TEMA 6

ACTIVIDAD 1

actividad1.py > ...

```python

# Configuración de logging para guardar en un archivo
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler("log_datos.log"), # Guardar logs en un archivo
    ]
)

#Datos
obras=[
    {"titulo": "La noche estrellada", "artista": "Van Gogh", "fechaCreacion": 1889, "tecnica": "Oleo", "museo": "MoMA"},
    {"titulo": "La Gioconda", "artista": "Leonardo da Vinci", "fechaCreacion": 1503, "tecnica": "sfumato", "museo": "Louvre"},
    {"titulo": "Las meninas", "artista": "Velazquez", "fechaCreacion": 1656, "tecnica": "alla prima", "museo": "El Prado"}
]

#Crea archivo json
gestor = DataManager("obras.json", tipo_archivo="json")

#Inicia transaccion, escribe los datos y confirma
gestor.iniciar_transaccion()
for obra in obras:
    gestor.escribir_dato(obra)
gestor.confirmar_transaccion()

#Cambiar configuracion para csv
gestor.actualizar_configuracion("obras.csv", nuevo_tipo="csv")

#Inicia transaccion y confirma
gestor.iniciar_transaccion()
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

log_datos.log

```
2024-11-21 09:32:20,769 - INFO - Transacción iniciada.
2024-11-21 09:32:20,769 - INFO - Dato agregado: {'titulo': 'La noche estrellada', 'artista': 'Van Gogh', 'fechaCreacion': 1889, 'tecn
2024-11-21 09:32:20,769 - INFO - Dato agregado: {'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': 1503, 'tec
2024-11-21 09:32:20,769 - INFO - Dato agregado: {'titulo': 'Las meninas', 'artista': 'Velazquez', 'fechaCreacion': 1656, 'tecnica': '
2024-11-21 09:32:20,769 - INFO - Transacción confirmada y cambios guardados.
2024-11-21 09:32:20,769 - INFO - Configuración actualizada. Nueva ruta del archivo: obras.csv
2024-11-21 09:32:20,769 - INFO - Transacción iniciada.
2024-11-21 09:32:20,769 - INFO - Archivo CSV guardado. Versión actual: 3
2024-11-21 09:32:20,770 - INFO - Transacción confirmada y cambios guardados.
```

ACTIVIDAD 2

actividad2.py

actividad2.py > DatabaseManager > eliminar_obraArte

```python
import logging
import mysql.connector
from mysql.connector import Error

# Configuración de logging
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s",
    handlers=[
        logging.FileHandler("databasemanager.log"), # Logs guardados en un archivo
        logging.StreamHandler(), # Logs también en consola
    ]
)

class DatabaseManager:
    def __init__(self, host, user, password, database):
        self.host = host
        self.user = user
        self.password = password
        self.database = database
        self.connection = None

    def conectar(self):
        """Conectar a la base de datos MySQL"""
        try:
            self.connection = mysql.connector.connect(
                host=self.host,
                user=self.user,
                password=self.password,
                database=self.database
            )
            if self.connection.is_connected():
                logging.info("Conexión exitosa a la base de datos.")
        except Error as e:
            logging.error(f"Error al conectar a la base de datos: {e}")
```

actividad2.py

Puede pegar la imagen desde el portapapeles.

actividad2.py > DatabaseManager > eliminar_obraArte

```python
class DatabaseManager:

    def desconectar(self):
        """Cerrar la conexión a la base de datos"""
        if self.connection.is_connected():
            self.connection.close()
            logging.info("Conexión cerrada.")

    def crear_obraArte(self, titulo, artista, fechaCreacion, tecnica, museo):
        """Insertar una nueva obra de arte en la base de datos"""
        try:
            cursor = self.connection.cursor()
            query = """
                INSERT INTO ObrasArte (titulo, artista, fechaCreacion, tecnica, museo)
                VALUES (%s, %s, %s, %s, %s)
            """
            cursor.execute(query, (titulo, artista, fechaCreacion, tecnica, museo))
            #self.connection.commit()
            logging.info(f"Obra de arte '{titulo}' insertada exitosamente.")
        except Error as e:
            logging.error(f"Error al insertar la obra de arte '{titulo}': {e}")

    def leer_obrasArte(self):
        """Leer todas las obras de arte de la base de datos"""
        try:
            cursor = self.connection.cursor(dictionary=True)
            cursor.execute("SELECT * FROM ObrasArte")
            obras = cursor.fetchall()
            logging.info("Obras de arte recuperadas:")
            for obra in obras:
                logging.info(obra)
            return obras
        except Error as e:
            logging.error(f"Error al leer las obras de arte: {e}")
            return None
```

actividad2.py

actividad2.py > DatabaseManager > eliminar_obraArte

```python
class DatabaseManager:
    def actualizar_obraArte(self, titulo, artista, fechaCreacion, tecnica, museo):
        """Actualizar una obra de arte en la base de datos"""
        try:
            cursor = self.connection.cursor()
            query = """
                UPDATE ObrasArte
                SET artista = %s, fechaCreacion = %s, tecnica = %s, museo = %s
                WHERE titulo = %s
            """
            cursor.execute(query, (artista, fechaCreacion, tecnica, museo, titulo))
            self.connection.commit()
            logging.info(f"Obra de arte con titulo{titulo} actualizada exitosamente.")
        except Error as e:
            logging.error(f"Error al actualizar la obra de arte con titulo {titulo}: {e}")

    def eliminar_obraArte(self, titulo):
        """Eliminar una obra de arte de la base de datos"""
        try:
            cursor = self.connection.cursor()
            query = "DELETE FROM ObrasArte WHERE titulo = %s"
            cursor.execute(query, (titulo,))
            self.connection.commit()
            logging.info(f"Obra de arte con titulo {titulo} eliminada exitosamente.")
        except Error as e:
            logging.error(f"Error al eliminar la obra de arte con titulo {titulo}: {e}")

    def iniciar_transaccion(self):
        """Iniciar una transacción"""
        try:
            if self.connection.is_connected():
                self.connection.start_transaction()
                logging.info("Transacción iniciada.")
        except Error as e:
            logging.error(f"Error al iniciar la transacción: {e}")
```

actividad2.py

actividad2.py > ...

```python
class DatabaseManager:
    def confirmar_transaccion(self):
        try:
            if self.connection.is_connected():
                self.connection.commit()
                logging.info("Transacción confirmada.")
        except Error as e:
            logging.error(f"Error al confirmar la transacción: {e}")

    def revertir_transaccion(self):
        """Revertir (rollback) una transacción"""
        try:
            if self.connection.is_connected():
                self.connection.rollback()
                logging.info("Transacción revertida.")
        except Error as e:
            logging.error(f"Error al revertir la transacción: {e}")


if __name__ == "__main__":
    db_manager = DatabaseManager("localhost", "usuario", "usuario", "1dam")
    db_manager.conectar()

    # Insertar una nueva obra de arte
    db_manager.crear_obraArte("La Gioconda", "Leonardo da Vinci", 1503, "sfumato", "Louvre")

    # Leer todas las obras de arte
    db_manager.leer_obrasArte()

    # Actualizar una obra de arte
    db_manager.actualizar_obraArte("La Gioconda", "Leonardo da Vinci", 1550, "sfumato", "Louvre")

    # Eliminar una obra de arte
    db_manager.eliminar_obraArte("El Guernica")

    # Gestionar transacciones
    db_manager.iniciar_transaccion()
    db_manager.crear_obraArte("La noche estrellada", "Van Gohg", 1889, "Oleo", "MoMA")
    db_manager.revertir_transaccion() # No se guardará la inserción
    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': '1
550', '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

databasemanager.log

```
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', 'tecni
2024-11-25 14:00:48,558 - INFO - {'id': 7, 'titulo': 'El Guernica', 'artista': 'Pablo Picasso', 'fechaCreacion': '1937', 'tecnica': '
2024-11-25 14:00:48,558 - INFO - {'id': 8, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1550', 'tecnica
2024-11-25 14:00:48,558 - INFO - {'id': 10, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1550', 'tecnic
2024-11-25 14:00:48,558 - INFO - {'id': 12, 'titulo': 'La Gioconda', 'artista': 'Leonardo da Vinci', 'fechaCreacion': '1503', 'tecnic
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.
```

ACTIVIDAD 3

actividad3.py > ...

```python
import logging
from peewee import Model, CharField, ForeignKeyField, MySQLDatabase

# Componente DatabaseManagerORM
class DatabaseManagerORM:
    def __init__(self):
        self.db = db

    def conectar(self):
        """Conecta la base de datos y crea las tablas."""
        self.db.connect()
        self.db.create_tables([Proveedor, Herramienta])
        logging.info("Conexión establecida y tablas creadas.")

    def desconectar(self):
        """Cierra la conexión a la base de datos."""
        if not self.db.is_closed():
            self.db.close()
            logging.info("Conexión cerrada.")

    def iniciar_transaccion(self):
        """Inicia una transacción."""
        self.db.begin()
        logging.info("Transacción iniciada.")

    def confirmar_transaccion(self):
        """Confirma (commit) una transacción."""
        self.db.commit()
        logging.info("Transacción confirmada.")

    def revertir_transaccion(self):
        """Revierte (rollback) una transacción."""
        self.db.rollback()
        logging.info("Transacción revertida.")

    def crear_proveedor(self, nombre, direccion):
        """Inserta un nuevo proveedor."""
        proveedor = Proveedor.create(nombre=nombre, direccion=direccion)
        logging.info(f"Proveedor creado: {proveedor.nombre} - {proveedor.direccion}")
        return proveedor
```

actividad3.py > ...

```python
class DatabaseManagerORM:
    def crear_herramienta(self, nombre, tipo, marca, uso, material, proveedor_nombre):
        """Inserta una nueva herramienta."""
        try:
            proveedor = Proveedor.get(Proveedor.nombre == proveedor_nombre)
            herramienta = Herramienta.create(
                nombre=nombre,
                tipo=tipo,
                marca=marca,
                uso=uso,
                material=material,
                proveedor=proveedor
            )
            logging.info(f"Herramienta creada: {herramienta.nombre} - {herramienta.tipo}")
            return herramienta
        except Proveedor.DoesNotExist:
            logging.error(f"El proveedor '{proveedor_nombre}' no existe.")
            raise

    def leer_herramientas(self):
        """Lee todas las herramientas."""
        herramientas = Herramienta.select()
        logging.info("Leyendo herramientas:")
        for herramienta in herramientas:
            logging.info(f"{herramienta.nombre} - {herramienta.tipo} ({herramienta.proveedor.nombre})")
        return herramientas

    def actualizar_proveedor(self, nombre, direccion):
        """Actualizar un proveedor en la base de datos"""
        proveedor = Proveedor.get(Proveedor.nombre == nombre)
        proveedor.direccion = direccion
        proveedor.save()
        return proveedor

    def actualizar_herramienta(self, nombre, tipo):
        """Actualizar una herramienta en la base de datos"""
        herramienta = Herramienta.get(Herramienta.nombre == nombre)
        herramienta.tipo = tipo
        herramienta.save()
        return herramienta
```

actividad3.py > ...

```python
class DatabaseManagerORM:
    def eliminar_herramienta(self, nombre):
        """Eliminar una herramienta de la base de datos"""
        herramienta = Herramienta.get(Herramienta.nombre == nombre)
        herramienta.delete_instance()
        return herramienta

    def eliminar_proveedor(self, nombre):
        """Eliminar un proveedor de la base de datos"""
        proveedor = Proveedor.get(Proveedor.nombre == nombre)
        proveedor.delete_instance()
        return proveedor

    def consultar_herramienta(self, proveedor_nombre):
        """Imprimir herramientas que pertenezcan a un proveedor"""
        try:
            proveedor = Proveedor.get(Proveedor.nombre == proveedor_nombre)
            herramientas = Herramienta.select().where(Herramienta.proveedor == proveedor)
            for herramienta in herramientas:
                logging.info(f"{herramienta.nombre} - {herramienta.tipo}")
        except Proveedor.DoesNotExist:
            logging.error(f"No existe el proveedor '{proveedor_nombre}'.")

# Configuración de logging
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s",
    handlers=[
        logging.FileHandler("databasemanager_orm.log"),
        logging.StreamHandler()
    ]
)

# Configuración de la base de datos MySQL
db = MySQLDatabase(
    "1dam", # Nombre de la base de datos
    user="usuario", # Usuario de MySQL
    password="usuario", # Contraseña de MySQL
    host="localhost", # Host
    port=3306 # Puerto por defecto de MySQL
```

actividad3.py > ...

```python
# Modelos de la base de datos
class Proveedor(Model):
    nombre = CharField()
    direccion = CharField()
    class Meta:
        database = db

class Herramienta(Model):
    nombre = CharField()
    tipo = CharField()
    marca = CharField()
    uso = CharField()
    material = CharField()
    proveedor = ForeignKeyField(Proveedor, backref='herramientas')
    class Meta:
        database = db

# Flujo principal
gestion = DatabaseManagerORM()
gestion.conectar()
# Gestion de Proveedores
print("Gestión de Proveedores:")
gestion.iniciar_transaccion()
gestion.crear_proveedor("Proveedor A", "Contacto 123-456-789")
gestion.crear_proveedor("Proveedor B", "Contacto 987-654-321")
gestion.confirmar_transaccion()

# Actualización de Proveedor
print("Cambio del dato por mi DNI 12345678A al proveedor A")
gestion.iniciar_transaccion()
gestion.actualizar_proveedor("Proveedor A", "24525402L")
gestion.confirmar_transaccion()

# Eliminar Proveedor
print("Eliminar al proveedor B")
gestion.iniciar_transaccion()
gestion.eliminar_proveedor("Proveedor B")
gestion.confirmar_transaccion()

# Gestión de Herramientas
```

actividad3.py > ...

```python
gestion.confirmar_transaccion()

# Gestión de Herramientas
print("Gestión de Herramientas:")
gestion.iniciar_transaccion()
gestion.crear_herramienta("Martillo", "Manual", "Facom", "Percusion", "Acero", "Proveedor A")
gestion.crear_herramienta("Taladro", "Eléctrico", "Facom", "Percusion", "Acero", "Proveedor A")
gestion.confirmar_transaccion()

# Consultar herramientas
print("Herramientas asociadas al proveedor Proveedor A:")
gestion.consultar_herramienta("Proveedor A")

# Actualizar herramienta
print("Actualizar herramienta Martillo a tipo reforzado")
gestion.iniciar_transaccion()
gestion.actualizar_herramienta("Martillo", "Reforzado")
gestion.confirmar_transaccion()

# Eliminar herramienta
print("Eliminar herramienta Taladro")
gestion.iniciar_transaccion()
gestion.eliminar_herramienta("Taladro")
gestion.confirmar_transaccion()

# Desconectar
gestion.desconectar()
```

```
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* ×

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

```python
import logging
from pymongo import MongoClient
from pymongo.errors import PyMongoError

import logging
from pymongo import MongoClient
from pymongo.errors import PyMongoError

# Configuración de logging
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s",
    handlers=[
        logging.FileHandler("databasemanager_documental.log"),  # Logs guardados en un archivo
        logging.StreamHandler(),  # Logs también en consola
    ]
)

class DatabaseManagerDocumental:
    def __init__(self, uri, database_name, collection_name):
        """Inicializa el componente DatabaseManagerDocumental."""
        self.uri = uri
        self.database_name = database_name
        self.collection_name = collection_name
        self.client = None
        self.db = None
        self.collection = None

    def conectar(self):
        """Conectar a la base de datos MongoDB"""
        try:
            self.client = MongoClient(self.uri)
            self.db = self.client[self.database_name]
            self.collection = self.db[self.collection_name]
            logging.info(f"Conectado a MongoDB: {self.database_name}.{self.collection_name}")
        except PyMongoError as e:
            logging.error(f"Error al conectar a MongoDB: {e}")

```

actividad4.py

```python
class DatabaseManagerDocumental:
    def desconectar(self):
        """Cerrar la conexión a MongoDB"""
        if self.client:
            self.client.close()
            logging.info("Conexión a MongoDB cerrada.")

    def crear_documento(self, documento):
        """Insertar un nuevo documento en la colección"""
        try:
            result = self.collection.insert_one(documento)
            logging.info(f"Documento insertado con ID: {result.inserted_id}")
            return result.inserted_id
        except PyMongoError as e:
            logging.error(f"Error al insertar el documento: {e}")

    def leer_documentos(self, filtro={}):
        """Leer documentos de la colección según un filtro"""
        try:
            documentos = list(self.collection.find(filtro))
            logging.info(f"Documentos recuperados: {len(documentos)}")
            for doc in documentos:
                logging.info(doc)
            return documentos
        except PyMongoError as e:
            logging.error(f"Error al leer los documentos: {e}")
            return []

    def actualizar_documento(self, filtro, actualizacion):
        """Actualizar un documento en la colección"""
        try:
            result = self.collection.update_one(filtro, {"$set": actualizacion})
            if result.modified_count > 0:
                logging.info(f"Documento actualizado: {filtro}")
            else:
                logging.warning(f"No se encontró documento para actualizar: {filtro}")
        except PyMongoError as e:
            logging.error(f"Error al actualizar el documento: {e}")
```

actividad4.py

```python
class DatabaseManagerDocumental:
    def eliminar_documento(self, filtro):
        """Eliminar un documento de la colección"""
        try:
            result = self.collection.delete_one(filtro)
            if result.deleted_count > 0:
                logging.info(f"Documento eliminado: {filtro}")
            else:
                logging.warning(f"No se encontró documento para eliminar: {filtro}")
        except PyMongoError as e:
            logging.error(f"Error al eliminar el documento: {e}")

    def iniciar_transaccion(self):
        """Iniciar una transacción"""
        try:
            self.session = self.client.start_session()
            self.session.start_transaction()
            logging.info("Transacción iniciada.")
        except PyMongoError as e:
            logging.error(f"Error al iniciar la transacción: {e}")

    def confirmar_transaccion(self):
        """Confirmar (commit) una transacción"""
        try:
            if self.session:
                self.session.commit_transaction()
                logging.info("Transacción confirmada.")
        except PyMongoError as e:
            logging.error(f"Error al confirmar la transacción: {e}")

    def revertir_transaccion(self):
        """Revertir (rollback) una transacción"""
        try:
            if self.session:
                self.session.abort_transaction()
                logging.info("Transacción revertida.")
        except PyMongoError as e:
            logging.error(f"Error al revertir la transacción: {e}")
```

actividad4.py

```python
if __name__ == "__main__":
    #Configurar el componente
    db_manager=DatabaseManagerDocumental(
        uri="mongodb://localhost:27017",
        database_name="1dam",
        collection_name="herramientas"
    )
    db_manager.conectar()

    try:
        # Crear documentos dentro de una transacción
        db_manager.iniciar_transaccion()
        db_manager.crear_documento({
            "titulo": "La Gioconda",
            "artista": "Leonardo da Vinci",
            "fechaCreacion": 1503,
            "tecnica": "sfumato",
            "museo": "Louvre"
        })
        db_manager.crear_documento({
            "titulo": "La noche estrellada",
            "artista": "Van Gogh",
            "fechaCreacion": 1889,
            "tecnica": "oleo",
            "museo": "MoMA"
        })
        db_manager.confirmar_transaccion()

        # Leer todos los documentos
        db_manager.leer_documentos()

        # Actualizar un documento
        db_manager.iniciar_transaccion()
        db_manager.actualizar_documento(
            {"titulo": "La Gioconda"},
            {"fechaCreacion": 1600}
        )
        db_manager.confirmar_transaccion()
```

```
        )
        db_manager.confirmar_transaccion()

        # Eliminar un documento
        db_manager.iniciar_transaccion()
        db_manager.eliminar_documento({"titulo": "La noche estrellada"})
        db_manager.confirmar_transaccion()

    except Exception as e:
        logging.error(f"Error general: {e}")
        db_manager.revertir_transaccion()

    finally:
        db_manager.desconectar()
```

```
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: 1dam.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*

databasemanager_documental.log

```
1   2024-11-25 14:34:09,440 - INFO - Conectado a MongoDB: 1dam.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

```python
import logging
import transaction
from ZODB import DB, FileStorage
from persistent import Persistent

# Configuración de logging
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s",
    handlers=[
        logging.FileHandler("databasemanager_object.log"),  # Logs guardados en un archivo
        logging.StreamHandler(),  # Logs también en consola
    ]
)

class ObraArte(Persistent):
    """Clase que representa una obra de arte."""
    def __init__(self, titulo, artista, fechaCreacion, tecnica, museo):
        self.titulo = titulo
        self.artista = artista
        self.fechaCreacion = fechaCreacion
        self.tecnica = tecnica
        self.museo = museo

class DatabaseManagerObject:
    """Componente para gestionar bases de datos orientadas a objetos con ZODB."""
    def __init__(self, filepath="1dam.fs"):
        self.filepath = filepath
        self.db = None
        self.connection = None
        self.root = None
        self.transaccion_iniciada = False

    def conectar(self):
        """Conecta a la base de datos ZODB."""
        try:
            storage = FileStorage.FileStorage(self.filepath)
            self.db = DB(storage)
```

actividad5.py > DatabaseManagerObject > eliminar_obraArte

```python
class DatabaseManagerObject:

    def conectar(self):
        """Conecta a la base de datos ZODB."""
        try:
            storage = FileStorage.FileStorage(self.filepath)
            self.db = DB(storage)
            self.connection = self.db.open()
            self.root = self.connection.root()
            if "obrasArte" not in self.root:
                self.root["obrasArte"] = {}
                transaction.commit()
            logging.info("Conexión establecida con ZODB.")
        except Exception as e:
            logging.error(f"Error al conectar a ZODB: {e}")

    def desconectar(self):
        """Cierra la conexión a la base de datos."""
        try:
            if self.connection:
                self.connection.close()
                self.db.close()
            logging.info("Conexión a ZODB cerrada.")
        except Exception as e:
            logging.error(f"Error al cerrar la conexión a ZODB: {e}")

    def iniciar_transaccion(self):
        """Inicia una transacción."""
        try:
            transaction.begin()
            self.transaccion_iniciada = True
            logging.info("Transacción iniciada.")
        except Exception as e:
            logging.error(f"Error al iniciar la transacción: {e}")

    def confirmar_transaccion(self):
        """Confirma la transacción."""
        if self.transaccion_iniciada:
            try:
```

actividad5.py > DatabaseManagerObject > eliminar_obraArte

```python
class DatabaseManagerObject:
    def confirmar_transaccion(self):
        if self.transaccion_iniciada:
            try:
                transaction.commit()
                self.transaccion_iniciada = False
                logging.info("Transacción confirmada.")
            except Exception as e:
                logging.error(f"Error al confirmar la transacción: {e}")

    def revertir_transaccion(self):
        """Revierte la transacción."""
        if self.transaccion_iniciada:
            try:
                transaction.abort()
                self.transaccion_iniciada = False
                logging.info("Transacción revertida.")
            except Exception as e:
                logging.error(f"Error al revertir la transacción: {e}")

    def crear_obraArte(self, id, titulo, artista, fechaCreacion, tecnica, museo):
        """Crea y almacena una nueva obra de arte."""
        try:
            if id in self.root["obrasArte"]:
                raise ValueError(f"Ya existe una obra de arte con ID {id}.")
            self.root["obrasArte"][id] = ObraArte(titulo, artista, fechaCreacion, tecnica, museo)
            logging.info(f"Obra de arte con ID {id} creada exitosamente.")
        except Exception as e:
            logging.error(f"Error al crear la obra de arte con ID {id}: {e}")
            raise ValueError(f"Ya existe una obra de arte con ID {id}.")

    def leer_obrasArte(self):
        """Lee y muestra todas las obras de arte almacenadas."""
        try:
            obras = self.root["obrasArte"]
            for id, obra in obras.items():
                logging.info(
                    f"ID: {id}, Titulo: {obra.titulo}, Artista: {obra.artista}, "
```

actividad5.py > DatabaseManagerObject > eliminar_obraArte

```python
class DatabaseManagerObject:

    def leer_obrasArte(self):
        """Lee y muestra todas las obras de arte almacenadas."""
        try:
            obras = self.root["obrasArte"]
            for id, obra in obras.items():
                logging.info(
                    f"ID: {id}, Titulo: {obra.titulo}, Artista: {obra.artista}, "
                    f"Fecha de creacion: {obra.fechaCreacion}, Tecnica: {obra.tecnica}, Museo: {obra.museo}"
                )
            return obras
        except Exception as e:
            logging.error(f"Error al leer las obras de arte: {e}")

    def actualizar_obrasArte(self, id, titulo, artista, fechaCreacion, tecnica, museo):
        """Actualiza los atributos de una obra de arte."""
        try:
            obra = self.root["obrasArte"].get(id)
            if not obra:
                raise ValueError(f"No existe una obra de arte con ID {id}.")
            obra.titulo = titulo
            obra.artista = artista
            obra.fechaCreacion = fechaCreacion
            obra.tecnica = tecnica
            obra.museo = museo
            logging.info(f"Obra de arte con ID {id} actualizada exitosamente.")
        except Exception as e:
            logging.error(f"Error al actualizar la obra de arte con ID {id}: {e}")

    def eliminar_obraArte(self, id):
        """Elimina una obra de arte por su ID."""
        try:
            if id not in self.root["obrasArte"]:
                raise ValueError(f"No existe una obra de arte con ID {id}.")
            del self.root["obrasArte"][id]
            logging.info(f"Obra de arte con ID {id} eliminada exitosamente.")
        except Exception as e:
            logging.error(f"Error al eliminar la obra de arte con ID {id}: {e}")
```

actividad5.py > DatabaseManagerObject > eliminar_obraArte

```python
class DatabaseManagerObject:

    def eliminar_obraArte(self, id):
        """Elimina una obra de arte por su ID."""
        try:
            if id not in self.root["obrasArte"]:
                raise ValueError(f"No existe una obra de arte con ID {id}.")
            del self.root["obrasArte"][id]
            logging.info(f"Obra de arte con ID {id} eliminada exitosamente.")
        except Exception as e:
            logging.error(f"Error al eliminar la obra ed arte con ID {id}: {e}")
            raise ValueError(f"No existe una obra de arte con ID {id}.")


if __name__ == "__main__":
    manager = DatabaseManagerObject()
    manager.conectar()

    # Crear obras de arte con transacción
    manager.iniciar_transaccion()
    manager.crear_obraArte(1, "La Gioconda", "Leonardo da Vinci", "1503", "sfumato", "Louvre")
    manager.crear_obraArte(2, "Las Meninas", "Velazquez", "1656", "alla prima", "El Prado")
    manager.crear_obraArte(3, "El Guernica", "Pablo Picasso", "1937", "oleo", "Reina Sofia")
    manager.confirmar_transaccion()

    # Leer obras de arte
    manager.leer_obrasArte()

    #Insertar obre de arte con id que ya existe para revertir la transaccion
    manager.iniciar_transaccion()
    try:
        manager.crear_obraArte(2, "La noche estrellada", "Van Gohg", "1889", "oleo", "MoMA")
        manager.confirmar_transaccion()
    except ValueError as e:
        manager.revertir_transaccion()

    #Mostrar obras de arte
    manager.leer_obrasArte()
```

```python
# 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, Titulo: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1503, Tecnica: sfumato, Museo: Louvre
2024-12-05 10:21:12,502 - INFO - ID: 2, Titulo: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo: El Prado
2024-12-05 10:21:12,502 - INFO - ID: 3, Titulo: 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, Titulo: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1503, Tecnica: sfumato, Museo: Louvre
2024-12-05 10:21:12,502 - INFO - ID: 2, Titulo: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo: El Prado
2024-12-05 10:21:12,502 - INFO - ID: 3, Titulo: 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, Titulo: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1600, Tecnica: oleo, Museo: Louvre
2024-12-05 10:21:12,502 - INFO - ID: 2, Titulo: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo: El Prado
2024-12-05 10:21:12,502 - INFO - ID: 3, Titulo: 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, Titulo: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1600, Tecnica: oleo, Museo: Louvre
2024-12-05 10:21:12,502 - INFO - ID: 2, Titulo: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo: El Prado
2024-12-05 10:21:12,502 - INFO - ID: 3, Titulo: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: Reina Sofia
act5usuario@usuario:~/ACD/Tema6/actividad5$
```

databasemanager_object.log

```
2024-12-05 10:23:58,692 - INFO - Conexión establecida con ZODB.
2024-12-05 10:23:58,692 - INFO - Transacción iniciada.
2024-12-05 10:23:58,693 - INFO - Obra de arte con ID 1 creada exitosamente.
2024-12-05 10:23:58,693 - INFO - Obra de arte con ID 2 creada exitosamente.
2024-12-05 10:23:58,693 - INFO - Obra de arte con ID 3 creada exitosamente.
2024-12-05 10:23:58,693 - INFO - Transacción confirmada.
2024-12-05 10:23:58,693 - INFO - ID: 1, Titulo: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1503, Tecnica: sfumato, M
2024-12-05 10:23:58,693 - INFO - ID: 2, Titulo: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo:
2024-12-05 10:23:58,693 - INFO - ID: 3, Titulo: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: R
2024-12-05 10:23:58,693 - INFO - Transacción iniciada.
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.
2024-12-05 10:23:58,693 - INFO - Transacción revertida.
2024-12-05 10:23:58,693 - INFO - ID: 1, Titulo: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1503, Tecnica: sfumato, M
2024-12-05 10:23:58,693 - INFO - ID: 2, Titulo: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo:
2024-12-05 10:23:58,693 - INFO - ID: 3, Titulo: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: R
2024-12-05 10:23:58,693 - INFO - Transacción iniciada.
2024-12-05 10:23:58,693 - INFO - Obra de arte con ID 1 actualizada exitosamente.
2024-12-05 10:23:58,693 - INFO - Transacción confirmada.
2024-12-05 10:23:58,693 - INFO - ID: 1, Titulo: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1600, Tecnica: oleo, Muse
2024-12-05 10:23:58,693 - INFO - ID: 2, Titulo: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo:
2024-12-05 10:23:58,693 - INFO - ID: 3, Titulo: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: R
2024-12-05 10:23:58,693 - INFO - Transacción iniciada.
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.
2024-12-05 10:23:58,693 - INFO - Transacción revertida.
2024-12-05 10:23:58,693 - INFO - ID: 1, Titulo: La Gioconda, Artista: Leonardo da Vinci, Fecha de creacion: 1600, Tecnica: oleo, Muse
2024-12-05 10:23:58,693 - INFO - ID: 2, Titulo: Las Meninas, Artista: Velazquez, Fecha de creacion: 1656, Tecnica: alla prima, Museo:
2024-12-05 10:23:58,693 - INFO - ID: 3, Titulo: El Guernica, Artista: Pablo Picasso, Fecha de creacion: 1937, Tecnica: oleo, Museo: R
```