

TEMA 6

ACTIVIDAD 1

```
actividad1.py > ...
104
105 # Configuración de logging para guardar en un archivo
106 logging.basicConfig(
107     level=logging.INFO,
108     format='%(asctime)s - %(levelname)s - %(message)s',
109     handlers=[
110         logging.FileHandler("log_datos.log"), # Guardar logs en un archivo
111     ]
112 )
113
114 #Datos
115 obras=[
116     {"titulo": "La noche estrellada", "artista": "Van Gogh", "fechaCreacion": 1889, "tecnica": "Oleo", "museo": "MoMA"},
117     {"titulo": "La Gioconda", "artista": "Leonardo da Vinci", "fechaCreacion": 1503, "tecnica": "sfumato", "museo": "Louvre"},
118     {"titulo": "Las meninas", "artista": "Velazquez", "fechaCreacion": 1656, "tecnica": "alla prima", "museo": "El Prado"}
119 ]
120
121 #Crea archivo json
122 gestor = DataManager("obras.json", tipo_archivo="json")
123
124 #Inicia transaccion, escribe los datos y confirma
125 gestor.iniciar_transaccion()
126 for obra in obras:
127     gestor.escribir_dato(obra)
128 gestor.confirmar_transaccion()
129
130 #Cambiar configuracion para csv
131 gestor.actualizar_configuracion("obras.csv", nuevo_tipo="csv")
132
133 #Inicia transaccion y confirma
134 gestor.iniciar_transaccion()
135 gestor.confirmar_transaccion()
```

```
● act1usuario@usuario:~/ACD/Tema6/actividad1$ /home/usuario/ACD/Tema6/actividad1/act1/bin/python /home/usuario/ACD/Tema6/actividad1/act
ividad1.py
○ act1usuario@usuario:~/ACD/Tema6/actividad1$
```

```
actividad1.py  log_datos.log X
log_datos.log
1  2024-11-21 09:32:20,769 - INFO - Transacción iniciada.
2  2024-11-21 09:32:20,769 - INFO - Dato agregado: {'titulo': 'La noche estrellada', 'artista': 'Van Gogh', 'fechaCreacion': 1889, 'tec
3  2024-11-21 09:32:20,769 - INFO - Dato agregado: {'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': 1503, 'tec
4  2024-11-21 09:32:20,769 - INFO - Dato agregado: {'titulo': 'Las meninas', 'artista': 'Velazquez', 'fechaCreacion': 1656, 'tecnica': '
5  2024-11-21 09:32:20,769 - INFO - Transacción confirmada y cambios guardados.
6  2024-11-21 09:32:20,769 - INFO - Configuración actualizada. Nueva ruta del archivo: obras.csv
7  2024-11-21 09:32:20,769 - INFO - Transacción iniciada.
8  2024-11-21 09:32:20,769 - INFO - Archivo CSV guardado. Versión actual: 3
9  2024-11-21 09:32:20,770 - INFO - Transacción confirmada y cambios guardados.
10
```

ACTIVIDAD 2

```
actividad2.py x
actividad2.py > DatabaseManager > eliminar_obraArte
1  import logging
2  import mysql.connector
3  from mysql.connector import Error
4
5  # Configuración de logging
6  logging.basicConfig(
7      level=logging.INFO,
8      format="%(asctime)s - %(levelname)s - %(message)s",
9      handlers=[
10         logging.FileHandler("databasemanager.log"), # Logs guardados en un archivo
11         logging.StreamHandler(), # Logs también en consola
12     ]
13 )
14
15 class DatabaseManager:
16     def __init__(self, host, user, password, database):
17         self.host = host
18         self.user = user
19         self.password = password
20         self.database = database
21         self.connection = None
22
23     def conectar(self):
24         """Conectar a la base de datos MySQL"""
25         try:
26             self.connection = mysql.connector.connect(
27                 host=self.host,
28                 user=self.user,
29                 password=self.password,
30                 database=self.database
31             )
32             if self.connection.is_connected():
33                 logging.info("Conexión exitosa a la base de datos.")
34         except Error as e:
35             logging.error(f"Error al conectar a la base de datos: {e}")
36
```

```
actividad2.py x
actividad2.py > DatabaseManager > eliminar_obraArte
15 class DatabaseManager:
36
37     def desconectar(self):
38         """Cerrar la conexión a la base de datos"""
39         if self.connection.is_connected():
40             self.connection.close()
41             logging.info("Conexión cerrada.")
42
43     def crear_obraArte(self, titulo, artista, fechaCreacion, tecnica, museo):
44         """Insertar una nueva obra de arte en la base de datos"""
45         try:
46             cursor = self.connection.cursor()
47             query = """
48                 INSERT INTO ObrasArte (titulo, artista, fechaCreacion, tecnica, museo)
49                 VALUES (%s, %s, %s, %s, %s)
50             """
51             cursor.execute(query, (titulo, artista, fechaCreacion, tecnica, museo))
52             #self.connection.commit()
53             logging.info(f"Obra de arte '{titulo}' insertada exitosamente.")
54         except Error as e:
55             logging.error(f"Error al insertar la obra de arte '{titulo}': {e}")
56
57     def leer_obrasArte(self):
58         """Leer todas las obras de arte de la base de datos"""
59         try:
60             cursor = self.connection.cursor(dictionary=True)
61             cursor.execute("SELECT * FROM ObrasArte")
62             obras = cursor.fetchall()
63             logging.info("Obras de arte recuperadas:")
64             for obra in obras:
65                 logging.info(obra)
66             return obras
67         except Error as e:
68             logging.error(f"Error al leer las obras de arte: {e}")
69         return None
70
```

actividad2.py X

actividad2.py > DatabaseManager > eliminar_obraArte

```
15 class DatabaseManager:
71     def actualizar_obraArte(self, titulo, artista, fechaCreacion, tecnica, museo):
72         """Actualizar una obra de arte en la base de datos"""
73         try:
74             cursor = self.connection.cursor()
75             query = """
76                 UPDATE ObrasArte
77                 SET artista = %s, fechaCreacion = %s, tecnica = %s, museo = %s
78                 WHERE titulo = %s
79             """
80             cursor.execute(query, (artista, fechaCreacion, tecnica, museo, titulo))
81             self.connection.commit()
82             logging.info(f"Obra de arte con titulo{titulo} actualizada exitosamente.")
83         except Error as e:
84             logging.error(f"Error al actualizar la obra de arte con titulo {titulo}: {e}")
85
86     def eliminar_obraArte(self, titulo):
87         """Eliminar una obra de arte de la base de datos"""
88         try:
89             cursor = self.connection.cursor()
90             query = "DELETE FROM ObrasArte WHERE titulo = %s"
91             cursor.execute(query, (titulo,))
92             self.connection.commit()
93             logging.info(f"Obra de arte con titulo {titulo} eliminada exitosamente.")
94         except Error as e:
95             logging.error(f"Error al eliminar la obra de arte con titulo {titulo}: {e}")
96
97     def iniciar_transaccion(self):
98         """Iniciar una transacción"""
99         try:
100             if self.connection.is_connected():
101                 self.connection.start_transaction()
102                 logging.info("Transacción iniciada.")
103         except Error as e:
104             logging.error(f"Error al iniciar la transacción: {e}")
```

actividad2.py X

actividad2.py > ...

```
15 class DatabaseManager:
106     def confirmar_transaccion(self):
107         try:
108             if self.connection.is_connected():
109                 self.connection.commit()
110                 logging.info("Transacción confirmada.")
111         except Error as e:
112             logging.error(f"Error al confirmar la transacción: {e}")
113
114     def revertir_transaccion(self):
115         """Revertir (rollback) una transacción"""
116         try:
117             if self.connection.is_connected():
118                 self.connection.rollback()
119                 logging.info("Transacción revertida.")
120         except Error as e:
121             logging.error(f"Error al revertir la transacción: {e}")
122
123
124
125 if __name__ == "__main__":
126     db_manager = DatabaseManager("localhost", "usuario", "usuario", "ldam")
127     db_manager.conectar()
128
129     # Insertar una nueva obra de arte
130     db_manager.crear_obraArte("La Gioconda", "Leonardo da Vinci", 1503, "sfumato", "Louvre")
131
132     # Leer todas las obras de arte
133     db_manager.leer_obrasArte()
134
135     # Actualizar una obra de arte
136     db_manager.actualizar_obraArte("La Gioconda", "Leonardo da Vinci", 1550, "sfumato", "Louvre")
137
138     # Eliminar una obra de arte
139     db_manager.eliminar_obraArte("El Guernica")
140
141     # Gestionar transacciones
142     db_manager.iniciar_transaccion()
143     db_manager.crear_obraArte("La noche estrellada", "Van Gogh", 1889, "Oleo", "MoMA")
144     db_manager.revertir_transaccion() # No se guardará la inserción
145     db_manager.desconectar()
```



```

● act2usuario@usuario:~/ACD/Tema6/actividad2$ /home/usuario/ACD/Tema6/actividad2/act2/bin/python /home/usuario/ACD/Tema6/actividad2/actividad2.py
2024-11-25 14:00:48,557 - INFO - Conexión exitosa a la base de datos.
2024-11-25 14:00:48,558 - INFO - Obra de arte 'La Gioconda' insertada exitosamente.
2024-11-25 14:00:48,558 - INFO - Obras de arte recuperadas:
2024-11-25 14:00:48,558 - INFO - {'id': 6, 'titulo': 'La Capilla Sixtina', 'artista': 'Miguel Ángel', 'fechaCreacion': '1508', 'tecnica': 'Fresco común', 'museo': 'El Vaticano'}
2024-11-25 14:00:48,558 - INFO - {'id': 7, 'titulo': 'El Guernica', 'artista': 'Pablo Picasso', 'fechaCreacion': '1937', 'tecnica': 'Óleo', 'museo': 'Reina Sofía'}
2024-11-25 14:00:48,558 - INFO - {'id': 8, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1550', 'tecnica': 'sfumato', 'museo': 'Louvre'}
2024-11-25 14:00:48,558 - INFO - {'id': 10, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1550', 'tecnica': 'sfumato', 'museo': 'Louvre'}
2024-11-25 14:00:48,558 - INFO - {'id': 12, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1503', 'tecnica': 'sfumato', 'museo': 'Louvre'}
2024-11-25 14:00:48,566 - INFO - Obra de arte con tituloLa Gioconda actualizada exitosamente.
2024-11-25 14:00:48,575 - INFO - Obra de arte con titulo El Guernica eliminada exitosamente.
2024-11-25 14:00:48,576 - INFO - Transacción iniciada.
2024-11-25 14:00:48,576 - INFO - Obra de arte 'La noche estrellada' insertada exitosamente.
2024-11-25 14:00:48,582 - INFO - Transacción revertida.
2024-11-25 14:00:48,583 - INFO - Conexión cerrada.
○ act2usuario@usuario:~/ACD/Tema6/actividad2$

```

```

actividad2.py  databasemanager.log x
databasemanager.log
1 2024-11-25 14:00:48,557 - INFO - Conexión exitosa a la base de datos.
2 2024-11-25 14:00:48,558 - INFO - Obra de arte 'La Gioconda' insertada exitosamente.
3 2024-11-25 14:00:48,558 - INFO - Obras de arte recuperadas:
4 2024-11-25 14:00:48,558 - INFO - {'id': 6, 'titulo': 'La Capilla Sixtina', 'artista': 'Miguel Ángel', 'fechaCreacion': '1508', 'tecnica': 'Fresco común', 'museo': 'El Vaticano'}
5 2024-11-25 14:00:48,558 - INFO - {'id': 7, 'titulo': 'El Guernica', 'artista': 'Pablo Picasso', 'fechaCreacion': '1937', 'tecnica': 'Óleo', 'museo': 'Reina Sofía'}
6 2024-11-25 14:00:48,558 - INFO - {'id': 8, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1550', 'tecnica': 'sfumato', 'museo': 'Louvre'}
7 2024-11-25 14:00:48,558 - INFO - {'id': 10, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1550', 'tecnica': 'sfumato', 'museo': 'Louvre'}
8 2024-11-25 14:00:48,558 - INFO - {'id': 12, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1503', 'tecnica': 'sfumato', 'museo': 'Louvre'}
9 2024-11-25 14:00:48,566 - INFO - Obra de arte con tituloLa Gioconda actualizada exitosamente.
10 2024-11-25 14:00:48,575 - INFO - Obra de arte con titulo El Guernica eliminada exitosamente.
11 2024-11-25 14:00:48,576 - INFO - Transacción iniciada.
12 2024-11-25 14:00:48,576 - INFO - Obra de arte 'La noche estrellada' insertada exitosamente.
13 2024-11-25 14:00:48,582 - INFO - Transacción revertida.
14 2024-11-25 14:00:48,583 - INFO - Conexión cerrada.

```

ACTIVIDAD 3

```

● actividad3.py > ...
1 import logging
2 from peewee import Model, CharField, ForeignKeyField, MySQLDatabase
3
4 # Componente DatabaseManagerORM
5 class DatabaseManagerORM:
6     def __init__(self):
7         self.db = db
8
9     def conectar(self):
10         """Conecta la base de datos y crea las tablas."""
11         self.db.connect()
12         self.db.create_tables([Proveedor, Herramienta])
13         logging.info("Conexión establecida y tablas creadas.")
14
15     def desconectar(self):
16         """Cierra la conexión a la base de datos."""
17         if not self.db.is_closed():
18             self.db.close()
19             logging.info("Conexión cerrada.")
20
21     def iniciar_transaccion(self):
22         """Inicia una transacción."""
23         self.db.begin()
24         logging.info("Transacción iniciada.")
25
26     def confirmar_transaccion(self):
27         """Confirma (commit) una transacción."""
28         self.db.commit()
29         logging.info("Transacción confirmada.")
30
31     def revertir_transaccion(self):
32         """Revierte (rollback) una transacción."""
33         self.db.rollback()
34         logging.info("Transacción revertida.")
35
36     def crear_proveedor(self, nombre, direccion):
37         """Inserta un nuevo proveedor."""
38         proveedor = Proveedor.create(nombre=nombre, direccion=direccion)
39         logging.info(f"Proveedor creado: {proveedor.nombre} - {proveedor.direccion}")
40         return proveedor

```

```

actividad3.py > ...
5 class DatabaseManagerORM:
42 def crear_herramienta(self, nombre, tipo, marca, uso, material, proveedor_nombre):
43     """Inserta una nueva herramienta."""
44     try:
45         proveedor = Proveedor.get(Proveedor.nombre == proveedor_nombre)
46         herramienta = Herramienta.create(
47             nombre=nombre,
48             tipo=tipo,
49             marca=marca,
50             uso=uso,
51             material=material,
52             proveedor=proveedor
53         )
54         logging.info(f"Herramienta creada: {herramienta.nombre} - {herramienta.tipo}")
55         return herramienta
56     except Proveedor.DoesNotExist:
57         logging.error(f"El proveedor '{proveedor_nombre}' no existe.")
58         raise
59
60 def leer_herramientas(self):
61     """Lee todas las herramientas."""
62     herramientas = Herramienta.select()
63     logging.info("Leyendo herramientas:")
64     for herramienta in herramientas:
65         logging.info(f"{herramienta.nombre} - {herramienta.tipo} ({herramienta.proveedor.nombre})")
66     return herramientas
67
68 def actualizar_proveedor(self, nombre, direccion):
69     """Actualizar un proveedor en la base de datos"""
70     proveedor = Proveedor.get(Proveedor.nombre == nombre)
71     proveedor.direccion = direccion
72     proveedor.save()
73     return proveedor
74
75 def actualizar_herramienta(self, nombre, tipo):
76     """Actualizar una herramienta en la base de datos"""
77     herramienta = Herramienta.get(Herramienta.nombre == nombre)
78     herramienta.tipo = tipo
79     herramienta.save()
80     return herramienta

```

```

actividad3.py > ...
5 class DatabaseManagerORM:
82 def eliminar_herramienta(self, nombre):
83     """Eliminar una herramienta de la base de datos"""
84     herramienta = Herramienta.get(Herramienta.nombre == nombre)
85     herramienta.delete_instance()
86     return herramienta
87
88 def eliminar_proveedor(self, nombre):
89     """Eliminar un proveedor de la base de datos"""
90     proveedor = Proveedor.get(Proveedor.nombre == nombre)
91     proveedor.delete_instance()
92     return proveedor
93
94 def consultar_herramienta(self, proveedor_nombre):
95     """Imprimir herramientas que pertenezcan a un proveedor"""
96     try:
97         proveedor = Proveedor.get(Proveedor.nombre == proveedor_nombre)
98         herramientas = Herramienta.select().where(Herramienta.proveedor == proveedor)
99         for herramienta in herramientas:
100             logging.info(f"{herramienta.nombre} - {herramienta.tipo}")
101     except Proveedor.DoesNotExist:
102         logging.error(f"No existe el proveedor '{proveedor_nombre}'.")
103
104 # Configuración de logging
105 logging.basicConfig(
106     level=logging.INFO,
107     format="%(asctime)s - %(levelname)s - %(message)s",
108     handlers=[
109         logging.FileHandler("databasemanager_orm.log"),
110         logging.StreamHandler()
111     ]
112 )
113
114 # Configuración de la base de datos MySQL
115 db = MySQLDatabase(
116     "ldam", # Nombre de la base de datos
117     user="usuario", # Usuario de MySQL
118     password="usuario", # Contraseña de MySQL
119     host="localhost", # Host
120     port=3306 # Puerto por defecto de MySQL

```

```

actividad3.py > ...
123 # Modelos de la base de datos
124 class Proveedor(Model):
125     nombre = CharField()
126     direccion = CharField()
127     class Meta:
128         database = db
129
130 class Herramienta(Model):
131     nombre = CharField()
132     tipo = CharField()
133     marca = CharField()
134     uso = CharField()
135     material = CharField()
136     proveedor = ForeignKeyField(Proveedor, backref='herramientas')
137     class Meta:
138         database = db
139
140 # Flujo principal
141 gestion = DatabaseManagerORM()
142 gestion.conectar()
143 # Gestión de Proveedores
144 print("Gestión de Proveedores:")
145 gestion.iniciar_transaccion()
146 gestion.crear_proveedor("Proveedor A", "Contacto 123-456-789")
147 gestion.crear_proveedor("Proveedor B", "Contacto 987-654-321")
148 gestion.confirmar_transaccion()
149
150 # Actualización de Proveedor
151 print("Cambio del dato por mi DNI 12345678A al proveedor A")
152 gestion.iniciar_transaccion()
153 gestion.actualizar_proveedor("Proveedor A", "24525402L")
154 gestion.confirmar_transaccion()
155
156 # Eliminar Proveedor
157 print("Eliminar al proveedor B")
158 gestion.iniciar_transaccion()
159 gestion.eliminar_proveedor("Proveedor B")
160 gestion.confirmar_transaccion()
161
162 # Gestión de Herramientas

```

```

actividad3.py > ...
160 gestion.confirmar_transaccion()
161
162 # Gestión de Herramientas
163 print("Gestión de Herramientas:")
164 gestion.iniciar_transaccion()
165 gestion.crear_herramienta("Martillo", "Manual", "Facom", "Percusion", "Acero", "Proveedor A")
166 gestion.crear_herramienta("Taladro", "Eléctrico", "Facom", "Percusion", "Acero", "Proveedor A")
167 gestion.confirmar_transaccion()
168
169 # Consultar herramientas
170 print("Herramientas asociadas al proveedor Proveedor A:")
171 gestion.consultar_herramienta("Proveedor A")
172
173 # Actualizar herramienta
174 print("Actualizar herramienta Martillo a tipo reforzado")
175 gestion.iniciar_transaccion()
176 gestion.actualizar_herramienta("Martillo", "Reforzado")
177 gestion.confirmar_transaccion()
178
179 # Eliminar herramienta
180 print("Eliminar herramienta Taladro")
181 gestion.iniciar_transaccion()
182 gestion.eliminar_herramienta("Taladro")
183 gestion.confirmar_transaccion()
184
185 # Desconectar
186 gestion.desconectar()
187

```



```

● act3usuario@usuario:~/ACD/Tema6/actividad3$ /home/usuario/ACD/Tema6/actividad3/act3/bin/python /home/usuario/ACD/
ividad3.py
2024-11-25 13:43:30,702 - INFO - Conexión establecida y tablas creadas.
Gestión de Proveedores:
2024-11-25 13:43:30,702 - INFO - Transacción iniciada.
2024-11-25 13:43:30,702 - INFO - Proveedor creado: Proveedor A - Contacto 123-456-789
2024-11-25 13:43:30,702 - INFO - Proveedor creado: Proveedor B - Contacto 987-654-321
2024-11-25 13:43:30,709 - INFO - Transacción confirmada.
Cambio del dato por mi DNI 12345678A al proveedor A
2024-11-25 13:43:30,709 - INFO - Transacción iniciada.
2024-11-25 13:43:30,710 - INFO - Transacción confirmada.
Eliminar al proveedor B
2024-11-25 13:43:30,710 - INFO - Transacción iniciada.
2024-11-25 13:43:30,720 - INFO - Transacción confirmada.
Gestión de Herramientas:
2024-11-25 13:43:30,720 - INFO - Transacción iniciada.
2024-11-25 13:43:30,721 - INFO - Herramienta creada: Martillo - Manual
2024-11-25 13:43:30,721 - INFO - Herramienta creada: Taladro - Eléctrico
2024-11-25 13:43:30,729 - INFO - Transacción confirmada.
Herramientas asociadas al proveedor Proveedor A:
2024-11-25 13:43:30,730 - INFO - Martillo - Manual
2024-11-25 13:43:30,730 - INFO - Taladro - Eléctrico
Actualizar herramienta Martillo a tipo reforzado
2024-11-25 13:43:30,730 - INFO - Transacción iniciada.
2024-11-25 13:43:30,739 - INFO - Transacción confirmada.
Eliminar herramienta Taladro
2024-11-25 13:43:30,739 - INFO - Transacción iniciada.
2024-11-25 13:43:30,746 - INFO - Transacción confirmada.
2024-11-25 13:43:30,746 - INFO - Conexión cerrada.

```

actividad3.py databasemanager_orm.log X

```

databasemanager_orm.log
1  2024-11-25 13:43:30,702 - INFO - Conexión establecida y tablas creadas.
2  2024-11-25 13:43:30,702 - INFO - Transacción iniciada.
3  2024-11-25 13:43:30,702 - INFO - Proveedor creado: Proveedor A - Contacto 123-456-789
4  2024-11-25 13:43:30,702 - INFO - Proveedor creado: Proveedor B - Contacto 987-654-321
5  2024-11-25 13:43:30,709 - INFO - Transacción confirmada.
6  2024-11-25 13:43:30,709 - INFO - Transacción iniciada.
7  2024-11-25 13:43:30,710 - INFO - Transacción confirmada.
8  2024-11-25 13:43:30,710 - INFO - Transacción iniciada.
9  2024-11-25 13:43:30,720 - INFO - Transacción confirmada.
10 2024-11-25 13:43:30,720 - INFO - Transacción iniciada.
11 2024-11-25 13:43:30,721 - INFO - Herramienta creada: Martillo - Manual
12 2024-11-25 13:43:30,721 - INFO - Herramienta creada: Taladro - Eléctrico
13 2024-11-25 13:43:30,729 - INFO - Transacción confirmada.
14 2024-11-25 13:43:30,730 - INFO - Martillo - Manual
15 2024-11-25 13:43:30,730 - INFO - Taladro - Eléctrico
16 2024-11-25 13:43:30,730 - INFO - Transacción iniciada.
17 2024-11-25 13:43:30,739 - INFO - Transacción confirmada.
18 2024-11-25 13:43:30,739 - INFO - Transacción iniciada.
19 2024-11-25 13:43:30,746 - INFO - Transacción confirmada.
20 2024-11-25 13:43:30,746 - INFO - Conexión cerrada.

```

ACTIVIDAD 4

```
actividad4.py X
actividad4.py > ...
1  import logging
2  from pymongo import MongoClient
3  from pymongo.errors import PyMongoError
4
5  import logging
6  from pymongo import MongoClient
7  from pymongo.errors import PyMongoError
8
9  # Configuración de logging
10 logging.basicConfig(
11     level=logging.INFO,
12     format="%(asctime)s - %(levelname)s - %(message)s",
13     handlers=[
14         logging.FileHandler("database_manager_documental.log"), # Logs guardados en un archivo
15         logging.StreamHandler(), # Logs también en consola
16     ]
17 )
18
19 class DatabaseManagerDocumental:
20     def __init__(self, uri, database_name, collection_name):
21         """Inicializa el componente DatabaseManagerDocumental."""
22         self.uri = uri
23         self.database_name = database_name
24         self.collection_name = collection_name
25         self.client = None
26         self.db = None
27         self.collection = None
28
29     def conectar(self):
30         """Conectar a la base de datos MongoDB"""
31         try:
32             self.client = MongoClient(self.uri)
33             self.db = self.client[self.database_name]
34             self.collection = self.db[self.collection_name]
35             logging.info(f"Conectado a MongoDB: {self.database_name}. {self.collection_name}")
36         except PyMongoError as e:
37             logging.error(f"Error al conectar a MongoDB: {e}")
38
```

```
actividad4.py X
actividad4.py > ...
19 class DatabaseManagerDocumental:
20     def desconectar(self):
21         """Cerrar la conexión a MongoDB"""
22         if self.client:
23             self.client.close()
24             logging.info("Conexión a MongoDB cerrada.")
25
26     def crear_documento(self, documento):
27         """Insertar un nuevo documento en la colección"""
28         try:
29             result = self.collection.insert_one(documento)
30             logging.info(f"Documento insertado con ID: {result.inserted_id}")
31             return result.inserted_id
32         except PyMongoError as e:
33             logging.error(f"Error al insertar el documento: {e}")
34
35     def leer_documentos(self, filtro={}):
36         """Leer documentos de la colección según un filtro"""
37         try:
38             documentos = list(self.collection.find(filtro))
39             logging.info(f"Documentos recuperados: {len(documentos)}")
40             for doc in documentos:
41                 logging.info(doc)
42             return documentos
43         except PyMongoError as e:
44             logging.error(f"Error al leer los documentos: {e}")
45             return []
46
47     def actualizar_documento(self, filtro, actualizacion):
48         """Actualizar un documento en la colección"""
49         try:
50             result = self.collection.update_one(filtro, {"$set": actualizacion})
51             if result.modified_count > 0:
52                 logging.info(f"Documento actualizado: {filtro}")
53             else:
54                 logging.warning(f"No se encontró documento para actualizar: {filtro}")
55         except PyMongoError as e:
56             logging.error(f"Error al actualizar el documento: {e}")
57
```



```

actividad4.py x
actividad4.py > ...
19 class DatabaseManagerDocumental:
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77 def eliminar_documento(self, filtro):
78     """Eliminar un documento de la colección"""
79     try:
80         result = self.collection.delete_one(filtro)
81         if result.deleted_count > 0:
82             logging.info(f"Documento eliminado: {filtro}")
83         else:
84             logging.warning(f"No se encontró documento para eliminar: {filtro}")
85     except PyMongoError as e:
86         logging.error(f"Error al eliminar el documento: {e}")
87
88 def iniciar_transaccion(self):
89     """Iniciar una transacción"""
90     try:
91         self.session = self.client.start_session()
92         self.session.start_transaction()
93         logging.info("Transacción iniciada.")
94     except PyMongoError as e:
95         logging.error(f"Error al iniciar la transacción: {e}")
96
97 def confirmar_transaccion(self):
98     """Confirmar (commit) una transacción"""
99     try:
100         if self.session:
101             self.session.commit_transaction()
102             logging.info("Transacción confirmada.")
103     except PyMongoError as e:
104         logging.error(f"Error al confirmar la transacción: {e}")
105
106 def revertir_transaccion(self):
107     """Revertir (rollback) una transacción"""
108     try:
109         if self.session:
110             self.session.abort_transaction()
111             logging.info("Transacción revertida.")
112     except PyMongoError as e:
113         logging.error(f"Error al revertir la transacción: {e}")

```

```

actividad4.py x
actividad4.py > ...
116 if __name__ == "__main__":
117     #Configurar el componente
118     db_manager=DatabaseManagerDocumental(
119         uri="mongodb://localhost:27017",
120         database_name="ldam",
121         collection_name="herramientas"
122     )
123     db_manager.conectar()
124
125     try:
126         # Crear documentos dentro de una transacción
127         db_manager.iniciar_transaccion()
128         db_manager.crear_documento({
129             "titulo": "La Gioconda",
130             "artista": "Leonardo da Vinci",
131             "fechaCreacion": 1503,
132             "tecnica": "sfumato",
133             "museo": "Louvre"
134         })
135         db_manager.crear_documento({
136             "titulo": "La noche estrellada",
137             "artista": "Van Gogh",
138             "fechaCreacion": 1889,
139             "tecnica": "oleo",
140             "museo": "MoMA"
141         })
142         db_manager.confirmar_transaccion()
143
144         # Leer todos los documentos
145         db_manager.leer_documentos()
146
147         # Actualizar un documento
148         db_manager.iniciar_transaccion()
149         db_manager.actualizar_documento(
150             {"titulo": "La Gioconda"},
151             {"fechaCreacion": 1600}
152         )
153         db_manager.confirmar_transaccion()
154

```

```

152     )
153     db_manager.confirmar_transaccion()
154
155     # Eliminar un documento
156     db_manager.iniciar_transaccion()
157     db_manager.eliminar_documento({"titulo": "La noche estrellada"})
158     db_manager.confirmar_transaccion()
159
160 except Exception as e:
161     logging.error(f"Error general: {e}")
162     db_manager.revertir_transaccion()
163
164 finally:
165     db_manager.desconectar()
166

```

```

● act4usuario@usuario:~/ACD/Tema6/actividad4$ /home/usuario/ACD/Tema6/actividad4/act4/bin/python /home/usuario/ACD/Tema6/actividad4/act
ividad4.py
2024-11-25 14:34:09,440 - INFO - Conectado a MongoDB: ldam.herramientas
2024-11-25 14:34:09,440 - INFO - Transacción iniciada.
2024-11-25 14:34:09,568 - INFO - Documento insertado con ID: 67447cd11c5f918e4ceba2ab
2024-11-25 14:34:09,569 - INFO - Documento insertado con ID: 67447cd11c5f918e4ceba2ac
2024-11-25 14:34:09,569 - INFO - Transacción confirmada.
2024-11-25 14:34:09,586 - INFO - Documentos recuperados: 2
2024-11-25 14:34:09,586 - INFO - {'_id': ObjectId('67447cd11c5f918e4ceba2ab'), 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci',
'fechaCreacion': 1503, 'tecnica': 'sfumato', 'museo': 'Louvre'}
2024-11-25 14:34:09,587 - INFO - {'_id': ObjectId('67447cd11c5f918e4ceba2ac'), 'titulo': 'La noche estrellada', 'artista': 'Van Gogh',
'fechaCreacion': 1889, 'tecnica': 'oleo', 'museo': 'MoMA'}
2024-11-25 14:34:09,587 - INFO - Transacción iniciada.
2024-11-25 14:34:09,603 - INFO - Documento actualizado: {'titulo': 'La Gioconda'}
2024-11-25 14:34:09,603 - INFO - Transacción confirmada.
2024-11-25 14:34:09,603 - INFO - Transacción iniciada.
2024-11-25 14:34:09,607 - INFO - Documento eliminado: {'titulo': 'La noche estrellada'}
2024-11-25 14:34:09,607 - INFO - Transacción confirmada.
2024-11-25 14:34:09,608 - INFO - Conexión a MongoDB cerrada.
○ act4usuario@usuario:~/ACD/Tema6/actividad4$

```

actividad4.py databasemanager_documental.log X

```

databasemanager_documental.log
1 2024-11-25 14:34:09,440 - INFO - Conectado a MongoDB: ldam.herramientas
2 2024-11-25 14:34:09,440 - INFO - Transacción iniciada.
3 2024-11-25 14:34:09,568 - INFO - Documento insertado con ID: 67447cd11c5f918e4ceba2ab
4 2024-11-25 14:34:09,569 - INFO - Documento insertado con ID: 67447cd11c5f918e4ceba2ac
5 2024-11-25 14:34:09,569 - INFO - Transacción confirmada.
6 2024-11-25 14:34:09,586 - INFO - Documentos recuperados: 2
7 2024-11-25 14:34:09,586 - INFO - {'_id': ObjectId('67447cd11c5f918e4ceba2ab'), 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci'
8 2024-11-25 14:34:09,587 - INFO - {'_id': ObjectId('67447cd11c5f918e4ceba2ac'), 'titulo': 'La noche estrellada', 'artista': 'Van Gogh'
9 2024-11-25 14:34:09,587 - INFO - Transacción iniciada.
10 2024-11-25 14:34:09,603 - INFO - Documento actualizado: {'titulo': 'La Gioconda'}
11 2024-11-25 14:34:09,603 - INFO - Transacción confirmada.
12 2024-11-25 14:34:09,603 - INFO - Transacción iniciada.
13 2024-11-25 14:34:09,607 - INFO - Documento eliminado: {'titulo': 'La noche estrellada'}
14 2024-11-25 14:34:09,607 - INFO - Transacción confirmada.
15 2024-11-25 14:34:09,608 - INFO - Conexión a MongoDB cerrada.
16

```

ACTIVIDAD 5

```

actividad5.py > DatabaseManagerObject > eliminar_obraArte
1  import logging
2  import transaction
3  from ZODB import DB, FileStorage
4  from persistent import Persistent
5
6  # Configuración de logging
7  logging.basicConfig(
8      level=logging.INFO,
9      format="%(asctime)s - %(levelname)s - %(message)s",
10     handlers=[
11         logging.FileHandler("databasemanager_object.log"), # Logs guardados en un archivo
12         logging.StreamHandler(), # Logs también en consola
13     ]
14 )
15
16 class ObraArte(Persistent):
17     """Clase que representa una obra de arte."""
18     def __init__(self, titulo, artista, fechaCreacion, tecnica, museo):
19         self.titulo = titulo
20         self.artista = artista
21         self.fechaCreacion = fechaCreacion
22         self.tecnica = tecnica
23         self.museo = museo
24
25 class DatabaseManagerObject:
26     """Componente para gestionar bases de datos orientadas a objetos con ZODB."""
27     def __init__(self, filepath="ldam.fs"):
28         self.filepath = filepath
29         self.db = None
30         self.connection = None
31         self.root = None
32         self.transaccion_iniciada = False
33
34     def conectar(self):
35         """Conecta a la base de datos ZODB."""
36         try:
37             storage = FileStorage.FileStorage(self.filepath)
38             self.db = DB(storage)
39             self.connection = self.db.open()
40             self.root = self.connection.root()
41             if "obrasArte" not in self.root:
42                 self.root["obrasArte"] = {}
43                 transaction.commit()
44             logging.info("Conexión establecida con ZODB.")
45         except Exception as e:
46             logging.error(f"Error al conectar a ZODB: {e}")
47
48     def desconectar(self):
49         """Cierra la conexión a la base de datos."""
50         try:
51             if self.connection:
52                 self.connection.close()
53             self.db.close()
54             logging.info("Conexión a ZODB cerrada.")
55         except Exception as e:
56             logging.error(f"Error al cerrar la conexión a ZODB: {e}")
57
58     def iniciar_transaccion(self):
59         """Inicia una transacción."""
60         try:
61             transaction.begin()
62             self.transaccion_iniciada = True
63             logging.info("Transacción iniciada.")
64         except Exception as e:
65             logging.error(f"Error al iniciar la transacción: {e}")
66
67     def confirmar_transaccion(self):
68         """Confirma la transacción."""
69         if self.transaccion_iniciada:
70             try:
71                 transaction.commit()
72                 self.transaccion_iniciada = False
73             except Exception as e:
74                 logging.error(f"Error al confirmar la transacción: {e}")
75
76     def eliminar_obraArte(self, titulo):
77         """Elimina una obra de arte por su título."""
78         try:
79             if self.transaccion_iniciada:
80                 transaction.abort()
81             self.conectar()
82             root = self.connection.root()
83             obrasArte = root["obrasArte"]
84             if titulo in obrasArte:
85                 del obrasArte[titulo]
86                 transaction.commit()
87             logging.info(f"Obra de arte '{titulo}' eliminada.")
88         except Exception as e:
89             logging.error(f"Error al eliminar obra de arte: {e}")
90         finally:
91             self.desconectar()
92
93     def __str__(self):
94         return f"DatabaseManagerObject(filepath={self.filepath})"
95
96 if __name__ == "__main__":
97     dm = DatabaseManagerObject()
98     dm.conectar()
99     root = dm.connection.root()
100     obrasArte = root["obrasArte"]
101     obrasArte["El Guernica"] = {}
102     transaction.commit()
103     dm.desconectar()
104     print("Obra de arte añadida correctamente.")
105
106 # Ejemplo de uso de la clase ObraArte
107 obra = ObraArte("El Guernica", "Pablo Picasso", "1937", "Pintura", "Museo de Arte Moderno")
108 dm.conectar()
109 dm.iniciar_transaccion()
110 dm.confirmar_transaccion()
111 dm.eliminar_obraArte("El Guernica")
112 dm.desconectar()
113

```

```

actividad5.py > DatabaseManagerObject > eliminar_obraArte
25 class DatabaseManagerObject:
33
34     def conectar(self):
35         """Conecta a la base de datos ZODB."""
36         try:
37             storage = FileStorage.FileStorage(self.filepath)
38             self.db = DB(storage)
39             self.connection = self.db.open()
40             self.root = self.connection.root()
41             if "obrasArte" not in self.root:
42                 self.root["obrasArte"] = {}
43                 transaction.commit()
44             logging.info("Conexión establecida con ZODB.")
45         except Exception as e:
46             logging.error(f"Error al conectar a ZODB: {e}")
47
48     def desconectar(self):
49         """Cierra la conexión a la base de datos."""
50         try:
51             if self.connection:
52                 self.connection.close()
53             self.db.close()
54             logging.info("Conexión a ZODB cerrada.")
55         except Exception as e:
56             logging.error(f"Error al cerrar la conexión a ZODB: {e}")
57
58     def iniciar_transaccion(self):
59         """Inicia una transacción."""
60         try:
61             transaction.begin()
62             self.transaccion_iniciada = True
63             logging.info("Transacción iniciada.")
64         except Exception as e:
65             logging.error(f"Error al iniciar la transacción: {e}")
66
67     def confirmar_transaccion(self):
68         """Confirma la transacción."""
69         if self.transaccion_iniciada:
70             try:
71                 transaction.commit()
72                 self.transaccion_iniciada = False
73             except Exception as e:
74                 logging.error(f"Error al confirmar la transacción: {e}")
75
76     def eliminar_obraArte(self, titulo):
77         """Elimina una obra de arte por su título."""
78         try:
79             if self.transaccion_iniciada:
80                 transaction.abort()
81             self.conectar()
82             root = self.connection.root()
83             obrasArte = root["obrasArte"]
84             if titulo in obrasArte:
85                 del obrasArte[titulo]
86                 transaction.commit()
87             logging.info(f"Obra de arte '{titulo}' eliminada.")
88         except Exception as e:
89             logging.error(f"Error al eliminar obra de arte: {e}")
90         finally:
91             self.desconectar()
92
93     def __str__(self):
94         return f"DatabaseManagerObject(filepath={self.filepath})"
95
96 if __name__ == "__main__":
97     dm = DatabaseManagerObject()
98     dm.conectar()
99     root = dm.connection.root()
100     obrasArte = root["obrasArte"]
101     obrasArte["El Guernica"] = {}
102     transaction.commit()
103     dm.desconectar()
104     print("Obra de arte añadida correctamente.")
105
106 # Ejemplo de uso de la clase ObraArte
107 obra = ObraArte("El Guernica", "Pablo Picasso", "1937", "Pintura", "Museo de Arte Moderno")
108 dm.conectar()
109 dm.iniciar_transaccion()
110 dm.confirmar_transaccion()
111 dm.eliminar_obraArte("El Guernica")
112 dm.desconectar()
113

```



```

actividad5.py > DatabaseManagerObject > eliminar_obraArte
25 class DatabaseManagerObject:
67     def confirmar_transaccion(self):
69         if self.transaccion_iniciada:
70             try:
71                 transaction.commit()
72                 self.transaccion_iniciada = False
73                 logging.info("Transacción confirmada.")
74             except Exception as e:
75                 logging.error(f"Error al confirmar la transacción: {e}")
76
77     def revertir_transaccion(self):
78         """Revierte la transacción."""
79         if self.transaccion_iniciada:
80             try:
81                 transaction.abort()
82                 self.transaccion_iniciada = False
83                 logging.info("Transacción revertida.")
84             except Exception as e:
85                 logging.error(f"Error al revertir la transacción: {e}")
86
87     def crear_obraArte(self, id, titulo, artista, fechaCreacion, tecnica, museo):
88         """Crea y almacena una nueva obra de arte."""
89         try:
90             if id in self.root["obrasArte"]:
91                 raise ValueError(f"Ya existe una obra de arte con ID {id}.")
92             self.root["obrasArte"][id] = ObraArte(titulo, artista, fechaCreacion, tecnica, museo)
93             logging.info(f"Obra de arte con ID {id} creada exitosamente.")
94         except Exception as e:
95             logging.error(f"Error al crear la obra de arte con ID {id}: {e}")
96             raise ValueError(f"Ya existe una obra de arte con ID {id}.")
97
98     def leer_obrasArte(self):
99         """Lee y muestra todas las obras de arte almacenadas."""
100         try:
101             obras = self.root["obrasArte"]
102             for id, obra in obras.items():
103                 logging.info(
104                     f"ID: {id}, Titulo: {obra.titulo}, Artista: {obra.artista}, "

```

```

actividad5.py > DatabaseManagerObject > eliminar_obraArte
25 class DatabaseManagerObject:
97
98     def leer_obrasArte(self):
99         """Lee y muestra todas las obras de arte almacenadas."""
100         try:
101             obras = self.root["obrasArte"]
102             for id, obra in obras.items():
103                 logging.info(
104                     f"ID: {id}, Titulo: {obra.titulo}, Artista: {obra.artista}, "
105                     f"Fecha de creacion: {obra.fechaCreacion}, Tecnica: {obra.tecnica}, Museo: {obra.museo}"
106                 )
107             return obras
108         except Exception as e:
109             logging.error(f"Error al leer las obras de arte: {e}")
110
111     def actualizar_obrasArte(self, id, titulo, artista, fechaCreacion, tecnica, museo):
112         """Actualiza los atributos de una obra de arte."""
113         try:
114             obra = self.root["obrasArte"].get(id)
115             if not obra:
116                 raise ValueError(f"No existe una obra de arte con ID {id}.")
117             obra.titulo = titulo
118             obra.artista = artista
119             obra.fechaCreacion = fechaCreacion
120             obra.tecnica = tecnica
121             obra.museo = museo
122             logging.info(f"Obra de arte con ID {id} actualizada exitosamente.")
123         except Exception as e:
124             logging.error(f"Error al actualizar la obra de arte con ID {id}: {e}")
125
126     def eliminar_obraArte(self, id):
127         """Elimina una obra de arte por su ID."""
128         try:
129             if id not in self.root["obrasArte"]:
130                 raise ValueError(f"No existe una obra de arte con ID {id}.")
131             del self.root["obrasArte"][id]
132             logging.info(f"Obra de arte con ID {id} eliminada exitosamente.")
133         except Exception as e:
134             logging.error(f"Error al eliminar la obra de arte con ID {id}: {e}")

```

```

actividad5.py > DatabaseManagerObject > eliminar_obraArte
25 class DatabaseManagerObject:
26
27     def eliminar_obraArte(self, id):
28         """Elimina una obra de arte por su ID."""
29         try:
30             if id not in self.root["obrasArte"]:
31                 raise ValueError(f"No existe una obra de arte con ID {id}.")
32             del self.root["obrasArte"][id]
33             logging.info(f"Obra de arte con ID {id} eliminada exitosamente.")
34         except Exception as e:
35             logging.error(f"Error al eliminar la obra ed arte con ID {id}: {e}")
36             raise ValueError(f"No existe una obra de arte con ID {id}.")
37
38 if __name__ == "__main__":
39     manager = DatabaseManagerObject()
40     manager.conectar()
41
42     # Crear obras de arte con transacción
43     manager.iniciar_transaccion()
44     manager.crear_obraArte(1, "La Gioconda", "Leonardo da Vinci", "1503", "sfumato", "Louvre")
45     manager.crear_obraArte(2, "Las Meninas", "Velazquez", "1656", "alla prima", "El Prado")
46     manager.crear_obraArte(3, "El Guernica", "Pablo Picasso", "1937", "oleo", "Reina Sofia")
47     manager.confirmar_transaccion()
48
49     # Leer obras de arte
50     manager.leer_obrasArte()
51
52     #Insertar obre de arte con id que ya existe para revertir la transaccion
53     manager.iniciar_transaccion()
54     try:
55         manager.crear_obraArte(2, "La noche estrellada", "Van Gohg", "1889", "oleo", "MoMA")
56         manager.confirmar_transaccion()
57     except ValueError as e:
58         manager.revertir_transaccion()
59
60     #Mostrar obras de arte
61     manager.leer_obrasArte()
62

```

```

# Actualizar una obra de arte con transacción
manager.iniciar_transaccion()
manager.actualizar_obrasArte(1, "La Gioconda", "Leonardo da Vinci", "1600", "oleo", "Louvre")
manager.confirmar_transaccion()

#Mostrar obras de arte
manager.leer_obrasArte()

# Eliminar una obra de arte con transacción
manager.iniciar_transaccion()
try:
    manager.eliminar_obraArte(5)
    manager.confirmar_transaccion()
except ValueError as e:
    manager.revertir_transaccion()

# Leer obras de arte nuevamente
manager.leer_obrasArte()

```

```

act5usuario@usuario:~/ACD/Tema6/actividad5$ /home/usuario/ACD/Tema6/actividad5/act5/bin/python /home/usuario/ACD/Tema6/actividad5/actividad5.py
2024-12-05 10:21:12,502 - INFO - Conexión establecida con ZODB.
2024-12-05 10:21:12,502 - INFO - Transacción iniciada.
2024-12-05 10:21:12,502 - INFO - Obra de arte con ID 1 creada exitosamente.
2024-12-05 10:21:12,502 - INFO - Obra de arte con ID 2 creada exitosamente.
2024-12-05 10:21:12,502 - INFO - Obra de arte con ID 3 creada exitosamente.
2024-12-05 10:21:12,502 - INFO - Transacción confirmada.
2024-12-05 10:21:12,502 - INFO - ID: 1, Título: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1503, Tecnica: sfumato, Museo: Louvre
2024-12-05 10:21:12,502 - INFO - ID: 2, Título: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo: El Prado
2024-12-05 10:21:12,502 - INFO - ID: 3, Título: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: Reina Sofia
2024-12-05 10:21:12,502 - INFO - Transacción iniciada.
2024-12-05 10:21:12,502 - ERROR - Error al crear la obra de arte con ID 2: Ya existe una obra de arte con ID 2.
2024-12-05 10:21:12,502 - INFO - Transacción revertida.
2024-12-05 10:21:12,502 - INFO - ID: 1, Título: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1503, Tecnica: sfumato, Museo: Louvre
2024-12-05 10:21:12,502 - INFO - ID: 2, Título: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo: El Prado
2024-12-05 10:21:12,502 - INFO - ID: 3, Título: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: Reina Sofia
2024-12-05 10:21:12,502 - INFO - Transacción iniciada.
2024-12-05 10:21:12,502 - INFO - Obra de arte con ID 1 actualizada exitosamente.
2024-12-05 10:21:12,502 - INFO - Transacción confirmada.
2024-12-05 10:21:12,502 - INFO - ID: 1, Título: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1600, Tecnica: oleo, Museo: Louvre
2024-12-05 10:21:12,502 - INFO - ID: 2, Título: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo: El Prado
2024-12-05 10:21:12,502 - INFO - ID: 3, Título: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: Reina Sofia
2024-12-05 10:21:12,502 - INFO - Transacción iniciada.
2024-12-05 10:21:12,502 - ERROR - Error al eliminar la obra ed arte con ID 5: No existe una obra de arte con ID 5.
2024-12-05 10:21:12,502 - INFO - Transacción revertida.
2024-12-05 10:21:12,502 - INFO - ID: 1, Título: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1600, Tecnica: oleo, Museo: Louvre
2024-12-05 10:21:12,502 - INFO - ID: 2, Título: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo: El Prado
2024-12-05 10:21:12,502 - INFO - ID: 3, Título: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: Reina Sofia
act5usuario@usuario:~/ACD/Tema6/actividad5$

```

database manager_objectlog

```

1 2024-12-05 10:23:58,692 - INFO - Conexión establecida con ZODB.
2 2024-12-05 10:23:58,692 - INFO - Transacción iniciada.
3 2024-12-05 10:23:58,693 - INFO - Obra de arte con ID 1 creada exitosamente.
4 2024-12-05 10:23:58,693 - INFO - Obra de arte con ID 2 creada exitosamente.
5 2024-12-05 10:23:58,693 - INFO - Obra de arte con ID 3 creada exitosamente.
6 2024-12-05 10:23:58,693 - INFO - Transacción confirmada.
7 2024-12-05 10:23:58,693 - INFO - ID: 1, Título: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1503, Tecnica: sfumato, M
8 2024-12-05 10:23:58,693 - INFO - ID: 2, Título: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo:
9 2024-12-05 10:23:58,693 - INFO - ID: 3, Título: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: R
10 2024-12-05 10:23:58,693 - INFO - Transacción iniciada.
11 2024-12-05 10:23:58,693 - ERROR - Error al crear la obra de arte con ID 2: Ya existe una obra de arte con ID 2.
12 2024-12-05 10:23:58,693 - INFO - Transacción revertida.
13 2024-12-05 10:23:58,693 - INFO - ID: 1, Título: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1503, Tecnica: sfumato, M
14 2024-12-05 10:23:58,693 - INFO - ID: 2, Título: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo:
15 2024-12-05 10:23:58,693 - INFO - ID: 3, Título: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: R
16 2024-12-05 10:23:58,693 - INFO - Transacción iniciada.
17 2024-12-05 10:23:58,693 - INFO - Obra de arte con ID 1 actualizada exitosamente.
18 2024-12-05 10:23:58,693 - INFO - Transacción confirmada.
19 2024-12-05 10:23:58,693 - INFO - ID: 1, Título: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1600, Tecnica: oleo, Muse
20 2024-12-05 10:23:58,693 - INFO - ID: 2, Título: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo:
21 2024-12-05 10:23:58,693 - INFO - ID: 3, Título: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: R
22 2024-12-05 10:23:58,693 - INFO - Transacción iniciada.
23 2024-12-05 10:23:58,693 - ERROR - Error al eliminar la obra ed arte con ID 5: No existe una obra de arte con ID 5.
24 2024-12-05 10:23:58,693 - INFO - Transacción revertida.
25 2024-12-05 10:23:58,693 - INFO - ID: 1, Título: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1600, Tecnica: oleo, Muse
26 2024-12-05 10:23:58,693 - INFO - ID: 2, Título: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo:
27 2024-12-05 10:23:58,693 - INFO - ID: 3, Título: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: R

```