Informe de Laboratorio 06

Tema: Django(Admin)

Nota

Estudiantes	Escuela	Asignatura
Anco Aymara Jean Pierre,Arocutipa Gutierrez Luis Edgar, Mendoza Contreras Giovani Angel, Suasaca Pacompia Alvaro Gustavo jancoay@unsa.edu.pe, gmendozaco@unsa.edu.pe, asuasaca@unsa.edu.pe, larocutipa@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 1702122

Laboratorio	Tema	Duración
06	Django(Admin)	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 06 Junio 2024	Al 11 Junio 2024

1. Enlaces

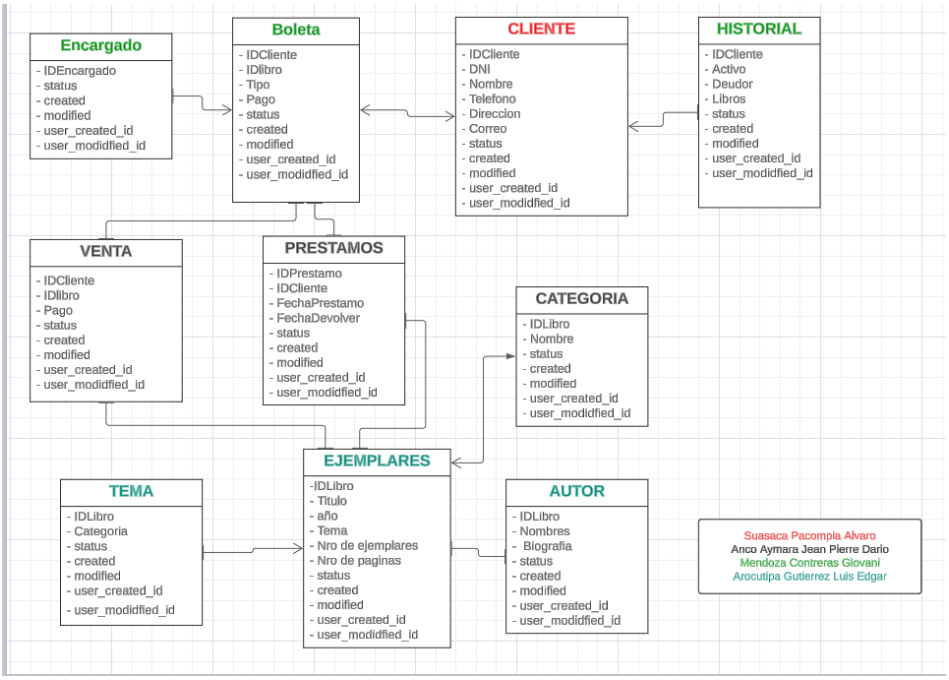
Url de Git-Hub: `https://github.com/alvaro865/pw2-24a`

Url para ver el modelo de datos: `https://lucid.app/lucidchart/510fb1fb-3be0-4ca7-aeac-2177ef40f857/edit?view_items=fbSIr~3nXQbd&invitationId=inv_1c0c2b14-e1e1-4b40-8372-f8209c086017`

2. Aplicacion

La aplicacion web es una biblioteca virtual en la cual una ves ya logeado puedas ver los ejemplares disponibles, podras comprar el libro o alquilarlo segun la preferencia.

3. Modelo de datos



Nuestro modelo de datos consta de 10 de tablas

Para empezar la tabla cliente tiene la relacion de uno a uno con la tabla historial ya que cada cliente tendra un solo historial, tambien tiene la relacion de uno a muchos con tabla boleta puesto que un cliente puede tener muchas boletas.

Ahora nuestra tabla boleta tiene una relacion de muchos a uno con la tabla encargado, tambien tiene una relacion de uno a muchos con la tabla venta ya que una boleta puede tener muchas ventas, tambien tiene una relacion de uno a muchos con la tabla prestamos.

Continuamos con nuestra tabla ejemplares la cual tiene una relacion de uno a muchos con las tablas ventas y prestamos, tambien tiene la relacion de muchos a uno con la tabla tema y categoria y autor.

4. Django(Admin)

4.1. Creacion del proyecto django

- Para empezar con la creacion de nuestro proyecto en django primero tenemos que verificar que tengamos lo necesario:

Listing 1: Usamos pip list

```
(entorno) D:\priv\lab_pweb2\proyecto>pip list
        Package Version
        -------- -------
        asgiref 3.8.1
        Django  5.0.6
        pip     24.0
        pygame  2.5.2
```

```
sqlparse 0.5.0
tzdata  2024.1
```

- Una ves verificado que tengamos django procedemos a crear nuestro proyecto llamado libreria de la sgte manera:

Listing 2: Usamos pip list

```
(entorno) D:\priv\lab_pweb2\proyecto> django-admin startproject libreria .
```

- En nuestro editor nos aparecera las carpetas necesarias para nuestro proyecto

Listing 3: En nuestro editor se veria asi

```
|-libreria
   |--- __init.py__
   |--- asgi.py
   |--- settings.py
   |--- urls.py
   |--- wsgi.py
|-manage.py
```

haremos una configuracion de la informacion en settings

Listing 4: libreria.py

```
LANGUAGE_CODE = 'es'
TIME_ZONE = 'America/Lima'
```

Ahora para verificar que no hay problemas haremos correr el servidor

Listing 5: Mostramos la pagina de inicio

```
(entorno) D:\priv\lab_pweb2\proyecto>python manage.py runserver
```

4.2. Creacion de la app libon

Ahora crearemos la app libon, se nos creara un directorio con el nombre de la app t los archivos necesarios.

```
(entorno) D:\priv\lab_pweb2\proyecto>django-admin startapp libon
```

Nos muestra la carpeta de la sgte manera

Listing 6: En nuestro editor se veria asi

```
|- libon
   |--- __init.py__
   |--- admin.py
   |--- apps.py
   |--- migrations
   |--- models.py
   |--- test.py
   |--- views.py
|-libreria
|-manage.py
```

Registramos nuestra aplicacion en nuestro proyecto.

Listing 7: libreria.py

```
INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'libon',
    ]
```

4.3. Creando modelos

Debemos de hacer obsoleto el archivo models.py para crear modelos en archivos individuales.

```
(entorno) D:\priv\lab_pweb2\proyecto>move models.py models.py.deprecated
```

ahora crearemos una nueva carpeta models la cual tendra loa archivos de los distintos modelos.

Listing 8: En nuestro editor se veria asi

```
|- libon
   |--- __pycahe__
   |--- migrations
   |--- models
   |      |--- autor.py
   |      |--- boleta.py
   |      |--- categoria.py
   |      |--- cliente.py
   |      |--- ejemplares.py
   |      |--- encargado.py
```

```
|     |--- historial.py
|     |--- prestamos.py
|     |--- tema.py
|     |--- venta.py
|--- __init.py__
|--- admin.py
|--- apps.py
|--- migrations
|--- models.py.deprecated
|--- test.py
|--- views.py
|-libreria
|-manage.py
```

Autor: Controlaremos la tabla de autores que tengan un ejemplar mediante el modelo Autor. Algunos de los atributos que tuvimos en cuenta para poder crear un autor son: nombre, apellido, nacionalidad y biografía. Todos estos atributos son ingresados mediante un tipo de campo CharField; a excepción del último, para el cual utilizamos un tipo de campo TextField ya que necesitamos más capacidad en el ingreso de datos.

Listing 9: Código para la tabla Autor

```
from django.db import models
from django.utils.translation import gettext_lazy as _

import uuid

class Autor(models.Model):
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    nacionalidad = models.CharField(max_length=50)
    biografia = models.TextField(null=True, blank=True)
    status = models.CharField(max_length=20)
    created = models.DateTimeField(auto_now_add=True)
    modified = models.DateTimeField(auto_now=True)
    user_created_id = models.TextField(max_length=40)
    user_modidfied_id = models.TextField(max_length=40)

    def __str__(self):
        return f'{self.nombre} {self.apellido}'
```

Boleta: tenemos los atributos IDCliente(sera la clave primaria), IDLibro(esta clave nos conectara a ejemplares), tipo(este atributo nos dira si es venta o prestamo), pago(nos indicara cuanto se ha pagado si fuese el caso venta), status(charfield), modified(charfield), created(charfield), status(charfield), userCreadtedID(charfield) y userModifiedID(charfield).

Listing 10: Codigo para la tabla Boleta

```
from django.db import models

class Boleta(models.Model):
    IDlibro = models.IntegerField()
    tipo = models.CharField(max_length=50)
    pago = models.CharField(max_length=50)
    status = models.CharField(max_length=50)
    created = models.CharField(max_length=50)
```

```
        modified = models.CharField(max_length=50)
        user_created_id = models.CharField(max_length=50)
        user_modidfied_id = models.CharField(max_length=50)
```

Categoria: La tabla categoria se represnta por la clase Categoria, en sus atributos necesitamos datos para poder manejar las llamadas de las tablas, entre estos tenemos los siguientes:

IDLibro como las importantes(sera la clave foranea para poder llamar a la tabla categoria), luego tenemos otras como Nombre(charfield), Created(charfield), Modified(charfield), UserModifiedId(charfield) y UserModifiedId(Charfield).

Listing 11: Codigo para la tabla Categoria

```
        class Categoria(models.Model):
            Nombre=models.CharField(max_length=50, null=False)
            Created=models.CharField(max_length=50, verbose_name="Created", null=False,
                blank=False)
            Modified=models.CharField(max_length=50, verbose_name="modified", null=True,
                blank=True)
            User_created_id=models.CharField(max_length=50, verbose_name="UserCreatedId",
                null=False, blank=False)
            User_modified_id=models.CharField(max_length=50,
                verbose_name="UserModifiedId", null=True, blank=True)

            def __str__(self):
                return f'Categoria: {self.Created} - {self.Nombre}'
```

La tabla de clientes estara representado por la clase Cliente tenemos sus atributos con los que trabajeremos los datos de este en el cual el atributo IDCliente(charfield) sera la llave que se conectara con otras tablas como boleta o historial, luego tenemos otros datos como el dni(charfield), nombre(charfield), telefono(integerfield), direccion(charfield), correo(emailfield) y otros.

```
from django.db import models
from django.utils.translation import gettext_lazy as _

import uuid

class Cliente(models.Model):
    IDCliente = models.CharField(max_length=20)
    DNI = models.CharField(max_length=8)
    Nombre = models.CharField(max_length=40)
    Telefono = models.IntegerField(max_length=10)
    Direccion = models.CharField(max_length=40)
    Correo = models.EmailField(unique=True)
    status = models.CharField(max_length=20)
    created = models.CharField(max_length=20)
    modified = models.CharField(max_length=20)
    user_created_id = models.CharField(max_length=20)
    user_modidfied_id = models.CharField(max_length=20)
```

Ejemplar: Para poder crear ejemplares para la biblioteca implementamos este modelo. Tuvimos en cuenta los siguientes atributos para la creación de un ejemplar:

- titulo: Este campo almacena el título del libro.

- categoría: Es un ForeignKey que establece relación con el modelo Categoria. Indica la categoría a la que pertenece el ejemplar.

- tema: Es un ManyToManyField que establece una relación con el modelo Tema. Un ejemplar puede estar asociado a múltiples temas y viceversa.

- año: Almacena el año de publicación del libro

- autor: Es un ManyToManyField que establece una relación con el modelo Autor. Un ejemplar puede estar asociado a múltiples autores y viceversa.

- sinopsis: Almacena una breve descripción del contenido del libro. Este campo es opcional y puede dejarse en blanco.

- paginas y stock: Indica el número de páginas y la cantidad de ejemplares disponibles.

Listing 12: Código para la tabla Ejemplar

```
from django.db import models
from django.utils.translation import gettext_lazy as _

import uuid
from .autor import Autor
from .categoria import Categoria
from .tema import Tema

class Ejemplar(models.Model):
    titulo = models.CharField(max_length=150)
    categoria = models.ForeignKey(Categoria, on_delete=models.CASCADE,
        related_name="ejemplares")
    tema = models.ManyToManyField(Tema , related_name="ejemplares")
     ao  = paginas = models.CharField(max_length=50)
    autor = models.ManyToManyField(Autor, related_name="ejemplares")
    sinopsis = models.TextField(null=True, blank=True)
    paginas = models.IntegerField()
    stock = models.IntegerField()
    status = models.CharField(max_length=20)
    created = models.DateTimeField(auto_now_add=True)
    modified = models.DateTimeField(auto_now=True)
    user_created_id = models.TextField(max_length=40)
    user_modidfied_id = models.TextField(max_length=40)

    def __str__(self):
        return self.titulo
```

Encargado: para la tabla encargado tenemos los atributos IDEncargado(sera la clave primaria), status(charfield), modified(charfield), created(charfield), status(charfield), userCreadtedID(charfield) y userModifiedID(charfield).

Listing 13: Codigo para la tabla Encargado

```
from django.db import models

class Encargado(models.Model):
    id_encargado = models.CharField(max_length=50)
    status = models.CharField(max_length=50)
    created = models.CharField(max_length=50)
    modified = models.CharField(max_length=50)
    user_created_id = models.CharField(max_length=50)
    user_modidfied_id = models.CharField(max_length=50)
```

Historial: Para la tabla historial tenemos IDCliente(clave primaria), activo(nos informara si esta activo en alguna opcion), deudor(con este atributo veremos si nos debe algo), libros(podremos observar que libros a leido), status(charfield), modified(charfield), created(charfield), status(charfield), userCreadtedID(charfield) y userModifiedID(charfield).

Listing 14: Codigo para la tabla Historial

```
from django.db import models

class Historial(models.Model):
    idCliente = models.CharField(max_length=40)
    activo = models.CharField(max_length=40)
    deudor = models.CharField(max_length=40)
    libros = models.CharField(max_length=50)
    status = models.CharField(max_length=50)
    created = models.CharField(max_length=50)
    modified = models.CharField(max_length=50)
    user_created_id = models.CharField(max_length=50)
    user_modidfied_id = models.CharField(max_length=50)
```

Prestamos: Para la tabla prestamos se añadio los siguinetes atributos: IDPrestamo(la clave primaria autoincrementable que se genera automaticamente), IDCliente(la clave foranea para poder llamra a los datos de la tabla cliente),FechaPrestamo y FechaDevolver(datos tipo DateField para el registro de prestamo y devolucion), created(charfield), status(booleanfield), modified(charfield), UserCreatedId(charfield) y UserModifedId(charfield).

Para el codigo tenemos lo siguiente:

Listing 15: Codigo para la tabla Prestamos

```
class Prestamos(models.Model):
    IDPrestamo =models.AutoField(primary_key=True)
    IDCliente=models.ForeignKey(Cliente, on_delete=models.CASCADE,
        verbose_name='Cliente')
    FechaPrestamo=models.DateField(verbose_name="FechaPrestamo")
    FechaDevolver=models.DateField(verbose_name="FechaDevolver")
    Created=models.CharField(max_length=50, verbose_name="Created", null=False,
        blank=False)
    Status=models.BooleanField(verbose_name="Status")
    Modified=models.CharField(max_length=50, verbose_name="modified", null=True,
        blank=True)
    User_created_id=models.CharField(max_length=50, verbose_name="UserCreatedId",
        null=False, blank=False)
    User_modified_id=models.CharField(max_length=50,
        verbose_name="UserModifiedId", null=True, blank=True)

    def __str__(self):
        return f'Prestamos: {self.IDCliente} - {self.FechaDevolver}'
```

Tema: Este modelo representa un tema que puede ser asignada a libro en una biblioteca. Para su implementación tuvimos en cuenta el campo "nombre", donde se almacena el nombre del tema. Además es único, lo que significa que no puede haber dos temas con el mismo nombre en la base de datos.

Listing 16: Código para la tabla Tema

```
from django.db import models
from django.utils.translation import gettext_lazy as _
```

```
import uuid

class Tema(models.Model):
    nombre = models.CharField(max_length=50, unique=True)
    status = models.CharField(max_length=20)
    created = models.DateTimeField(auto_now_add=True)
    modified = models.DateTimeField(auto_now=True)
    user_created_id = models.TextField(max_length=40)
    user_modidfied_id = models.TextField(max_length=40)

    def __str__(self):
        return self.nombre
```

Venta: para el tema de venta se creo los siguientes atributos en la tabla Ventas mediante DJnago: IDCliente(la clave foranea para poder pedir los datos de la tabla cliente), IDLibro(otra clave foranea para poder pedir los datos de la tabla Ejemplares),Pago(un decimalField que registrara el monto total del pago), Created(Charfield), Modified(Charfield), UserCretedId(Charfield) y UserModifiedId(Charfield).

para los datos atneriores se creo el siguiente codigo:

Listing 17: Codigo para la tabla Prestamos

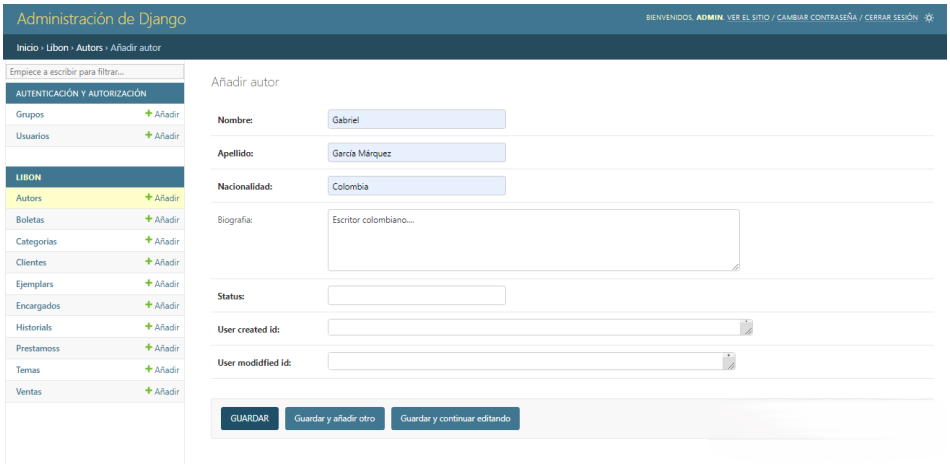
```
class Venta(models.Model):
    IDCliente=models.ForeignKey(Cliente, on_delete=models.CASCADE,
        verbose_name='IDCliente')
    IDLibro=models.ForeignKey(Ejemplar, on_delete=models.CASCADE,
        verbose_name='IDLibro')
    Pago=models.DecimalField(max_digits=10, decimal_places=2)
    Created=models.CharField(max_length=50, verbose_name="Created", null=False,
        blank=False)
    Modified=models.CharField(max_length=50, verbose_name="modified", null=True,
        blank=True)
    User_created_id=models.CharField(max_length=50, verbose_name="UserCreatedId",
        null=False, blank=False)
    User_modified_id=models.CharField(max_length=50,
        verbose_name="UserModifiedId", null=True, blank=True)

    def __str__(self):
        return f'Venta: {self.IDCliente} - {self.IDLibro} - {self.Created}'
```

5. Implementacion del Django administrador

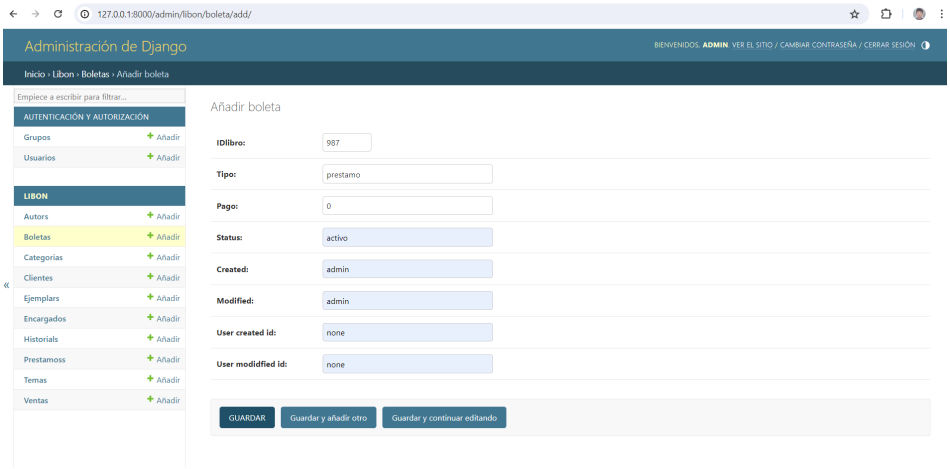
5.1. Autor

Ahora crearemos un autor desde el administrados de django

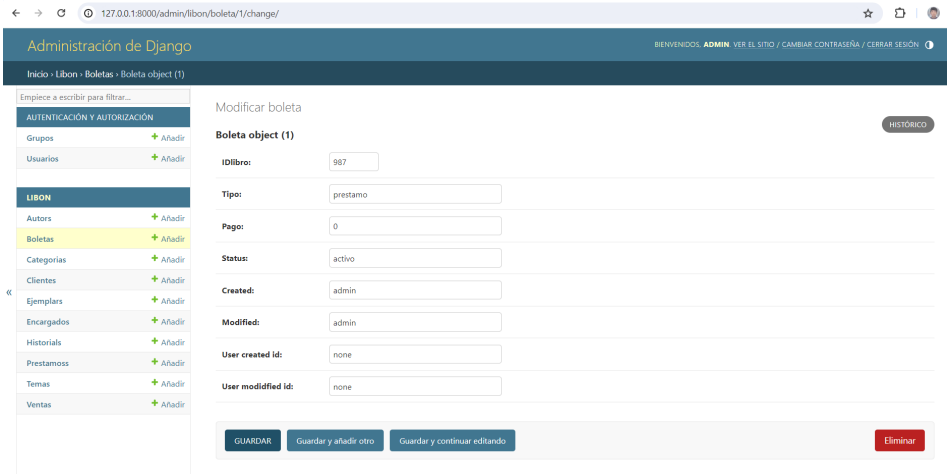


5.2. Boleta

Ahora crearemos una boleta desde el administrados de django

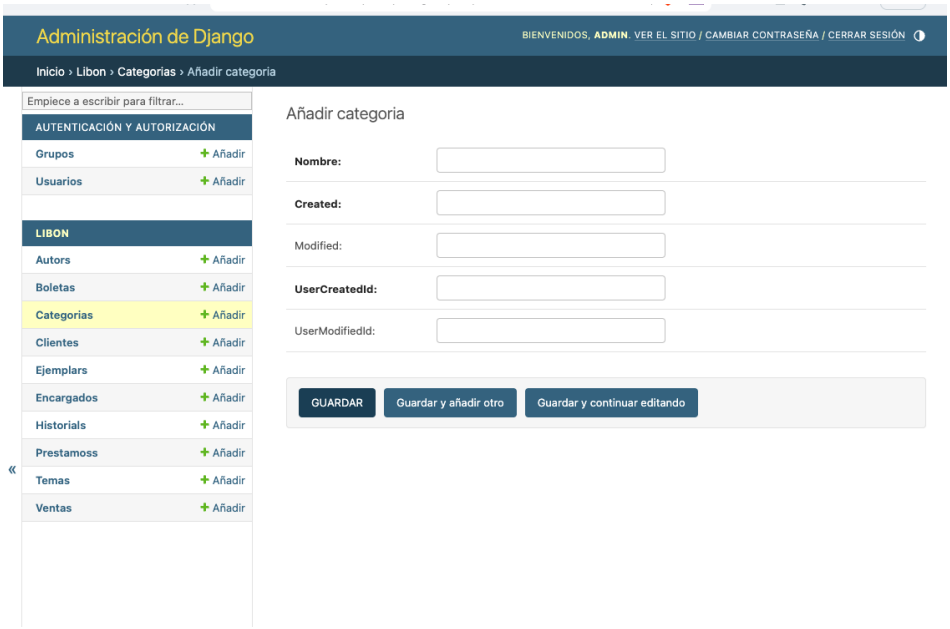


Cuando ingresamos en el objeto creado no aparece la informacion como tambien modificar y botones de opciones como guardado, guardado y añadir otro, guardar y continuar editando, y eliminar.

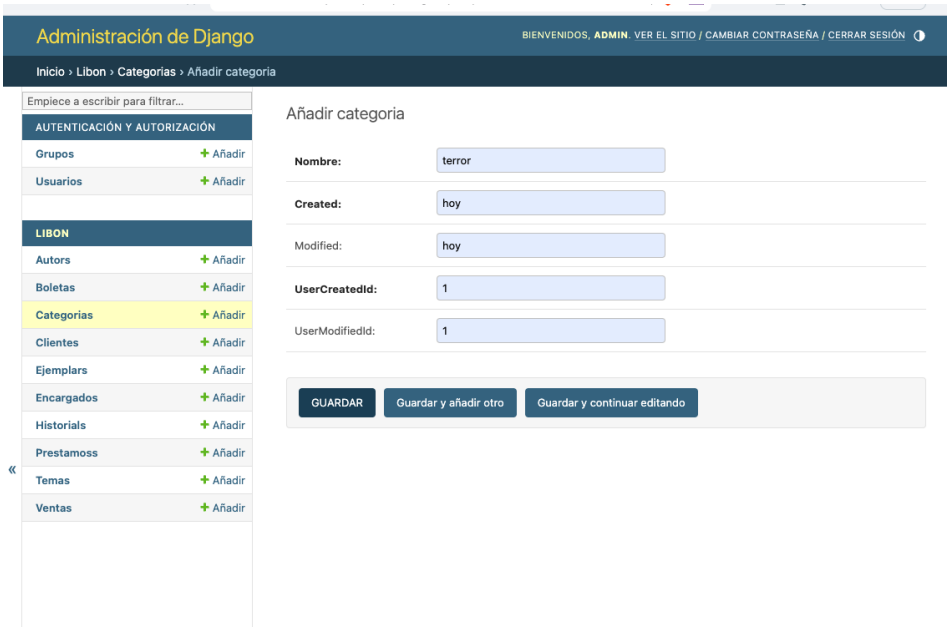


5.3. Categoria

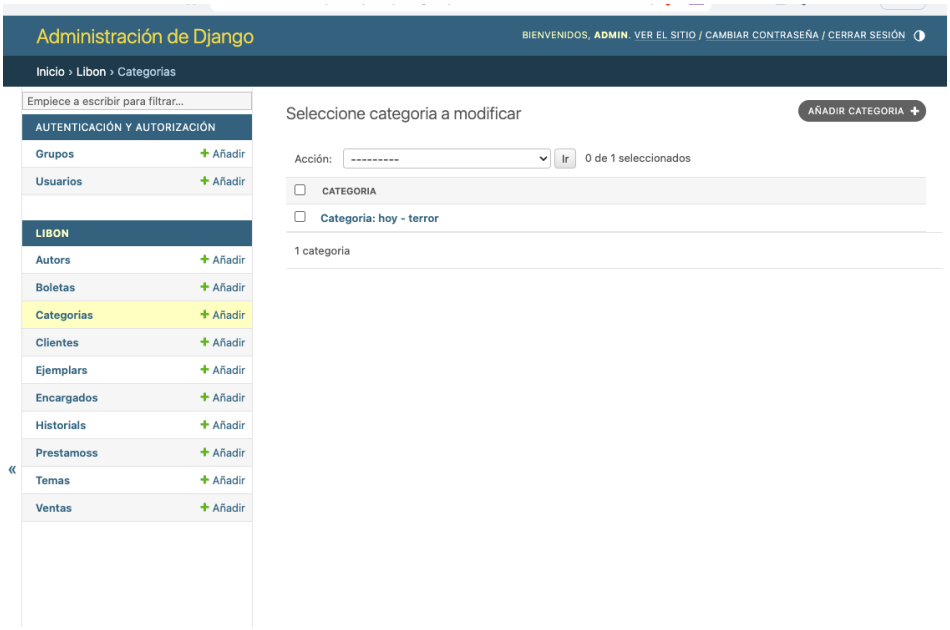
Para ver como se crea una categoria desde el administrador de django se hace lo siguiente:



Como se puede observar en el adminitrador nos deja agregar todos los atributos de la clase categoria exeptuando aquellos que necesitan las claves foraneas de otras clases que en este caso nos pedira escoger de una lista de objetos ya creados.

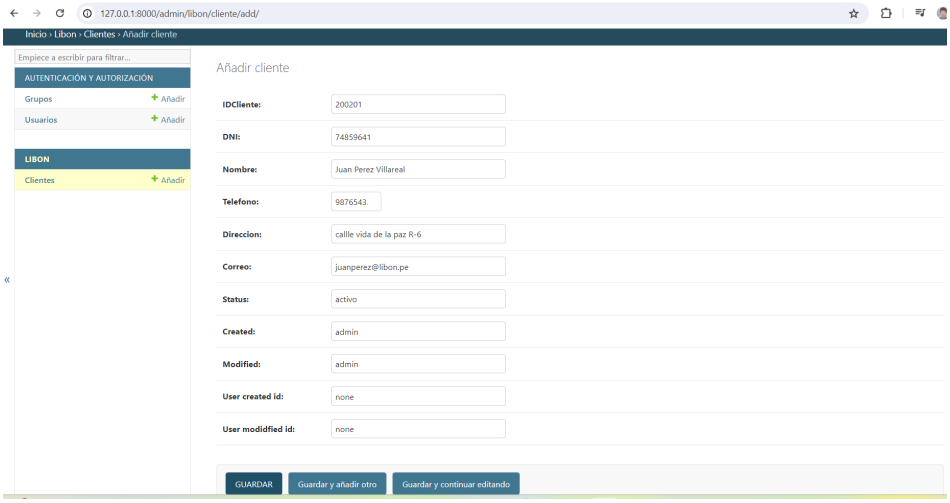


Para rellenar esto de ejemplo hemos tomado nombres al hazar y creado objetos de otras clses necesarias y el resultado es el que se ve en las imagenes:

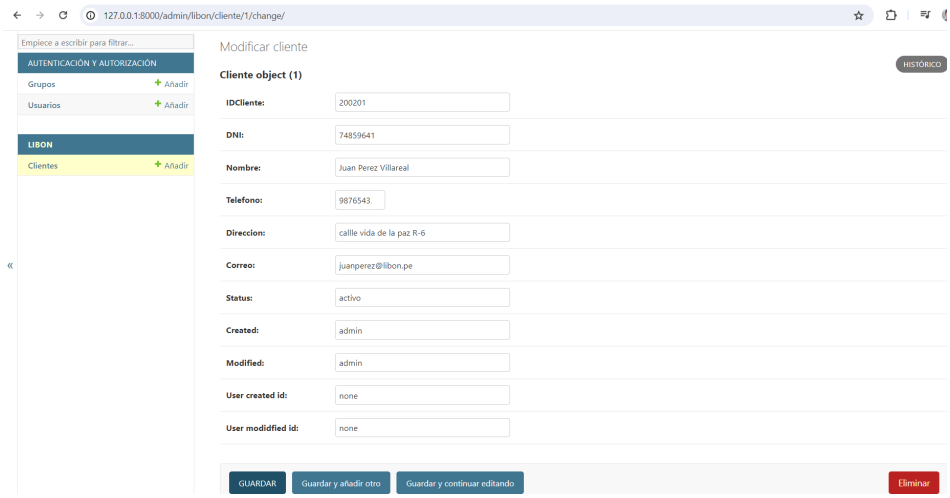


5.4. Cliente

Ahora crearemos un cliente desde el administrados de django

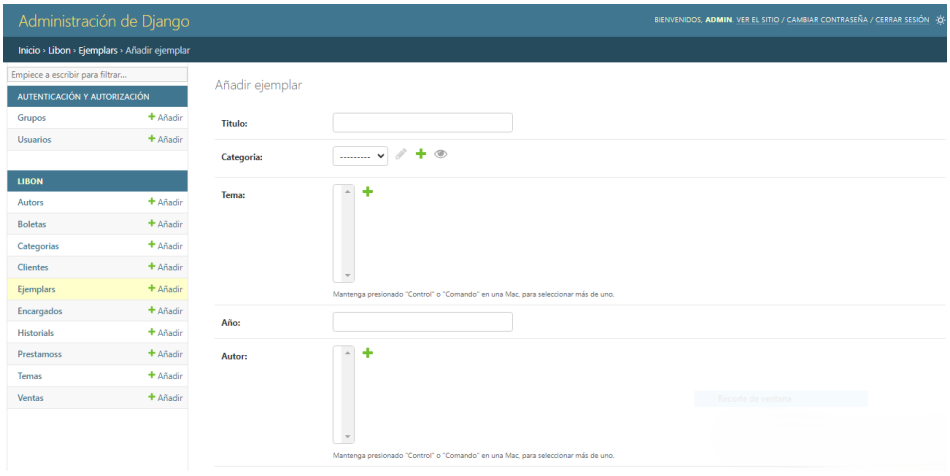


Cuando ingresamos en el objeto creado no aparece la informacion como tambien modificar y botones de opciones como guardado, guardado y añadir otro, guardar y continuar editando, y eliminar.



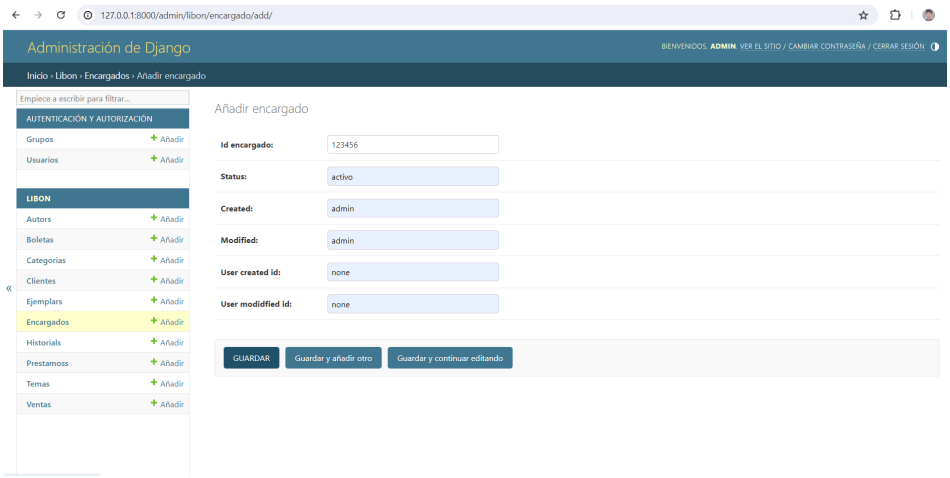
5.5. Ejemplares

Ahora crearemos un ejemplar desde el administrados de django

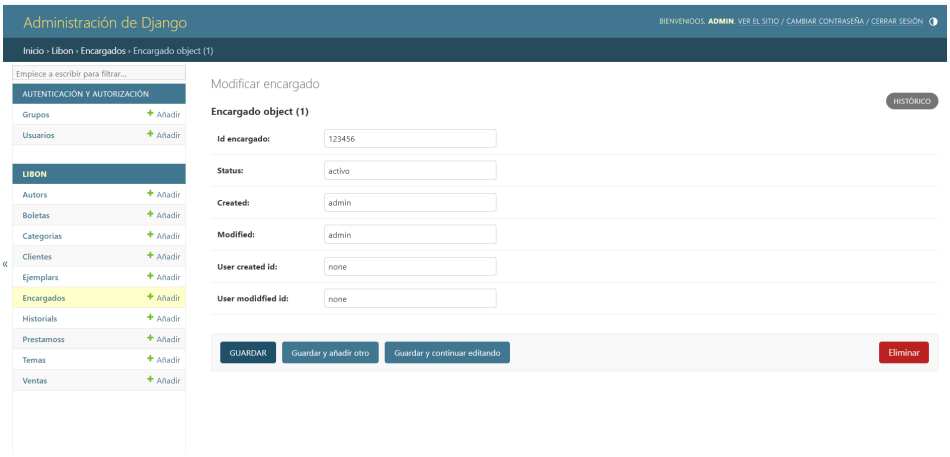


5.6. Encargado

Ahora crearemos un encargado desde el administrados de django

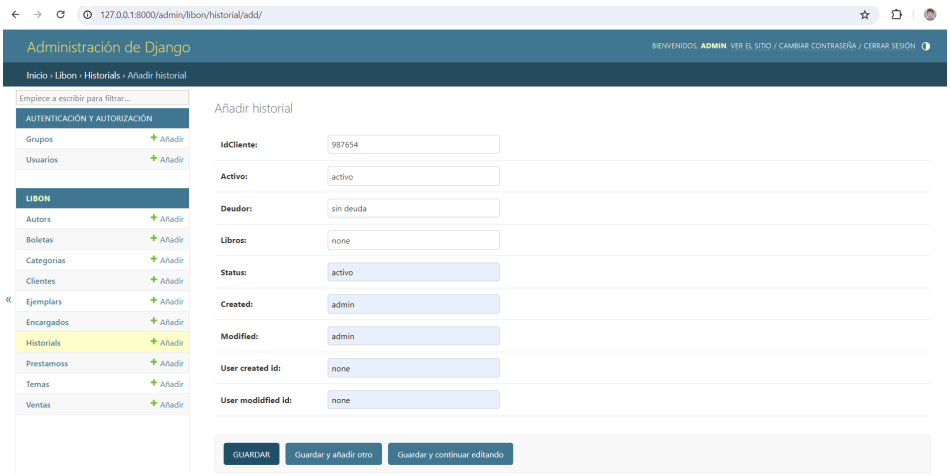


Cuando ingresamos en el objeto creado no aparece la informacion como tambien modificar y botones de opciones como guardado, guardado y añadir otro, guardar y continuar editando, y eliminar.

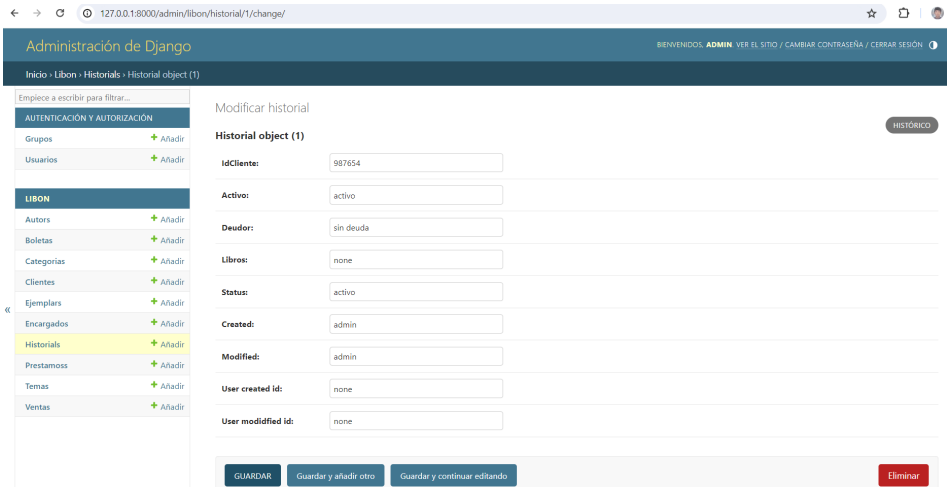


5.7. Historial

Ahora crearemos un historial desde el administrados de django

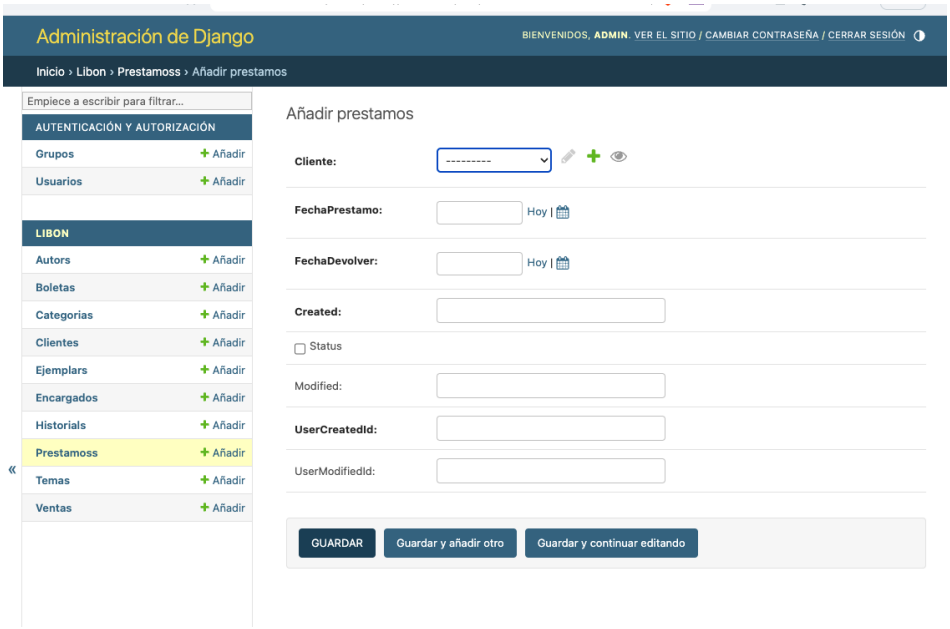


Cuando ingresamos en el objeto creado no aparece la informacion como tambien modificar y botones de opciones como guardado, guardado y añadir otro, guardar y continuar editando, y eliminar.

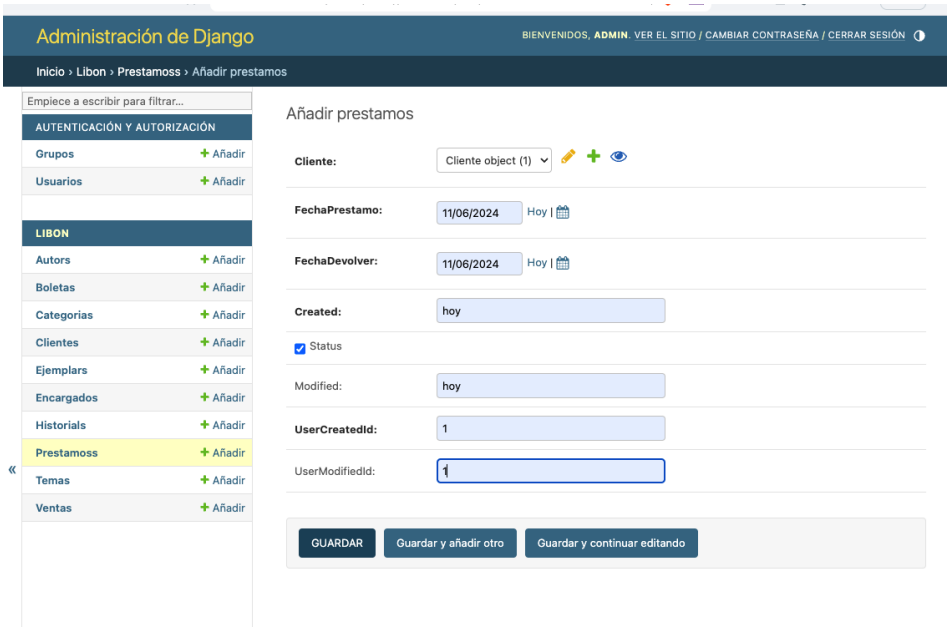


5.8. Prestamos

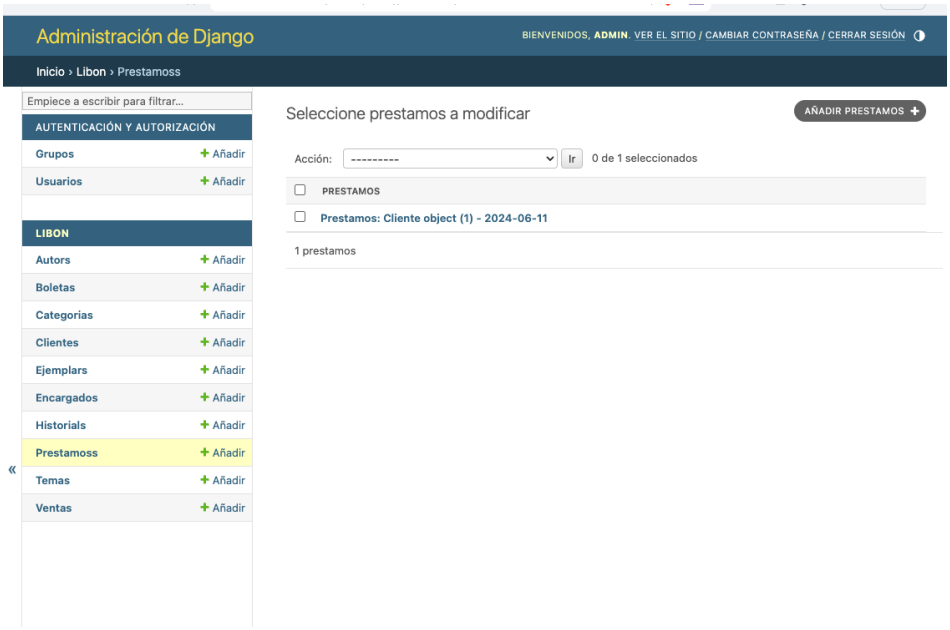
Para la trabla prestamo es similar a las anteriores teniendo que añadir los diferentes datos que nos piden en este caso obligatios para la corrrecta insercion en la tabla:



Como se puede ver en la tabla que nos pide llenar estan los datos que acordamos en el codigo al momento de la creacion de la tabla con django, la clave primaria no es necesaria ingresarla ya qu la pusimos autofield pero las claves foraneas nos siguen siendo necesarias para la correcta union de las tablas.

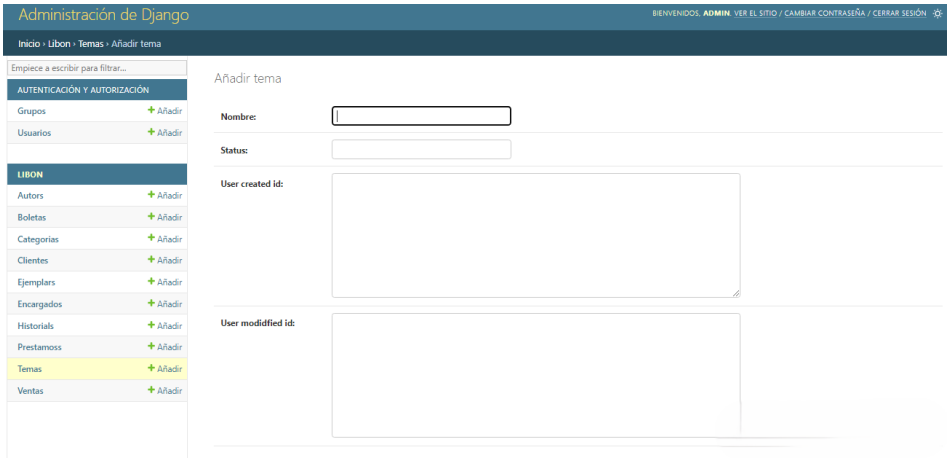


Una vez ya llenados todos las cajas nos deberia de quedar el registro de la siguiente manera:



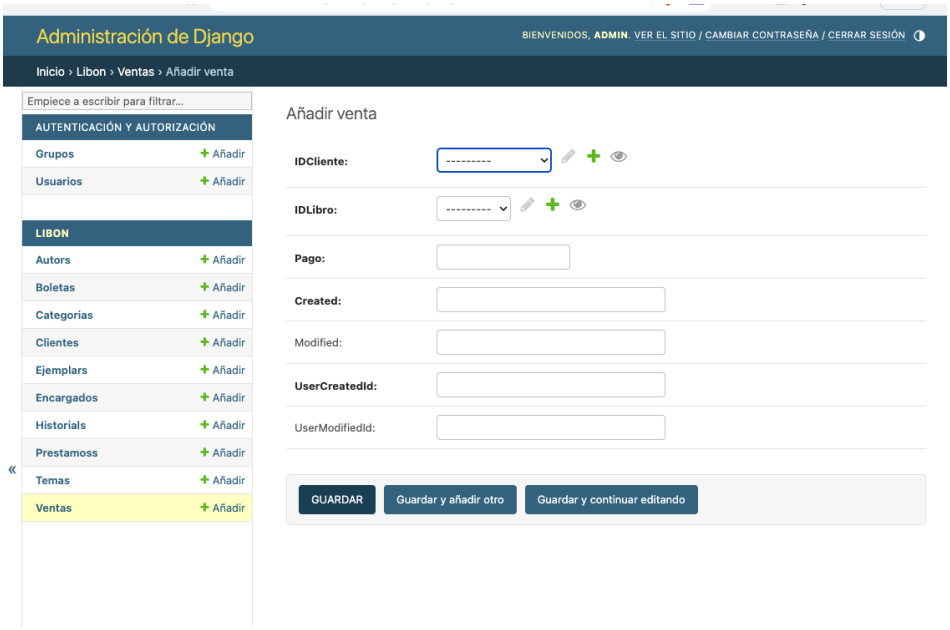
5.9. Tema

Ahora crearemos un tema desde el administrados de django

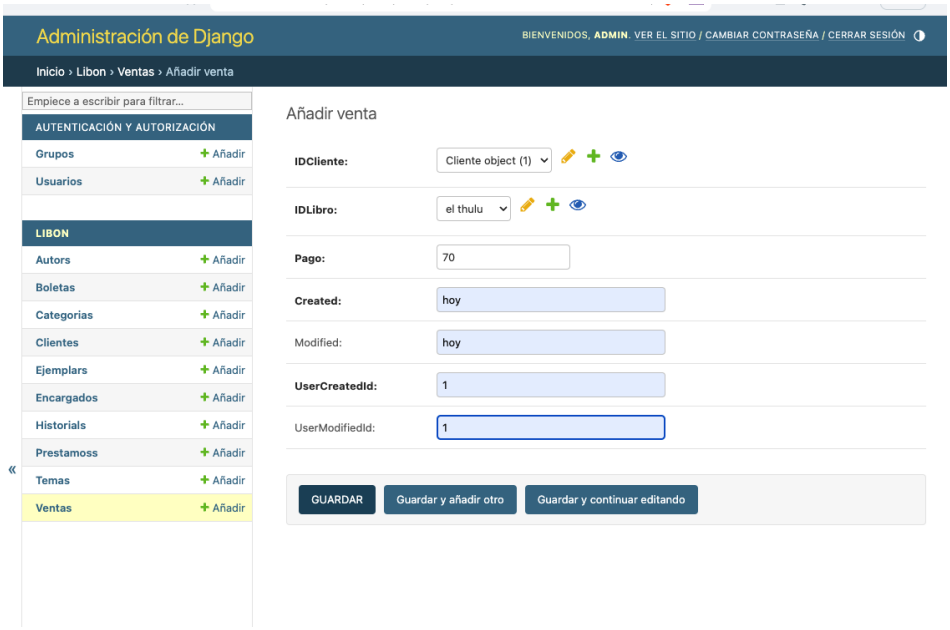


5.10. Venta

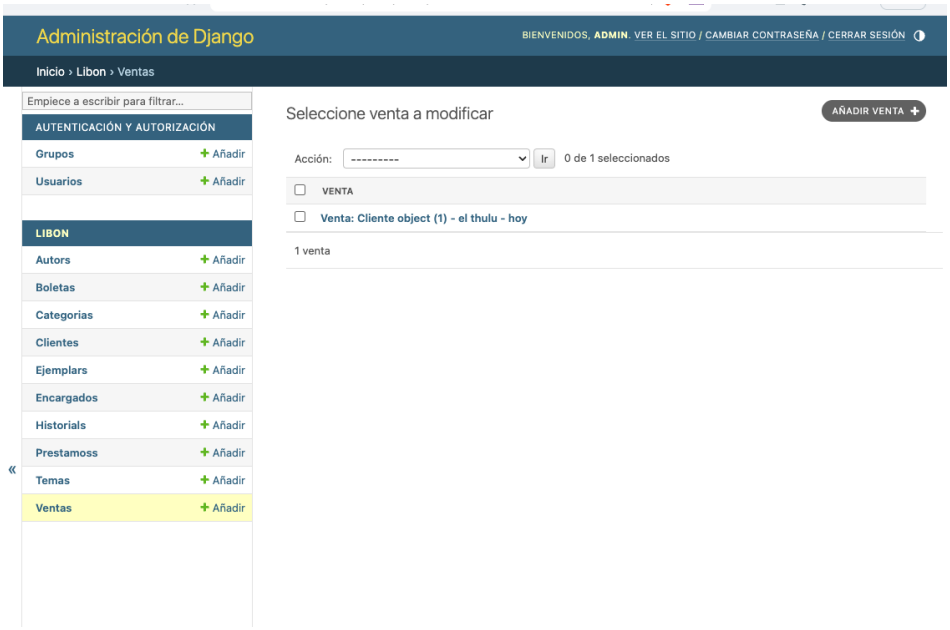
Para la tabla Venta el funcionamiento nos pide dos claves foraneas que son las del cliente y el ejemplar que se quiere vender, para eso tenemos lo siguiente:



Para el correcto registro del dato en la tabla se inserta los datos que nos piden, y las claves foraneas se escogen de los objetos que ya hemos registrado previamnete(Cliente y Ejemplar):



Al final el registro exitoso nos quedaria asi y mostrado por la clase str del codgio definido al principio:



6. Pregunta: Por cada integrante del equipo, resalte un aprendizaje que adquirió al momento de estudiar esta primera parte de Django (Admin). No se reprima de ser detallista. Coloque su nombre entre parentesis para saber que es su aporte.

Es de mucha ayuda el framework django al momento de crear nuestra aplicacion web pues para empezar al momento de crear el proyecto libreria nos creo una carpeta con los archivos necesarios y luego cuando creamos nuestra app libon podemos observar que no crea incovenientes pues se vuelve mas intuitivo ya que crea sus propios archivos, me gusta la parte con la que solo tenemos que registrar nuestra app en settings sin tener que estar mandando url o direccion de nuestro archivo, pues me parece mas facil estar registrando mediante nombres e importaciones ya sea una app o una clase models.Aunque tambien debemos de tener cuidado de subir archivos binarios como el pycache o la base de datos pues al momento de hacer git pull nos muestra que hay conflictos por lo que primeramente se deberia de ver como trabajar el archivo gitignore para evitar conflictos al hacer git pull en nuestro proyecto.(Suasaca Pacompia Alvaro Gustavo)

El framework de django es una manera muy rapida y sencilla de crear paginas web algo mas complicadoas que requieran un manejo exaustivo de base de datos, al momento de crear las libreiras necesarias para el proyecto se puede sentir cierto nivel de abruma pero una vez que se entiendo que es lo que se esta haciendo la creacion, modificacion y insercion en lo que es el DB se vuelve intuitivo, con las creaciones de las tablas y el uso de las claves primarias y foraneas se puede con entrelazar los datos con mas facilidad y gracia a que django cuneta con un administrador grafico lo hace mas sencillo de entender(Anco Aymara Jean Pierre).

En Django, las relaciones entre los modelos son fundamentales para organizar los datos en una aplicación web de manera eficiente. Estas relaciones nos permiten establecer conexiones lógicas entres los diferentes tipos de datos, lo que nos facilita el acceso y la manipulación de la información en la base de datos. Una de las relaciones más comunes en Django es la relación de clave externa o ForeignKey, esta establece una relación donde un modelo puede pertenecer a otro. Otra relación importante es la ManyToManyField, que permite establecer una relación de muchos a muchos entre dos modelos. Esto significa que un modelo puede estar asociado a varios objetos de otro modelo y viceversa. (Arocutipa Gutierrez Luis Edgar)

El framework de django me parece intuitivo pues ya tiene un orden predeterminado como cuando creamos el proyecto o cuando iniciamos una nueva app y tiene esa facilidad de mostrar el administrador de django ya ordenado y configurado con solo registrar en el admin de la app y me parece tambien que podremos crear muchas apps y todas ellas estaran en orden(Mendoza Contreras Giovani Angel).

7. Estructura de directorios y archivos

```
pw2-24a
|---proyecto
    |--- latex
    |    |--- laboratorio06_pw2.tex
    |    |--- laboratorio06_pw2.pdf
    |--- libon/
    |    |--- __pycache__
    |    |--- migrations
    |    |--- models
    |    |     |--- autor.py
    |    |     |--- boleta.py
```

```
|    |     |--- categoria.py
|    |     |--- cliente.py
|    |     |--- ejemplar.py
|    |     |--- encargado.py
|    |     |--- historial.py
|    |     |--- prestamos.py
|    |     |--- tema.py
|    |     |--- venta.py
|    |--- __init__.py
|    |--- admin.py
|    |--- apps.py
|    |--- models.py.deprecated
|    |--- tests.py
|    |--- views.py
|--- libreria
|    |--- __pycache__
|    |--- __init.py
|    |--- asgi.py
|    |--- settings.py
|    |--- urls.py
|    |--- wsgi.py
|--- .gitignore
|--- db.sqlite3
|--- manage.py
|--- README.md
|--- requirements.txt
```

8. Rúbricas

8.1. 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.

8.2. 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	2	
3. Codigo 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 proffesor puede preguntar para refrentar calificacion).	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. Ortografia	El documento no muestra errores ortograficos.	2	X	1	
8. Madurez	El informe muestra de manera general una evolucion de la madurez del codigo fuente con explicaciones puntuales pero presisas, y un acabado impecable.(El profesor puede preguntar para refrendar calificacion).	4	X	2	
Total		20		15	

9. Referencias

- https://www.w3schools.com/python/python_lists.asp
- https://stackoverflow.com/questions/16869024/what-is-pycache