```
# Archivo: modules/db_manager.py

import json

import streamlit as st
from sqlalchemy import create_engine, text
from sqlalchemy.exc import SQLAlchemyError

# Obtener la URL de conexión desde secrets.toml
try:
    DATABASE_URL = st.secrets["DATABASE_URL"]
except Exception:
    DATABASE_URL = None


def get_engine():
    """Crea y retorna el motor de conexión SQLAlchemy."""
    if not DATABASE_URL:
        return None
    try:
        # echo=False para producción
        engine = create_engine(DATABASE_URL, echo=False)
        return engine
    except Exception as e:
        st.error(f"Error creando engine: {e}")
        return None


def init_db():
    """
    Inicializa la tabla de preferencias en PostgreSQL si no existe.
    """
    engine = get_engine()
    if engine is not None:
        try:
            with engine.connect() as conn:
                # Sintaxis PostgreSQL
                conn.execute(
                    text(
                        """
                    CREATE TABLE IF NOT EXISTS user_preferences (
                        id SERIAL PRIMARY KEY,
                        username TEXT NOT NULL,
                        preference_key TEXT NOT NULL,
                        preference_value TEXT,
                        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
                    );
                """
                    )
                )
                # Índice para búsquedas rápidas
                conn.execute(
                    text(
                        """
                    CREATE INDEX IF NOT EXISTS idx_user_pref ON user_preferences (username, preference_key);
                """
                    )
                )
                conn.commit()
        except SQLAlchemyError as e:
            st.error(f"Error inicializando DB: {e}")


def save_user_preference(username, key, value):
    """
    Guarda o actualiza una preferencia.
    """
    engine = get_engine()
    if engine is not None:
        try:
            # Serializar si es objeto complejo
            if isinstance(value, (dict, list)):
                val_str = json.dumps(value)
            else:
                val_str = str(value)

            with engine.connect() as conn:
                # Lógica UPSERT simple: Borrar e Insertar
                conn.execute(
                    text(
                        """
                    DELETE FROM user_preferences
                    WHERE username = :user AND preference_key = :key
                """
                    ),
                    {"user": username, "key": key},
                )

                conn.execute(
                    text(
                        """
                    INSERT INTO user_preferences (username, preference_key, preference_value)
                    VALUES (:user, :key, :val)
                """
                    ),
                    {"user": username, "key": key, "val": val_str},
                )

                conn.commit()
            return True
        except SQLAlchemyError as e:
            st.error(f"Error guardando preferencia: {e}")
            return False
    return False


def get_user_preference(username, key, default=None):
    """
    Recupera una preferencia específica.
    """
    engine = get_engine()
    if engine is not None:
        try:
            with engine.connect() as conn:
                result = conn.execute(
                    text(
                        """
                    SELECT preference_value FROM user_preferences
                    WHERE username = :user AND preference_key = :key
                    LIMIT 1
                """
                    ),
                    {"user": username, "key": key},
                ).fetchone()

                if result:
                    val = result[0]
                    # Intentar deserializar JSON
                    try:
                        return json.loads(val)
                    except:
                        return val
        except SQLAlchemyError:
            # st.error(f"Error leyendo DB: {e}") # Opcional: silenciar en producción
            pass

    return default
```