolute\_url(): este método devuelve la URL absoluta para acceder a la vista de detalle del producto.
  - b) \_\_str\_\_(): este método devuelve una representación en forma de cadena del objeto Producto, que en este caso es el nombre del producto.

Listing 2: clase Producto

```
1 class Producto(models.Model):
2     nombre = models.CharField(max_length=100)
3     precioVenta = models.DecimalField(max_digits=6, decimal_places=2)
4     precioCompra = models.DecimalField(max_digits=6, decimal_places=2)
5     cantidad = models.PositiveIntegerField(default=0)
6     categoria = models.ForeignKey('Categoría', on_delete=models.SET_NULL, null=True)
7
8     #para redirigir su url -> se necesita para la vista productoCreate,update
9     def get_absolute_url(self):
10         return reverse('producto-detail', args=[str(self.id)])
11
12     def __str__(self):
13         return self.nombre
```

Listing 3: Commit: agregando modelos del inventario

```
$ git add .  
$ git commit -m "agregando modelos del inventario"  
$ git push -u origin main
```

- 2. La clase Categoría representa una categoría de productos.
- Tiene un campo llamado nombre que almacena el nombre de la categoría.
- El método `__str__()` devuelve una representación en forma de cadena del objeto Categoría, que en este caso es el nombre de la categoría.

Listing 4: clase Categoría

```
1 class Categoría(models.Model):  
2     nombre = models.CharField(max_length=100)  
3  
4     def __str__(self):  
5         return self.nombre
```

Listing 5: Commit: agregando modelos del inventario2

```
$ git add .  
$ git commit -m "agregando modelos del inventario2"  
$ git push -u origin main
```

- 3. La clase Venta representa una venta realizada en la tienda. Tiene los siguientes campos:
- reporte: una relación de clave externa (ForeignKey) con la clase Reporte, indicando a qué informe pertenece la venta.
- producto: una relación de clave externa (ForeignKey) con la clase Producto, indicando qué producto se vendió.
- cantidad: un campo de tipo entero positivo que almacena la cantidad de productos vendidos.
- monto-total: un campo de tipo decimal que almacena el monto total de la venta.

Listing 6: clase Venta

```
1 class Venta(models.Model):  
2     reporte = models.ForeignKey('Reporte', on_delete=models.CASCADE)  
3     producto = models.ForeignKey('Producto', on_delete=models.CASCADE)  
4     cantidad = models.PositiveIntegerField()  
5     monto_total = models.DecimalField(max_digits=5, decimal_places=2)
```

Listing 7: Commit: se agrego aplicacion inventario

```
$ git add .  
$ git commit -m "se agrego aplicacion inventario"  
$ git push -u origin main
```

- La clase Reporte representa un informe financiero generado en la tienda. Tiene los siguientes campos:
- fecha: un campo de tipo fecha que almacena la fecha del informe.
- saldo-inicial: un campo de tipo decimal que almacena el saldo inicial del informe.
- saldo-final: un campo de tipo decimal que almacena el saldo final del informe.
- ingreso-total: un campo de tipo decimal que almacena el ingreso total del informe.
- inversion-total: un campo de tipo decimal que almacena la inversión total del informe.
- ganancia-neta: un campo de tipo decimal que almacena la ganancia neta del informe.

Listing 8: clase Reporte

```
1 class Reporte(models.Model):
2     fecha = models.DateField()
3     saldo_inicial = models.DecimalField(max_digits=5,decimal_places=2)
4     saldo_final = models.DecimalField(max_digits=5,decimal_places=2)
5     ingreso_total = models.DecimalField(max_digits=5,decimal_places=2)
6     inversion_total = models.DecimalField(max_digits =5, decimal_places=2)
7     ganancia_neta = models.DecimalField(max_digits=5,decimal_places=2)
```

Listing 9: Commit: se agrego formulario para la creacion de un nuevo producto

```
$ git add .
$ git commit -m "se agrego formulario para la creacion de un nuevo producto"
$ git push -u origin main
```

INVENTARIO		
Categorías	+ Añadir	✎ Modificar
Productos	+ Añadir	✎ Modificar
Reportes	+ Añadir	✎ Modificar
Ventas	+ Añadir	✎ Modificar

Figura 2: Tabla

## 5.2. Registrando los modelos

Listing 10: Creando admin.py

```
$ vim lab05/inventario/admin.py
```

- El código tiene las siguientes líneas:
- from django.contrib import admin: Esta línea importa el módulo admin de Django, que proporciona las clases y funciones necesarias para crear la interfaz de administración.

- `from . import models`: Esta línea importa los modelos definidos en el archivo `models.py` del mismo directorio. El punto `(.)` representa el directorio actual.
- `admin.site.register(models.Producto)`: Esta línea registra el modelo `Producto` en la interfaz de administración. Al hacer esto, se crea una sección en la interfaz de administración donde se pueden gestionar los productos, como agregar, editar y eliminar registros de productos.
- `admin.site.register(models.Categoria)`: Esta línea registra el modelo `Categoria` en la interfaz de administración. Esto permite administrar las categorías de productos de manera similar a cómo se gestionan los productos.
- En resumen, este código registra los modelos `Producto`, `Categoria`, `Reporte` y `Venta` en la interfaz de administración de Django. Al hacerlo, se crea una interfaz de administración que permite realizar tareas de gestión y manipulación de datos para estos modelos, como agregar, editar y eliminar registros.

Listing 11: `admin.py`

```
1 from django.contrib import admin
2 from . import models
3 # Register your models here.
4 admin.site.register(models.Producto)
5 admin.site.register(models.Categoria)
6 admin.site.register(models.Reporte)
7 admin.site.register(models.Venta)
```

Listing 12: Commit: Creando corrigiendo admin

```
$ git add .
$ git commit -m "corrigiendo admin"
$ git push -u origin main
```



Figura 3: Django

### 5.3. Estructura de laboratorio 05

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab05/
|--- ministock
|   |--- inventario
|   |--- ministock
|   |--- manage.py
|--- requirements.txt
|--- latex
|   |--- img
|   |   |--- logo_abet.png
|   |   |--- logo_episunsa.png
|   |   |--- logo_unsa.jpg
|   |   |--- minimarketFoto.jpg
|   |--- programacion_lab05_.pdf
|   |--- programacion_lab05_.tex
|   |--- src
|       |--- admin.py
|       |--- categoria.py
|       |--- producto.py
|       |--- reporte.py
|       |--- venta.py
```

## 6. Pregunta: Por cada integrante del equipo, resalte un aprendizaje que adquiri o al momento de estudiar Django. No se reprima de ser detallista. Coloque su nombre entre parentesis para saber que es su aporte.

- (Frank Duks): Django es un framework de desarrollo web versátil y potente que permite crear aplicaciones web de manera eficiente y segura. Sus características, como la productividad, la arquitectura MVC, la seguridad, el soporte de bases de datos, hacen de este framework muy interesante.
- (Rutbel Carlos Ttito): Django permite la creacion de modelos a traves de clases en python. Ademas ofrece una interfaz de administracion (/admin) para manipular facilmente los registros de nuestros modelos.

## 7. Rúbricas

### 7.1. 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.

## 7.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple 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 los ítems.
- 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 están 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	
<b>Total</b>		20		20	



## 8. Referencias

- <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>
- <https://docs.djangoproject.com/en/4.2/intro/tutorial01/>