HydroFlow Manager v2.0

Manual Técnico

Versión del Software: 2.0 **Fecha de Publicación:** Noviembre 2025 **Empresa:** Artanda Ingeniería y Consultoría **Audiencia:** Administradores de sistemas, Personal técnico de IT

Tabla de Contenidos

1. Introducción

1.1 Propósito del Manual

Este manual técnico está dirigido a administradores de sistemas y personal de IT responsable de la instalación, configuración, mantenimiento y optimización de HydroFlow Manager v2.0.

No es un manual de usuario final. Para uso de la aplicación, consulte el **Manual de Usuario**.

1.2 Conocimientos Previos Requeridos

- Administración de sistemas Windows/Linux
- MySQL/MariaDB (creación de esquemas, usuarios, permisos)
- Conceptos de networking (TCP/IP, puertos, firewalls)
- Línea de comandos (CMD, PowerShell, Bash)
- Variables de entorno
- Conceptos básicos de Python (opcional pero recomendado)

1.3 Convenciones del Documento

```
# Comandos de terminal (Linux/Mac)
```

```
REM Comandos de terminal (Windows)
```

```
-- Comandos SQL
```

```
# Código Python
```

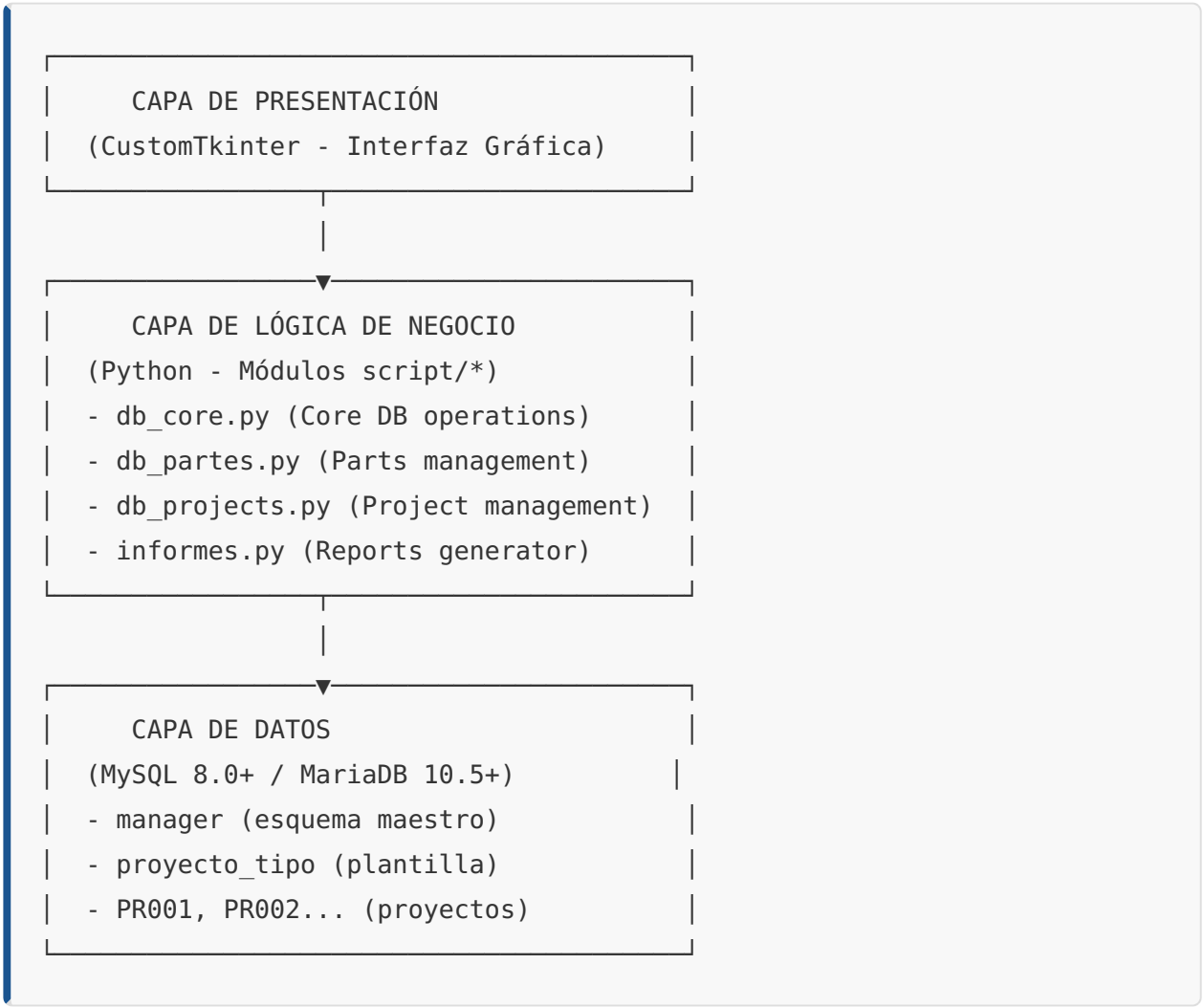
⚠️ **Advertencia:** Operaciones críticas que requieren precaución 💡 **Nota:** Información adicional importante 🔧 **Configuración:** Parámetros configurables

---

2. Arquitectura del Sistema

2.1 Vista General

HydroFlow Manager v2.0 sigue una arquitectura **cliente-servidor de tres capas**:



2.2 Componentes Principales

Componente	Tecnología	Función
Frontend	CustomTkinter 5.0+	Interfaz de usuario moderna
Backend	Python 3.8+	Lógica de negocio
Base de Datos	MySQL 8.0+	Persistencia de datos
Connection Layer	mysql-connector-python	Conexión con pooling
Export Engine	ReportLab, python-docx, openpyxl	Generación de documentos

2.3 Diagrama de Módulos

```
hydroflow/
├── main.py (Punto de entrada)
├── interface/ (46 módulos de UI)
│   ├── login_interfaz.py
│   ├── parts_manager_interfaz.py
│   ├── informes_interfaz.py
│   └── ...
├── script/ (Lógica de negocio)
│   ├── db_config.py (Configuración centralizada)
│   ├── db_connection.py (Pool de conexiones)
│   ├── db_core.py (Operaciones base)
│   ├── db_partes.py (Gestión de partes)
│   ├── db_projects.py (Gestión de proyectos)
│   ├── informes.py (Generador de informes)
│   └── modulo_db.py (API unificada)
└── resources/ (Recursos estáticos)
    ├── images/
    └── plantillas/
```

2.4 Flujo de Datos

Ejemplo: Crear un Parte

```
Usuario (UI)
    ↓
[parts_manager_interfaz.py] → Captura datos del formulario
    ↓
[db_partes.py::add_parte_with_code()] → Validación y lógica
    ↓
[db_connection.py::get_project_connection()] → Obtiene conexión del pool
    ↓
[MySQL] → INSERT INTO tbl_partes
    ↓
[db_connection.py] → Devuelve conexión al pool
    ↓
[UI] → Actualiza TreeView con nuevo parte
```

---

3. Instalación Avanzada

3.1 Instalación desde Código Fuente (Desarrollo)

```
# 1. Clonar repositorio
git clone https://github.com/empresa/hydroflow.git
cd hydroflow

# 2. Crear entorno virtual
python -m venv venv
source venv/bin/activate  # Linux/Mac
venv\Scripts\activate     # Windows

# 3. Instalar dependencias
pip install -r requirements.txt

# 4. Configurar .env
cp .env.example .env
nano .env  # Editar configuración

# 5. Ejecutar aplicación
python main.py
```

3.2 Compilación con PyInstaller

Para crear un ejecutable standalone:

```
# Instalación de PyInstaller
pip install pyinstaller

# Compilar (usando spec file)
pyinstaller HydroFlowManager.spec

# El ejecutable se genera en:
# dist/HydroFlowManager.exe
```

Contenido de HydroFlowManager.spec:

```
# -*- mode: python ; coding: utf-8 -*-

block_cipher = None

a = Analysis(
    ['main.py'],
    pathex=[],
    binaries=[],
    datas=[
        ('resources', 'resources'),
        ('informes_guardados', 'informes_guardados'),
    ],
    hiddenimports=[
        'mysql.connector',
        'tkcalendar',
        'customtkinter',
        'CTkMessagebox',
        'PIL',
        'matplotlib',
        'openpyxl',
        'docx',
        'reportlab',
        'pandas',
        'script.informes_storage'
    ],
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=block_cipher,
    noarchive=False,
)

pyz = PYZ(a.pure, a.zipped_data, cipher=block_cipher)

exe = EXE(
    pyz,
    a.scripts,
```

```
    a.binaries,
    a.zipfiles,
    a.datas,
    [],
    name='HydroFlowManager',
    debug=False,
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
    upx_exclude=[],
    runtime_tmpdir=None,
    console=False,
    disable_windowed_traceback=False,
    argv_emulation=False,
    target_arch=None,
    codesign_identity=None,
    entitlements_file=None,
    icon='source/logo.ico'
)
```

3.3 Instalación en Servidor (Producción)

Para instalación en servidor Synology u otro:

```
# 1. Transferir archivos al servidor
scp -r hydroflow/ admin@192.168.1.100:/volume1/applications/

# 2. SSH al servidor
ssh admin@192.168.1.100

# 3. Instalar Python y dependencias
sudo apt-get update
sudo apt-get install python3 python3-pip mysql-server

# 4. Configurar MySQL
sudo mysql_secure_installation

# 5. Crear base de datos
mysql -u root -p < backup/estructura_base.sql

# 6. Instalar dependencias Python
cd /volume1/applications/hydroflow
pip3 install -r requirements-prod.txt

# 7. Configurar .env
cp .env.produccion.template .env
nano .env

# 8. Ejecutar aplicación
python3 main.py
```

---

4. Configuración de Base de Datos

4.1 Instalación de MySQL

4.1.1 Windows

```
REM Descargar MySQL Community Server 8.0
REM https://dev.mysql.com/downloads/mysql/

REM Ejecutar instalador
mysql-installer-community-8.0.xx.msi

REM Configuración recomendada:
REM - Server Type: Development Computer
REM - Port: 3306
REM - Authentication: Strong Password Encryption
```

4.1.2 Linux (Ubuntu/Debian)

```
# Actualizar repositorios
sudo apt-get update

# Instalar MySQL
sudo apt-get install mysql-server

# Verificar instalación
sudo systemctl status mysql

# Configurar seguridad
sudo mysql_secure_installation
```

4.2 Crear Usuario de Aplicación

⚠️ **Importante:** NO usar `root` en producción.

```
-- Conectar como root
mysql -u root -p

-- Crear usuario para la aplicación
CREATE USER 'hydroflow_app'@'localhost'
IDENTIFIED BY 'contraseña_segura_aqui';

-- Para permitir conexiones remotas:
CREATE USER 'hydroflow_app'@'%'
IDENTIFIED BY 'contraseña_segura_aqui';

-- Otorgar permisos necesarios
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX
ON *.*
TO 'hydroflow_app'@'localhost';

-- Aplicar cambios
FLUSH PRIVILEGES;
```

4.3 Configuración de Puertos

Puerto estándar MySQL: `3306`

Si necesita cambiar el puerto:

```
# Editar /etc/mysql/mysql.conf.d/mysqld.cnf (Linux)
# O my.ini (Windows)

[mysqld]
port = 3307

# Reiniciar MySQL
sudo systemctl restart mysql
```

Actualizar `.env`:

```
DB_PORT=3307
```

4.4 Configuración de Conexiones Remotas

Para permitir conexiones desde otros equipos:

```
-- MySQL 8.0+
CREATE USER 'hydroflow_app'@'192.168.1.%'
IDENTIFIED BY 'contraseña';

GRANT ALL PRIVILEGES ON hydroflow.*
TO 'hydroflow_app'@'192.168.1.%';

FLUSH PRIVILEGES;
```

Configurar firewall:

```
# Linux (UFW)
sudo ufw allow 3306/tcp

# Windows
netsh advfirewall firewall add rule name="MySQL" dir=in action=allow protocol=TCP loca
```

Editar `my.cnf`:

```
[mysqld]
bind-address = 0.0.0.0  # Permitir todas las IPs
# O específicamente:
bind-address = 192.168.1.100
```

---

5. Arquitectura Multi-Esquema

5.1 Concepto Fundamental

HydroFlow Manager utiliza una arquitectura **multi-esquema** única:

- **Un esquema por cada proyecto** creado
- **Aislamiento total** de datos entre proyectos

- **Backup independiente** por proyecto
- **Permisos granulares** por proyecto

5.2 Esquemas del Sistema

5.2.1 Esquema `manager` (Maestro)

Contiene: - 👥 Usuarios del sistema (`tbl_usuarios`) - 📋 Registro de todos los proyectos (`tbl_proyectos`) - 📚 Catálogos de referencia compartidos - `tbl_catalogo` (partidas presupuestarias) - `tbl_familia` , `tbl_tipo_hidraulica` - `tbl_marca` , `tbl_caracteristica` - 🌐 Dimensiones compartidas - `dim_red` , `dim_tipo_trabajo` - `dim_provincias` , `dim_comarcas` - 🗺️ Listados geográficos - `list_municipios` (Todos los municipios) - `list_comarcas`

Backup necesario:

```
mysqldump -u root -p manager > manager_con_catalogos.sql
```

5.2.2 Esquema `proyecto_tipo` (Plantilla)

Contiene: - 📋 SOLO estructura de 79 tablas (vacías) - 🚫 NO contiene datos transaccionales - 🚫 NO contiene catálogos (se acceden por vistas desde manager)

¿Por qué es crítico?

Cada vez que un usuario crea un proyecto nuevo: 1. Se ejecuta `CREATE SCHEMA [codigo_proyecto]` 2. Se copian todas las tablas vacías desde `proyecto_tipo` 3. Se crean vistas que apuntan a catálogos en `manager` 4. Se copian datos geográficos específicos del proyecto

Backup necesario:

```
mysqldump -u root -p --no-data proyecto_tipo > proyecto_tipo_estructura.sql
```

5.2.3 Esquemas de Proyectos (PR001, PR002, etc.)

Contiene: - 📊 Datos transaccionales del proyecto: - Partes de trabajo (`tbl_partes`) - OTs (`tbl_ots`) - Presupuestos (`tbl_presupuesto` , `tbl_pres_precios`) - Certificaciones (`tbl_certificacion` , `tbl_cert_lineas`) - Inventario (`tbl_inv_elementos`) - 👁️ Vistas que apuntan a `manager` - `vw_catalogo_hidraulica` - `tbl_proyectos` (vista) - 🗺️ Municipios filtrados por provincia del proyecto

Ejemplo de estructura:

```
-- Ver todos los esquemas
SHOW DATABASES;

-- Resultado:
-- +--------------------+
-- | Database           |
-- +--------------------+
-- | manager            |
-- | proyecto_tipo      |
-- | PR001              |
-- | PR002              |
-- | CERT_2024_001      |
-- +--------------------+
```

5.3 Flujo de Creación de Proyecto Nuevo

```
-- Aplicación ejecuta automáticamente:

-- 1. Crear esquema
CREATE SCHEMA IF NOT EXISTS PR003;

-- 2. Copiar estructura desde proyecto_tipo
-- (79 tablas vacías)

-- 3. Crear vistas a manager
CREATE VIEW PR003.tbl_proyectos AS
SELECT * FROM manager.tbl_proyectos;

CREATE VIEW PR003.vw_catalogo AS
SELECT * FROM manager.tbl_catalogo;

-- 4. Copiar municipios de la provincia
INSERT INTO PR003.tbl_municipios
SELECT * FROM manager.list_municipios
WHERE provincia_id = 20;  -- Gipuzkoa

-- 5. Crear FKs y relaciones
ALTER TABLE PR003.tbl_partes
ADD CONSTRAINT fk_municipio
FOREIGN KEY (id_municipio)
REFERENCES PR003.tbl_municipios(id);
```

5.4 Implicaciones para el Administrador

✅ Ventajas

- ✅ Cada proyecto es independiente (backup/restore selectivo)
- ✅ Borrar un proyecto = `DROP SCHEMA` (no afecta otros)
- ✅ Permisos granulares (usuario solo accede a sus proyectos)
- ✅ Escalabilidad (proyectos en diferentes servidores)

⚠️ Consideraciones Críticas

- ⚠️ `proyecto_tipo` DEBE estar limpio (sin datos de test)

- ⚠️ Cambios en estructura afectan solo proyectos nuevos (no existentes)
- ⚠️ Actualizaciones de catálogos en `manager` afectan a TODOS los proyectos
- ⚠️ Backup debe incluir TODOS los esquemas (manager + proyecto_tipo + PRxxx)

5.5 Comandos Útiles para Verificación

```
-- Ver tamaño de cada esquema
SELECT
  TABLE_SCHEMA as 'Esquema',
  ROUND(SUM(DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024, 2) as 'Tamaño (MB)',
  COUNT(TABLE_NAME) as 'Num Tablas'
FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('information_schema', 'mysql', 'performance_schema', 'sys')
GROUP BY TABLE_SCHEMA
ORDER BY SUM(DATA_LENGTH + INDEX_LENGTH) DESC;

-- Verificar que proyecto_tipo esté vacío
SELECT
  TABLE_NAME,
  TABLE_ROWS
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'proyecto_tipo'
  AND TABLE_ROWS > 0;
-- Debe retornar 0 filas

-- Ver todos los proyectos activos
SELECT codigo, nombre, provincia, fecha_inicio
FROM manager.tbl_proyectos
WHERE activo = 1;

-- Listar todas las vistas en un proyecto
SELECT TABLE_NAME
FROM information_schema.VIEWS
WHERE TABLE_SCHEMA = 'PR001';
```

---

6. Sistema de Configuración

6.1 Configuración Centralizada (v2.0)

Novedad de v2.0: Sistema de configuración flexible sin valores hardcodeados.

6.1.1 Jerarquía de Configuración

La configuración se carga con la siguiente prioridad (de mayor a menor):

```
1. Variables de entorno (más alta)
   ↓
2. Archivo .env
   ↓
3. db_user_config.py (configuración persistente del usuario)
   ↓
4. Valores por defecto en db_config.py (más baja)
```

6.1.2 Archivo `.env`

```
# Configuración de Base de Datos
DB_HOST=localhost
DB_PORT=3306
DB_MANAGER_SCHEMA=manager
DB_EXAMPLE_SCHEMA=cert_dev

# Connection pooling
DB_USE_POOLING=true

# Directorios
INFORMES_DIR=./informes_guardados
EXPORT_DIR=./exportados
BACKUP_DIR=./backups

# Logging
LOG_LEVEL=INFO

# Backup automático
AUTO_BACKUP_ENABLED=false
BACKUP_FREQUENCY_HOURS=24
```

⚠️ **Seguridad:** El archivo `.env` está en `.gitignore` y NUNCA debe commitearse.

6.1.3 Variables de Entorno (Windows)

```
REM Establecer variables de entorno temporalmente
set DB_HOST=192.168.1.100
set DB_PORT=3307
python main.py

REM Establecer permanentemente (usuario actual)
setx DB_HOST "192.168.1.100"
setx DB_PORT "3307"
```

6.1.4 Variables de Entorno (Linux)

```
# Temporal (solo esta sesión)
export DB_HOST=192.168.1.100
export DB_PORT=3307
python3 main.py

# Permanente (agregar a ~/.bashrc o ~/.profile)
echo 'export DB_HOST=192.168.1.100' >> ~/.bashrc
echo 'export DB_PORT=3307' >> ~/.bashrc
source ~/.bashrc
```

6.2 Configuración Interactiva del Usuario

Ubicación: - Windows: `%APPDATA%/HydroFlow/connection.json` - Linux: `~/.config/hydroflow/connection.json`

```
# Ejecutar configuración interactiva
python -m script.db_user_config --configure

# Ver configuración actual
python -m script.db_user_config --show

# Resetear configuración
python -m script.db_user_config --reset

# Establecer un valor específico
python -m script.db_user_config --set host 192.168.1.100
python -m script.db_user_config --set port 3307
```

Contenido de connection.json:

```
{
    "host": "192.168.1.100",
    "port": 3307,
    "user": "hidroflow_app",
    "remember_user": true,
    "connection_type": "remote"
}
```

💡 **Nota:** La contraseña NUNCA se guarda por seguridad.

6.3 Connection Pooling

¿Qué es?

Un pool de conexiones reutiliza conexiones existentes en lugar de crear nuevas en cada operación.

Beneficios: - ⚡ Latencia reducida: ~1ms vs ~50ms - 🚀 Mejor rendimiento en operaciones múltiples - 💾 Uso eficiente de recursos

Configuración:

```
DB_USE_POOLING=true  # Activar pooling (recomendado)
```

Parámetros del pool (en código):

```
# script/db_connection.py
DEFAULT_POOL_SIZE = 5  # Conexiones por pool
```

Monitoreo:

```
from script.db_connection import ConnectionPoolManager

# Obtener estadísticas de pools
stats = ConnectionPoolManager.get_pool_stats()
print(stats)

# Salida:
# {
#     'root/manager': {
#         'pool_name': 'hydroflow_pool_root_manager',
#         'pool_size': 5
#     },
#     'root/PR001': {
#         'pool_name': 'hydroflow_pool_root_PR001',
#         'pool_size': 5
#     }
# }
```

---

7. Seguridad

7.1 Gestión de Credenciales

7.1.1 NO Almacenar Contraseñas en Código

❌ NUNCA hacer esto:

```
password = "mi_contraseña_123"  # ❌ MAL
```

✅ Hacer esto:

```
import os
password = os.getenv('DB_PASSWORD')  # ✅ BIEN
```

7.1.2 Protección del Archivo `.env`

```
# Linux: Establecer permisos restrictivos
chmod 600 .env

# Verificar que está en .gitignore
cat .gitignore | grep .env
# Debe mostrar: .env
```

Windows:

```
REM Establecer archivo como oculto
attrib +h .env

REM Configurar permisos (solo usuario actual)
icacls .env /inheritance:r
icacls .env /grant:r "%USERNAME%:F"
```

7.2 Seguridad de Base de Datos

7.2.1 Crear Usuario con Permisos Mínimos

```
-- Usuario de aplicación (NO administrador)
CREATE USER 'hydroflow_app'@'localhost'
IDENTIFIED BY 'contraseña_compleja_8FgH!23$';

-- Permisos SOLO en esquemas necesarios
GRANT SELECT, INSERT, UPDATE, DELETE
ON manager.*
TO 'hydroflow_app'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE
ON PR001.*
TO 'hydroflow_app'@'localhost';

-- NO otorgar:
-- - DROP, CREATE TABLE, ALTER (excepto para migraciones)
-- - SUPER, FILE, PROCESS
-- - Acceso a mysql.user

FLUSH PRIVILEGES;
```

7.2.2 Conexiones SSL/TLS (Avanzado)

Para conexiones cifradas:

```
# En script/db_connection.py
ssl_config = {
    'ssl_ca': '/path/to/ca.pem',
    'ssl_cert': '/path/to/client-cert.pem',
    'ssl_key': '/path/to/client-key.pem'
}

connection = mysql.connector.connect(
    host=host,
    port=port,
    user=user,
    password=password,
    ssl_ca=ssl_config['ssl_ca'],
    ssl_cert=ssl_config['ssl_cert'],
    ssl_key=ssl_config['ssl_key']
)
```

7.3 Seguridad de Aplicación

7.3.1 Validación de Entrada

Todas las entradas de usuario se validan:

```
# Ejemplo: Validación de código de parte
def validate_parte_code(code):
    """Valida formato de código de parte"""
    import re
    pattern = r'^P-\d{4}-\d{3}$'
    if not re.match(pattern, code):
        raise ValueError("Código de parte inválido")
    return code
```

7.3.2 Prevención de SQL Injection

Siempre usar consultas parametrizadas:

```
# ❌ MAL (vulnerable a SQL injection)
query = f"SELECT * FROM tbl_partes WHERE codigo = '{user_input}'"
cursor.execute(query)

# ✅ BIEN (seguro)
query = "SELECT * FROM tbl_partes WHERE codigo = %s"
cursor.execute(query, (user_input,))
```

7.4 Auditoría y Logs

Configurar nivel de log según entorno:

```
# Desarrollo
LOG_LEVEL=DEBUG

# Producción
LOG_LEVEL=INFO

# Solo errores críticos
LOG_LEVEL=ERROR
```

Ubicación de logs: - **Windows:** `%APPDATA%/HydroFlow/logs/` - **Linux:** `~/.config/hydroflow/logs/`

---

8. Backup y Restauración

8.1 Estrategia de Backup

8.1.1 Niveles de Backup

Tipo	Frecuencia	Contenido	Retención
Completo	Semanal	Todos los esquemas	4 semanas
Incremental	Diario	Solo cambios	7 días
Esquema manager	Mensual	Catálogos y usuarios	6 meses
Esquema proyecto	Por demanda	Proyecto específico	Permanente

8.1.2 Script de Backup Automático

```
#!/bin/bash
# backup_mysql_hydroflow.sh

# Configuración
BACKUP_DIR="/volume1/backups/hydroflow"
MYSQL_USER="root"
MYSQL_PASS="contraseña_root"
DATE=$(date +%Y%m%d_%H%M%S)

# Crear directorio de backup
mkdir -p $BACKUP_DIR/$DATE

# Backup esquema manager (con datos)
mysqldump -u$MYSQL_USER -p$MYSQL_PASS manager \
  > $BACKUP_DIR/$DATE/manager_$DATE.sql

# Backup proyecto_tipo (solo estructura)
mysqldump -u$MYSQL_USER -p$MYSQL_PASS --no-data proyecto_tipo \
  > $BACKUP_DIR/$DATE/proyecto_tipo_estructura_$DATE.sql

# Backup de cada proyecto
for SCHEMA in $(mysql -u$MYSQL_USER -p$MYSQL_PASS -e "SHOW DATABASES LIKE 'PR%'" -s --
do
  echo "Backing up $SCHEMA..."
  mysqldump -u$MYSQL_USER -p$MYSQL_PASS $SCHEMA \
    > $BACKUP_DIR/$DATE/${SCHEMA}_$DATE.sql
done

# Comprimir backups
tar -czf $BACKUP_DIR/backup_$DATE.tar.gz -C $BACKUP_DIR $DATE

# Eliminar archivos sin comprimir
rm -rf $BACKUP_DIR/$DATE

# Eliminar backups antiguos (>30 días)
find $BACKUP_DIR -name "backup_*.tar.gz" -mtime +30 -delete

echo "Backup completado: backup_$DATE.tar.gz"
```

8.1.3 Script de Backup Windows

```
@echo off
REM backup_hydroflow.bat

SET BACKUP_DIR=D:\Backups\HydroFlow
SET MYSQL_BIN="C:\Program Files\MySQL\MySQL Server 8.0\bin"
SET MYSQL_USER=root
SET MYSQL_PASS=contraseña

SET DATE=%date:~-4,4%%date:~-7,2%%date:~-10,2%_%time:~0,2%%time:~3,2%%time:~6,2%
SET DATE=%DATE: =0%

mkdir %BACKUP_DIR%\%DATE%

%MYSQL_BIN%\mysqldump -u%MYSQL_USER% -p%MYSQL_PASS% manager > %BACKUP_DIR%\%DATE%\mana

%MYSQL_BIN%\mysqldump -u%MYSQL_USER% -p%MYSQL_PASS% --no-data proyecto_tipo > %BACKUP_

echo Backup completado en %BACKUP_DIR%\%DATE%
```

8.2 Restauración

8.2.1 Restaurar Esquema Completo

```
# Restaurar manager
mysql -u root -p manager < backup/manager_20241120.sql

# Restaurar proyecto_tipo
mysql -u root -p proyecto_tipo < backup/proyecto_tipo_estructura.sql

# Restaurar proyecto específico
mysql -u root -p PR001 < backup/PR001_20241120.sql
```

8.2.2 Restaurar Solo Algunas Tablas

```
# Extraer una tabla específica del backup
sed -n '/^-- Table structure for table `tbl_partes`/,/^-- Table structure for table/p'
  backup/PR001.sql > tbl_partes_only.sql

# Restaurar solo esa tabla
mysql -u root -p PR001 < tbl_partes_only.sql
```

8.3 Script de Backup desde Python

El sistema incluye un script de backup:

```
# Backup de un esquema específico
python dev_tools/verificacion/crear_backup.py backup_manager manager "Backup diario"

# El archivo se guarda en:
# dev_tools/backup/backup_manager.sql
```

---

9. Optimización y Rendimiento

9.1 Índices de Base de Datos

9.1.1 Índices Recomendados

```
-- tbl_partes (tabla principal de partes)
CREATE INDEX idx_fecha ON tbl_partes(fecha);
CREATE INDEX idx_estado ON tbl_partes(estado);
CREATE INDEX idx_municipio ON tbl_partes(id_municipio);
CREATE INDEX idx_red ON tbl_partes(id_red);
CREATE INDEX idx_codigo ON tbl_partes(codigo);

-- tbl_certificacion
CREATE INDEX idx_fecha_cert ON tbl_certificacion(fecha_certificacion);
CREATE INDEX idx_proyecto ON tbl_certificacion(id_proyecto);

-- tbl_presupuesto
CREATE INDEX idx_codigo_partida ON tbl_presupuesto(codigo_partida);
CREATE INDEX idx_capitulo ON tbl_presupuesto(id_capitulo);
```

9.1.2 Verificar Índices Existentes

```
-- Ver todos los índices en una tabla
SHOW INDEX FROM tbl_partes;

-- Verificar uso de índices en una consulta
EXPLAIN SELECT * FROM tbl_partes WHERE fecha > '2024-01-01';
```

9.2 Optimización de Consultas

9.2.1 Consultas Lentas

Identificar consultas lentas:

```
-- Activar slow query log
SET GLOBAL slow_query_log = 'ON';
SET GLOBAL long_query_time = 2;  -- Consultas >2 segundos

-- Ver consultas lentas
SELECT * FROM mysql.slow_log
ORDER BY query_time DESC
LIMIT 10;
```

9.2.2 Caché de Consultas

```
-- Verificar tamaño de caché
SHOW VARIABLES LIKE 'query_cache_size';

-- Estadísticas de caché
SHOW STATUS LIKE 'Qcache%';
```

9.3 Optimización de Connection Pool

Ajustar tamaño del pool según carga:

```
# script/db_connection.py

# Para pocos usuarios (1-5)
DEFAULT_POOL_SIZE = 3

# Para usuarios moderados (5-20)
DEFAULT_POOL_SIZE = 5

# Para muchos usuarios (20+)
DEFAULT_POOL_SIZE = 10
```

9.4 Mantenimiento de Tablas

```
-- Analizar tablas (actualiza estadísticas)
ANALYZE TABLE tbl_partes;
ANALYZE TABLE tbl_presupuesto;

-- Optimizar tablas (desfragmenta)
OPTIMIZE TABLE tbl_partes;
OPTIMIZE TABLE tbl_certificacion;

-- Reparar tabla (si está corrupta)
REPAIR TABLE tbl_partes;

-- Verificar integridad
CHECK TABLE tbl_partes;
```

---

10. Mantenimiento

10.1 Tareas Diarias

- [ ] Verificar espacio en disco
- [ ] Revisar logs de errores
- [ ] Verificar backups automáticos

```
# Verificar espacio en disco
df -h | grep mysql

# Ver últimos errores de MySQL
tail -n 50 /var/log/mysql/error.log

# Verificar último backup
ls -lht /volume1/backups/hydroflow | head -5
```

10.2 Tareas Semanales

- [ ] Ejecutar backup completo

- [ ] Optimizar tablas grandes
- [ ] Revisar consultas lentas
- [ ] Actualizar estadísticas

```
-- Optimizar todas las tablas de un esquema
USE manager;
SELECT CONCAT('OPTIMIZE TABLE ', table_name, ';')
FROM information_schema.TABLES
WHERE table_schema = 'manager';
```

10.3 Tareas Mensuales

- [ ] Revisar crecimiento de base de datos
- [ ] Archivar datos antiguos
- [ ] Actualizar documentación
- [ ] Verificar integridad de backups

```
-- Ver crecimiento de esquemas
SELECT
  TABLE_SCHEMA,
  ROUND(SUM(DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024, 2) AS 'Size (MB)'
FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('information_schema', 'mysql', 'performance_schema')
GROUP BY TABLE_SCHEMA
ORDER BY SUM(DATA_LENGTH + INDEX_LENGTH) DESC;
```

10.4 Actualización de Software

10.4.1 Actualización de HydroFlow Manager

```
# 1. Backup completo ANTES de actualizar
python dev_tools/verificacion/crear_backup.py backup_pre_update manager

# 2. Descargar nueva versión
# wget https://releases.artanda.com/hydroflow/v2.1.0.zip

# 3. Detener aplicación
pkill -f "python.*main.py"

# 4. Backup de configuración actual
cp .env .env.backup

# 5. Descomprimir nueva versión
unzip v2.1.0.zip -d /tmp/hydroflow_new

# 6. Copiar configuración
cp .env /tmp/hydroflow_new/

# 7. Reemplazar archivos
rm -rf /opt/hydroflow_old
mv /opt/hydroflow /opt/hydroflow_old
mv /tmp/hydroflow_new /opt/hydroflow

# 8. Ejecutar migraciones de BD si es necesario
python /opt/hydroflow/migrations/migrate_v2.0_to_v2.1.py

# 9. Iniciar aplicación
python /opt/hydroflow/main.py
```

---

11. Troubleshooting Avanzado

11.1 Problemas de Conexión

11.1.1 Error: "Can't connect to MySQL server"

Diagnóstico:

```
# ¿MySQL está ejecutándose?
sudo systemctl status mysql

# ¿Puerto está abierto?
netstat -tulpn | grep 3306

# ¿Firewall bloqueando?
sudo ufw status
```

Soluciones:

```
# Iniciar MySQL
sudo systemctl start mysql

# Permitir puerto en firewall
sudo ufw allow 3306/tcp

# Verificar bind-address
sudo grep bind-address /etc/mysql/mysql.conf.d/mysqld.cnf
```

11.1.2 Error: "Access denied for user"

Diagnóstico:

```
-- Verificar usuario existe
SELECT User, Host FROM mysql.user WHERE User = 'hydroflow_app';

-- Verificar permisos
SHOW GRANTS FOR 'hydroflow_app'@'localhost';
```

Solución:

```
-- Recrear usuario con permisos correctos
DROP USER IF EXISTS 'hydroflow_app'@'localhost';
CREATE USER 'hydroflow_app'@'localhost' IDENTIFIED BY 'nueva_contraseña';
GRANT ALL PRIVILEGES ON manager.* TO 'hydroflow_app'@'localhost';
FLUSH PRIVILEGES;
```

11.2 Problemas de Rendimiento

11.2.1 Aplicación Lenta

Diagnóstico:

```
# Activar profiling en la aplicación
# Agregar al inicio de main.py
import cProfile
import pstats

profiler = cProfile.Profile()
profiler.enable()

# ... código de la aplicación ...

profiler.disable()
stats = pstats.Stats(profiler)
stats.sort_stats('cumulative')
stats.print_stats(20)  # Top 20 funciones más lentas
```

Verificar: - RAM disponible - Connection pooling activado - Índices en tablas grandes

11.2.2 MySQL Usando Mucha RAM

```
-- Verificar configuración de InnoDB buffer pool
SHOW VARIABLES LIKE 'innodb_buffer_pool_size';

-- Recomendación: 70% de RAM disponible para MySQL
-- Para servidor con 8GB RAM:
SET GLOBAL innodb_buffer_pool_size = 5368709120;  -- 5GB
```

11.3 Problemas de Datos

11.3.1 Datos Inconsistentes

```
-- Verificar integridad referencial
SELECT p.codigo, p.id_municipio
FROM tbl_partes p
LEFT JOIN tbl_municipios m ON p.id_municipio = m.id
WHERE m.id IS NULL;

-- Si hay filas, hay FKs rotas
```

11.3.2 Duplicados

```
-- Encontrar códigos de parte duplicados
SELECT codigo, COUNT(*) as cantidad
FROM tbl_partes
GROUP BY codigo
HAVING cantidad > 1;

-- Eliminar duplicados (conservar el más reciente)
DELETE p1 FROM tbl_partes p1
INNER JOIN tbl_partes p2
WHERE p1.codigo = p2.codigo
  AND p1.id < p2.id;
```

---

12. Integración y APIs

12.1 API de Base de Datos

El módulo `script/modulo_db.py` expone una API unificada:

```
from script.modulo_db import (
    # Autenticación
    login_db,

    # Proyectos
    get_schemas_db,
    create_schemas_db,
    add_project_item,

    # Partes
    add_parte_with_code,
    list_partes,
    update_parte,

    # Presupuestos
    add_presupuesto_item,
    list_presupuesto,

    # Certificaciones
    add_certificacion,
    list_certificaciones
)
```

12.2 Exportar Datos a JSON

```
# script/export_to_json.py
import json
from script.db_connection import get_project_connection

def export_partes_to_json(user, password, schema, output_file):
    """Exporta todos los partes a JSON"""
    with get_project_connection(user, password, schema) as conn:
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT * FROM tbl_partes")
        partes = cursor.fetchall()

    with open(output_file, 'w', encoding='utf-8') as f:
        json.dump(partes, f, indent=2, default=str)

    print(f"Exportados {len(partes)} partes a {output_file}")
```

12.3 Importar desde API Externa

```
# Ejemplo: Importar desde API REST
import requests
from script.modulo_db import add_parte_with_code

def sync_from_external_api(user, password, schema, api_url):
    """Sincroniza partes desde API externa"""
    # Obtener datos de API
    response = requests.get(f"{api_url}/partes")
    external_partes = response.json()

    # Importar cada parte
    for parte_data in external_partes:
        add_parte_with_code(
            user=user,
            password=password,
            schema=schema,
            codigo=parte_data['codigo'],
            fecha=parte_data['fecha'],
            # ... más campos
        )
```

---

13. Migración de Datos

13.1 Migración desde Access

El sistema incluye un script para migrar desde Microsoft Access:

```
# Windows (con pyodbc)
python dev_tools/importacion/importar_partes_access.py "APLICACION.accdb"

# Linux (exportar primero a CSV)
# 1. En Windows: Exportar tablas de Access a CSV
# 2. Transferir CSVs a servidor Linux
# 3. Importar
python dev_tools/importacion/importar_partes_access.py partes.csv csv
```

13.2 Migración desde Excel

```
# script/import_from_excel.py
import pandas as pd
from script.db_connection import get_project_connection

def import_partes_from_excel(user, password, schema, excel_file):
    """Importa partes desde Excel"""
    # Leer Excel
    df = pd.read_excel(excel_file)

    # Conectar a BD
    with get_project_connection(user, password, schema) as conn:
        cursor = conn.cursor()

        # Insertar cada fila
        for _, row in df.iterrows():
            sql = """
                INSERT INTO tbl_partes
                (codigo, fecha, municipio, descripcion)
                VALUES (%s, %s, %s, %s)
            """
            cursor.execute(sql, (
                row['codigo'],
                row['fecha'],
                row['municipio'],
                row['descripcion']
            ))

        conn.commit()

    print(f"Importados {len(df)} partes")
```

13.3 Migración entre Versiones

Para migrar de v1.x a v2.0:

```
# 1. Backup completo de v1.x
mysqldump -u root -p --all-databases > backup_v1_complete.sql

# 2. Ejecutar script de migración
python migrations/migrate_v1_to_v2.py

# 3. Verificar migración
python migrations/verify_v2_migration.py
```

---

14. Anexos Técnicos

14.1 Estructura de Tablas Principales

tbl_partes

Campo	Tipo	Descripción
id	INT	PK auto-increment
codigo	VARCHAR(50)	Código único del parte
fecha	DATE	Fecha de realización
id_red	INT	FK a dim_red
id_municipio	INT	FK a tbl_municipios
id_registro	INT	FK a tbl_inventario
descripcion	TEXT	Descripción del trabajo
estado	ENUM	Pendiente/EnCurso/Finalizado
coord_x	DECIMAL(10,2)	Coordenada X (UTM)
coord_y	DECIMAL(10,2)	Coordenada Y (UTM)

tbl_presupuesto

Campo	Tipo	Descripción
id	INT	PK
id_parte	INT	FK a tbl_partes
codigo_partida	VARCHAR(50)	Código de partida
cantidad	DECIMAL(10,2)	Cantidad
precio_unitario	DECIMAL(10,2)	Precio/unidad
importe	DECIMAL(10,2)	Cantidad × PU

14.2 Vistas Importantes

```
-- vw_partes_resumen: Vista resumen de partes
CREATE VIEW vw_partes_resumen AS
SELECT
    p.id,
    p.codigo,
    p.fecha,
    m.NAMEUNIT as municipio,
    r.nombre as red,
    p.estado,
    COALESCE(SUM(pre.importe), 0) as importe_total
FROM tbl_partes p
LEFT JOIN tbl_municipios m ON p.id_municipio = m.id
LEFT JOIN dim_red r ON p.id_red = r.id
LEFT JOIN tbl_presupuesto pre ON p.id = pre.id_parte
GROUP BY p.id;

-- vw_certificacion_resumen: Vista de certificaciones
CREATE VIEW vw_certificacion_resumen AS
SELECT
    c.id,
    c.codigo,
    c.fecha_certificacion,
    COUNT(cl.id) as num_lineas,
    SUM(cl.importe) as importe_total,
    p.nombre as proyecto
FROM tbl_certificacion c
LEFT JOIN tbl_cert_lineas cl ON c.id = cl.id_certificacion
LEFT JOIN tbl_proyectos p ON c.id_proyecto = p.id
GROUP BY c.id;
```

14.3 Procedimientos Almacenados

```
-- Procedimiento para calcular totales de certificación
DELIMITER $$

CREATE PROCEDURE sp_calcular_totales_certificacion(
    IN p_id_certificacion INT,
    OUT p_pem DECIMAL(10,2),
    OUT p_total DECIMAL(10,2)
)
BEGIN
    DECLARE v_gg DECIMAL(5,2);
    DECLARE v_bi DECIMAL(5,2);
    DECLARE v_iva DECIMAL(5,2);

    -- Obtener parámetros económicos del proyecto
    SELECT gastos_generales, beneficio_industrial, iva
    INTO v_gg, v_bi, v_iva
    FROM tbl_proy_presupuesto pp
    INNER JOIN tbl_certificacion c ON c.id_proyecto = pp.id_proyecto
    WHERE c.id = p_id_certificacion;

    -- Calcular PEM
    SELECT SUM(importe) INTO p_pem
    FROM tbl_cert_lineas
    WHERE id_certificacion = p_id_certificacion;

    -- Calcular total
    SET p_total = p_pem * (1 + v_gg/100) * (1 + v_bi/100) * (1 + v_iva/100);
END$$

DELIMITER ;
```

14.4 Comandos Útiles de MySQL

```
-- Ver procesos activos
SHOW PROCESSLIST;

-- Matar proceso bloqueado
KILL 123;

-- Ver variables de sistema
SHOW VARIABLES;

-- Ver estado del servidor
SHOW STATUS;

-- Ver motores de almacenamiento disponibles
SHOW ENGINES;

-- Ver tamaño de todas las tablas
SELECT
    TABLE_NAME,
    ROUND(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) AS 'Size (MB)'
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'manager'
ORDER BY (DATA_LENGTH + INDEX_LENGTH) DESC;

-- Ver conexiones abiertas
SELECT
    User,
    Host,
    db,
    Command,
    Time,
    State,
    Info
FROM information_schema.PROCESSLIST
WHERE User != 'system user';
```

Fin del Manual Técnico

Para más información, contacte con el departamento de IT de Artanda Ingeniería.

Versión del documento: 2.0 **Última actualización:** Noviembre 2025 **Autor:** Equipo técnico de Artanda