UNIVERSIDAD NACIONAL DE COLOMBIA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS E INDUSTRIAL

Entrega 04

Fecha: 24/10/2025

Joan Camilo Betancourt Gonzalez (jobetancourtg@unal.edu.co)
Nicolás Alejandro Diosa Benavides (ndiosab@unal.edu.co)
Jonathan Felipe López Nuñez (jolopezn@unal.edu.co)
Raúl Felipe Rodríguez Hernández (rrodriguezhe@unal.edu.co)

---

3. Patrones de diseño:

a. Singleton:

Python

```
from core.models import *
from django.db.models import Model

class DB_Manager:
    instancia = None
    def __new__(cls):
        if cls.instancia is None:
            cls.instancia = super().__new__(cls) #Crea el objeto volviendo a la superclase object y a su método new que es el que instancia la clase
        return cls.instancia

    '''INICIO CREATE'''

    def create_usuario(cls, nombre_usuario, email, contrasena, nombre, bio, foto_perfil):
```

```
        Usuario.objects.create( nombre_usuario=nombre_usuario,
email=email, contrasena=contrasena, nombre=nombre, bio=bio,
foto_perfil=foto_perfil)

    def create_actividad(cls, id_creador, nombre_actividad,
descripcion, categoria, ubicacion, lat, lng, fecha_hora_inicio,
fecha_hora_fin, cupos, foto_actividad, estado,
fecha_hora_creacion, fecha_hora_actualizacion):
        Actividad.objects.create( id_creador=id_creador,
nombre_actividad=nombre_actividad, descripcion=descripcion,
categoria=categoria, ubicacion=ubicacion, lat=lat, lng=lng,
fecha_hora_inicio=fecha_hora_inicio,
fecha_hora_fin=fecha_hora_fin, cupos=cupos,
foto_actividad=foto_actividad, estado=estado,
fecha_hora_creacion=fecha_hora_creacion,
fecha_hora_actualizacion=fecha_hora_actualizacion)

    def create_part_actividad(cls, id_actividad, id_usuario,
hora_llegada, hora_salida, estado_participante):
        ParticipanteActividad.objects.create(id_actividad =
id_actividad , id_usuario = id_usuario,
hora_llegada=hora_llegada, hora_salida=hora_salida,
estado_participante=estado_participante)

    def create_materia(cls, id_usuario, nombre_materia, semestre,
horario_materia, prioridad, estado_materia):
        Materia.objects.create( id_usuario=id_usuario,
nombre_materia=nombre_materia, semestre=semestre,
horario_materia=horario_materia, prioridad=prioridad,
estado_materia=estado_materia)

    def create_evento_calendario(cls, id_usuario, id_materia,
nombre_evento, fecha_hora_inicio, fecha_hora_fin, prioridad):
        EventoCalendario.objects.create( id_usuario=id_usuario,
id_materia=id_materia, nombre_evento=nombre_evento,
```

```
fecha_hora_inicio=fecha_hora_inicio,
fecha_hora_fin=fecha_hora_fin, prioridad=prioridad)

    def create_tarea(cls, id_usuario, id_materia, nombre_tarea,
descripcion_tarea, prioridad, fecha_vencimiento, es_recurrente,
recurrencia, estado_tarea, creacion_tarea, completada_en):
        Tarea.objects.create( id_usuario=id_usuario,
id_materia=id_materia, nombre_tarea=nombre_tarea,
descripcion_tarea=descripcion_tarea, prioridad=prioridad,
fecha_vencimiento=fecha_vencimiento, es_recurrente=es_recurrente,
recurrencia=recurrencia, estado_tarea=estado_tarea,
creacion_tarea=creacion_tarea, completada_en=completada_en)

    def create_chat(cls, id_actividad, id_emisor, contenido,
hora_creacion):
        Chat.objects.create( id_actividad=id_actividad,
id_emisor=id_emisor, contenido=contenido,
hora_creacion=hora_creacion)

    '''INICIO READ'''

    def read_all(cls, tabla: Model):
        return tabla.objects.all()

    # -------------------- USUARIOS -------------------

    def get_usuario_by_nombre_usuario(cls, nombre_usuario):
        return Usuario.objects.get(nombre_usuario=nombre_usuario)



    # -------------------- ACTIVIDADES --------------------

    def get_actividad_by_nombre_actividad(cls, nombre_actividad):
        return
Actividad.objects.get(nombre_actividad=nombre_actividad)
```

```
    # -------------------- PARTICIPANTES ACTIVIDAD
--------------------

    def get_participante_actividad(cls, id_actividad,
id_usuario):
        return
ParticipanteActividad.objects.get(id_actividad=id_actividad,
id_usuario=id_usuario)


    # -------------------- MATERIAS --------------------

    def get_materia_by_nombre_materia(cls, nombre_materia):
        return Materia.objects.get(nombre_materia=nombre_materia)


    # -------------------- EVENTOS CALENDARIO
--------------------

    def get_evento_calendario_by_nombre_evento(cls,
nombre_evento):
        return
EventoCalendario.objects.get(nombre_evento=nombre_evento)


    # -------------------- TAREAS --------------------

    def get_tarea_by_nombre_tarea(cls, nombre_tarea):
        return Tarea.objects.get(nombre_tarea=nombre_tarea)


    # -------------------- CHATS --------------------
    def get_chat_by_id_actividad(cls, id_actividad):
```

```
        return Chat.objects.get(id_actividad=id_actividad)

    '''INICIO UPDATE'''

    def update(cls, tabla: Model, campo, valor, **kwargs):
        tabla.objects.filter(**{campo: valor}).update(**kwargs)

    '''INICIO DELETE'''

    def delete(cls, tabla: Model, campo, valor):
        tabla.objects.filter(**{campo: valor}).delete()
```

Para garantizar un único punto de acceso a las operaciones de base de datos dentro del proyecto, se implementó el patrón de diseño Singleton a través de la clase DB_Manager.
Este patrón asegura que solo exista una única instancia de esta clase durante la ejecución de la aplicación, evitando la creación redundante de objetos y asegurando la consistencia en el manejo de los datos.

La clase DB_Manager tiene las operaciones CRUD sobre los modelos que se definieron en el core de la aplicación.

b.Observer:

En el proyecto se propone la utilización del patrón de diseño Observer para el manejo de las notificaciones de cambios en las actividades.

El modelo Actividad actúa como sujeto observable y mantiene una lista de observadores que serían los usuarios interesados o participantes.

Cuando la actividad cambia su estado, por ejemplo, modificación de fecha, lugar o cupos, el sistema notifica automáticamente a todos los observadores registrados.

Este enfoque favorece la propagación eficiente de los eventos y mejora la experiencia del usuario al mantenerlo informado en tiempo real sin necesidad de consultas manuales.