Apéndices

Apéndice A. *Fotografías de herramientas de corte*

Figura de plaquita del código 982510194 para procesos de torneado.

Figura de plaquita del código TPKR 2204 para procesos de torneado.

Figura de plaquita del código TPKR 2204 para procesos de torneado.

Figura de plaquita del código TPKR 1603 para procesos de torneado.

Figura de plaquita del código TPKR 1603 PDRS para procesos de torneado.

Figura de plaquita del código APKT 1604 para procesos de fresado.

Apéndice B. *Factura de compra de herramientas de corte*

BITPAL FYF LTDA
N.I.T: 900.270.220-6
I. V. A. Régimen Común
Tarifa ICA 11.04 x 1000
Act. Económica 4669
CLL 36 sur No. 68 I-36
TELEFAX : 60 - 1 - 2389122
7240282

30/05/2024 08:52:16

Factura Electrónica De Venta No
FVE 2456
FECHA FACTURA
30-may.-24

Nombre del Cliente	MECANIZADOS INDUSTRIALES PRECISION LIMITADA		
NIT o C.C.	800251415 1	**Fecha de Vencimiento**	29-jul.-24
CIUDAD	Duitama	**Orden de Compra**	
TELEFONO	7619067	**Vendedor**	JOSE PLINIO RAMOS RAMOS
Dirección Oficina	CLL 21B 40 102		

Item	*Código*	*Descripción del Producto*	*Cant*	*Precio Unit.*	*%Dto.*	*Total*
1	111060968	INSERTO DE FRESADO TPKN 2204 PDSR PH6920	10	29.750	0,00%	297.500
2	1121328V5	INSERTO DE TORNEADO DNMG 150608-MR PHG140	10	35.000	0,00%	350.000

Total líneas o ítems: 2

FAVOR CONSIGNAR EN LA CTA CTE BANCOLOMBIA # 10848954199

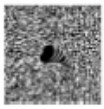
DIAN - RESOLUCION DE FAC. No.18764050405159 AUTORIZA DEL FVE 2001 AL FVE 3000 DEL 14/06/2023 VIG. 12 MESES

Esta Factura se asimila es sus efectos juridicos y legales a una letra de cambio conforme a los Art. 621, 773 y 774 del C.C.
No se aceptan devoluciones pasado 3 (tres) dias habiles despues de recibida la mercancia

SON: SETECIENTOS SETENTA MIL QUINIENTOS VEINTICINCO PESOS M/CTE

Descuento	0
Subtotal	647.500
I.V.A	123.025
Rete Fuente	0
Rete IVA	0
Rete Ica	0
Total	770.525

Atentamente



Firma del Cliente

Recibida y aceptada Nit. O C. 800251415 1

CUFE: f71ff595cc65e872a7fcdbd89d11f43df267fba3a496625db9e5c739f8b4949a41d1b6e38ba32282e2c03ebcae478ec--Expedición:30/05/2024 08:56 AM
Fabricante y Proveedor Tecnológico: World Office Colombia SAS NIT 900534356-3 Software: World Office (wo_2)

Nota: Orden de compra del 30 de mayo por insertos TPKN y DNMG para fresado y torneado respectivamente.

bitpal f y f ltda

BITPAL FYF LTDA
N.I.T: 900.270.220-6
I. V. A. Régimen Común
Tarifa ICA 11.04 x 1000
Act. Económica 4662
CLL 36 sur No. 68 I-36
TELEFAX : 60 - 1 - 2389122
7240282

21/03/2023 08:30:07

Factura Electrónica De Venta No
FVE 1889
FECHA FACTURA
21-mar-23

Nombre del Cliente	MECANIZADOS INDUSTRIALES PRECISION LIMITADA		
NIT o C.C.	800251415 1	Fecha de Vencimiento	20-may-23
CIUDAD	Duitama	Orden de Compra	
TELEFONO	7619067	Vendedor	JOSE PLINIO RAMOS RAMOS
Dirección Oficina	CLL 21B 40 102		

Item	Código	Descripción del Producto	Cant	Precio Unit.	%Dto.	Total
1	1120749L8	INSERTO DE TORNEADO TPMR 160308-13 PH5125	10	24.030	0,00%	240.300
2	112118041	INSERTO DE TORNEADO CNMG 120404 MR PH2135	10	30.240	0,00%	302.400
3	112118156	INSERTO DE TORNEADO SNMG 120408-ST PH6215	10	20.000	0,00%	200.000

Total lineas o items: 3

FAVOR CONSIGNAR EN LA CTA CTE BANCOLOMBIA # 10648954199

DIAN - RESOLUCION DE FAC. No.18764035982002 AUTORIZA DEL FVE 1622 AL FVE 2000 DEL 12/09/2022 VIG. 12 MESES

Esta Factura se asimila es sus efectos juridicos y legales a una letra de cambio conforme a los Art. 621, 773 y 774 del C.C.
No se aceptan devoluciones pasado 3 (tres) dias habiles despues de recibida la mercancia

SON: OCHOCIENTOS OCHENTA Y TRES MIL OCHOCIENTOS TRECE PESOS M/CTE

Descuento	0
Subtotal	742.700
I.V.A	141.113
Rete Fuente	0
Rete IVA	0
Rete Ica	0
Total	883.813

Atentamente

Firma del Cliente

Recibida y aceptada Nit. O C. 800251415 1

CUFE: b987c09067ef9623c157042b805ac51ac5d8b3371789bae77977f6396da0b75f6d4b250c809c7a4a8ac789bfe0844e58~Expedición:21/03/2023 08:34 AM
Fabricante y Proveedor Tecnológico: World Office Colombia SAS NIT 900534356-3 Software: World Office (wo_2)

Nota: Orden de compra del 21 de marzo por insertos TPMR, CNMG, SNMG para procesos de torneado.

BITPAL FYF LTDA
N.I.T: 900.270.220-6
I. V. A. Régimen Común
Tarifa ICA 11.04 x 1000
Act. Económica 4662
CLL 36 sur No. 68 I-36
TELEFAX : 60 - 1 - 2389122
7240282

07/07/2023 08:44:18

Factura Electrónica De Venta No
FVE 2024
FECHA FACTURA
07-jul-23

Nombre del Cliente	MECANIZADOS INDUSTRIALES PRECISION LIMITADA		
NIT o C.C.	800251415 1	**Fecha de Vencimiento**	05-sept-23
CIUDAD	Duitama	**Orden de Compra**	
TELEFONO	7619067	**Vendedor**	JOSE PLINIO RAMOS RAMOS
Dirección Oficina	CLL 21B 40 102		

Item	*Código*	*Descripción del Producto*	*Cant*	*Precio Unit.*	*%Dto.*	*Total*
1	1121254V5	INSERTO DE TORNEADO DNMG 150608-HR PHG140	10	44.460	0,00%	444.600
2	1121201G4	INSERTO DE TORNEADO CNMG 120408-SS PH7920	10	30.240	0,00%	302.400
3	111215968	INSERTO DE FRESADO APKT 160408 PDER-X1 PH6920	10	39.240	0,00%	392.400
4	1121193V5	INSERTO DE TORNEADO CNMG 120408-HR PHG140	10	30.240	0,00%	302.400
5	212127000	PLACA BASE CC120301 (PCLNR/L M12)	3	55.170	0,00%	165.510

Total lineas o items: 5

FAVOR CONSIGNAR EN LA CTA CTE BANCOLOMBIA # 10848954199

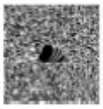
DIAN - RESOLUCION DE FAC. No.18764050405159 AUTORIZA DEL FVE 2001 AL FVE 3000 DEL 14/06/2023 VIG. 12 MESES

Esta Factura se asimila es sus efectos juridicos y legales a una letra de cambio conforme a los Art. 621, 773 y 774 del C.C.
No se aceptan devoluciones pasado 3 (tres) dias habiles despues de recibida la mercancia

SON: UN MILLON OCHOCIENTOS SETENTA Y DOS MIL QUINIENTOS DIECISEIS PESOS M/CTE

Descuento	0
Subtotal	1.607.310
I.V.A	305.389
Rete Fuente	40.183
Rete IVA	0
Rete Ica	0
Total	1.872.516

Atentamente



Firma del Cliente

Recibida y aceptada Nit. O C. 800251415 1

CUFE: 4410f523947dfb8d7c118adccdd247d8c28ba2b7b9a3a2bf2af0fc69297cfa12b53cb258b339ad0a01e80a829a2320f7--Expedición:07/07/2023 08:54 AM
Fabricante y Proveedor Tecnológico: World Office Colombia SAS NIT 900534356-3 Software: World Office (wo_2)

Nota: Orden de compra del 5 de septiembre por insertos DNMG, CNMG para torneado y APKT para procesos de fresado.

bitpal f y f ltda

BITPAL FYF LTDA
N.I.T: 900.270.220-6
I. V. A. Régimen Común
Tarifa ICA 11.04 x 1000
Act. Económica 4669
CLL 36 sur No. 68 I-36
TELEFAX : 60 - 1 - 2389122
7240282

16/02/2024 08:01:06

Factura Electrónica De Venta No
FVE 2309
FECHA FACTURA
16-feb.-24

Nombre del Cliente	MECANIZADOS INDUSTRIALES PRECISION LIMITADA		
NIT o C.C.	800251415 1	Fecha de Vencimiento	16-abr.-24
CIUDAD	Duitama	Orden de Compra	
TELEFONO	7619067	Vendedor	JOSE PLINIO RAMOS RAMOS
Dirección Oficina	CLL 21B 40 102		

Item	Código	Descripción del Producto	Cant	Precio Unit.	%Dto.	Total
1	188043168	INSERTO DE ROSCADO EXTERNO 16 ER G60 PH6920	10	50.235	0,00%	502.350
2	181034700	FRESA 050A09945-04-20-U022040(SEHT13T3)	1	310.000	0,00%	310.000
3	111093154	INSERTO DE FRESADO SEHT 13T3 AGTN PH6910	10	29.000	0,00%	290.000
4	111093154	INSERTO DE FRESADO SEHT 13T3 AGTN PH6910	10	29.000	0,00%	290.000

Total líneas o ítems: 4

FAVOR CONSIGNAR EN LA CTA CTE BANCOLOMBIA # 10848954199

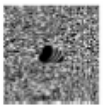
DIAN - RESOLUCION DE FAC. No.18764050405159 AUTORIZA DEL FVE 2001 AL FVE 3000 DEL 14/06/2023 VIG. 12 MESES

Esta Factura se asimila es sus efectos juridicos y legales a una letra de cambio conforme a los Art. 621, 773 y 774 del C.C.
No se aceptan devoluciones pasado 3 (tres) dias habiles despues de recibida la mercancia

SON: UN MILLON SEISCIENTOS VEINTIDOS MIL OCHENTA Y OCHO PESOS M/CTE

Descuento	0
Subtotal	1.392.350
I.V.A	264.547
Rete Fuente	34.809
Rete IVA	0
Rete Ica	0
Total	1.622.088

Atentamente



Firma del Cliente

Recibida y aceptada Nit. O C. 800251415 1

CUFE: 1b02e5c8382472680dafbe53f5caba8f2ff33505e65347473e7052c83e20a86ecd2f4925aa1400bf2525bce88e33d14--Expedición:16/02/2024 08:16 AM
Fabricante y Proveedor Tecnológico: World Office Colombia SAS NIT 900534356-3 Software: World Office (wo_2)

Nota: Orden de compra del 16 de abril por insertos SEHT para procesos de fresado.

Apéndice C. *Código del sistema*

```
import os
import sqlite3

DB_PATH = os.path.join(os.path.dirname(__file__), "data", "herramientas.db")

print("✅ CHECK_DB arrancó")
print("📁 BD usada:", DB_PATH)

if not os.path.exists(DB_PATH):
    print("❌ No existe la BD en esa ruta. Revisa que exista /data/herramientas.db")
    raise SystemExit

con = sqlite3.connect(DB_PATH)
cur = con.cursor()

cur.execute("SELECT name FROM sqlite_master WHERE type='table' ORDER BY
name;")
tables = [t[0] for t in cur.fetchall()]
print("📌 Tablas encontradas:", tables)

if "herramientas" not in tables:
    print("❌ No existe la tabla 'herramientas'. Entonces nunca se creó o estás en otra BD.")
else:
    cur.execute("SELECT COUNT(*) FROM herramientas;")
    total = cur.fetchone()[0]
    print("🔢 Total herramientas:", total)

    cur.execute("SELECT id, codigo, nombre FROM herramientas ORDER BY id DESC
LIMIT 10;")
    ultimas = cur.fetchall()
    print("📄 Últimas 10 herramientas:")
    for row in ultimas:
        print(" -", row)

con.close()
print("✅ CHECK_DB terminó bien")
```

```
import os
```

```
import sqlite3

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
DATA_DIR = os.path.join(BASE_DIR, "data")
DB_PATH = os.path.join(DATA_DIR, "herramientas.db")


def get_conn():
    os.makedirs(DATA_DIR, exist_ok=True)
    con = sqlite3.connect(DB_PATH, timeout=15)
    con.execute("PRAGMA foreign_keys = ON;")
    con.execute("PRAGMA busy_timeout = 15000;")
    con.execute("PRAGMA journal_mode = WAL;")
    return con


def _column_exists(cur, table: str, column: str) -> bool:
    cur.execute(f"PRAGMA table_info({table})")
    cols = [r[1] for r in cur.fetchall()]
    return column in cols


def _fk_points_to(cur, table: str, target: str) -> bool:
    cur.execute(f"PRAGMA foreign_key_list({table})")
    for row in cur.fetchall():
        if str(row[2]).strip().lower() == target.strip().lower():
            return True
    return False


def init_db():
    con = get_conn()
    cur = con.cursor()

    cur.execute("""
    CREATE TABLE IF NOT EXISTS maquinas (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nombre TEXT NOT NULL UNIQUE,
        imagen TEXT
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS materiales (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nombre TEXT NOT NULL UNIQUE,
        imagen TEXT
```

```
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS material_piezas (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        material_padre_id INTEGER NOT NULL,
        codigo TEXT NOT NULL UNIQUE,
        nombre TEXT NOT NULL,
        iso_min INTEGER NOT NULL DEFAULT 0,
        iso_max INTEGER NOT NULL DEFAULT 0,
        hb_min REAL NOT NULL DEFAULT 0,
        hb_max REAL NOT NULL DEFAULT 0,
        imagen TEXT,
        FOREIGN KEY(material_padre_id) REFERENCES materiales(id) ON DELETE
CASCADE
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS niveles (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nombre TEXT NOT NULL UNIQUE,
        imagen TEXT
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS usuarios (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nombre TEXT NOT NULL UNIQUE
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS operaciones (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        maquina_id INTEGER NOT NULL,
        nombre TEXT NOT NULL,
        imagen TEXT,
        FOREIGN KEY(maquina_id) REFERENCES maquinas(id) ON DELETE
CASCADE,
        UNIQUE(maquina_id, nombre)
    )
    """)

    cur.execute("""
```

```
    CREATE TABLE IF NOT EXISTS herramientas (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        codigo TEXT NOT NULL,
        nombre TEXT NOT NULL,
        maquina_id INTEGER NOT NULL,
        nivel_id INTEGER NOT NULL,
        operacion_id INTEGER NOT NULL,
        stock INTEGER NOT NULL DEFAULT 0,
        stock_nuevo INTEGER NOT NULL DEFAULT 0,
        stock_usado INTEGER NOT NULL DEFAULT 0,
        ubicacion TEXT,
        imagen TEXT,
        created_at TEXT DEFAULT (datetime('now')),
        ap_min REAL NOT NULL DEFAULT 0.0,
        ap_max REAL NOT NULL DEFAULT 0.0,
        fn_min REAL NOT NULL DEFAULT 0.0,
        fn_max REAL NOT NULL DEFAULT 0.0,
        vc_min REAL NOT NULL DEFAULT 0.0,
        vc_max REAL NOT NULL DEFAULT 0.0,
        fn_1 REAL NOT NULL DEFAULT 0.0,
        fn_2 REAL NOT NULL DEFAULT 0.0,
        fn_3 REAL NOT NULL DEFAULT 0.0,
        vc1_min REAL NOT NULL DEFAULT 0.0,
        vc1_max REAL NOT NULL DEFAULT 0.0,
        vc2_min REAL NOT NULL DEFAULT 0.0,
        vc2_max REAL NOT NULL DEFAULT 0.0,
        vc3_min REAL NOT NULL DEFAULT 0.0,
        vc3_max REAL NOT NULL DEFAULT 0.0,
        FOREIGN KEY(maquina_id) REFERENCES maquinas(id) ON DELETE
RESTRICT,
        FOREIGN KEY(nivel_id) REFERENCES niveles(id) ON DELETE RESTRICT,
        FOREIGN KEY(operacion_id) REFERENCES operaciones(id) ON DELETE
RESTRICT
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS portaherramientas (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nombre TEXT NOT NULL UNIQUE,
        imagen TEXT
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS portaherramienta_herramientas (
        portaherramienta_id INTEGER NOT NULL,
```

```
        herramienta_id INTEGER NOT NULL,
        PRIMARY KEY (portaherramienta_id, herramienta_id),
        FOREIGN KEY(portaherramienta_id) REFERENCES portaherramientas(id) ON
DELETE CASCADE,
        FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
CASCADE
    )
    """)

    migrations = [
        ("ap_min", "REAL NOT NULL DEFAULT 0.0"),
        ("ap_max", "REAL NOT NULL DEFAULT 0.0"),
        ("fn_min", "REAL NOT NULL DEFAULT 0.0"),
        ("fn_max", "REAL NOT NULL DEFAULT 0.0"),
        ("vc_min", "REAL NOT NULL DEFAULT 0.0"),
        ("vc_max", "REAL NOT NULL DEFAULT 0.0"),
        ("fn_1", "REAL NOT NULL DEFAULT 0.0"),
        ("fn_2", "REAL NOT NULL DEFAULT 0.0"),
        ("fn_3", "REAL NOT NULL DEFAULT 0.0"),
        ("vc1_min", "REAL NOT NULL DEFAULT 0.0"),
        ("vc1_max", "REAL NOT NULL DEFAULT 0.0"),
        ("vc2_min", "REAL NOT NULL DEFAULT 0.0"),
        ("vc2_max", "REAL NOT NULL DEFAULT 0.0"),
        ("vc3_min", "REAL NOT NULL DEFAULT 0.0"),
        ("vc3_max", "REAL NOT NULL DEFAULT 0.0"),
        ("stock_nuevo", "INTEGER NOT NULL DEFAULT 0"),
        ("stock_usado", "INTEGER NOT NULL DEFAULT 0"),
    ]
    for col, ddl in migrations:
        if not _column_exists(cur, "herramientas", col):
            cur.execute(f"ALTER TABLE herramientas ADD COLUMN {col} {ddl}")

    image_migrations = [
        ("maquinas", "imagen", "TEXT"),
        ("materiales", "imagen", "TEXT"),
        ("material_piezas", "imagen", "TEXT"),
        ("niveles", "imagen", "TEXT"),
        ("operaciones", "imagen", "TEXT"),
        ("herramientas", "imagen", "TEXT"),
        ("portaherramientas", "imagen", "TEXT"),
    ]
    for table, col, ddl in image_migrations:
        if not _column_exists(cur, table, col):
            cur.execute(f"ALTER TABLE {table} ADD COLUMN {col} {ddl}")

    if not _column_exists(cur, "material_piezas", "hb"):
```

```
        cur.execute("ALTER TABLE material_piezas ADD COLUMN hb REAL NOT
NULL DEFAULT 0")
    if not _column_exists(cur, "material_piezas", "hb_min"):
        cur.execute("ALTER TABLE material_piezas ADD COLUMN hb_min REAL NOT
NULL DEFAULT 0")
    if not _column_exists(cur, "material_piezas", "hb_max"):
        cur.execute("ALTER TABLE material_piezas ADD COLUMN hb_max REAL NOT
NULL DEFAULT 0")
    if not _column_exists(cur, "material_piezas", "iso_min"):
        cur.execute("ALTER TABLE material_piezas ADD COLUMN iso_min INTEGER
NOT NULL DEFAULT 0")
    if not _column_exists(cur, "material_piezas", "iso_max"):
        cur.execute("ALTER TABLE material_piezas ADD COLUMN iso_max INTEGER
NOT NULL DEFAULT 0")
    if not _column_exists(cur, "material_piezas", "codigo"):
        cur.execute("ALTER TABLE material_piezas ADD COLUMN codigo TEXT")

    cur.execute("SELECT sql FROM sqlite_master WHERE type='table' AND
name='material_piezas'")
    row_sql = cur.fetchone()
    table_sql = (row_sql[0] if row_sql and row_sql[0] else "").replace(" ", "").lower()
    if "unique(material_padre_id,nombre)" in table_sql:
        cur.execute("PRAGMA foreign_keys = OFF")
        cur.execute("""
            CREATE TABLE IF NOT EXISTS material_piezas_new (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                material_padre_id INTEGER NOT NULL,
                codigo TEXT NOT NULL UNIQUE,
                nombre TEXT NOT NULL,
                iso_min INTEGER NOT NULL DEFAULT 0,
                iso_max INTEGER NOT NULL DEFAULT 0,
                hb_min REAL NOT NULL DEFAULT 0,
                hb_max REAL NOT NULL DEFAULT 0,
                imagen TEXT,
                hb REAL NOT NULL DEFAULT 0,
                FOREIGN KEY(material_padre_id) REFERENCES materiales(id) ON DELETE
CASCADE
            )
        """)
        cur.execute("""
            INSERT INTO material_piezas_new
                (id, material_padre_id, codigo, nombre, iso_min, iso_max, hb_min, hb_max,
imagen, hb)
            SELECT
                id,
                material_padre_id,
                COALESCE(codigo, '),
```

```
            nombre,
            COALESCE(iso_min, 0),
            COALESCE(iso_max, 0),
            COALESCE(hb_min, 0),
            COALESCE(hb_max, 0),
            imagen,
            COALESCE(hb, 0)
          FROM material_piezas
        """)
        cur.execute("ALTER TABLE material_piezas RENAME TO material_piezas_old")
        cur.execute("ALTER TABLE material_piezas_new RENAME TO material_piezas")
        cur.execute("DROP TABLE material_piezas_old")
        cur.execute("PRAGMA foreign_keys = ON")

    cur.execute("SELECT id, COALESCE(codigo,") FROM material_piezas ORDER BY
id")
    rows_mp = cur.fetchall()
    seen_codes = set()
    for mp_id, raw_code in rows_mp:
        code = (raw_code or "").strip().upper()
        if not code or code in seen_codes:
            code = f"MP{mp_id}"
            while code in seen_codes:
                code = f"MP{mp_id}_{len(seen_codes)+1}"
            cur.execute("UPDATE material_piezas SET codigo=? WHERE id=?", (code,
mp_id))
        seen_codes.add(code)

    cur.execute("CREATE UNIQUE INDEX IF NOT EXISTS ux_material_piezas_codigo
ON material_piezas(codigo)")

    cur.execute("""
        UPDATE material_piezas
        SET hb_min = COALESCE(hb, 0),
            hb_max = COALESCE(hb, 0)
        WHERE COALESCE(hb, 0) > 0
          AND COALESCE(hb_min, 0) = 0
          AND COALESCE(hb_max, 0) = 0
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS herramienta_materiales (
        herramienta_id INTEGER NOT NULL,
        material_id INTEGER NOT NULL,
        PRIMARY KEY (herramienta_id, material_id),
        FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
CASCADE,
```

```
        FOREIGN KEY(material_id) REFERENCES materiales(id) ON DELETE
RESTRICT
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS herramienta_material_parametros (
        herramienta_id INTEGER NOT NULL,
        material_id INTEGER NOT NULL,
        fn_1 REAL NOT NULL DEFAULT 0.0,
        vc1_min REAL NOT NULL DEFAULT 0.0,
        vc1_max REAL NOT NULL DEFAULT 0.0,
        fn_2 REAL NOT NULL DEFAULT 0.0,
        vc2_min REAL NOT NULL DEFAULT 0.0,
        vc2_max REAL NOT NULL DEFAULT 0.0,
        fn_3 REAL NOT NULL DEFAULT 0.0,
        vc3_min REAL NOT NULL DEFAULT 0.0,
        vc3_max REAL NOT NULL DEFAULT 0.0,
        PRIMARY KEY (herramienta_id, material_id),
        FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
CASCADE,
        FOREIGN KEY(material_id) REFERENCES materiales(id) ON DELETE
RESTRICT
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS herramienta_material_pieza_parametros (
        herramienta_id INTEGER NOT NULL,
        material_pieza_id INTEGER NOT NULL,
        fn_1 REAL NOT NULL DEFAULT 0.0,
        vc1_min REAL NOT NULL DEFAULT 0.0,
        vc1_max REAL NOT NULL DEFAULT 0.0,
        fn_2 REAL NOT NULL DEFAULT 0.0,
        vc2_min REAL NOT NULL DEFAULT 0.0,
        vc2_max REAL NOT NULL DEFAULT 0.0,
        fn_3 REAL NOT NULL DEFAULT 0.0,
        vc3_min REAL NOT NULL DEFAULT 0.0,
        vc3_max REAL NOT NULL DEFAULT 0.0,
        PRIMARY KEY (herramienta_id, material_pieza_id),
        FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
CASCADE,
        FOREIGN KEY(material_pieza_id) REFERENCES material_piezas(id) ON DELETE
RESTRICT
    )
    """)
```

```
    cur.execute("""
    CREATE TABLE IF NOT EXISTS herramienta_material_piezas (
        herramienta_id INTEGER NOT NULL,
        material_pieza_id INTEGER NOT NULL,
        PRIMARY KEY (herramienta_id, material_pieza_id),
        FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
CASCADE,
        FOREIGN KEY(material_pieza_id) REFERENCES material_piezas(id) ON DELETE
RESTRICT
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS herramienta_operaciones (
        herramienta_id INTEGER NOT NULL,
        operacion_id INTEGER NOT NULL,
        PRIMARY KEY (herramienta_id, operacion_id),
        FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
CASCADE,
        FOREIGN KEY(operacion_id) REFERENCES operaciones(id) ON DELETE
RESTRICT
    )
    """)

    if _fk_points_to(cur, "herramienta_material_piezas", "material_piezas_old"):
        cur.execute("PRAGMA foreign_keys = OFF")
        cur.execute("""
            CREATE TABLE IF NOT EXISTS herramienta_material_piezas_new (
                herramienta_id INTEGER NOT NULL,
                material_pieza_id INTEGER NOT NULL,
                PRIMARY KEY (herramienta_id, material_pieza_id),
                FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
CASCADE,
                FOREIGN KEY(material_pieza_id) REFERENCES material_piezas(id) ON
DELETE RESTRICT
            )
        """)
        cur.execute("""
            INSERT OR IGNORE INTO herramienta_material_piezas_new (herramienta_id,
material_pieza_id)
            SELECT herramienta_id, material_pieza_id
            FROM herramienta_material_piezas
        """)
        cur.execute("DROP TABLE herramienta_material_piezas")
        cur.execute("ALTER TABLE herramienta_material_piezas_new RENAME TO
herramienta_material_piezas")
        cur.execute("PRAGMA foreign_keys = ON")
```

```
    if _fk_points_to(cur, "herramienta_material_pieza_parametros", "material_piezas_old"):
        cur.execute("PRAGMA foreign_keys = OFF")
        cur.execute("""
            CREATE TABLE IF NOT EXISTS herramienta_material_pieza_parametros_new (
                herramienta_id INTEGER NOT NULL,
                material_pieza_id INTEGER NOT NULL,
                fn_1 REAL NOT NULL DEFAULT 0.0,
                vc1_min REAL NOT NULL DEFAULT 0.0,
                vc1_max REAL NOT NULL DEFAULT 0.0,
                fn_2 REAL NOT NULL DEFAULT 0.0,
                vc2_min REAL NOT NULL DEFAULT 0.0,
                vc2_max REAL NOT NULL DEFAULT 0.0,
                fn_3 REAL NOT NULL DEFAULT 0.0,
                vc3_min REAL NOT NULL DEFAULT 0.0,
                vc3_max REAL NOT NULL DEFAULT 0.0,
                PRIMARY KEY (herramienta_id, material_pieza_id),
                FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
CASCADE,
                FOREIGN KEY(material_pieza_id) REFERENCES material_piezas(id) ON
DELETE RESTRICT
            )
        """)
        cur.execute("""
            INSERT OR IGNORE INTO herramienta_material_pieza_parametros_new
            (herramienta_id, material_pieza_id, fn_1, vc1_min, vc1_max, fn_2, vc2_min,
vc2_max, fn_3, vc3_min, vc3_max)
            SELECT herramienta_id, material_pieza_id, fn_1, vc1_min, vc1_max, fn_2,
vc2_min, vc2_max, fn_3, vc3_min, vc3_max
            FROM herramienta_material_pieza_parametros
        """)
        cur.execute("DROP TABLE herramienta_material_pieza_parametros")
        cur.execute("ALTER TABLE herramienta_material_pieza_parametros_new
RENAME TO herramienta_material_pieza_parametros")
        cur.execute("PRAGMA foreign_keys = ON")

    cur.execute("""
    CREATE TABLE IF NOT EXISTS historial_selecciones (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        fecha_hora TEXT NOT NULL DEFAULT (datetime('now')),
        herramienta_id INTEGER NOT NULL,
        maquina_id INTEGER NOT NULL,
        operacion_id INTEGER NOT NULL,
        nivel_id INTEGER NOT NULL,
        material_id INTEGER NOT NULL,
        material_pieza TEXT,
        usuario TEXT NOT NULL,
```

```
        tipo_salida TEXT NOT NULL DEFAULT 'NUEVA',
        cantidad INTEGER NOT NULL DEFAULT 1,
        FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
RESTRICT,
        FOREIGN KEY(maquina_id) REFERENCES maquinas(id) ON DELETE
RESTRICT,
        FOREIGN KEY(operacion_id) REFERENCES operaciones(id) ON DELETE
RESTRICT,
        FOREIGN KEY(nivel_id) REFERENCES niveles(id) ON DELETE RESTRICT,
        FOREIGN KEY(material_id) REFERENCES materiales(id) ON DELETE
RESTRICT
    )
    """)

    cur.execute("""
    CREATE TABLE IF NOT EXISTS herramientas_activas (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        fecha_salida TEXT NOT NULL DEFAULT (datetime('now')),
        fecha_cierre TEXT,
        estado TEXT NOT NULL DEFAULT 'EN_USO',
        tipo_salida TEXT NOT NULL DEFAULT 'NUEVA',
        motivo_cierre TEXT,
        devuelta_a_stock INTEGER NOT NULL DEFAULT 0,
        herramienta_id INTEGER NOT NULL,
        maquina_id INTEGER NOT NULL,
        operacion_id INTEGER NOT NULL,
        nivel_id INTEGER NOT NULL,
        material_id INTEGER NOT NULL,
        material_pieza TEXT,
        usuario TEXT NOT NULL,
        FOREIGN KEY(herramienta_id) REFERENCES herramientas(id) ON DELETE
RESTRICT,
        FOREIGN KEY(maquina_id) REFERENCES maquinas(id) ON DELETE
RESTRICT,
        FOREIGN KEY(operacion_id) REFERENCES operaciones(id) ON DELETE
RESTRICT,
        FOREIGN KEY(nivel_id) REFERENCES niveles(id) ON DELETE RESTRICT,
        FOREIGN KEY(material_id) REFERENCES materiales(id) ON DELETE
RESTRICT
    )
    """)

    usage_migrations = [
        ("historial_selecciones", "tipo_salida", "TEXT NOT NULL DEFAULT 'NUEVA'"),
        ("herramientas_activas", "tipo_salida", "TEXT NOT NULL DEFAULT 'NUEVA'"),
        ("historial_selecciones", "material_pieza", "TEXT"),
        ("herramientas_activas", "material_pieza", "TEXT"),
```

```
    ]
    for table, col, ddl in usage_migrations:
        if not _column_exists(cur, table, col):
            cur.execute(f"ALTER TABLE {table} ADD COLUMN {col} {ddl}")

    cur.execute("""
        UPDATE herramientas
        SET
            fn_1 = CASE WHEN COALESCE(fn_1,0)=0 THEN COALESCE(fn_min,0) ELSE
fn_1 END,
            fn_2 = CASE WHEN COALESCE(fn_2,0)=0 THEN
                    CASE
                        WHEN COALESCE(fn_min,0)>0 AND COALESCE(fn_max,0)>0 THEN
(COALESCE(fn_min,0)+COALESCE(fn_max,0))/2.0
                        ELSE COALESCE(fn_2,0)
                    END
                ELSE fn_2 END,
            fn_3 = CASE WHEN COALESCE(fn_3,0)=0 THEN COALESCE(fn_max,0)
ELSE fn_3 END,
            vc1_min = CASE WHEN COALESCE(vc1_min,0)=0 THEN
COALESCE(vc_min,0) ELSE vc1_min END,
            vc1_max = CASE WHEN COALESCE(vc1_max,0)=0 THEN
COALESCE(vc_max,0) ELSE vc1_max END,
            vc2_min = CASE WHEN COALESCE(vc2_min,0)=0 THEN
COALESCE(vc_min,0) ELSE vc2_min END,
            vc2_max = CASE WHEN COALESCE(vc2_max,0)=0 THEN
COALESCE(vc_max,0) ELSE vc2_max END,
            vc3_min = CASE WHEN COALESCE(vc3_min,0)=0 THEN
COALESCE(vc_min,0) ELSE vc3_min END,
            vc3_max = CASE WHEN COALESCE(vc3_max,0)=0 THEN
COALESCE(vc_max,0) ELSE vc3_max END
    """)

    cur.execute("""
        UPDATE herramientas
        SET stock_nuevo = stock
        WHERE COALESCE(stock_nuevo, 0) = 0 AND COALESCE(stock_usado, 0) = 0
AND COALESCE(stock, 0) > 0
    """)
    cur.execute("UPDATE herramientas SET stock = COALESCE(stock_nuevo,0) +
COALESCE(stock_usado,0)")

    cur.execute("""
        INSERT OR IGNORE INTO herramienta_material_parametros
        (herramienta_id, material_id, fn_1, vc1_min, vc1_max, fn_2, vc2_min, vc2_max,
fn_3, vc3_min, vc3_max)
        SELECT
```

```
        hm.herramienta_id,
        hm.material_id,
        COALESCE(h.fn_1, COALESCE(h.fn_min,0)),
        COALESCE(h.vc1_min, COALESCE(h.vc_min,0)),
        COALESCE(h.vc1_max, COALESCE(h.vc_max,0)),
        COALESCE(h.fn_2, CASE
          WHEN COALESCE(h.fn_min,0)>0 AND COALESCE(h.fn_max,0)>0 THEN
(COALESCE(h.fn_min,0)+COALESCE(h.fn_max,0))/2.0
          ELSE 0
        END),
        COALESCE(h.vc2_min, COALESCE(h.vc_min,0)),
        COALESCE(h.vc2_max, COALESCE(h.vc_max,0)),
        COALESCE(h.fn_3, COALESCE(h.fn_max,0)),
        COALESCE(h.vc3_min, COALESCE(h.vc_min,0)),
        COALESCE(h.vc3_max, COALESCE(h.vc_max,0))
      FROM herramienta_materiales hm
      JOIN herramientas h ON h.id = hm.herramienta_id
    """)

    cur.execute("""
      INSERT OR IGNORE INTO herramienta_material_pieza_parametros
      (herramienta_id, material_pieza_id, fn_1, vc1_min, vc1_max, fn_2, vc2_min,
vc2_max, fn_3, vc3_min, vc3_max)
      SELECT
        hmp.herramienta_id,
        hmp.material_pieza_id,
        COALESCE(hmpm.fn_1, hm.fn_1, COALESCE(h.fn_1,
COALESCE(h.fn_min,0))),
        COALESCE(hmpm.vc1_min, hm.vc1_min, COALESCE(h.vc1_min,
COALESCE(h.vc_min,0))),
        COALESCE(hmpm.vc1_max, hm.vc1_max, COALESCE(h.vc1_max,
COALESCE(h.vc_max,0))),
        COALESCE(hmpm.fn_2, hm.fn_2, COALESCE(h.fn_2, CASE
          WHEN COALESCE(h.fn_min,0)>0 AND COALESCE(h.fn_max,0)>0 THEN
(COALESCE(h.fn_min,0)+COALESCE(h.fn_max,0))/2.0
          ELSE 0
        END)),
        COALESCE(hmpm.vc2_min, hm.vc2_min, COALESCE(h.vc2_min,
COALESCE(h.vc_min,0))),
        COALESCE(hmpm.vc2_max, hm.vc2_max, COALESCE(h.vc2_max,
COALESCE(h.vc_max,0))),
        COALESCE(hmpm.fn_3, hm.fn_3, COALESCE(h.fn_3,
COALESCE(h.fn_max,0))),
        COALESCE(hmpm.vc3_min, hm.vc3_min, COALESCE(h.vc3_min,
COALESCE(h.vc_min,0))),
        COALESCE(hmpm.vc3_max, hm.vc3_max, COALESCE(h.vc3_max,
COALESCE(h.vc_max,0)))
```

```
        FROM herramienta_material_piezas hmp
        JOIN herramientas h ON h.id = hmp.herramienta_id
        LEFT JOIN material_piezas mp ON mp.id = hmp.material_pieza_id
        LEFT JOIN herramienta_material_parametros hm
            ON hm.herramienta_id = hmp.herramienta_id AND hm.material_id =
mp.material_padre_id
        LEFT JOIN herramienta_material_pieza_parametros hmpm
            ON hmpm.herramienta_id = hmp.herramienta_id AND hmpm.material_pieza_id
= hmp.material_pieza_id
    """)

    cur.execute("""
        INSERT OR IGNORE INTO herramienta_operaciones (herramienta_id, operacion_id)
        SELECT id, operacion_id
        FROM herramientas
        WHERE operacion_id IS NOT NULL
    """)

    con.commit()

    cur.execute("SELECT COUNT(*) FROM maquinas")
    if cur.fetchone()[0] == 0:
        cur.executemany("INSERT INTO maquinas(nombre) VALUES (?)", [
            ("Torno",), ("Fresadora",)
        ])

    cur.execute("SELECT COUNT(*) FROM niveles")
    if cur.fetchone()[0] == 0:
        cur.executemany("INSERT INTO niveles(nombre) VALUES (?)", [
            ("Desbaste",), ("Acabado",)
        ])

    cur.execute("SELECT COUNT(*) FROM materiales")
    if cur.fetchone()[0] == 0:
        cur.executemany("INSERT INTO materiales(nombre) VALUES (?)", [
            ("P",), ("M",), ("K",), ("N",), ("S",), ("H",)
        ])

    cur.execute("SELECT COUNT(*) FROM usuarios")
    if cur.fetchone()[0] == 0:
        cur.executemany("INSERT INTO usuarios(nombre) VALUES (?)", [
            ("Operador 1",), ("Operador 2",), ("Supervisor",)
        ])

    con.commit()
    con.close()
    print(f"BD lista en: {DB_PATH}")
```

```
def fetchall(sql: str, params=()):
    con = None
    try:
        con = get_conn()
        cur = con.cursor()
        cur.execute(sql, params)
        rows = cur.fetchall()
        return rows
    finally:
        if con is not None:
            con.close()


def execute(sql: str, params=()):
    con = None
    try:
        con = get_conn()
        cur = con.cursor()
        cur.execute(sql, params)
        con.commit()
    except Exception:
        if con is not None:
            con.rollback()
        raise
    finally:
        if con is not None:
            con.close()


def insert_and_get_id(sql: str, params=()):
    con = None
    try:
        con = get_conn()
        cur = con.cursor()
        cur.execute(sql, params)
        con.commit()
        return cur.lastrowid
    except Exception:
        if con is not None:
            con.rollback()
        raise
    finally:
        if con is not None:
            con.close()
```

```
def get_next_herramienta_id() -> int:
    con = get_conn()
    cur = con.cursor()
    cur.execute("""
        SELECT
            CASE
                WHEN NOT EXISTS (SELECT 1 FROM herramientas WHERE id = 1) THEN
1
                ELSE COALESCE(
                    (
                        SELECT MIN(a.id) + 1
                        FROM herramientas a
                        WHERE NOT EXISTS (
                            SELECT 1
                            FROM herramientas b
                            WHERE b.id = a.id + 1
                        )
                    ),
                    1
                )
            END
    """)
    row = cur.fetchone()
    con.close()
    try:
        return int(row[0]) if row and row[0] is not None else 1
    except Exception:
        return 1
```

```
import os
import sys
import sqlite3
import math
from datetime import datetime
from typing import Optional

from PySide6.QtCore import Qt, Signal, QDate, QTimer
from PySide6.QtGui import QPixmap, QPainter, QValidator, QFont, QColor
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLabel,
    QPushButton, QStackedWidget, QTabWidget, QListWidget, QListWidgetItem,
    QDialog, QFormLayout, QLineEdit, QComboBox, QSpinBox, QMessageBox,
    QTableWidget, QTableWidgetItem, QHeaderView, QAbstractItemView, QFileDialog,
    QDateEdit, QDoubleSpinBox, QButtonGroup, QFrame, QStyledItemDelegate,
    QGridLayout, QScrollArea
)
try:
    from PySide6.QtCharts import (
        QChart, QChartView, QBarSeries, QBarSet, QBarCategoryAxis, QValueAxis
    )
    HAS_QT_CHARTS = True
except Exception:
    HAS_QT_CHARTS = False

from openpyxl import Workbook, load_workbook

from db import init_db, fetchall, execute, insert_and_get_id, get_next_herramienta_id,
DB_PATH

def info(parent, msg):
    QMessageBox.information(parent, "Info", msg)
```

```
def error(parent, msg):
    QMessageBox.critical(parent, "Error", msg)


def confirm(parent, msg) -> bool:
    return QMessageBox.question(parent, "Confirmar", msg, QMessageBox.Yes |
QMessageBox.No) == QMessageBox.Yes


def choose_image(parent, current_path: str = "") -> str:
    path, _ = QFileDialog.getOpenFileName(
        parent,
        "Seleccionar imagen",
        os.path.dirname(current_path) if current_path else "",
        "Imagenes (*.png *.jpg *.jpeg *.bmp *.webp)"
    )
    return path or ""


def update_image_preview(lbl: QLabel, image_path: str, size=320):
    if image_path and os.path.isfile(image_path):
        pix = QPixmap(image_path)
        if not pix.isNull():
            lbl.setPixmap(pix.scaled(size, size, Qt.KeepAspectRatio,
Qt.SmoothTransformation))
            lbl.setText("")
            return
    lbl.setPixmap(QPixmap())
    lbl.setText("Sin imagen")


def dark_theme(app: QApplication):
    app.setStyleSheet("""
        /* =======================
          Base global
        ======================= */
        QWidget {
            background: #0a0d16;
            color: #EAF0FF;
            font-family: Segoe UI;
            font-size: 13px;
        }

        QMainWindow {
            background: #090c15;
```

```
}

QLabel#h1 {
    font-size: 30px;
    font-weight: 950;
    color: #FFFFFF;
    letter-spacing: 0.5px;
}

QLabel#h2 {
    font-size: 20px;
    font-weight: 900;
    color: #F8FBFF;
}

QLabel#h3 {
    font-size: 15px;
    font-weight: 800;
    color: #FFFFFF;
}

QFrame#moduleHeader {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 rgba(23,35,57,0.92),
                    stop:1 rgba(13,20,34,0.92));
    border: 1px solid rgba(112, 139, 194, 0.35);
    border-radius: 14px;
}

QLabel#moduleEyebrow {
    color: #9FB8E8;
    font-size: 10px;
    font-weight: 800;
    letter-spacing: 1px;
}

QLabel#moduleTitle {
    color: #FFFFFF;
    font-size: 24px;
    font-weight: 950;
}

QLabel#moduleSub {
    color: #B7C8EC;
    font-size: 11px;
    font-weight: 600;
```

```
}

QLabel#muted {
    color: #A9B7D3;
    font-size: 14px;
}

QLabel#stepTitle {
    font-size: 25px;
    font-weight: 950;
    color: #FFFFFF;
}

QLabel#stepSub {
    color: #B9C7E6;
    font-size: 14px;
}

/* ========================
   Fondos premium
======================== */
QWidget#mainBg {
    background:
        qradialgradient(cx:0.72, cy:0.16, radius:1.25,
                        fx:0.72, fy:0.16,
                        stop:0 #2D3E4F,
                        stop:0.22 #22303D,
                        stop:0.48 #171F2B,
                        stop:0.72 #101722,
                        stop:1 #0A1119);
}

QFrame#sidePanel {
    background:
        qlineargradient(x1:0, y1:0, x2:0, y2:1,
                        stop:0 rgba(18,26,36,0.97),
                        stop:1 rgba(10,14,22,0.99));
    border: 1px solid rgba(110, 141, 171, 0.38);
    border-radius: 22px;
}

QFrame#sideBrandCard {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:1,
                        stop:0 rgba(49,71,108,0.96),
                        stop:1 rgba(26,38,62,0.96));
    border: 1px solid rgba(150, 182, 242, 0.52);
```

```
    border-radius: 18px;
}

QLabel#sideBrandTitle {
    font-size: 23px;
    font-weight: 950;
    color: #FFFFFF;
}

QLabel#sideBrandSub {
    color: #D0DEFF;
    font-size: 13px;
    font-weight: 600;
}

QLabel#sideBrandPending {
    color: #EAF2FF;
    font-size: 10px;
    font-weight: 800;
    background: rgba(10, 18, 40, 0.55);
    border: 1px solid rgba(163, 190, 255, 0.52);
    border-radius: 8px;
    padding: 3px 8px;
}

QLabel#sideSection {
    color: #AFC4F0;
    font-size: 12px;
    font-weight: 800;
    letter-spacing: 1px;
    padding: 2px 4px;
}

QLabel#sideVersion {
    color: #95AAD8;
    font-size: 12px;
    font-weight: 700;
}

QLabel#sideAlert {
    color: #FFE8B3;
    font-size: 12px;
    font-weight: 800;
    background: rgba(64, 45, 18, 0.70);
    border: 1px solid rgba(238, 192, 112, 0.55);
    border-radius: 10px;
    padding: 8px 10px;
```

```
}

QFrame#heroPanel {
   background:
      qradialgradient(cx:0.76, cy:0.12, radius:1.25,
                  fx:0.76, fy:0.12,
                  stop:0 rgba(72,102,210,0.30),
                  stop:0.30 rgba(37,56,122,0.22),
                  stop:0.62 rgba(14,20,39,0.90),
                  stop:1 rgba(8,11,20,0.96));
   border: 1px solid rgba(93, 119, 199, 0.28);
   border-radius: 22px;
}

QFrame#card {
   background: rgba(16, 22, 40, 0.92);
   border: 1px solid rgba(79, 101, 170, 0.32);
   border-radius: 18px;
}

QFrame#glassCard {
   background:
      qlineargradient(x1:0, y1:0, x2:1, y2:1,
                  stop:0 rgba(28,42,87,0.95),
                  stop:1 rgba(16,24,49,0.95));
   border: 1px solid rgba(91, 121, 214, 0.42);
   border-radius: 18px;
}

/* ========================
  Botones
======================== */
QPushButton {
   background: rgba(18, 24, 42, 0.95);
   border: 1px solid #2D3B63;
   padding: 10px 12px;
   border-radius: 12px;
   text-align: left;
   font-weight: 700;
}

QPushButton:hover {
   background: rgba(24, 32, 56, 1);
   border-color: #4F67A7;
}

QPushButton:pressed {
```

```
    background: rgba(14, 19, 35, 1);
}

QPushButton#primary {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:0,
                        stop:0 #1F9E8E,
                        stop:1 #56C8B8);
    border: 1px solid rgba(145, 231, 219, 0.34);
    color: white;
    font-weight: 900;
}

QPushButton#primary:hover {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:0,
                        stop:0 #27B3A1,
                        stop:1 #66DACC);
}

QPushButton#danger {
    background: rgba(48, 18, 25, 0.96);
    border: 1px solid #6A2C38;
}

QPushButton#danger:hover {
    background: rgba(60, 22, 30, 1);
    border-color: #8D3A4A;
}

QPushButton#ghost {
    background: rgba(12, 16, 28, 0.35);
    border: 1px solid rgba(77, 101, 169, 0.55);
    font-weight: 800;
}

QPushButton#ghost:hover {
    background: rgba(21, 30, 55, 0.75);
    border: 1px solid #6E8FFF;
}

QPushButton#navBtn {
    padding: 18px 20px;
    border-radius: 16px;
    border: 1px solid rgba(114, 143, 191, 0.56);
    background: rgba(16, 24, 38, 0.88);
    font-weight: 900;
```

```
    font-size: 16px;
    text-align: left;
}

QPushButton#navBtn:hover {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 rgba(45,67,108,0.82),
                    stop:1 rgba(24,38,66,0.82));
    border: 1px solid rgba(167, 198, 255, 0.80);
}

QPushButton#navBtn:checked {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 rgba(66,97,165,0.96),
                    stop:1 rgba(31,48,91,0.96));
    border: 1px solid #B7D2FF;
    color: #FFFFFF;
}

QPushButton#cardAction {
    background: rgba(10, 18, 34, 0.50);
    border: 1px solid rgba(92, 119, 204, 0.65);
    border-radius: 12px;
    padding: 10px 14px;
    font-weight: 900;
    text-align: center;
}

QPushButton#cardAction:hover {
    background: rgba(20, 34, 66, 0.95);
    border: 1px solid #7FA2FF;
}

/* =======================
   Inputs
======================= */
QLineEdit, QComboBox, QSpinBox, QDateEdit, QDoubleSpinBox {
    background: rgba(10, 15, 28, 0.95);
    border: 1px solid #2D3B63;
    padding: 9px 10px;
    border-radius: 12px;
    selection-background-color: #2F6FED;
    color: #F5F8FF;
}
```

```
    QLineEdit:focus, QComboBox:focus, QSpinBox:focus, QDateEdit:focus,
QDoubleSpinBox:focus {
        border: 1px solid #6C8CFF;
    }

    QComboBox::drop-down {
        border: 0px;
        width: 28px;
    }

    /* ========================
      Tabs
    ======================== */
    QTabWidget::pane {
        border: 0px;
    }

    QTabBar::tab {
        background: rgba(19, 26, 45, 0.96);
        border: 1px solid rgba(72, 92, 154, 0.50);
        padding: 10px 16px;
        border-radius: 14px;
        margin-right: 8px;
        font-weight: 900;
    }

    QTabBar::tab:hover {
        background: rgba(26, 35, 63, 1);
    }

    QTabBar::tab:selected {
        background:
            qlineargradient(x1:0, y1:0, x2:1, y2:0,
                        stop:0 rgba(42,67,155,0.96),
                        stop:1 rgba(29,42,85,0.96));
        border: 1px solid #6E8FFF;
    }

    /* ========================
      Tablas
    ======================== */
    QTableWidget {
        background: rgba(8, 13, 24, 0.94);
        border: 1px solid rgba(74, 96, 166, 0.35);
        border-radius: 18px;
        gridline-color: rgba(53, 66, 109, 0.55);
    }
```

```
QHeaderView::section {
    background: rgba(24, 34, 45, 0.98);
    border: 0px;
    padding: 11px;
    font-weight: 900;
    color: #F4F7FF;
}

QTableWidget::item {
    padding: 8px;
    border: 0px;
}

QTableWidget::item:selected {
    background: rgba(36, 53, 106, 0.95);
    color: #FFFFFF;
    border: 0px;
    outline: none;
}

QTableWidget::item:focus,
QTableView::item:focus {
    border: 0px;
    outline: none;
}

/* ========================
   Listas
======================== */
QListWidget {
    background: rgba(9, 14, 26, 0.96);
    border: 1px solid rgba(73, 96, 163, 0.34);
    border-radius: 16px;
    padding: 6px;
}

QListWidget::item {
    padding: 10px 10px;
    border-radius: 10px;
}

QListWidget::item:hover {
    background: rgba(24, 34, 60, 0.95);
}

QListWidget::item:selected {
```

```
    background: rgba(36, 53, 106, 0.95);
}

/* ========================
   Scrollbars
======================== */
QScrollBar:vertical {
    background: transparent;
    width: 12px;
    margin: 6px;
}

QScrollBar::handle:vertical {
    background: #31406A;
    border-radius: 6px;
    min-height: 30px;
}

QScrollBar::handle:vertical:hover {
    background: #47609A;
}

QScrollBar::add-line:vertical, QScrollBar::sub-line:vertical {
    height: 0px;
}

QScrollBar::add-page:vertical, QScrollBar::sub-page:vertical {
    background: transparent;
}

/* ========================
   Dashboard de catálogos
======================== */
QWidget#catalogBg {
    background:
        qradialgradient(cx:0.78, cy:0.18, radius:1.18,
                        fx:0.78, fy:0.18,
                        stop:0 rgba(49,104,108,0.28),
                        stop:0.32 rgba(33,67,73,0.22),
                        stop:0.68 rgba(14,23,30,0.92),
                        stop:1 rgba(9,13,20,0.98));
    border: 1px solid rgba(88, 132, 130, 0.30);
    border-radius: 20px;
}

QFrame#catalogDashboardCard {
    background:
```

```
        qlineargradient(x1:0, y1:0, x2:1, y2:1,
                    stop:0 rgba(46,74,79,0.98),
                    stop:1 rgba(20,34,39,0.98));
    border: 1px solid rgba(116, 172, 165, 0.52);
    border-radius: 20px;
}

QLabel#catalogCardIcon {
    min-width: 74px;
    max-width: 74px;
    min-height: 74px;
    max-height: 74px;
    border-radius: 37px;
    border: 1px solid rgba(146, 182, 255, 0.62);
    background: rgba(11, 18, 37, 0.82);
    font-size: 24px;
    font-weight: 900;
    qproperty-alignment: AlignCenter;
}

QLabel#catalogCardTitle {
    font-size: 21px;
    font-weight: 900;
    background: transparent;
    qproperty-alignment: AlignCenter;
}

QLabel#catalogCardDesc {
    font-size: 13px;
    color: #BFD0F8;
    background: transparent;
    qproperty-alignment: AlignCenter;
}

QLabel#catalogCardMeta {
    font-size: 12px;
    color: #DCE6FF;
    background: transparent;
    qproperty-alignment: AlignCenter;
}

QFrame#catalogDialogHeader {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 rgba(46,71,158,0.92),
                    stop:1 rgba(21,31,64,0.92));
    border: 1px solid rgba(118, 148, 238, 0.46);
```

```
    border-radius: 16px;
}

QLabel#catalogBreadcrumb {
    color: #C9D8FF;
    font-size: 11px;
    font-weight: 700;
}

/* ========================
   Flujo de selección
======================== */
QLabel#progressNode {
    background: rgba(15, 20, 35, 0.65);
    border: 1px solid rgba(71, 94, 165, 0.45);
    border-radius: 12px;
    padding: 9px 16px;
    font-weight: 800;
    font-size: 15px;
    color: #D8E3FF;
}

QLabel#progressNodeActive {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:0,
                        stop:0 rgba(41,66,155,0.95),
                        stop:1 rgba(25,38,80,0.95));
    border: 1px solid #8FB2FF;
    border-radius: 12px;
    padding: 9px 16px;
    font-weight: 900;
    font-size: 15px;
    color: #FFFFFF;
}

QLabel#progressArrow {
    color: #7CF2A3;
    font-size: 20px;
    font-weight: 900;
    padding: 0 4px;
    background: transparent;
}

QLabel#summaryChip {
    background: rgba(16, 22, 38, 0.92);
    border: 1px solid rgba(118, 142, 120, 0.42);
    border-radius: 10px;
```

```
    padding: 4px 8px;
    color: #E1EAFF;
    font-weight: 700;
    font-size: 11px;
}

QFrame#stepCard {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:1,
                    stop:0 rgba(29,43,91,0.98),
                    stop:1 rgba(16,24,49,0.98));
    border: 1px solid rgba(101, 132, 226, 0.42);
    border-radius: 20px;
    padding: 0px;
}

QFrame#stepCard:hover {
    background:
        qlineargradient(x1:0, y1:0, x2:1, y2:1,
                    stop:0 rgba(38,59,126,0.98),
                    stop:1 rgba(20,30,61,0.98));
    border: 1px solid #88A9FF;
}

QLabel#stepCardIcon {
    font-size: 92px;
    background: transparent;
    color: #FFFFFF;
}

QLabel#stepCardTitle {
    font-size: 17px;
    font-weight: 950;
    background: transparent;
    color: #FFFFFF;
}

QLabel#stepCardDesc {
    color: #D2DEF9;
    background: transparent;
    font-size: 13px;
}

QLabel#stepCardAction, QPushButton#stepCardAction {
    background: rgba(9, 18, 36, 0.45);
    border: 1px solid rgba(98, 129, 223, 0.70);
    border-radius: 12px;
```

```
            padding: 8px 14px;
            font-weight: 900;
            color: white;
            text-align: center;
        }

        QPushButton#stepCardAction:hover {
            background: rgba(20, 34, 66, 0.95);
            border: 1px solid #7FA2FF;
        }

        QLabel#stockGood {
            background: rgba(25, 129, 70, 0.95);
            border: 1px solid rgba(115, 232, 164, 0.35);
            border-radius: 10px;
            padding: 4px 10px;
            color: white;
            font-weight: 900;
        }

        QLabel#stockLow {
            background: rgba(191, 134, 22, 0.95);
            border: 1px solid rgba(255, 220, 136, 0.35);
            border-radius: 10px;
            padding: 4px 10px;
            color: white;
            font-weight: 900;
        }

        QLabel#stockZero {
            background: rgba(166, 48, 59, 0.96);
            border: 1px solid rgba(255, 162, 170, 0.30);
            border-radius: 10px;
            padding: 4px 10px;
            color: white;
            font-weight: 900;
        }
    """)


def make_card(widget: QWidget) -> QFrame:
    card = QFrame()
    card.setObjectName("card")
    lay = QVBoxLayout(card)
    lay.setContentsMargins(16, 16, 16, 16)
    lay.setSpacing(12)
    lay.addWidget(widget)
```

```
    return card


def make_glass_card(widget: QWidget) -> QFrame:
    card = QFrame()
    card.setObjectName("glassCard")
    lay = QVBoxLayout(card)
    lay.setContentsMargins(18, 18, 18, 18)
    lay.setSpacing(12)
    lay.addWidget(widget)
    return card


def make_module_header(title: str, subtitle: str = "", eyebrow: str = "MÓDULO") ->
QFrame:
    header = QFrame()
    header.setObjectName("moduleHeader")
    lay = QVBoxLayout(header)
    lay.setContentsMargins(16, 12, 16, 12)
    lay.setSpacing(4)

    show_eyebrow = bool(str(eyebrow).strip())
    show_subtitle = bool(str(subtitle).strip())

    lbl_eyebrow = QLabel(eyebrow)
    lbl_eyebrow.setObjectName("moduleEyebrow")
    lbl_eyebrow.setAlignment(Qt.AlignCenter)
    lbl_eyebrow.setVisible(show_eyebrow)
    lbl_title = QLabel(title)
    lbl_title.setObjectName("moduleTitle")
    lbl_title.setAlignment(Qt.AlignCenter)
    lbl_sub = QLabel(subtitle)
    lbl_sub.setObjectName("moduleSub")
    lbl_sub.setWordWrap(False)
    lbl_sub.setAlignment(Qt.AlignCenter)
    lbl_sub.setVisible(show_subtitle)

    if show_eyebrow:
        lay.addWidget(lbl_eyebrow, 0, Qt.AlignCenter)
    lay.addWidget(lbl_title, 0, Qt.AlignCenter)
    if show_subtitle:
        lay.addWidget(lbl_sub, 0, Qt.AlignCenter)
    return header


def apply_dialog_chrome(
    dialog: QDialog,
```

```
    title: str,
    subtitle: str,
    body_widget: QWidget,
    actions_layout: QHBoxLayout
):
    header = make_module_header(title, subtitle, "")
    root = QVBoxLayout(dialog)
    root.setContentsMargins(16, 16, 16, 16)
    root.setSpacing(12)
    root.addWidget(header)
    root.addWidget(make_card(body_widget), 1)
    root.addLayout(actions_layout)


class CenterAlignDelegate(QStyledItemDelegate):
    def initStyleOption(self, option, index):
        super().initStyleOption(option, index)
        option.displayAlignment = Qt.AlignCenter


def apply_table_polish(table: QTableWidget):
    table.setAlternatingRowColors(True)
    table.setItemDelegate(CenterAlignDelegate(table))
    table.horizontalHeader().setDefaultAlignment(Qt.AlignCenter)
    table.setStyleSheet(table.styleSheet() + """
        QTableWidget::item:alternate { background: rgba(13, 19, 35, 0.96); }
    """)

class BlankDateEdit(QDateEdit):
    def __init__(self, min_date: QDate, parent=None):
        super().__init__(parent)
        self._min_date = min_date

        self.setCalendarPopup(True)
        self.setDisplayFormat("dd/MM/yyyy")
        self.setMinimumDate(self._min_date)
        self.setSpecialValueText("")

        self.setDate(self._min_date)

        if self.lineEdit():
            self.lineEdit().setPlaceholderText("dd/MM/yyyy")
            self.lineEdit().setText("")

        self.editingFinished.connect(self._sync_empty_state)

    def _sync_empty_state(self):
```

```
        if self.lineEdit() and self.lineEdit().text().strip() == "":
            self.setDate(self._min_date)
            self.lineEdit().setText("")

    def showEvent(self, event):
        super().showEvent(event)
        if self.date() == self._min_date and self.lineEdit():
            self.lineEdit().setText("")

    def mousePressEvent(self, event):
        if self.date() == self._min_date:
            self.setDate(QDate.currentDate())
        super().mousePressEvent(event)

    def focusInEvent(self, event):
        if self.date() == self._min_date:
            self.setDate(QDate.currentDate())
        super().focusInEvent(event)


class DualDecimalSpinBox(QDoubleSpinBox):
    def validate(self, text: str, pos: int):
        t = (text or "").strip()
        if t == "":
            return (QValidator.Intermediate, text, pos)

        norm = t.replace(",", ".")
        if norm in ("+", "-", ".", "+.", "-."):
            return (QValidator.Intermediate, text, pos)

        try:
            float(norm)
        except Exception:
            return (QValidator.Invalid, text, pos)

        return (QValidator.Acceptable, text, pos)

    def valueFromText(self, text: str) -> float:
        t = (text or "").strip()
        if t == "":
            return float(self.minimum())
        try:
            return float(t.replace(",", "."))
        except Exception:
            return float(self.minimum())

    def wheelEvent(self, event):
```

```
        event.ignore()

class AddSimpleDialog(QDialog):
    def __init__(
        self,
        title: str,
        placeholder: str,
        parent=None,
        initial_name: str = "",
        initial_image: str = "",
        require_image: bool = False
    ):
        super().__init__(parent)
        self.setWindowTitle(title)
        self.setModal(True)
        self.setMinimumWidth(420)
        self.require_image = require_image
        self.image_path = initial_image or ""

        self.txt = QLineEdit()
        self.txt.setPlaceholderText(placeholder)
        self.txt.setText(initial_name)

        self.txt_image = QLineEdit()
        self.txt_image.setReadOnly(True)
        self.txt_image.setPlaceholderText("Selecciona una imagen")
        self.txt_image.setText(self.image_path)

        self.lbl_preview = QLabel("Sin imagen")
        self.lbl_preview.setAlignment(Qt.AlignCenter)
        self.lbl_preview.setFixedSize(320, 320)
        self.lbl_preview.setStyleSheet("border: 1px solid #2D3B63; border-radius: 10px;")
        update_image_preview(self.lbl_preview, self.image_path)

        self.btn_pick_image = QPushButton("Seleccionar imagen")
        self.btn_pick_image.setObjectName("ghost")
        self.btn_pick_image.clicked.connect(self._pick_image)

        self.btn_clear_image = QPushButton("Quitar")
        self.btn_clear_image.setObjectName("ghost")
        self.btn_clear_image.clicked.connect(self._clear_image)

        form = QFormLayout()
        form.addRow("Nombre *:", self.txt)
        if self.require_image:
            image_row = QHBoxLayout()
            image_row.addWidget(self.txt_image, 1)
```

```
            image_row.addWidget(self.btn_pick_image)
            image_row.addWidget(self.btn_clear_image)
            image_widget = QWidget()
            image_widget.setLayout(image_row)
            form.addRow("Imagen *:", image_widget)
            form.addRow("", self.lbl_preview)

        btn_ok = QPushButton("Guardar")
        btn_ok.setObjectName("primary")
        btn_ok.clicked.connect(self._on_accept)

        btn_cancel = QPushButton("Cancelar")
        btn_cancel.clicked.connect(self.reject)

        row = QHBoxLayout()
        row.addStretch(1)
        row.addWidget(btn_cancel)
        row.addWidget(btn_ok)

        body = QWidget()
        body.setLayout(form)
        apply_dialog_chrome(self, title, "Completa los campos y guarda.", body, row)

    def _pick_image(self):
        picked = choose_image(self, self.image_path)
        if picked:
            self.image_path = picked
            self.txt_image.setText(picked)
            update_image_preview(self.lbl_preview, self.image_path)

    def _clear_image(self):
        self.image_path = ""
        self.txt_image.clear()
        update_image_preview(self.lbl_preview, "")

    def _on_accept(self):
        if not self.txt.text().strip():
            return error(self, "Debes escribir un nombre.")
        if self.require_image and not self.image_path:
            return error(self, "Debes seleccionar una imagen.")
        self.accept()

    def values(self):
        return self.txt.text().strip(), self.image_path.strip()


class AddOperacionDialog(QDialog):
```

```
def __init__(
    self,
    parent=None,
    initial_maquina_id: Optional[int] = None,
    initial_nombre: str = "",
    initial_image: str = ""
):
    super().__init__(parent)
    self.setWindowTitle("Agregar operación")
    self.setModal(True)
    self.setMinimumWidth(520)
    self.image_path = initial_image or ""

    self.cb_maquina = QComboBox()
    for mid, n in fetchall("SELECT id, nombre FROM maquinas ORDER BY nombre"):
        self.cb_maquina.addItem(n, mid)
    if initial_maquina_id is not None:
        idx = self.cb_maquina.findData(initial_maquina_id)
        if idx >= 0:
            self.cb_maquina.setCurrentIndex(idx)

    self.txt = QLineEdit()
    self.txt.setPlaceholderText("Ej: Cilindrado / Planeado / Ranurado")
    self.txt.setText(initial_nombre)

    self.txt_image = QLineEdit()
    self.txt_image.setReadOnly(True)
    self.txt_image.setPlaceholderText("Selecciona una imagen")
    self.txt_image.setText(self.image_path)

    self.lbl_preview = QLabel("Sin imagen")
    self.lbl_preview.setAlignment(Qt.AlignCenter)
    self.lbl_preview.setFixedSize(320, 320)
    self.lbl_preview.setStyleSheet("border: 1px solid #2D3B63; border-radius: 10px;")
    update_image_preview(self.lbl_preview, self.image_path)

    self.btn_pick_image = QPushButton("Seleccionar imagen")
    self.btn_pick_image.setObjectName("ghost")
    self.btn_pick_image.clicked.connect(self._pick_image)

    self.btn_clear_image = QPushButton("Quitar")
    self.btn_clear_image.setObjectName("ghost")
    self.btn_clear_image.clicked.connect(self._clear_image)

    form = QFormLayout()
    form.addRow("Máquina *:", self.cb_maquina)
    form.addRow("Operación *:", self.txt)
```

```
        image_row = QHBoxLayout()
        image_row.addWidget(self.txt_image, 1)
        image_row.addWidget(self.btn_pick_image)
        image_row.addWidget(self.btn_clear_image)
        image_widget = QWidget()
        image_widget.setLayout(image_row)
        form.addRow("Imagen *:", image_widget)
        form.addRow("", self.lbl_preview)

        btn_ok = QPushButton("Guardar")
        btn_ok.setObjectName("primary")
        btn_ok.clicked.connect(self._on_accept)

        btn_cancel = QPushButton("Cancelar")
        btn_cancel.clicked.connect(self.reject)

        row = QHBoxLayout()
        row.addStretch(1)
        row.addWidget(btn_cancel)
        row.addWidget(btn_ok)

        body = QWidget()
        body.setLayout(form)
        apply_dialog_chrome(self, "Operación", "Registra una operación con su imagen.",
body, row)

    def _pick_image(self):
        picked = choose_image(self, self.image_path)
        if picked:
            self.image_path = picked
            self.txt_image.setText(picked)
            update_image_preview(self.lbl_preview, self.image_path)

    def _clear_image(self):
        self.image_path = ""
        self.txt_image.clear()
        update_image_preview(self.lbl_preview, "")

    def _on_accept(self):
        if not self.txt.text().strip():
            return error(self, "Debes escribir el nombre de la operación.")
        if not self.image_path:
            return error(self, "Debes seleccionar una imagen.")
        self.accept()

    def values(self):
        return self.cb_maquina.currentData(), self.txt.text().strip(), self.image_path.strip()
```

```
class AddMaterialPiezaDialog(QDialog):
    def __init__(
        self,
        parent=None,
        initial_material_padre_id: Optional[int] = None,
        initial_codigo: str = "",
        initial_nombre: str = "",
        initial_iso_min: int = 0,
        initial_iso_max: int = 0,
        initial_hb_min: float = 0.0,
        initial_hb_max: float = 0.0
    ):
        super().__init__(parent)
        self.setWindowTitle("Agregar material de pieza")
        self.setModal(True)
        self.setMinimumWidth(520)

        self.cb_material_padre = QComboBox()
        self.cb_material_padre.addItem("Seleccione...", None)
        for mid, n in fetchall("SELECT id, nombre FROM materiales ORDER BY nombre"):
            self.cb_material_padre.addItem(n, mid)
        if initial_material_padre_id is not None:
            idx = self.cb_material_padre.findData(initial_material_padre_id)
            if idx >= 0:
                self.cb_material_padre.setCurrentIndex(idx)
        else:
            self.cb_material_padre.setCurrentIndex(0)

        self.txt_codigo = QLineEdit()
        self.txt_codigo.setPlaceholderText("Ej: MP001 / H10_01 / P25-A")
        self.txt_codigo.setText((initial_codigo or "").strip())
        self.txt_codigo.returnPressed.connect(self._on_accept)

        self.txt = QLineEdit()
        self.txt.setPlaceholderText("Ej: Acero 1045 / Inox 304 / Fundición gris")
        self.txt.setText(initial_nombre)
        self.txt.returnPressed.connect(self._on_accept)

        self.spin_iso_min = QSpinBox()
        self.spin_iso_min.setRange(0, 99)
        self.spin_iso_min.setSingleStep(1)
        self.spin_iso_min.setValue(int(initial_iso_min or 0))

        self.spin_iso_max = QSpinBox()
        self.spin_iso_max.setRange(0, 99)
```

```
self.spin_iso_max.setSingleStep(1)
self.spin_iso_max.setValue(int(initial_iso_max or 0))

self.spin_hb_min = DualDecimalSpinBox()
self.spin_hb_min.setRange(0.0, 1200.0)
self.spin_hb_min.setDecimals(1)
self.spin_hb_min.setSingleStep(1.0)
self.spin_hb_min.setValue(float(initial_hb_min or 0.0))

self.spin_hb_max = DualDecimalSpinBox()
self.spin_hb_max.setRange(0.0, 1200.0)
self.spin_hb_max.setDecimals(1)
self.spin_hb_max.setSingleStep(1.0)
self.spin_hb_max.setValue(float(initial_hb_max or 0.0))

form = QFormLayout()
form.addRow("Material padre *:", self.cb_material_padre)
form.addRow("ID material *:", self.txt_codigo)
form.addRow("Material de pieza *:", self.txt)
iso_row = QHBoxLayout()
iso_row.addWidget(QLabel("Mín:"))
iso_row.addWidget(self.spin_iso_min, 1)
iso_row.addSpacing(8)
iso_row.addWidget(QLabel("Máx:"))
iso_row.addWidget(self.spin_iso_max, 1)
iso_box = QWidget()
iso_box.setLayout(iso_row)
form.addRow("Rango ISO de aplicación *:", iso_box)
hb_row = QHBoxLayout()
hb_row.addWidget(QLabel("Mín:"))
hb_row.addWidget(self.spin_hb_min, 1)
hb_row.addSpacing(8)
hb_row.addWidget(QLabel("Máx:"))
hb_row.addWidget(self.spin_hb_max, 1)
hb_box = QWidget()
hb_box.setLayout(hb_row)
form.addRow("Dureza Brinell (HB) *:", hb_box)

self.btn_ok = QPushButton("Guardar")
self.btn_ok.setObjectName("primary")
self.btn_ok.setAutoDefault(True)
self.btn_ok.setDefault(True)
self.btn_ok.clicked.connect(self._on_accept)

btn_cancel = QPushButton("Cancelar")
btn_cancel.clicked.connect(self.reject)
```

```
        row = QHBoxLayout()
        row.addStretch(1)
        row.addWidget(btn_cancel)
        row.addWidget(self.btn_ok)

        body = QWidget()
        body.setLayout(form)
        apply_dialog_chrome(self, "Material de pieza", "Define el rango ISO de aplicación y
la dureza HB.", body, row)

    def _on_accept(self):
        if self.cb_material_padre.currentData() is None:
            return error(self, "Debes seleccionar un material padre.")
        if not self.txt_codigo.text().strip():
            return error(self, "Debes escribir el ID del material.")
        if not self.txt.text().strip():
            return error(self, "Debes escribir el nombre del material de pieza.")
        iso_min = int(self.spin_iso_min.value())
        iso_max = int(self.spin_iso_max.value())
        if iso_min <= 0 or iso_max <= 0:
            return error(self, "El Rango ISO de aplicación (mín/máx) es obligatorio y debe ser
mayor que 0.")
        if iso_max < iso_min:
            return error(self, "Rango inválido: ISO máximo no puede ser menor que ISO
mínimo.")
        hb_min = float(self.spin_hb_min.value())
        hb_max = float(self.spin_hb_max.value())
        if hb_min <= 0 or hb_max <= 0:
            return error(self, "La Dureza Brinell (HB) mínima y máxima son obligatorias y
deben ser mayores que 0.")
        if hb_max < hb_min:
            return error(self, "Rango inválido: HB máximo no puede ser menor que HB
mínimo.")
        self.accept()

    def values(self):
        return (
            self.cb_material_padre.currentData(),
            self.txt_codigo.text().strip().upper(),
            self.txt.text().strip(),
            int(self.spin_iso_min.value()),
            int(self.spin_iso_max.value()),
            float(self.spin_hb_min.value()),
            float(self.spin_hb_max.value())
        )
```

```
class SelectMaterialPiezaDialog(QDialog):
    def __init__(self, material_padre_id: int, material_padre_nombre: str, parent=None):
        super().__init__(parent)
        self.material_padre_id = material_padre_id
        self.material_padre_nombre = material_padre_nombre
        self.setWindowTitle(f"Seleccionar material de pieza - {material_padre_nombre}")
        self.setModal(True)
        self.resize(900, 620)

        self.table = QTableWidget(0, 5)
        self.table.setHorizontalHeaderLabels(["ID material", "Material de pieza", "Rango
ISO", "HB mín", "HB máx"])
        self.table.setSelectionBehavior(QAbstractItemView.SelectRows)
        self.table.setSelectionMode(QAbstractItemView.MultiSelection)
        self.table.setEditTriggers(QAbstractItemView.NoEditTriggers)
        self.table.setFocusPolicy(Qt.NoFocus)
        self.table.verticalHeader().setVisible(False)
        self.table.verticalHeader().setDefaultSectionSize(40)
        self.table.setShowGrid(True)
        self.table.horizontalHeader().setSectionResizeMode(0, QHeaderView.Fixed)
        self.table.horizontalHeader().setSectionResizeMode(1, QHeaderView.Stretch)
        self.table.horizontalHeader().setSectionResizeMode(2, QHeaderView.Fixed)
        self.table.horizontalHeader().setSectionResizeMode(3, QHeaderView.Fixed)
        self.table.horizontalHeader().setSectionResizeMode(4, QHeaderView.Fixed)
        self.table.setColumnWidth(0, 150)
        self.table.setColumnWidth(2, 170)
        self.table.setColumnWidth(3, 130)
        self.table.setColumnWidth(4, 130)
        self.table.doubleClicked.connect(self._on_accept)
        apply_table_polish(self.table)

        self.lbl_count = QLabel("")
        self.lbl_count.setObjectName("muted")
        self.lbl_count.setAlignment(Qt.AlignCenter)
        self.lbl_familia = QLabel(f"Familia: {material_padre_nombre}")
        self.lbl_familia.setAlignment(Qt.AlignCenter)

        info_box = QFrame()
        info_box.setObjectName("glassCard")
        info_l = QVBoxLayout(info_box)
        info_l.setContentsMargins(12, 10, 12, 10)
        info_l.setSpacing(4)
        info_l.addWidget(self.lbl_familia, 0, Qt.AlignCenter)
        info_l.addWidget(self.lbl_count, 0, Qt.AlignCenter)

        btn_cancel = QPushButton("Cancelar")
        btn_cancel.clicked.connect(self.reject)
```

```
    self.btn_ok = QPushButton("Seleccionar")
    self.btn_ok.setObjectName("primary")
    self.btn_ok.clicked.connect(self._on_accept)

    row = QHBoxLayout()
    row.addStretch(1)
    row.addWidget(btn_cancel)
    row.addWidget(self.btn_ok)

    body = QWidget()
    body_l = QVBoxLayout(body)
    body_l.setContentsMargins(0, 0, 0, 0)
    body_l.setSpacing(10)
    body_l.addWidget(info_box)
    body_l.addWidget(self.table, 1)

    apply_dialog_chrome(
        self,
        "Seleccionar material de pieza",
        "Elige el material específico de la familia seleccionada.",
        body,
        row
    )

    self._load_rows()

def _load_rows(self):
    rows = fetchall("""
        SELECT
            id,
            COALESCE(codigo,''),
            nombre,
            COALESCE(iso_min,0),
            COALESCE(iso_max,0),
            COALESCE(hb_min, COALESCE(hb,0)),
            COALESCE(hb_max, COALESCE(hb,0))
        FROM material_piezas
        WHERE material_padre_id=?
        ORDER BY UPPER(TRIM(COALESCE(codigo,''))), UPPER(nombre)
    """, (self.material_padre_id,))

    self.table.setRowCount(0)
    if not rows:
        self.btn_ok.setEnabled(False)
        self.lbl_count.setText("Registros: 0")
        self.table.setRowCount(1)
```

```
            it_msg = QTableWidgetItem("No hay materiales de pieza para esta familia.")
            it_msg.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(0, 0, it_msg)
            it_dash = QTableWidgetItem("-")
            it_dash.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(0, 1, it_dash)
            it_dash2 = QTableWidgetItem("-")
            it_dash2.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(0, 2, it_dash2)
            it_dash3 = QTableWidgetItem("-")
            it_dash3.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(0, 3, it_dash3)
            it_dash4 = QTableWidgetItem("-")
            it_dash4.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(0, 4, it_dash4)
            self.table.setRowHeight(0, 44)
            return

        self.lbl_count.setText(f"Registros: {len(rows)}")
        for pid, codigo, nombre, iso_min, iso_max, hb_min, hb_max in rows:
            r = self.table.rowCount()
            self.table.insertRow(r)

            it_codigo = QTableWidgetItem(str(codigo or ""))
            it_codigo.setData(Qt.UserRole, pid)
            it_codigo.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 0, it_codigo)

            it_nombre = QTableWidgetItem(str(nombre))
            it_nombre.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 1, it_nombre)

            iso_min_val = int(iso_min or 0)
            iso_max_val = int(iso_max or 0)
            iso_prefix = (self.material_padre_nombre or "").strip().split(" ")[0].upper()[:1]
            iso_prefix = iso_prefix if iso_prefix in ("P", "M", "K", "N", "S", "H") else "ISO"
            iso_txt = f"{iso_prefix}{iso_min_val}-{iso_prefix}{iso_max_val}" if iso_min_val
> 0 and iso_max_val > 0 else "-"
            it_iso = QTableWidgetItem(iso_txt)
            it_iso.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 2, it_iso)

            hb_min_val = float(hb_min or 0.0)
            hb_max_val = float(hb_max or 0.0)
            hb_min_txt = f"{int(hb_min_val)}" if hb_min_val.is_integer() else
f"{hb_min_val:.1f}"
```

```
        hb_max_txt = f"{int(hb_max_val)}" if hb_max_val.is_integer() else
f"{hb_max_val:.1f}"

        it_hb_min = QTableWidgetItem(hb_min_txt)
        it_hb_min.setTextAlignment(Qt.AlignCenter)
        self.table.setItem(r, 3, it_hb_min)

        it_hb_max = QTableWidgetItem(hb_max_txt)
        it_hb_max.setTextAlignment(Qt.AlignCenter)
        self.table.setItem(r, 4, it_hb_max)
        self.table.setRowHeight(r, 42)

    def _on_accept(self):
        selected = self.selected_data()
        if not selected:
            return info(self, "Selecciona al menos un material de pieza.")
        self.accept()

    def selected_data(self):
        rows = sorted({idx.row() for idx in self.table.selectionModel().selectedRows()})
        selected = []
        for r in rows:
            it = self.table.item(r, 0)
            if not it:
                continue
            pid = it.data(Qt.UserRole)
            if pid is None:
                continue
            codigo = it.text().strip()
            nombre = self.table.item(r, 1).text().strip() if self.table.item(r, 1) else ""
            iso_txt = self.table.item(r, 2).text().strip() if self.table.item(r, 2) else "-"
            hb_min_txt = self.table.item(r, 3).text().strip() if self.table.item(r, 3) else "0"
            hb_max_txt = self.table.item(r, 4).text().strip() if self.table.item(r, 4) else "0"
            iso_min, iso_max = 0, 0
            try:
                if "-" in iso_txt and iso_txt != "-":
                    left, right = iso_txt.split("-", 1)
                    iso_min = int("".join(ch for ch in left if ch.isdigit()) or "0")
                    iso_max = int("".join(ch for ch in right if ch.isdigit()) or "0")
            except Exception:
                iso_min, iso_max = 0, 0
            try:
                hb_min = float(hb_min_txt)
            except Exception:
                hb_min = 0.0
            try:
                hb_max = float(hb_max_txt)
```

```
            except Exception:
                hb_max = 0.0
            selected.append((int(pid), codigo, nombre, iso_min, iso_max, hb_min, hb_max))
        return selected


class AddPortaHerramientaDialog(QDialog):
    def __init__(
        self,
        parent=None,
        initial_nombre: str = "",
        initial_image: str = "",
        initial_tool_ids: Optional[list] = None
    ):
        super().__init__(parent)
        self.setWindowTitle("Agregar portaherramienta")
        self.setModal(True)
        self.setMinimumWidth(760)
        self.setMinimumHeight(520)
        screen = QApplication.primaryScreen()
        if screen:
            geo = screen.availableGeometry()
            target_w = min(980, max(760, geo.width() - 220))
            target_h = min(690, max(520, geo.height() - 160))
            self.resize(target_w, target_h)
        else:
            self.resize(900, 620)
        self.image_path = initial_image or ""
        self.initial_tool_ids = set(initial_tool_ids or [])

        self.txt = QLineEdit()
        self.txt.setPlaceholderText("Ej: BT40 / HSK63 / Portapinzas ER32")
        self.txt.setText(initial_nombre)

        self.txt_image = QLineEdit()
        self.txt_image.setReadOnly(True)
        self.txt_image.setPlaceholderText("Selecciona una imagen")
        self.txt_image.setText(self.image_path)

        self.lbl_preview = QLabel("Sin imagen")
        self.lbl_preview.setAlignment(Qt.AlignCenter)
        self.lbl_preview.setFixedSize(300, 300)
        self.lbl_preview.setStyleSheet("border: 1px solid #2D3B63; border-radius: 10px;")
        update_image_preview(self.lbl_preview, self.image_path)

        self.btn_pick_image = QPushButton("Seleccionar imagen")
        self.btn_pick_image.setObjectName("ghost")
```

```
self.btn_pick_image.clicked.connect(self._pick_image)

self.btn_clear_image = QPushButton("Quitar")
self.btn_clear_image.setObjectName("ghost")
self.btn_clear_image.clicked.connect(self._clear_image)

self.list_herramientas = QListWidget()
self.list_herramientas.setSelectionMode(QAbstractItemView.NoSelection)
self.list_herramientas.setMinimumHeight(180)
self.list_herramientas.setUniformItemSizes(False)
self.list_herramientas.setStyleSheet("""
   QListWidget::item {
      padding: 8px 10px;
      margin: 3px 0px;
   }
   QListWidget::indicator {
      width: 18px;
      height: 18px;
   }
""")
self.lbl_tools_sel = QLabel("Seleccionadas: 0")
self.lbl_tools_sel.setObjectName("muted")
self._load_herramientas()
self.list_herramientas.itemChanged.connect(self._on_tools_checked_changed)

form = QFormLayout()
form.addRow("Nombre *:", self.txt)

image_row = QHBoxLayout()
image_row.addWidget(self.txt_image, 1)
image_row.addWidget(self.btn_pick_image)
image_row.addWidget(self.btn_clear_image)
image_widget = QWidget()
image_widget.setLayout(image_row)
form.addRow("Imagen *:", image_widget)
form.addRow("", self.lbl_preview)
tools_box = QWidget()
tools_lay = QVBoxLayout(tools_box)
tools_lay.setContentsMargins(0, 0, 0, 0)
tools_lay.setSpacing(6)
tools_lay.addWidget(self.list_herramientas)
tools_lay.addWidget(self.lbl_tools_sel)
form.addRow(QLabel("Herramientas asociadas (1 o más) *:"), tools_box)

btn_ok = QPushButton("Guardar")
btn_ok.setObjectName("primary")
btn_ok.clicked.connect(self._on_accept)
```

```
        btn_cancel = QPushButton("Cancelar")
        btn_cancel.clicked.connect(self.reject)

        row = QHBoxLayout()
        row.addStretch(1)
        row.addWidget(btn_cancel)
        row.addWidget(btn_ok)

        form_widget = QWidget()
        form_widget.setLayout(form)

        scroll = QScrollArea()
        scroll.setWidgetResizable(True)
        scroll.setFrameShape(QFrame.NoFrame)
        scroll.setWidget(form_widget)

        apply_dialog_chrome(
            self,
            "Portaherramienta",
            "Asocia este portaherramienta con una o más herramientas.",
            scroll,
            row
        )

    def _load_herramientas(self):
        self.list_herramientas.clear()
        rows = fetchall("SELECT id, codigo, nombre FROM herramientas ORDER BY
nombre, codigo")
        for hid, codigo, nombre in rows:
            it = QListWidgetItem(f"{codigo} - {nombre}")
            it.setData(Qt.UserRole, hid)
            it.setFlags(it.flags() | Qt.ItemIsUserCheckable)
            it.setCheckState(Qt.Checked if hid in self.initial_tool_ids else Qt.Unchecked)
            self.list_herramientas.addItem(it)
        self._update_tools_counter()

    def _checked_tool_ids(self):
        ids = []
        for i in range(self.list_herramientas.count()):
            it = self.list_herramientas.item(i)
            if it.checkState() == Qt.Checked:
                ids.append(it.data(Qt.UserRole))
        return ids

    def _update_tools_counter(self):
        total = self.list_herramientas.count()
```

```
        checked = len(self._checked_tool_ids())
        self.lbl_tools_sel.setText(f"Seleccionadas: {checked} de {total}")

    def _on_tools_checked_changed(self, _item: QListWidgetItem):
        self._update_tools_counter()

    def _pick_image(self):
        picked = choose_image(self, self.image_path)
        if picked:
            self.image_path = picked
            self.txt_image.setText(picked)
            update_image_preview(self.lbl_preview, self.image_path)

    def _clear_image(self):
        self.image_path = ""
        self.txt_image.clear()
        update_image_preview(self.lbl_preview, "")

    def _on_accept(self):
        if not self.txt.text().strip():
            return error(self, "Debes escribir el nombre del portaherramienta.")
        if not self.image_path:
            return error(self, "Debes seleccionar una imagen.")
        tool_ids = self._checked_tool_ids()
        if not tool_ids:
            return error(self, "Debes asociar al menos una herramienta.")
        self.accept()

    def values(self):
        tool_ids = self._checked_tool_ids()
        return self.txt.text().strip(), self.image_path.strip(), tool_ids


class AddToolDialog(QDialog):
    saved = Signal()

    def __init__(self, parent=None, tool_id: Optional[int] = None):
        super().__init__(parent)
        self.tool_id = tool_id
        self.stock_usado_current = 0

        self.setWindowTitle("Editar herramienta" if self.tool_id else "Agregar herramienta")
        self.setModal(True)
        self.setMinimumWidth(980)
        self.setMinimumHeight(620)
        screen = QApplication.primaryScreen()
        if screen:
```

```
    geo = screen.availableGeometry()
    target_w = min(1240, max(1000, geo.width() - 140))
    target_h = min(760, max(620, geo.height() - 120))
    self.resize(target_w, target_h)
else:
    self.resize(1180, 720)

self.txt_codigo = QLineEdit()
self.txt_codigo.setPlaceholderText("Ej: 1 / A-1023 / COD-XYZ")

self.txt_nombre = QLineEdit()
self.txt_nombre.setPlaceholderText("Ej: Herramienta 10mm")

self.cb_maquina = QComboBox()
self.list_operaciones = QListWidget()
self.list_operaciones.setSelectionMode(QAbstractItemView.NoSelection)
self.list_operaciones.setMinimumHeight(120)
self.list_operaciones.setUniformItemSizes(False)
self.list_operaciones.setStyleSheet("""
    QListWidget::item {
        padding: 8px 10px;
        margin: 3px 0px;
    }
    QListWidget::indicator {
        width: 18px;
        height: 18px;
    }
""")
self.lbl_operaciones_sel = QLabel("Seleccionadas: 0")
self.lbl_operaciones_sel.setObjectName("muted")
self.current_nivel_id = None

self.spin_stock = QSpinBox()
self.spin_stock.setRange(0, 100000)
self.spin_stock.setValue(0)

self.txt_ubic = QLineEdit()
self.txt_ubic.setPlaceholderText("Opcional (Estante A3, Gaveta 2...)")

self.image_path = ""
self.txt_image = QLineEdit()
self.txt_image.setReadOnly(True)
self.txt_image.setPlaceholderText("Selecciona una imagen")

self.lbl_preview = QLabel("Sin imagen")
self.lbl_preview.setAlignment(Qt.AlignCenter)
self.lbl_preview.setFixedSize(320, 320)
```

```
self.lbl_preview.setStyleSheet("border: 1px solid #2D3B63; border-radius: 10px;")
update_image_preview(self.lbl_preview, "")

self.btn_pick_image = QPushButton("Seleccionar imagen")
self.btn_pick_image.setObjectName("ghost")
self.btn_pick_image.clicked.connect(self._pick_image)

self.btn_clear_image = QPushButton("Quitar")
self.btn_clear_image.setObjectName("ghost")
self.btn_clear_image.clicked.connect(self._clear_image)

def make_dspin(minv, maxv, dec=2, step=0.1):
    w = DualDecimalSpinBox()
    w.setRange(minv, maxv)
    w.setDecimals(dec)
    w.setSingleStep(step)
    w.setKeyboardTracking(False)
    w.setSpecialValueText("")
    w.setValue(minv)
    if w.lineEdit() and float(minv) == 0.0:
        w.lineEdit().clear()
    return w

self.ap_min = make_dspin(0.0, 500.0, dec=3, step=0.1)
self.ap_max = make_dspin(0.0, 500.0, dec=3, step=0.1)
self.fn_1 = make_dspin(0.0, 50.0, dec=4, step=0.01)
self.fn_2 = make_dspin(0.0, 50.0, dec=4, step=0.01)
self.fn_3 = make_dspin(0.0, 50.0, dec=4, step=0.01)
self.vc1_min = make_dspin(0.0, 10000.0, dec=2, step=1.0)
self.vc1_max = make_dspin(0.0, 10000.0, dec=2, step=1.0)
self.vc2_min = make_dspin(0.0, 10000.0, dec=2, step=1.0)
self.vc2_max = make_dspin(0.0, 10000.0, dec=2, step=1.0)
self.vc3_min = make_dspin(0.0, 10000.0, dec=2, step=1.0)
self.vc3_max = make_dspin(0.0, 10000.0, dec=2, step=1.0)

self.list_materiales = QListWidget()
self.list_materiales.setSelectionMode(QAbstractItemView.NoSelection)
self.list_materiales.setMinimumHeight(160)
self.list_materiales.setUniformItemSizes(False)
self.list_materiales.setStyleSheet("""
    QListWidget::item {
        padding: 8px 10px;
        margin: 3px 0px;
    }
    QListWidget::indicator {
        width: 18px;
        height: 18px;
```

```
        }
    """)
    self.lbl_materiales_sel = QLabel("Seleccionados: 0")
    self.lbl_materiales_sel.setObjectName("muted")

    self.tbl_materiales_pieza_sel = QTableWidget(0, 6)
    self.tbl_materiales_pieza_sel.setHorizontalHeaderLabels(["Sel.", "ID", "Familia",
"Material de pieza", "ISO", "HB"])
    self.tbl_materiales_pieza_sel.setSelectionMode(QAbstractItemView.NoSelection)
    self.tbl_materiales_pieza_sel.setSelectionBehavior(QAbstractItemView.SelectRows)
    self.tbl_materiales_pieza_sel.setEditTriggers(QAbstractItemView.AllEditTriggers)
    self.tbl_materiales_pieza_sel.verticalHeader().setVisible(False)
    self.tbl_materiales_pieza_sel.horizontalHeader().setSectionResizeMode(0,
QHeaderView.Fixed)
    self.tbl_materiales_pieza_sel.horizontalHeader().setSectionResizeMode(1,
QHeaderView.Fixed)
    self.tbl_materiales_pieza_sel.horizontalHeader().setSectionResizeMode(2,
QHeaderView.ResizeToContents)
    self.tbl_materiales_pieza_sel.horizontalHeader().setSectionResizeMode(3,
QHeaderView.Stretch)
    self.tbl_materiales_pieza_sel.horizontalHeader().setSectionResizeMode(4,
QHeaderView.Fixed)
    self.tbl_materiales_pieza_sel.horizontalHeader().setSectionResizeMode(5,
QHeaderView.Fixed)
    self.tbl_materiales_pieza_sel.setColumnWidth(0, 64)
    self.tbl_materiales_pieza_sel.setColumnWidth(1, 90)
    self.tbl_materiales_pieza_sel.setColumnWidth(4, 130)
    self.tbl_materiales_pieza_sel.setColumnWidth(5, 120)
    self.tbl_materiales_pieza_sel.verticalHeader().setDefaultSectionSize(34)
    self.tbl_materiales_pieza_sel.setMinimumHeight(200)
    apply_table_polish(self.tbl_materiales_pieza_sel)
    self.lbl_materiales_pieza_sel = QLabel("Seleccionados: 0")
    self.lbl_materiales_pieza_sel.setObjectName("muted")
    self.material_params_cache = {}
    self.tbl_material_params = QTableWidget(0, 14)
    self.tbl_material_params.setHorizontalHeaderLabels([
        "ID", "Familia", "Material de pieza", "ISO", "HB",
        "Fn1", "Vc1 mín", "Vc1 máx", "Fn2", "Vc2 mín", "Vc2 máx", "Fn3", "Vc3 mín",
"Vc3 máx"
    ])
    self.tbl_material_params.setSelectionMode(QAbstractItemView.NoSelection)
    self.tbl_material_params.setEditTriggers(QAbstractItemView.NoEditTriggers)
    self.tbl_material_params.verticalHeader().setVisible(False)
    self.tbl_material_params.horizontalHeader().setSectionResizeMode(0,
QHeaderView.Fixed)
    self.tbl_material_params.horizontalHeader().setSectionResizeMode(1,
QHeaderView.Fixed)
```

```
        self.tbl_material_params.horizontalHeader().setSectionResizeMode(2,
QHeaderView.Fixed)
        self.tbl_material_params.horizontalHeader().setSectionResizeMode(3,
QHeaderView.Fixed)
        self.tbl_material_params.horizontalHeader().setSectionResizeMode(4,
QHeaderView.Fixed)
        self.tbl_material_params.setColumnWidth(0, 70)
        self.tbl_material_params.setColumnWidth(1, 170)
        self.tbl_material_params.setColumnWidth(2, 260)
        self.tbl_material_params.setColumnWidth(3, 115)
        self.tbl_material_params.setColumnWidth(4, 105)
        for c in range(5, 14):
            self.tbl_material_params.horizontalHeader().setSectionResizeMode(c,
QHeaderView.Fixed)
            self.tbl_material_params.setColumnWidth(c, 102)
        self.tbl_material_params.verticalHeader().setDefaultSectionSize(40)
        self.tbl_material_params.setMinimumHeight(280)
        self.tbl_material_params.setWordWrap(False)
        apply_table_polish(self.tbl_material_params)

        form = QFormLayout()

        form.addRow("Código *:", self.txt_codigo)
        form.addRow("Nombre *:", self.txt_nombre)

        form.addRow("Máquina *:", self.cb_maquina)
        form.addRow("Operaciones (multi) *:", self._list_with_counter(self.list_operaciones,
self.lbl_operaciones_sel))

        form.addRow(QLabel("Materiales de herramienta compatibles (multi) *:"),
self._list_with_counter(self.list_materiales, self.lbl_materiales_sel))
        form.addRow(QLabel("Materiales de pieza compatibles (multi) *:"),
self._table_with_counter(self.tbl_materiales_pieza_sel, self.lbl_materiales_pieza_sel))
        form.addRow(QLabel("Parámetros por material *:"), self.tbl_material_params)

        form.addRow(QLabel("Ap (mm) rango *:"), self._two(self.ap_min, self.ap_max,
"min", "max"))
        image_row = QHBoxLayout()
        image_row.addWidget(self.txt_image, 1)
        image_row.addWidget(self.btn_pick_image)
        image_row.addWidget(self.btn_clear_image)
        image_widget = QWidget()
        image_widget.setLayout(image_row)
        form.addRow("Imagen *:", image_widget)
        form.addRow("", self.lbl_preview)

        form.addRow("Stock:", self.spin_stock)
```

```
        form.addRow("Ubicación:", self.txt_ubic)

        btn_ok = QPushButton("Guardar cambios" if self.tool_id else "Guardar")
        btn_ok.setObjectName("primary")
        btn_ok.clicked.connect(self.on_save)

        btn_cancel = QPushButton("Cancelar")
        btn_cancel.clicked.connect(self.reject)

        row = QHBoxLayout()
        row.addStretch(1)
        row.addWidget(btn_cancel)
        row.addWidget(btn_ok)

        form_widget = QWidget()
        form_widget.setLayout(form)

        scroll = QScrollArea()
        scroll.setWidgetResizable(True)
        scroll.setFrameShape(QFrame.NoFrame)
        scroll.setWidget(form_widget)

        apply_dialog_chrome(
            self,
            "Herramienta",
            "Completa los datos de la herramienta y su configuración.",
            scroll,
            row
        )

        self.load_catalogs()
        self.list_materiales.itemChanged.connect(self._on_materiales_checked_changed)

self.tbl_materiales_pieza_sel.itemChanged.connect(self._on_materiales_pieza_checked_ch
anged)

self.tbl_materiales_pieza_sel.cellClicked.connect(self._toggle_material_pieza_check_on_cl
ick)
        self.list_operaciones.itemChanged.connect(self._on_operaciones_checked_changed)
        self.cb_maquina.currentIndexChanged.connect(self.load_operaciones_por_maquina)

        if self.tool_id:
            self.load_tool(self.tool_id)

    def _two(self, w1, w2, l1="min", l2="max"):
        row = QHBoxLayout()
        row.addWidget(QLabel(l1))
```

```
        row.addWidget(w1, 1)
        row.addSpacing(10)
        row.addWidget(QLabel(l2))
        row.addWidget(w2, 1)
        box = QWidget()
        box.setLayout(row)
        return box

    def _list_with_counter(self, list_widget: QListWidget, counter_label: QLabel) ->
QWidget:
        box = QWidget()
        lay = QVBoxLayout(box)
        lay.setContentsMargins(0, 0, 0, 0)
        lay.setSpacing(6)
        lay.addWidget(list_widget)
        lay.addWidget(counter_label)
        return box

    def _table_with_counter(self, table_widget: QTableWidget, counter_label: QLabel) ->
QWidget:
        box = QWidget()
        lay = QVBoxLayout(box)
        lay.setContentsMargins(0, 0, 0, 0)
        lay.setSpacing(6)
        lay.addWidget(table_widget)
        lay.addWidget(counter_label)
        return box

    def _is_checked(self, item: QListWidgetItem) -> bool:
        return item.checkState() == Qt.Checked

    def _checked_ids(self, list_widget: QListWidget):
        ids = []
        for i in range(list_widget.count()):
            it = list_widget.item(i)
            if self._is_checked(it):
                ids.append(it.data(Qt.UserRole))
        return ids

    def _checked_labels(self, list_widget: QListWidget):
        labels = []
        for i in range(list_widget.count()):
            it = list_widget.item(i)
            if self._is_checked(it):
                labels.append(it.text())
        return labels
```

```
    def _update_checked_counter(self, list_widget: QListWidget, counter_label: QLabel):
        total = list_widget.count()
        checked = len(self._checked_ids(list_widget))
        counter_label.setText(f"Seleccionados: {checked} de {total}")

    def _checked_materiales_pieza_ids(self):
        ids = []
        for r in range(self.tbl_materiales_pieza_sel.rowCount()):
            chk = self.tbl_materiales_pieza_sel.item(r, 0)
            if chk and chk.checkState() == Qt.Checked:
                ids.append(chk.data(Qt.UserRole))
        return ids

    def _checked_materiales_pieza_label_by_id(self):
        out = {}
        for r in range(self.tbl_materiales_pieza_sel.rowCount()):
            chk = self.tbl_materiales_pieza_sel.item(r, 0)
            if not chk:
                continue
            mpid = chk.data(Qt.UserRole)
            id_txt = (self.tbl_materiales_pieza_sel.item(r, 1).text() if
self.tbl_materiales_pieza_sel.item(r, 1) else "-").strip()
            fam_txt = (self.tbl_materiales_pieza_sel.item(r, 2).text() if
self.tbl_materiales_pieza_sel.item(r, 2) else "").strip()
            mat_txt = (self.tbl_materiales_pieza_sel.item(r, 3).text() if
self.tbl_materiales_pieza_sel.item(r, 3) else "").strip()
            iso_txt = (self.tbl_materiales_pieza_sel.item(r, 4).text() if
self.tbl_materiales_pieza_sel.item(r, 4) else "").strip()
            hb_txt = (self.tbl_materiales_pieza_sel.item(r, 5).text() if
self.tbl_materiales_pieza_sel.item(r, 5) else "").strip()
            out[mpid] = f"[{id_txt}] {mat_txt} ({fam_txt} | {iso_txt} | {hb_txt})"
        return out

    def _update_materiales_pieza_counter(self):
        total = self.tbl_materiales_pieza_sel.rowCount()
        checked = len(self._checked_materiales_pieza_ids())
        self.lbl_materiales_pieza_sel.setText(f"Seleccionados: {checked} de {total}")

    def _on_materiales_checked_changed(self, _item: QListWidgetItem):
        self._update_checked_counter(self.list_materiales, self.lbl_materiales_sel)
        self.load_materiales_pieza_por_materiales()

    def _checked_operacion_ids(self):
        ids = []
        for i in range(self.list_operaciones.count()):
            it = self.list_operaciones.item(i)
            if self._is_checked(it):
```

```
            ids.append(it.data(Qt.UserRole))
    return ids

def _on_operaciones_checked_changed(self, _item: QListWidgetItem):
    self._update_checked_counter(self.list_operaciones, self.lbl_operaciones_sel)

def _on_materiales_pieza_checked_changed(self, item):
    if not item or item.column() != 0:
        return
    self._update_materiales_pieza_counter()
    self.sync_material_params_table()

def _toggle_material_pieza_check_on_click(self, row: int, col: int):
    if col == 0:
        return
    chk = self.tbl_materiales_pieza_sel.item(row, 0)
    if not chk:
        return
    chk.setCheckState(Qt.Unchecked if chk.checkState() == Qt.Checked else Qt.Checked)

def _build_avances_widget(self):
    box = QFrame()
    box.setObjectName("glassCard")
    grid = QGridLayout(box)
    grid.setContentsMargins(10, 10, 10, 10)
    grid.setHorizontalSpacing(10)
    grid.setVerticalSpacing(8)

    headers = ["Avance", "Fn (mm/rev)", "Vc mín (m/min)", "Vc máx (m/min)"]
    for c, txt in enumerate(headers):
        lbl = QLabel(txt)
        lbl.setStyleSheet("font-weight: 900; color: #DDE8FF;")
        grid.addWidget(lbl, 0, c)

    rows = [
        ("Avance 1", self.fn_1, self.vc1_min, self.vc1_max),
        ("Avance 2", self.fn_2, self.vc2_min, self.vc2_max),
        ("Avance 3", self.fn_3, self.vc3_min, self.vc3_max),
    ]
    for r, (name, fnw, vminw, vmaxw) in enumerate(rows, start=1):
        name_lbl = QLabel(name)
        name_lbl.setStyleSheet("font-weight: 800;")
        grid.addWidget(name_lbl, r, 0)
        grid.addWidget(fnw, r, 1)
        grid.addWidget(vminw, r, 2)
        grid.addWidget(vmaxw, r, 3)
    return box
```

```
    def _pick_image(self):
        picked = choose_image(self, self.image_path)
        if picked:
            self.image_path = picked
            self.txt_image.setText(picked)
            update_image_preview(self.lbl_preview, self.image_path)

    def _clear_image(self):
        self.image_path = ""
        self.txt_image.clear()
        update_image_preview(self.lbl_preview, "")

    def load_catalogs(self):
        self.cb_maquina.clear()
        self.list_operaciones.clear()
        self.list_materiales.clear()
        self.tbl_materiales_pieza_sel.setRowCount(0)

        self.cb_maquina.addItem("Selecciona una máquina...", None)
        for mid, n in fetchall("SELECT id, nombre FROM maquinas ORDER BY nombre"):
            self.cb_maquina.addItem(n, mid)

        self.current_nivel_id = self._default_nivel_id()

        for mat_id, n in fetchall("SELECT id, nombre FROM materiales ORDER BY
nombre"):
            it = QListWidgetItem(n)
            it.setData(Qt.UserRole, mat_id)
            it.setFlags(it.flags() | Qt.ItemIsUserCheckable)
            it.setCheckState(Qt.Unchecked)
            self.list_materiales.addItem(it)

        self.load_operaciones_por_maquina()
        self.load_materiales_pieza_por_materiales()
        self.sync_material_params_table()
        self._update_checked_counter(self.list_operaciones, self.lbl_operaciones_sel)
        self._update_checked_counter(self.list_materiales, self.lbl_materiales_sel)
        self._update_materiales_pieza_counter()

    def _default_nivel_id(self):
        rows = fetchall("SELECT id FROM niveles ORDER BY id LIMIT 1")
        return int(rows[0][0]) if rows else None

    def _fmt_hb(self, hb_min, hb_max) -> str:
        a = float(hb_min or 0.0)
        b = float(hb_max or 0.0)
```

```
    if a <= 0 or b <= 0:
        return "HB -"
    at = f"{int(a)}" if a.is_integer() else f"{a:.1f}"
    bt = f"{int(b)}" if b.is_integer() else f"{b:.1f}"
    return f"HB {at}-{bt}"

def _fmt_iso(self, pref: str, iso_min, iso_max) -> str:
    a = int(iso_min or 0)
    b = int(iso_max or 0)
    p = (pref or "").strip().upper()[:1]
    p = p if p in ("P", "M", "K", "N", "S", "H") else "ISO"
    if a <= 0 or b <= 0:
        return "-"
    return f"{p}{a}-{p}{b}"

def _selected_material_ids(self):
    return self._checked_ids(self.list_materiales)

def _make_cut_spin(self, minv, maxv, dec=2, step=0.1):
    w = DualDecimalSpinBox()
    w.setRange(minv, maxv)
    w.setDecimals(dec)
    w.setSingleStep(step)
    w.setKeyboardTracking(False)
    w.setSpecialValueText("")
    w.setValue(minv)
    if w.lineEdit() and float(minv) == 0.0:
        w.lineEdit().clear()
    w.setAlignment(Qt.AlignCenter)
    w.setFixedHeight(32)
    w.setMinimumWidth(90)
    return w

def _capture_material_params_from_table(self):
    for r in range(self.tbl_material_params.rowCount()):
        it = self.tbl_material_params.item(r, 0)
        if not it:
            continue
        mpid = it.data(Qt.UserRole)
        if mpid is None:
            continue
        vals = []
        for c in range(5, 14):
            w = self.tbl_material_params.cellWidget(r, c)
            vals.append(float(w.value()) if isinstance(w, QDoubleSpinBox) else 0.0)
        self.material_params_cache[mpid] = tuple(vals)
```

```
    def sync_material_params_table(self):
        self._capture_material_params_from_table()
        selected = []
        for r in range(self.tbl_materiales_pieza_sel.rowCount()):
            chk = self.tbl_materiales_pieza_sel.item(r, 0)
            if not chk or chk.checkState() != Qt.Checked:
                continue
            mpid = chk.data(Qt.UserRole)
            id_txt = (self.tbl_materiales_pieza_sel.item(r, 1).text() if
self.tbl_materiales_pieza_sel.item(r, 1) else "-").strip()
            fam_txt = (self.tbl_materiales_pieza_sel.item(r, 2).text() if
self.tbl_materiales_pieza_sel.item(r, 2) else "").strip()
            mat_txt = (self.tbl_materiales_pieza_sel.item(r, 3).text() if
self.tbl_materiales_pieza_sel.item(r, 3) else "").strip()
            iso_txt = (self.tbl_materiales_pieza_sel.item(r, 4).text() if
self.tbl_materiales_pieza_sel.item(r, 4) else "").strip()
            hb_txt = (self.tbl_materiales_pieza_sel.item(r, 5).text() if
self.tbl_materiales_pieza_sel.item(r, 5) else "").strip()
            selected.append((mpid, id_txt, fam_txt, mat_txt, iso_txt, hb_txt))
        selected.sort(key=lambda x: (x[2].lower(), x[3].lower(), x[1]))

        self.tbl_material_params.setRowCount(0)
        for mpid, id_txt, fam_txt, mat_txt, iso_txt, hb_txt in selected:
            r = self.tbl_material_params.rowCount()
            self.tbl_material_params.insertRow(r)

            it_id = QTableWidgetItem(id_txt)
            it_id.setData(Qt.UserRole, mpid)
            it_id.setTextAlignment(Qt.AlignCenter)
            it_fam = QTableWidgetItem(fam_txt)
            it_fam.setTextAlignment(Qt.AlignVCenter | Qt.AlignLeft)
            it_mat = QTableWidgetItem(mat_txt)
            it_mat.setTextAlignment(Qt.AlignVCenter | Qt.AlignLeft)
            it_iso = QTableWidgetItem(iso_txt)
            it_iso.setTextAlignment(Qt.AlignCenter)
            it_hb = QTableWidgetItem(hb_txt)
            it_hb.setTextAlignment(Qt.AlignCenter)

            self.tbl_material_params.setItem(r, 0, it_id)
            self.tbl_material_params.setItem(r, 1, it_fam)
            self.tbl_material_params.setItem(r, 2, it_mat)
            self.tbl_material_params.setItem(r, 3, it_iso)
            self.tbl_material_params.setItem(r, 4, it_hb)

            values = self.material_params_cache.get(mpid, (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0))
            specs = [
```

```
            (0.0, 50.0, 4, 0.01),
            (0.0, 10000.0, 2, 1.0),
            (0.0, 10000.0, 2, 1.0),
            (0.0, 50.0, 4, 0.01),
            (0.0, 10000.0, 2, 1.0),
            (0.0, 10000.0, 2, 1.0),
            (0.0, 50.0, 4, 0.01),
            (0.0, 10000.0, 2, 1.0),
            (0.0, 10000.0, 2, 1.0),
        ]
        for c, (minv, maxv, dec, step) in enumerate(specs, start=5):
            w = self._make_cut_spin(minv, maxv, dec=dec, step=step)
            cache_idx = c - 5
            w.setValue(float(values[cache_idx] if cache_idx < len(values) else 0.0))
            self.tbl_material_params.setCellWidget(r, c, w)

    def _collect_selected_material_params(self):
        self._capture_material_params_from_table()
        out = {}
        for mpid in self._checked_materiales_pieza_ids():
            out[mpid] = self.material_params_cache.get(mpid, (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0))
        return out

    def load_materiales_pieza_por_materiales(self, preserve_selected: bool = True,
preselect_ids=None):
        keep_ids = set(preselect_ids or [])
        if preserve_selected and not keep_ids:
            keep_ids = set(self._checked_materiales_pieza_ids())

        self.tbl_materiales_pieza_sel.blockSignals(True)
        self.tbl_materiales_pieza_sel.setRowCount(0)
        mat_ids = self._selected_material_ids()
        if not mat_ids:
            self.tbl_materiales_pieza_sel.blockSignals(False)
            self._update_materiales_pieza_counter()
            self.tbl_material_params.setRowCount(0)
            return

        placeholders = ",".join("?" for _ in mat_ids)
        rows = fetchall(f"""
            SELECT
                mp.id,
                COALESCE(mp.codigo,''),
                mp.nombre,
                m.nombre,
                COALESCE(mp.iso_min,0),
```

```
            COALESCE(mp.iso_max,0),
            COALESCE(mp.hb_min, COALESCE(mp.hb,0)),
            COALESCE(mp.hb_max, COALESCE(mp.hb,0))
        FROM material_piezas mp
        JOIN materiales m ON m.id = mp.material_padre_id
        WHERE mp.material_padre_id IN ({placeholders})
        ORDER BY m.nombre, UPPER(TRIM(COALESCE(mp.codigo,''))), mp.nombre
    """, tuple(mat_ids))

    for mp_id, mp_codigo, mp_nombre, mat_nombre, iso_min, iso_max, hb_min, hb_max
in rows:
        iso_txt = self._fmt_iso(mat_nombre, iso_min, iso_max)
        hb_txt = self._fmt_hb(hb_min, hb_max)
        cod_txt = (mp_codigo or "").strip() or "-"
        r = self.tbl_materiales_pieza_sel.rowCount()
        self.tbl_materiales_pieza_sel.insertRow(r)

        chk = QTableWidgetItem("")
        chk.setFlags((chk.flags() | Qt.ItemIsUserCheckable) & ~Qt.ItemIsEditable)
        chk.setCheckState(Qt.Checked if mp_id in keep_ids else Qt.Unchecked)
        chk.setData(Qt.UserRole, mp_id)
        chk.setTextAlignment(Qt.AlignCenter)
        self.tbl_materiales_pieza_sel.setItem(r, 0, chk)

        id_item = QTableWidgetItem(cod_txt)
        id_item.setTextAlignment(Qt.AlignCenter)
        fam_item = QTableWidgetItem(mat_nombre)
        mat_item = QTableWidgetItem(mp_nombre)
        iso_item = QTableWidgetItem(iso_txt)
        iso_item.setTextAlignment(Qt.AlignCenter)
        hb_item = QTableWidgetItem(hb_txt)
        hb_item.setTextAlignment(Qt.AlignCenter)

        self.tbl_materiales_pieza_sel.setItem(r, 1, id_item)
        self.tbl_materiales_pieza_sel.setItem(r, 2, fam_item)
        self.tbl_materiales_pieza_sel.setItem(r, 3, mat_item)
        self.tbl_materiales_pieza_sel.setItem(r, 4, iso_item)
        self.tbl_materiales_pieza_sel.setItem(r, 5, hb_item)

    self.tbl_materiales_pieza_sel.blockSignals(False)
    self._update_materiales_pieza_counter()
    self.sync_material_params_table()

def load_operaciones_por_maquina(self):
    keep_ids = set(self._checked_operacion_ids())
    self.list_operaciones.blockSignals(True)
    self.list_operaciones.clear()
```

```
        maquina_id = self.cb_maquina.currentData()
        if maquina_id is None:
            self.list_operaciones.blockSignals(False)
            self._update_checked_counter(self.list_operaciones, self.lbl_operaciones_sel)
            return
        for oid, n in fetchall("SELECT id, nombre FROM operaciones WHERE
maquina_id=? ORDER BY nombre", (maquina_id,)):
            it = QListWidgetItem(str(n))
            it.setData(Qt.UserRole, int(oid))
            it.setFlags(it.flags() | Qt.ItemIsUserCheckable)
            it.setCheckState(Qt.Checked if int(oid) in keep_ids else Qt.Unchecked)
            self.list_operaciones.addItem(it)
        self.list_operaciones.blockSignals(False)
        self._update_checked_counter(self.list_operaciones, self.lbl_operaciones_sel)

    def load_tool(self, hid: int):
        rows = fetchall("""
            SELECT codigo, nombre, maquina_id, nivel_id, operacion_id, stock,
COALESCE(stock_usado,0), COALESCE(ubicacion,''),
                   COALESCE(imagen,''),
                   COALESCE(ap_min,0), COALESCE(ap_max,0),
                   COALESCE(fn_1, COALESCE(fn_min,0)),
                   COALESCE(fn_2, CASE WHEN COALESCE(fn_min,0)>0 AND
COALESCE(fn_max,0)>0 THEN (COALESCE(fn_min,0)+COALESCE(fn_max,0))/2.0
ELSE 0 END),
                   COALESCE(fn_3, COALESCE(fn_max,0)),
                   COALESCE(vc1_min, COALESCE(vc_min,0)), COALESCE(vc1_max,
COALESCE(vc_max,0)),
                   COALESCE(vc2_min, COALESCE(vc_min,0)), COALESCE(vc2_max,
COALESCE(vc_max,0)),
                   COALESCE(vc3_min, COALESCE(vc_min,0)), COALESCE(vc3_max,
COALESCE(vc_max,0))
            FROM herramientas
            WHERE id=?
            LIMIT 1
        """, (hid,))
        if not rows:
            return error(self, "No se encontró la herramienta seleccionada.")

        (codigo, nombre, maq_id, niv_id, op_id, stock, stock_usado, ubic, img,
         apmin, apmax, fn1, fn2, fn3, vc1min, vc1max, vc2min, vc2max, vc3min, vc3max) =
rows[0]

        self.txt_codigo.setText(str(codigo))
        self.txt_nombre.setText(str(nombre))
        self.spin_stock.setValue(int(stock) if stock is not None else 0)
        self.stock_usado_current = int(stock_usado or 0)
```

```
        self.txt_ubic.setText(ubic or "")
        self.image_path = img or ""
        self.txt_image.setText(self.image_path)
        update_image_preview(self.lbl_preview, self.image_path)

        self.ap_min.setValue(float(apmin or 0))
        self.ap_max.setValue(float(apmax or 0))
        self.fn_1.setValue(float(fn1 or 0))
        self.fn_2.setValue(float(fn2 or 0))
        self.fn_3.setValue(float(fn3 or 0))
        self.vc1_min.setValue(float(vc1min or 0))
        self.vc1_max.setValue(float(vc1max or 0))
        self.vc2_min.setValue(float(vc2min or 0))
        self.vc2_max.setValue(float(vc2max or 0))
        self.vc3_min.setValue(float(vc3min or 0))
        self.vc3_max.setValue(float(vc3max or 0))

        idx_maq = self.cb_maquina.findData(maq_id)
        if idx_maq >= 0:
            self.cb_maquina.setCurrentIndex(idx_maq)

        self.load_operaciones_por_maquina()

        self.current_nivel_id = int(niv_id) if niv_id is not None else self._default_nivel_id()

        op_ids = set([x[0] for x in fetchall(
            "SELECT operacion_id FROM herramienta_operaciones WHERE
herramienta_id=?",
            (hid,)
        )])
        if not op_ids and op_id is not None:
            op_ids = {int(op_id)}

        self.list_operaciones.blockSignals(True)
        for i in range(self.list_operaciones.count()):
            it = self.list_operaciones.item(i)
            oid = it.data(Qt.UserRole)
            it.setCheckState(Qt.Checked if oid in op_ids else Qt.Unchecked)
        self.list_operaciones.blockSignals(False)
        self._update_checked_counter(self.list_operaciones, self.lbl_operaciones_sel)

        mat_ids = set([x[0] for x in fetchall(
            "SELECT material_id FROM herramienta_materiales WHERE herramienta_id=?",
            (hid,)
        )])
        params_rows = fetchall("""
```

```
            SELECT material_pieza_id, fn_1, vc1_min, vc1_max, fn_2, vc2_min, vc2_max,
fn_3, vc3_min, vc3_max
            FROM herramienta_material_pieza_parametros
            WHERE herramienta_id=?
        """, (hid,))
        self.material_params_cache = {
            int(mpid): (
                float(fn1 or 0), float(vc1min or 0), float(vc1max or 0),
                float(fn2 or 0), float(vc2min or 0), float(vc2max or 0),
                float(fn3 or 0), float(vc3min or 0), float(vc3max or 0)
            )
            for (mpid, fn1, vc1min, vc1max, fn2, vc2min, vc2max, fn3, vc3min, vc3max) in
params_rows
        }
        mat_pieza_ids = set([x[0] for x in fetchall(
            "SELECT material_pieza_id FROM herramienta_material_piezas WHERE
herramienta_id=?",
            (hid,)
        )])
        if not mat_pieza_ids and self.material_params_cache:
            mat_pieza_ids = set(self.material_params_cache.keys())

        if not mat_ids and mat_pieza_ids:
            placeholders = ",".join("?" for _ in mat_pieza_ids)
            inferred = fetchall(
                f"SELECT DISTINCT material_padre_id FROM material_piezas WHERE id IN
({placeholders})",
                tuple(mat_pieza_ids)
            )
            mat_ids = {int(x[0]) for x in inferred if x and x[0] is not None}

        self.list_materiales.blockSignals(True)
        for i in range(self.list_materiales.count()):
            it = self.list_materiales.item(i)
            checked = it.data(Qt.UserRole) in mat_ids
            it.setCheckState(Qt.Checked if checked else Qt.Unchecked)
        self.list_materiales.blockSignals(False)
        self._update_checked_counter(self.list_materiales, self.lbl_materiales_sel)
        self.load_materiales_pieza_por_materiales(preserve_selected=False,
preselect_ids=mat_pieza_ids)
        for mpid in mat_pieza_ids:
            if mpid not in self.material_params_cache:
                self.material_params_cache[mpid] = (
                    float(fn1 or 0), float(vc1min or 0), float(vc1max or 0),
                    float(fn2 or 0), float(vc2min or 0), float(vc2max or 0),
                    float(fn3 or 0), float(vc3min or 0), float(vc3max or 0)
                )
```

```
        self.sync_material_params_table()

    def _validate_ranges(self):
        apmin, apmax = self.ap_min.value(), self.ap_max.value()
        if apmin <= 0 or apmax <= 0:
            return False, "Ap es obligatorio y debe ser > 0 (min y max)."
        if apmax < apmin:
            return False, "Rango inválido: Ap_max no puede ser menor que Ap_min."

        params_map = self._collect_selected_material_params()
        if not params_map:
            return False, "Debes seleccionar al menos 1 material de pieza."

        name_by_id = self._checked_materiales_pieza_label_by_id()
        for mpid, vals in params_map.items():
            fn1, vc1min, vc1max, fn2, vc2min, vc2max, fn3, vc3min, vc3max = vals
            label = name_by_id.get(mpid, f"ID {mpid}")
            lanes = [
                (1, float(fn1 or 0), float(vc1min or 0), float(vc1max or 0)),
                (2, float(fn2 or 0), float(vc2min or 0), float(vc2max or 0)),
                (3, float(fn3 or 0), float(vc3min or 0), float(vc3max or 0)),
            ]

            active_fn_values = []
            for idx, fnv, vmin, vmax in lanes:
                any_value = fnv > 0 or vmin > 0 or vmax > 0
                if not any_value:
                    continue
                if fnv <= 0:
                    return False, f"En material '{label}', si usas Vc{idx} debes definir Fn{idx} >
0."
                if vmin <= 0 or vmax <= 0:
                    return False, f"En material '{label}', Vc{idx} mínimo y máximo deben ser >
0."
                if vmax < vmin:
                    return False, f"En material '{label}', Vc{idx}_max no puede ser menor que
Vc{idx}_min."
                active_fn_values.append(fnv)

            if not active_fn_values:
                return False, f"En material '{label}', debes definir al menos 1 avance (Fn + Vc
min/máx)."

            for i in range(1, len(active_fn_values)):
                if active_fn_values[i] <= active_fn_values[i - 1]:
                    return False, (
                        f"En material '{label}', los avances activos deben ser crecientes "
```

```
                    f"(Fn1 < Fn2 < Fn3 si se usan)."
                )
        return True, ""

    def on_save(self):
        codigo = self.txt_codigo.text().strip()
        nombre = self.txt_nombre.text().strip()

        maquina_id = self.cb_maquina.currentData()
        nivel_id = self.current_nivel_id if self.current_nivel_id is not None else
self._default_nivel_id()
        operacion_ids = self._checked_operacion_ids()
        operacion_id = operacion_ids[0] if operacion_ids else None
        stock = int(self.spin_stock.value())
        ubic = (self.txt_ubic.text().strip() or "")

        mats = self._checked_ids(self.list_materiales)
        mats_pieza = self._checked_materiales_pieza_ids()

        if not codigo:
            return error(self, "Código es obligatorio.")
        if not nombre:
            return error(self, "Nombre es obligatorio.")
        if maquina_id is None:
            return error(self, "Selecciona una máquina.")
        if nivel_id is None:
            return error(self, "No hay niveles creados en la base de datos.")
        if not operacion_ids:
            return error(self, "Selecciona al menos una operación para la herramienta.")
        if not mats:
            return error(self, "Selecciona al menos 1 material compatible.")
        if not mats_pieza:
            return error(self, "Selecciona al menos 1 material de pieza compatible.")
        if not self.image_path:
            return error(self, "Debes seleccionar una imagen.")
        if stock < self.stock_usado_current:
            return error(self, f"El stock total no puede ser menor al stock de previo uso
({self.stock_usado_current}).")

        ok, msg = self._validate_ranges()
        if not ok:
            return error(self, msg)

        apmin, apmax = self.ap_min.value(), self.ap_max.value()
        params_map = self._collect_selected_material_params()
        first_mpid = mats_pieza[0] if mats_pieza else None
        first_vals = params_map.get(first_mpid, (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0))
```

```
        fn1, vc1min, vc1max, fn2, vc2min, vc2max, fn3, vc3min, vc3max = first_vals
        active_lanes = [
            (float(fn1 or 0), float(vc1min or 0), float(vc1max or 0)),
            (float(fn2 or 0), float(vc2min or 0), float(vc2max or 0)),
            (float(fn3 or 0), float(vc3min or 0), float(vc3max or 0)),
        ]
        active_lanes = [x for x in active_lanes if x[0] > 0 and x[1] > 0 and x[2] > 0]
        if active_lanes:
            fnmin_legacy = min(x[0] for x in active_lanes)
            fnmax_legacy = max(x[0] for x in active_lanes)
            vcmin_legacy = min(x[1] for x in active_lanes)
            vcmax_legacy = max(x[2] for x in active_lanes)
        else:
            fnmin_legacy = 0.0
            fnmax_legacy = 0.0
            vcmin_legacy = 0.0
            vcmax_legacy = 0.0
        stock_nuevo = stock - self.stock_usado_current

        try:
            if self.tool_id:
                dup = fetchall("SELECT id FROM herramientas WHERE codigo=? AND id<>?
LIMIT 1", (codigo, self.tool_id))
                if dup:
                    return error(self, f"Ya existe otra herramienta con código '{codigo}'.")

                execute("""
                    UPDATE herramientas
                    SET codigo=?, nombre=?, maquina_id=?, nivel_id=?, operacion_id=?,
                        stock=?, stock_nuevo=?, ubicacion=?,
                        imagen=?, ap_min=?, ap_max=?, fn_min=?, fn_max=?, vc_min=?,
vc_max=?,
                        fn_1=?, fn_2=?, fn_3=?, vc1_min=?, vc1_max=?, vc2_min=?, vc2_max=?,
vc3_min=?, vc3_max=?
                    WHERE id=?
                """, (codigo, nombre, maquina_id, nivel_id, operacion_id, stock, stock_nuevo,
                      ubic, self.image_path, apmin, apmax, fnmin_legacy, fnmax_legacy,
vcmin_legacy, vcmax_legacy,
                      fn1, fn2, fn3, vc1min, vc1max, vc2min, vc2max, vc3min, vc3max,
self.tool_id))

                execute("DELETE FROM herramienta_operaciones WHERE herramienta_id=?",
(self.tool_id,))
                for oid in operacion_ids:
                    execute(
                        "INSERT OR IGNORE INTO herramienta_operaciones (herramienta_id,
operacion_id) VALUES (?, ?)",
```

```
                    (self.tool_id, int(oid))
                )

            execute("DELETE FROM herramienta_materiales WHERE herramienta_id=?",
(self.tool_id,))
            for mat_id in mats:
                execute("INSERT OR IGNORE INTO herramienta_materiales
(herramienta_id, material_id) VALUES (?, ?)",
                        (self.tool_id, mat_id))
            execute("DELETE FROM herramienta_material_pieza_parametros WHERE
herramienta_id=?", (self.tool_id,))
            for mp_id in mats_pieza:
                p = params_map.get(mp_id, (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0))
                execute("""
                    INSERT OR REPLACE INTO herramienta_material_pieza_parametros
                    (herramienta_id, material_pieza_id, fn_1, vc1_min, vc1_max, fn_2,
vc2_min, vc2_max, fn_3, vc3_min, vc3_max)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
                """, (self.tool_id, mp_id,
                      float(p[0]), float(p[1]), float(p[2]),
                      float(p[3]), float(p[4]), float(p[5]),
                      float(p[6]), float(p[7]), float(p[8])))
            execute("DELETE FROM herramienta_material_piezas WHERE
herramienta_id=?", (self.tool_id,))
            for mp_id in mats_pieza:
                execute(
                    "INSERT OR IGNORE INTO herramienta_material_piezas
(herramienta_id, material_pieza_id) VALUES (?, ?)",
                    (self.tool_id, mp_id)
                )
        else:
            next_hid = get_next_herramienta_id()
            hid = insert_and_get_id("""
                INSERT INTO herramientas
                (id, codigo, nombre, maquina_id, nivel_id, operacion_id, stock, stock_nuevo,
stock_usado, ubicacion,
                 imagen, ap_min, ap_max, fn_min, fn_max, vc_min, vc_max,
                 fn_1, fn_2, fn_3, vc1_min, vc1_max, vc2_min, vc2_max, vc3_min, vc3_max)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?, 0, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
            """, (next_hid, codigo, nombre, maquina_id, nivel_id, operacion_id, stock, stock,
                  ubic, self.image_path, apmin, apmax, fnmin_legacy, fnmax_legacy,
vcmin_legacy, vcmax_legacy,
                  fn1, fn2, fn3, vc1min, vc1max, vc2min, vc2max, vc3min, vc3max))

            for oid in operacion_ids:
                execute(
```

```
                    "INSERT OR IGNORE INTO herramienta_operaciones (herramienta_id,
operacion_id) VALUES (?, ?)",
                    (hid, int(oid))
                )

            for mat_id in mats:
                execute("INSERT OR IGNORE INTO herramienta_materiales
(herramienta_id, material_id) VALUES (?, ?)",
                        (hid, mat_id))
            for mp_id in mats_pieza:
                p = params_map.get(mp_id, (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0))
                execute("""
                    INSERT OR REPLACE INTO herramienta_material_pieza_parametros
                    (herramienta_id, material_pieza_id, fn_1, vc1_min, vc1_max, fn_2,
vc2_min, vc2_max, fn_3, vc3_min, vc3_max)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
                """, (hid, mp_id,
                      float(p[0]), float(p[1]), float(p[2]),
                      float(p[3]), float(p[4]), float(p[5]),
                      float(p[6]), float(p[7]), float(p[8])))
            for mp_id in mats_pieza:
                execute(
                    "INSERT OR IGNORE INTO herramienta_material_piezas
(herramienta_id, material_pieza_id) VALUES (?, ?)",
                    (hid, mp_id)
                )

        except sqlite3.IntegrityError as e:
            return error(self, f"No se pudo guardar (integridad):\n{e}")
        except Exception as e:
            return error(self, f"No se pudo guardar:\n{e}")

        self.saved.emit()
        self.accept()


class ConfirmToolSelectionDialog(QDialog):
    def __init__(self, tool_info: dict, usuarios: list, parent=None):
        super().__init__(parent)
        self.tool_info = tool_info
        self.porta_nombre = str(self.tool_info.get("porta_nombre", "-"))
        self.porta_imagen = str(self.tool_info.get("porta_imagen", ""))
        self.setWindowTitle("Confirmar seleccion de herramienta")
        self.setModal(True)
        self.resize(1020, 700)

        title = QLabel("Confirmacion de herramienta")
```

```
        title.setObjectName("h2")
        subtitle = QLabel("Revisa el conjunto herramienta + portaherramienta y confirma la
salida.")
        subtitle.setObjectName("muted")
        subtitle.setWordWrap(True)

        def build_image_card(section_title: str, section_name: str, image_path: str):
            card = QFrame()
            card.setObjectName("card")
            cl = QVBoxLayout(card)
            cl.setContentsMargins(12, 12, 12, 12)
            cl.setSpacing(8)

            title_lbl = QLabel(section_title)
            title_lbl.setStyleSheet("font-size: 16px; font-weight: 900; color: #FFFFFF;")

            image_lbl = QLabel("Sin imagen")
            image_lbl.setAlignment(Qt.AlignCenter)
            image_lbl.setMinimumHeight(190)
            image_lbl.setStyleSheet("border: 1px solid #2D3B63; border-radius: 10px;")
            update_image_preview(image_lbl, image_path, size=200)

            name_lbl = QLabel(section_name or "-")
            name_lbl.setStyleSheet("font-size: 15px; font-weight: 800; color: #EAF0FF;")
            name_lbl.setWordWrap(True)

            cl.addWidget(title_lbl)
            cl.addWidget(image_lbl)
            cl.addWidget(name_lbl)
            return card

        left = QVBoxLayout()
        left.setSpacing(12)
        left.setContentsMargins(0, 0, 0, 0)
        left.addWidget(build_image_card("Herramienta", str(self.tool_info.get("nombre", "")),
self.tool_info.get("imagen", "")))
        left.addWidget(build_image_card("Portaherramienta", self.porta_nombre,
self.porta_imagen))
        left_w = QWidget()
        left_w.setLayout(left)

        def _fmt_num(v, d=3):
            try:
                return f"{float(v):.{d}f}"
            except Exception:
                return "0"
```

```
def _make_section_title(text: str):
    lbl = QLabel(text)
    lbl.setStyleSheet("font-size: 15px; font-weight: 900; color: #FFFFFF;")
    return lbl

def _make_kv_table(rows):
    t = QTableWidget(len(rows), 2)
    t.setHorizontalHeaderLabels(["Campo", "Valor"])
    t.verticalHeader().setVisible(False)
    t.setSelectionMode(QAbstractItemView.NoSelection)
    t.setFocusPolicy(Qt.NoFocus)
    t.setEditTriggers(QAbstractItemView.NoEditTriggers)
    t.setShowGrid(True)
    t.horizontalHeader().setSectionResizeMode(0, QHeaderView.ResizeToContents)
    t.horizontalHeader().setSectionResizeMode(1, QHeaderView.Stretch)
    t.setMinimumHeight(34 * (len(rows) + 1))
    t.setMaximumHeight(36 * (len(rows) + 1))
    apply_table_polish(t)

    for r, (k, v) in enumerate(rows):
        it_k = QTableWidgetItem(str(k))
        it_k.setTextAlignment(Qt.AlignCenter)
        it_k.setFlags(it_k.flags() & ~Qt.ItemIsSelectable)
        t.setItem(r, 0, it_k)

        it_v = QTableWidgetItem(str(v))
        it_v.setTextAlignment(Qt.AlignCenter)
        it_v.setFlags(it_v.flags() & ~Qt.ItemIsSelectable)
        t.setItem(r, 1, it_v)
    return t

details_card = QFrame()
details_card.setObjectName("card")
details_l = QVBoxLayout(details_card)
details_l.setContentsMargins(14, 14, 14, 14)
details_l.setSpacing(8)
details_l.addWidget(_make_section_title("Resumen"))
details_l.addWidget(_make_kv_table([
    ("Código", self.tool_info.get("codigo", "")),
    ("Máquina", self.tool_info.get("maquina", "")),
    ("Operación", self.tool_info.get("operacion", "")),
    ("Material", self.tool_info.get("materiales", "")),
    ("Portaherramienta", self.porta_nombre),
    ("Ubicación", self.tool_info.get("ubicacion", "")),
]))

inv_card = QFrame()
```

```
        inv_card.setObjectName("card")
        inv_l = QVBoxLayout(inv_card)
        inv_l.setContentsMargins(14, 14, 14, 14)
        inv_l.setSpacing(8)
        inv_l.addWidget(_make_section_title("Inventario"))
        inv_l.addWidget(_make_kv_table([
            ("Stock total", self.tool_info.get("stock", 0)),
            ("Stock nuevo", self.tool_info.get("stock_nuevo", 0)),
            ("Stock previo uso", self.tool_info.get("stock_usado", 0)),
        ]))

        spec_card = QFrame()
        spec_card.setObjectName("card")
        spec_l = QVBoxLayout(spec_card)
        spec_l.setContentsMargins(14, 14, 14, 14)
        spec_l.setSpacing(8)
        spec_l.addWidget(_make_section_title("Especificaciones de corte"))

        spec_table = QTableWidget(4, 4)
        spec_table.setHorizontalHeaderLabels(["Línea", "Fn (mm/rev)", "Vc mín (m/min)",
"Vc máx (m/min)"])
        spec_table.verticalHeader().setVisible(False)
        spec_table.setSelectionMode(QAbstractItemView.NoSelection)
        spec_table.setFocusPolicy(Qt.NoFocus)
        spec_table.setEditTriggers(QAbstractItemView.NoEditTriggers)
        spec_table.horizontalHeader().setSectionResizeMode(0,
QHeaderView.ResizeToContents)
        spec_table.horizontalHeader().setSectionResizeMode(1, QHeaderView.Stretch)
        spec_table.horizontalHeader().setSectionResizeMode(2, QHeaderView.Stretch)
        spec_table.horizontalHeader().setSectionResizeMode(3, QHeaderView.Stretch)
        spec_table.setMinimumHeight(190)
        spec_table.setMaximumHeight(230)
        apply_table_polish(spec_table)

        spec_rows = [
            ("Ap", f"{_fmt_num(self.tool_info.get('ap_min', 0), 3)}",
f"{_fmt_num(self.tool_info.get('ap_max', 0), 3)}"),
            ("Fn1", _fmt_num(self.tool_info.get("fn_1", 0), 4),
f"{_fmt_num(self.tool_info.get('vc1_min', 0), 2)}",
f"{_fmt_num(self.tool_info.get('vc1_max', 0), 2)}"),
            ("Fn2", _fmt_num(self.tool_info.get("fn_2", 0), 4),
f"{_fmt_num(self.tool_info.get('vc2_min', 0), 2)}",
f"{_fmt_num(self.tool_info.get('vc2_max', 0), 2)}"),
            ("Fn3", _fmt_num(self.tool_info.get("fn_3", 0), 4),
f"{_fmt_num(self.tool_info.get('vc3_min', 0), 2)}",
f"{_fmt_num(self.tool_info.get('vc3_max', 0), 2)}"),
        ]
```

```
for r, data in enumerate(spec_rows):
    if r == 0:
        values = [data[0], "-", data[1], data[2]]
    else:
        values = [data[0], data[1], data[2], data[3]]
    for c, v in enumerate(values):
        it = QTableWidgetItem(str(v))
        it.setTextAlignment(Qt.AlignCenter)
        it.setFlags(it.flags() & ~Qt.ItemIsSelectable)
        spec_table.setItem(r, c, it)
spec_l.addWidget(spec_table)

self.cb_usuario = QComboBox()
self.cb_usuario.addItem("Selecciona un usuario...", "")
for u in usuarios:
    self.cb_usuario.addItem(u)
self.cb_usuario.setCurrentIndex(0)

self.cb_tipo = QComboBox()
self.cb_tipo.addItem("Selecciona el tipo...", "")
self.cb_tipo.addItem("Nueva", "NUEVA")
if int(self.tool_info.get("stock_usado", 0) or 0) > 0:
    self.cb_tipo.addItem("Con desgaste (previo uso)", "DESGASTE")
self.cb_tipo.setCurrentIndex(0)

confirm_card = QFrame()
confirm_card.setObjectName("card")
confirm_form = QFormLayout(confirm_card)
confirm_form.setContentsMargins(14, 14, 14, 14)
confirm_form.setVerticalSpacing(10)
confirm_form.addRow("Usuario responsable *:", self.cb_usuario)
confirm_form.addRow("Tipo de herramienta *:", self.cb_tipo)

right = QVBoxLayout()
right.setSpacing(12)
right.setContentsMargins(0, 0, 0, 0)
right.addWidget(details_card)
right.addWidget(inv_card)
right.addWidget(spec_card)
right.addWidget(confirm_card)
right.addStretch(1)
right_w = QWidget()
right_w.setLayout(right)

center = QHBoxLayout()
center.setSpacing(14)
```

```
        center.addWidget(left_w, 1)
        center.addWidget(right_w, 1)

        btn_cancel = QPushButton("Cancelar")
        btn_cancel.clicked.connect(self.reject)
        btn_ok = QPushButton("Confirmar seleccion")
        btn_ok.setObjectName("primary")
        btn_ok.clicked.connect(self._on_confirm)

        actions = QHBoxLayout()
        actions.addStretch(1)
        actions.addWidget(btn_cancel)
        actions.addWidget(btn_ok)

        content = QWidget()
        content_lay = QVBoxLayout(content)
        content_lay.setContentsMargins(0, 0, 0, 0)
        content_lay.setSpacing(12)
        content_lay.addWidget(title)
        content_lay.addWidget(subtitle)
        content_lay.addLayout(center)

        scroll = QScrollArea()
        scroll.setWidgetResizable(True)
        scroll.setFrameShape(QFrame.NoFrame)
        scroll.setWidget(content)

        lay = QVBoxLayout(self)
        lay.setContentsMargins(16, 16, 16, 16)
        lay.setSpacing(10)
        lay.addWidget(scroll, 1)
        lay.addLayout(actions)

    def _on_confirm(self):
        if not self.selected_usuario():
            return info(self, "Debes seleccionar un usuario responsable.")
        tipo = self.selected_tipo_salida()
        if not tipo:
            return info(self, "Debes seleccionar el tipo de herramienta.")
        if tipo == "NUEVA" and int(self.tool_info.get("stock_nuevo", 0) or 0) <= 0:
            return info(self, "No hay stock nuevo disponible para esta herramienta.")
        if tipo == "DESGASTE" and int(self.tool_info.get("stock_usado", 0) or 0) <= 0:
            return info(self, "No hay stock de previo uso disponible para esta herramienta.")
        self.accept()

    def selected_usuario(self) -> str:
```

```
        val = str(self.cb_usuario.currentData() if self.cb_usuario.currentData() is not None
else "").strip()
        if val:
            return val
        txt = (self.cb_usuario.currentText() or "").strip()
        if txt.lower().startswith("selecciona"):
            return ""
        return txt

    def selected_tipo_salida(self) -> str:
        return str(self.cb_tipo.currentData() or "").strip()

class SimpleCatalogPage(QWidget):
    def __init__(self, title: str, table: str, placeholder: str):
        super().__init__()
        self.table = table
        self.placeholder = placeholder
        self.require_image = self.table != "usuarios"

        self.lbl = QLabel(title)
        self.lbl.setObjectName("h2")

        self.list = QListWidget()
        self.list.setSelectionMode(QAbstractItemView.SingleSelection)

        self.btn_add = QPushButton("Agregar")
        self.btn_add.setObjectName("primary")
        self.btn_edit = QPushButton("Editar")
        self.btn_edit.setObjectName("primary")
        self.btn_del = QPushButton("Eliminar")
        self.btn_del.setObjectName("danger")

        self.btn_add.clicked.connect(self.add)
        self.btn_edit.clicked.connect(self.edit)
        self.btn_del.clicked.connect(self.delete)

        row = QHBoxLayout()
        row.addWidget(self.btn_add)
        row.addWidget(self.btn_edit)
        row.addWidget(self.btn_del)
        row.addStretch(1)

        lay = QVBoxLayout(self)
        lay.addWidget(self.lbl)
        lay.addWidget(self.list, 1)
        lay.addLayout(row)
```

```
        self.refresh()

    def refresh(self):
        self.list.clear()
        if self.require_image:
            rows = fetchall(f"SELECT id, nombre, COALESCE(imagen,'') FROM {self.table}
ORDER BY nombre")
        else:
            rows = [(rid, n, "") for rid, n in fetchall(f"SELECT id, nombre FROM {self.table}
ORDER BY nombre")]

        for rid, n, img in rows:
            it = QListWidgetItem(n)
            it.setData(Qt.UserRole, rid)
            it.setData(Qt.UserRole + 1, img)
            self.list.addItem(it)

    def add(self):
        dlg = AddSimpleDialog(
            "Agregar",
            self.placeholder,
            self,
            require_image=self.require_image
        )
        if dlg.exec() != QDialog.Accepted:
            return
        name, image_path = dlg.values()

        try:
            if self.require_image:
                execute(f"INSERT INTO {self.table}(nombre, imagen) VALUES (?, ?)", (name,
image_path))
            else:
                execute(f"INSERT INTO {self.table}(nombre) VALUES (?)", (name,))
        except Exception as e:
            return error(self, f"No se pudo guardar:\n{e}")
        self.refresh()

    def edit(self):
        it = self.list.currentItem()
        if not it:
            return info(self, "Selecciona un registro.")

        rid = it.data(Qt.UserRole)
        name = it.text()
        image_path = it.data(Qt.UserRole + 1) or ""
```

```
        dlg = AddSimpleDialog(
            "Editar",
            self.placeholder,
            self,
            initial_name=name,
            initial_image=image_path,
            require_image=self.require_image
        )
        if dlg.exec() != QDialog.Accepted:
            return

        new_name, new_image = dlg.values()
        try:
            if self.require_image:
                execute(f"UPDATE {self.table} SET nombre=?, imagen=? WHERE id=?",
(new_name, new_image, rid))
            else:
                execute(f"UPDATE {self.table} SET nombre=? WHERE id=?", (new_name,
rid))
        except Exception as e:
            return error(self, f"No se pudo actualizar:\n{e}")
        self.refresh()

    def delete(self):
        it = self.list.currentItem()
        if not it:
            return info(self, "Selecciona un registro.")
        rid = it.data(Qt.UserRole)
        name = it.text()

        if not confirm(self, f"¿Eliminar '{name}'?"):
            return
        try:
            execute(f"DELETE FROM {self.table} WHERE id=?", (rid,))
        except Exception as e:
            return error(self, f"No se pudo eliminar:\n{e}")
        self.refresh()


class OperacionesPage(QWidget):
    def __init__(self):
        super().__init__()

        self.lbl = QLabel("Operaciones (dependen de máquina)")
        self.lbl.setObjectName("h2")

        self.cb_maquina = QComboBox()
```

```
        self.cb_maquina.currentIndexChanged.connect(self.refresh)

        self.list = QListWidget()
        self.list.setSelectionMode(QAbstractItemView.SingleSelection)

        self.btn_add = QPushButton("Agregar")
        self.btn_add.setObjectName("primary")
        self.btn_edit = QPushButton("Editar")
        self.btn_edit.setObjectName("primary")
        self.btn_del = QPushButton("Eliminar")
        self.btn_del.setObjectName("danger")

        self.btn_add.clicked.connect(self.add)
        self.btn_edit.clicked.connect(self.edit)
        self.btn_del.clicked.connect(self.delete)

        top = QHBoxLayout()
        top.addWidget(QLabel("Máquina:"))
        top.addWidget(self.cb_maquina, 1)

        row = QHBoxLayout()
        row.addWidget(self.btn_add)
        row.addWidget(self.btn_edit)
        row.addWidget(self.btn_del)
        row.addStretch(1)

        lay = QVBoxLayout(self)
        lay.addWidget(self.lbl)
        lay.addLayout(top)
        lay.addWidget(self.list, 1)
        lay.addLayout(row)

        self.reload_maquinas()
        self.refresh()

    def reload_maquinas(self):
        self.cb_maquina.clear()
        for mid, n in fetchall("SELECT id, nombre FROM maquinas ORDER BY nombre"):
            self.cb_maquina.addItem(n, mid)

    def refresh(self):
        self.list.clear()
        maquina_id = self.cb_maquina.currentData()
        if maquina_id is None:
            return
        for oid, n, img in fetchall(
```

```
            "SELECT id, nombre, COALESCE(imagen,'') FROM operaciones WHERE
maquina_id=? ORDER BY nombre",
            (maquina_id,)
        ):
            it = QListWidgetItem(n)
            it.setData(Qt.UserRole, oid)
            it.setData(Qt.UserRole + 1, img or "")
            self.list.addItem(it)

    def add(self):
        dlg = AddOperacionDialog(self)
        if dlg.exec() != QDialog.Accepted:
            return
        maquina_id, nombre, image_path = dlg.values()
        try:
            execute("INSERT INTO operaciones(maquina_id, nombre, imagen) VALUES (?, ?,
?)", (maquina_id, nombre, image_path))
        except Exception as e:
            return error(self, f"No se pudo guardar:\n{e}")

        idx = self.cb_maquina.findData(maquina_id)
        if idx >= 0:
            self.cb_maquina.setCurrentIndex(idx)
        self.refresh()

    def edit(self):
        it = self.list.currentItem()
        if not it:
            return info(self, "Selecciona una operación.")

        oid = it.data(Qt.UserRole)
        nombre = it.text()
        imagen = it.data(Qt.UserRole + 1) or ""
        maquina_id = self.cb_maquina.currentData()

        dlg = AddOperacionDialog(
            self,
            initial_maquina_id=maquina_id,
            initial_nombre=nombre,
            initial_image=imagen
        )
        if dlg.exec() != QDialog.Accepted:
            return
        new_maquina_id, new_nombre, new_imagen = dlg.values()

        try:
            execute(
```

```
            "UPDATE operaciones SET maquina_id=?, nombre=?, imagen=? WHERE
id=?",
            (new_maquina_id, new_nombre, new_imagen, oid)
          )
        except Exception as e:
          return error(self, f"No se pudo actualizar:\n{e}")

        idx = self.cb_maquina.findData(new_maquina_id)
        if idx >= 0:
          self.cb_maquina.setCurrentIndex(idx)
        self.refresh()

    def delete(self):
        it = self.list.currentItem()
        if not it:
          return info(self, "Selecciona una operación.")
        oid = it.data(Qt.UserRole)
        name = it.text()

        if not confirm(self, f"¿Eliminar operación '{name}'?"):
          return
        try:
          execute("DELETE FROM operaciones WHERE id=?", (oid,))
        except Exception as e:
          return error(self, f"No se pudo eliminar:\n{e}")
        self.refresh()


class MaterialesPiezaPage(QWidget):
    def __init__(self):
        super().__init__()

        self.lbl = QLabel("Materiales de pieza")
        self.lbl.setObjectName("h2")

        self.cb_filter_padre = QComboBox()
        self.cb_filter_padre.currentIndexChanged.connect(self.refresh)

        self.lbl_count = QLabel("")
        self.lbl_count.setObjectName("muted")

        self.table = QTableWidget(0, 6)
        self.table.setHorizontalHeaderLabels(["ID material", "Material padre", "Material de
pieza", "Rango ISO", "HB mín", "HB máx"])
        self.table.setSelectionBehavior(QAbstractItemView.SelectRows)
        self.table.setSelectionMode(QAbstractItemView.SingleSelection)
        self.table.setEditTriggers(QAbstractItemView.NoEditTriggers)
```

```
        self.table.verticalHeader().setVisible(False)
        self.table.horizontalHeader().setSectionResizeMode(0, QHeaderView.Fixed)
        self.table.horizontalHeader().setSectionResizeMode(1, QHeaderView.Stretch)
        self.table.horizontalHeader().setSectionResizeMode(2, QHeaderView.Stretch)
        self.table.horizontalHeader().setSectionResizeMode(3, QHeaderView.Fixed)
        self.table.horizontalHeader().setSectionResizeMode(4, QHeaderView.Fixed)
        self.table.horizontalHeader().setSectionResizeMode(5, QHeaderView.Fixed)
        self.table.setColumnWidth(0, 150)
        self.table.setColumnWidth(3, 170)
        self.table.setColumnWidth(4, 130)
        self.table.setColumnWidth(5, 130)
        apply_table_polish(self.table)

        self.btn_add = QPushButton("Agregar")
        self.btn_add.setObjectName("primary")
        self.btn_edit = QPushButton("Editar")
        self.btn_edit.setObjectName("primary")
        self.btn_del = QPushButton("Eliminar")
        self.btn_del.setObjectName("danger")

        self.btn_add.clicked.connect(self.add)
        self.btn_edit.clicked.connect(self.edit)
        self.btn_del.clicked.connect(self.delete)

        filter_row = QHBoxLayout()
        filter_row.addWidget(QLabel("Filtrar por material padre:"))
        filter_row.addWidget(self.cb_filter_padre, 1)
        filter_row.addWidget(self.lbl_count)

        row = QHBoxLayout()
        row.addWidget(self.btn_add)
        row.addWidget(self.btn_edit)
        row.addWidget(self.btn_del)
        row.addStretch(1)

        lay = QVBoxLayout(self)
        lay.addWidget(self.lbl)
        lay.addLayout(filter_row)
        lay.addWidget(self.table, 1)
        lay.addLayout(row)

        self._load_filter_padres()
        self.refresh()

    def _display_text(self, value: str) -> str:
        s = (value or "").strip()
        if not s:
```

```
            return ""
        return s[0].upper() + s[1:].lower()

    def _load_filter_padres(self):
        self.cb_filter_padre.clear()
        self.cb_filter_padre.addItem("Todos", None)
        for mid, n in fetchall("SELECT id, nombre FROM materiales ORDER BY nombre"):
            self.cb_filter_padre.addItem(self._display_text(n), mid)

    def refresh(self):
        self.table.setRowCount(0)
        material_padre_id = self.cb_filter_padre.currentData()
        if material_padre_id is None:
            rows = fetchall(
                """
                SELECT
                    mp.id,
                    COALESCE(mp.codigo,''),
                    mp.nombre,
                    m.id,
                    m.nombre,
                    COALESCE(mp.iso_min,0),
                    COALESCE(mp.iso_max,0),
                    COALESCE(mp.hb_min, COALESCE(mp.hb,0)),
                    COALESCE(mp.hb_max, COALESCE(mp.hb,0))
                FROM material_piezas mp
                JOIN materiales m ON m.id = mp.material_padre_id
                ORDER BY
                    CASE
                        WHEN TRIM(COALESCE(mp.codigo,'')) GLOB '[0-9]*'
                            AND TRIM(COALESCE(mp.codigo,'')) <> '' THEN 0
                        ELSE 1
                    END,
                    CASE
                        WHEN TRIM(COALESCE(mp.codigo,'')) GLOB '[0-9]*'
                            AND TRIM(COALESCE(mp.codigo,'')) <> '' THEN
CAST(TRIM(mp.codigo) AS INTEGER)
                        ELSE 0
                    END,
                    UPPER(TRIM(COALESCE(mp.codigo,''))),
                    UPPER(mp.nombre)
                """
            )
        else:
            rows = fetchall(
                """
                SELECT
```

```
                mp.id,
                COALESCE(mp.codigo,''),
                mp.nombre,
                m.id,
                m.nombre,
                COALESCE(mp.iso_min,0),
                COALESCE(mp.iso_max,0),
                COALESCE(mp.hb_min, COALESCE(mp.hb,0)),
                COALESCE(mp.hb_max, COALESCE(mp.hb,0))
            FROM material_piezas mp
            JOIN materiales m ON m.id = mp.material_padre_id
            WHERE m.id=?
            ORDER BY
                CASE
                    WHEN TRIM(COALESCE(mp.codigo,'')) GLOB '[0-9]*'
                        AND TRIM(COALESCE(mp.codigo,'')) <> '' THEN 0
                    ELSE 1
                END,
                CASE
                    WHEN TRIM(COALESCE(mp.codigo,'')) GLOB '[0-9]*'
                        AND TRIM(COALESCE(mp.codigo,'')) <> '' THEN
CAST(TRIM(mp.codigo) AS INTEGER)
                    ELSE 0
                END,
                UPPER(TRIM(COALESCE(mp.codigo,''))),
                UPPER(mp.nombre)
            """,
            (material_padre_id,)
        )

        self.lbl_count.setText(f"Registros: {len(rows)}")

        for pid, codigo, n, mid, mname, iso_min, iso_max, hb_min, hb_max in rows:
            r = self.table.rowCount()
            self.table.insertRow(r)

            it_codigo = QTableWidgetItem(str(codigo or ""))
            it_codigo.setData(Qt.UserRole, pid)
            it_codigo.setData(Qt.UserRole + 1, mid)
            it_codigo.setData(Qt.UserRole + 2, str(codigo or ""))
            it_codigo.setData(Qt.UserRole + 3, n)
            it_codigo.setData(Qt.UserRole + 4, int(iso_min or 0))
            it_codigo.setData(Qt.UserRole + 5, int(iso_max or 0))
            it_codigo.setData(Qt.UserRole + 6, float(hb_min or 0))
            it_codigo.setData(Qt.UserRole + 7, float(hb_max or 0))
            it_codigo.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 0, it_codigo)
```

```
            it_padre = QTableWidgetItem(self._display_text(mname))
            it_padre.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 1, it_padre)

            it_pieza = QTableWidgetItem(self._display_text(n))
            it_pieza.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 2, it_pieza)

            iso_prefix = (mname or "").strip().split(" ")[0].upper()[:1]
            iso_prefix = iso_prefix if iso_prefix in ("P", "M", "K", "N", "S", "H") else "ISO"
            iso_min_val = int(iso_min or 0)
            iso_max_val = int(iso_max or 0)
            iso_txt = f"{iso_prefix}{iso_min_val}-{iso_prefix}{iso_max_val}" if iso_min_val
> 0 and iso_max_val > 0 else "-"
            it_iso = QTableWidgetItem(iso_txt)
            it_iso.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 3, it_iso)

            hb_min_val = float(hb_min or 0.0)
            hb_max_val = float(hb_max or 0.0)
            hb_min_txt = "-" if hb_min_val <= 0 else (f"{int(hb_min_val)}" if
hb_min_val.is_integer() else f"{hb_min_val:.1f}")
            hb_max_txt = "-" if hb_max_val <= 0 else (f"{int(hb_max_val)}" if
hb_max_val.is_integer() else f"{hb_max_val:.1f}")
            it_hb_min = QTableWidgetItem(hb_min_txt)
            it_hb_min.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 4, it_hb_min)
            it_hb_max = QTableWidgetItem(hb_max_txt)
            it_hb_max.setTextAlignment(Qt.AlignCenter)
            self.table.setItem(r, 5, it_hb_max)

    def add(self):
        dlg = AddMaterialPiezaDialog(self)
        if dlg.exec() != QDialog.Accepted:
            return
        material_padre_id, codigo, nombre, iso_min, iso_max, hb_min, hb_max = dlg.values()
        try:
            execute(
                "INSERT INTO material_piezas(material_padre_id, codigo, nombre, iso_min,
iso_max, hb_min, hb_max) VALUES (?, ?, ?, ?, ?, ?, ?)",
                (material_padre_id, codigo, nombre, iso_min, iso_max, hb_min, hb_max)
            )
        except Exception as e:
            msg = str(e)
            if "material_piezas.codigo" in msg or "ux_material_piezas_codigo" in msg:
```

```
            return error(self, f"No se pudo guardar:\nEl ID material '{codigo}' ya existe.
Debe ser único.")
        return error(self, f"No se pudo guardar:\n{e}")
    self.refresh()

  def edit(self):
    row = self.table.currentRow()
    if row < 0:
        return info(self, "Selecciona un material de pieza.")

    it = self.table.item(row, 0)
    if not it:
        return info(self, "Selecciona un material de pieza.")

    pid = it.data(Qt.UserRole)
    material_padre_id = it.data(Qt.UserRole + 1)
    codigo = it.data(Qt.UserRole + 2) or ""
    nombre = it.data(Qt.UserRole + 3) or ""
    iso_min = int(it.data(Qt.UserRole + 4) or 0)
    iso_max = int(it.data(Qt.UserRole + 5) or 0)
    hb_min = float(it.data(Qt.UserRole + 6) or 0.0)
    hb_max = float(it.data(Qt.UserRole + 7) or 0.0)

    dlg = AddMaterialPiezaDialog(
        self,
        initial_material_padre_id=material_padre_id,
        initial_codigo=codigo,
        initial_nombre=nombre,
        initial_iso_min=iso_min,
        initial_iso_max=iso_max,
        initial_hb_min=hb_min,
        initial_hb_max=hb_max
    )
    if dlg.exec() != QDialog.Accepted:
        return
    new_material_padre_id, new_codigo, new_nombre, new_iso_min, new_iso_max,
new_hb_min, new_hb_max = dlg.values()
    try:
        execute(
            "UPDATE material_piezas SET material_padre_id=?, codigo=?, nombre=?,
iso_min=?, iso_max=?, hb_min=?, hb_max=? WHERE id=?",
            (new_material_padre_id, new_codigo, new_nombre, new_iso_min,
new_iso_max, new_hb_min, new_hb_max, pid)
        )
    except Exception as e:
        msg = str(e)
        if "material_piezas.codigo" in msg or "ux_material_piezas_codigo" in msg:
```

```
            return error(self, f"No se pudo actualizar:\nEl ID material '{new_codigo}' ya
existe. Debe ser único.")
            return error(self, f"No se pudo actualizar:\n{e}")
        self.refresh()

    def delete(self):
        row = self.table.currentRow()
        if row < 0:
            return info(self, "Selecciona un material de pieza.")
        it0 = self.table.item(row, 0)
        it1 = self.table.item(row, 2)
        if not it0:
            return info(self, "Selecciona un material de pieza.")
        pid = it0.data(Qt.UserRole)
        padre = it0.text().strip()
        name = it1.text().strip() if it1 else ""

        if not confirm(self, f"¿Eliminar material de pieza '{name}' de la familia '{padre}'?"):
            return
        try:
            execute("DELETE FROM material_piezas WHERE id=?", (pid,))
        except Exception as e:
            return error(self, f"No se pudo eliminar:\n{e}")
        self.refresh()


class PortaHerramientasPage(QWidget):
    def __init__(self):
        super().__init__()

        self.lbl = QLabel("Portaherramientas")
        self.lbl.setObjectName("h2")

        self.list = QListWidget()
        self.list.setSelectionMode(QAbstractItemView.SingleSelection)
        self.list.currentItemChanged.connect(self._refresh_herramientas_asociadas)

        self.list_tools = QListWidget()
        self.list_tools.setSelectionMode(QAbstractItemView.NoSelection)

        self.btn_add = QPushButton("Agregar")
        self.btn_add.setObjectName("primary")
        self.btn_edit = QPushButton("Editar")
        self.btn_edit.setObjectName("primary")
        self.btn_del = QPushButton("Eliminar")
        self.btn_del.setObjectName("danger")
```

```
        self.btn_add.clicked.connect(self.add)
        self.btn_edit.clicked.connect(self.edit)
        self.btn_del.clicked.connect(self.delete)

        left_col = QVBoxLayout()
        left_col.addWidget(QLabel("Portaherramientas registrados"))
        left_col.addWidget(self.list, 1)

        row = QHBoxLayout()
        row.addWidget(self.btn_add)
        row.addWidget(self.btn_edit)
        row.addWidget(self.btn_del)
        row.addStretch(1)
        left_col.addLayout(row)

        right_col = QVBoxLayout()
        right_col.addWidget(QLabel("Herramientas asociadas"))
        right_col.addWidget(self.list_tools, 1)

        body = QHBoxLayout()
        body.addLayout(left_col, 3)
        body.addLayout(right_col, 2)

        lay = QVBoxLayout(self)
        lay.addWidget(self.lbl)
        lay.addLayout(body, 1)

        self.refresh()

    def _associated_tool_ids(self, portah_id: int) -> list:
        return [
            x[0] for x in fetchall(
                "SELECT herramienta_id FROM portaherramienta_herramientas WHERE
portaherramienta_id=?",
                (portah_id,)
            )
        ]

    def refresh(self):
        self.list.clear()
        rows = fetchall("""
            SELECT p.id, p.nombre, COALESCE(p.imagen,''), COUNT(ph.herramienta_id)
AS total_herr
            FROM portaherramientas p
            LEFT JOIN portaherramienta_herramientas ph ON ph.portaherramienta_id = p.id
            GROUP BY p.id, p.nombre, p.imagen
            ORDER BY p.nombre
```

```
        """)

        for pid, nombre, imagen, total in rows:
            it = QListWidgetItem(f"{nombre}  |  {total} herramienta(s)")
            it.setData(Qt.UserRole, pid)
            it.setData(Qt.UserRole + 1, nombre)
            it.setData(Qt.UserRole + 2, imagen or "")
            self.list.addItem(it)

        self._refresh_herramientas_asociadas()

    def _refresh_herramientas_asociadas(self):
        self.list_tools.clear()
        it = self.list.currentItem()
        if not it:
            self.list_tools.addItem("Sin selección")
            return

        pid = it.data(Qt.UserRole)
        rows = fetchall("""
            SELECT h.codigo, h.nombre
            FROM portaherramienta_herramientas ph
            JOIN herramientas h ON h.id = ph.herramienta_id
            WHERE ph.portaherramienta_id=?
            ORDER BY h.nombre, h.codigo
        """, (pid,))

        if not rows:
            self.list_tools.addItem("Sin herramientas asociadas")
            return

        for codigo, nombre in rows:
            self.list_tools.addItem(f"{codigo} - {nombre}")

    def add(self):
        has_tools = fetchall("SELECT 1 FROM herramientas LIMIT 1")
        if not has_tools:
            return info(self, "Primero debes crear al menos una herramienta en el catálogo de
herramientas.")

        dlg = AddPortaHerramientaDialog(self)
        if dlg.exec() != QDialog.Accepted:
            return

        nombre, imagen, tool_ids = dlg.values()
        try:
            pid = insert_and_get_id(
```

```
                "INSERT INTO portaherramientas(nombre, imagen) VALUES (?, ?)",
                (nombre, imagen)
            )
            for hid in tool_ids:
                execute(
                    "INSERT OR IGNORE INTO portaherramienta_herramientas
(portaherramienta_id, herramienta_id) VALUES (?, ?)",
                    (pid, hid)
                )
        except Exception as e:
            return error(self, f"No se pudo guardar:\n{e}")
        self.refresh()

    def edit(self):
        it = self.list.currentItem()
        if not it:
            return info(self, "Selecciona un portaherramienta.")

        pid = it.data(Qt.UserRole)
        nombre = it.data(Qt.UserRole + 1) or ""
        imagen = it.data(Qt.UserRole + 2) or ""
        tool_ids = self._associated_tool_ids(pid)

        dlg = AddPortaHerramientaDialog(
            self,
            initial_nombre=nombre,
            initial_image=imagen,
            initial_tool_ids=tool_ids
        )
        if dlg.exec() != QDialog.Accepted:
            return

        new_nombre, new_imagen, new_tool_ids = dlg.values()
        try:
            execute(
                "UPDATE portaherramientas SET nombre=?, imagen=? WHERE id=?",
                (new_nombre, new_imagen, pid)
            )
            execute("DELETE FROM portaherramienta_herramientas WHERE
portaherramienta_id=?", (pid,))
            for hid in new_tool_ids:
                execute(
                    "INSERT OR IGNORE INTO portaherramienta_herramientas
(portaherramienta_id, herramienta_id) VALUES (?, ?)",
                    (pid, hid)
                )
        except Exception as e:
```

```
            return error(self, f"No se pudo actualizar:\n{e}")
        self.refresh()

    def delete(self):
        it = self.list.currentItem()
        if not it:
            return info(self, "Selecciona un portaherramienta.")
        pid = it.data(Qt.UserRole)
        name = it.data(Qt.UserRole + 1) or it.text()

        if not confirm(self, f"¿Eliminar portaherramienta '{name}'?"):
            return
        try:
            execute("DELETE FROM portaherramientas WHERE id=?", (pid,))
        except Exception as e:
            return error(self, f"No se pudo eliminar:\n{e}")
        self.refresh()


class HerramientasPage(QWidget):
    def __init__(self):
        super().__init__()

        self.lbl = QLabel("Herramientas")
        self.lbl.setObjectName("h2")

        self.table = QTableWidget(0, 22)
        self.table.setHorizontalHeaderLabels([
            "ID", "Código", "Nombre", "Máquina", "Nivel",
            "Operación", "Materiales herramienta", "Materiales pieza", "Stock", "Ubicación",
"Imagen",
            "Ap_min", "Ap_max",
            "Fn1", "Vc1 mín", "Vc1 máx",
            "Fn2", "Vc2 mín", "Vc2 máx",
            "Fn3", "Vc3 mín", "Vc3 máx"
        ])
        self.table.setSelectionBehavior(QAbstractItemView.SelectRows)
        self.table.setSelectionMode(QAbstractItemView.SingleSelection)
        self.table.setEditTriggers(QAbstractItemView.NoEditTriggers)
        self.table.verticalHeader().setVisible(False)
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Interactive)
        self.table.horizontalHeader().setStretchLastSection(False)
        self.table.horizontalHeader().setMinimumSectionSize(90)
        self.table.setHorizontalScrollMode(QAbstractItemView.ScrollPerPixel)
        self.table.setHorizontalScrollBarPolicy(Qt.ScrollBarAsNeeded)
        apply_table_polish(self.table)
```

```
        self.btn_add = QPushButton("Agregar")
        self.btn_add.setObjectName("primary")
        self.btn_edit = QPushButton("Editar")
        self.btn_edit.setObjectName("primary")
        self.btn_del = QPushButton("Eliminar")
        self.btn_del.setObjectName("danger")

        self.btn_export = QPushButton("Descargar Excel")
        self.btn_export.setObjectName("primary")
        self.btn_import = QPushButton("Cargar Excel")
        self.btn_import.setObjectName("primary")

        self.btn_add.clicked.connect(self.add)
        self.btn_edit.clicked.connect(self.edit)
        self.btn_del.clicked.connect(self.delete)
        self.btn_export.clicked.connect(self.export_excel)
        self.btn_import.clicked.connect(self.import_excel)

        row = QHBoxLayout()
        row.addWidget(self.btn_add)
        row.addWidget(self.btn_edit)
        row.addWidget(self.btn_del)
        row.addSpacing(16)
        row.addWidget(self.btn_export)
        row.addWidget(self.btn_import)
        row.addStretch(1)

        content = QWidget()
        lay = QVBoxLayout(content)
        lay.setContentsMargins(0, 0, 0, 0)
        lay.addWidget(self.lbl)
        lay.addWidget(self.table, 1)
        lay.addLayout(row)

        root = QVBoxLayout(self)
        root.addWidget(make_card(content), 1)

        self._apply_column_widths()

        self.refresh()

    def _apply_column_widths(self):
        widths = {
            0: 70,
            1: 150,
            2: 230,
            3: 150,
```

```
            4: 120,
            5: 170,
            6: 230,
            7: 280,
            8: 90,
            9: 170,
            10: 210,
            11: 95,
            12: 95,
            13: 95,
            14: 95,
            15: 95,
            16: 95,
            17: 95,
            18: 95,
            19: 95,
            20: 95,
            21: 95,
        }
        for col, width in widths.items():
            self.table.setColumnWidth(col, width)

    def refresh(self):
        rows = fetchall("""
            SELECT
                h.id, h.codigo, h.nombre,
                mq.nombre, nv.nombre,
                COALESCE(
                    (
                        SELECT GROUP_CONCAT(o2.nombre, ', ')
                        FROM herramienta_operaciones ho
                        JOIN operaciones o2 ON o2.id = ho.operacion_id
                        WHERE ho.herramienta_id = h.id
                    ),
                    op.nombre
                ) AS operaciones_txt,
                h.stock, COALESCE(h.ubicacion,''), COALESCE(h.imagen,''),
                COALESCE(h.ap_min,0), COALESCE(h.ap_max,0),
                COALESCE(h.fn_1, COALESCE(h.fn_min,0)),
                COALESCE(h.fn_2, CASE WHEN COALESCE(h.fn_min,0)>0 AND
COALESCE(h.fn_max,0)>0 THEN
(COALESCE(h.fn_min,0)+COALESCE(h.fn_max,0))/2.0 ELSE 0 END),
                COALESCE(h.fn_3, COALESCE(h.fn_max,0)),
                COALESCE(h.vc1_min, COALESCE(h.vc_min,0)), COALESCE(h.vc1_max,
COALESCE(h.vc_max,0)),
                COALESCE(h.vc2_min, COALESCE(h.vc_min,0)), COALESCE(h.vc2_max,
COALESCE(h.vc_max,0)),
```

```
            COALESCE(h.vc3_min, COALESCE(h.vc_min,0)), COALESCE(h.vc3_max,
COALESCE(h.vc_max,0))
        FROM herramientas h
        JOIN maquinas mq ON mq.id=h.maquina_id
        JOIN niveles nv ON nv.id=h.nivel_id
        LEFT JOIN operaciones op ON op.id=h.operacion_id
        ORDER BY h.id DESC
    """)

    self.table.setRowCount(0)

    for (hid, codigo, nombre, maq, niv, oper, stock, ubic, imagen,
         apmin, apmax, fn1, fn2, fn3, vc1min, vc1max, vc2min, vc2max, vc3min, vc3max)
in rows:
        mats = fetchall("""
            SELECT m.nombre
            FROM herramienta_materiales hm
            JOIN materiales m ON m.id = hm.material_id
            WHERE hm.herramienta_id=?
            ORDER BY m.nombre
        """, (hid,))
        mats_txt = ",".join([x[0] for x in mats])
        mats_pieza = fetchall("""
            SELECT mp.nombre
            FROM herramienta_material_piezas hmp
            JOIN material_piezas mp ON mp.id = hmp.material_pieza_id
            WHERE hmp.herramienta_id=?
            ORDER BY mp.nombre
        """, (hid,))
        mats_pieza_txt = ",".join([x[0] for x in mats_pieza])

        r = self.table.rowCount()
        self.table.insertRow(r)

        def fmt_num(v, dec=2):
            vv = float(v or 0.0)
            if vv <= 0:
                return "-"
            txt = f"{vv:.{dec}f}".rstrip("0").rstrip(".")
            return txt
        values = [
            hid, codigo, nombre, maq, niv, oper,
            mats_txt, mats_pieza_txt, stock, ubic, imagen,
            fmt_num(apmin, 3), fmt_num(apmax, 3),
            fmt_num(fn1, 4), fmt_num(vc1min, 2), fmt_num(vc1max, 2),
            fmt_num(fn2, 4), fmt_num(vc2min, 2), fmt_num(vc2max, 2),
            fmt_num(fn3, 4), fmt_num(vc3min, 2), fmt_num(vc3max, 2)
```

```
            ]

            for c, v in enumerate(values):
                it = QTableWidgetItem(str(v))
                if c in (0, 8):
                    it.setTextAlignment(Qt.AlignCenter)
                self.table.setItem(r, c, it)

        self._apply_column_widths()

    def current_id(self):
        row = self.table.currentRow()
        if row < 0:
            return None
        return int(self.table.item(row, 0).text())

    def add(self):
        dlg = AddToolDialog(self)
        dlg.saved.connect(self.refresh)
        dlg.exec()

    def edit(self):
        hid = self.current_id()
        if hid is None:
            return info(self, "Selecciona una herramienta para editar.")
        dlg = AddToolDialog(self, tool_id=hid)
        dlg.saved.connect(self.refresh)
        dlg.exec()

    def delete(self):
        hid = self.current_id()
        if hid is None:
            return info(self, "Selecciona una herramienta para eliminar.")

        nombre = self.table.item(self.table.currentRow(), 2).text()
        if not confirm(self, f"¿Eliminar '{nombre}' (ID {hid})?\n\nTambién se eliminará su
historial y registros activos asociados."):
            return
        con = None
        try:
            con = sqlite3.connect(DB_PATH, timeout=15)
            con.execute("PRAGMA foreign_keys = ON;")
            con.execute("PRAGMA busy_timeout = 15000;")
            cur = con.cursor()
            cur.execute("BEGIN")
```

```
            cur.execute("DELETE FROM herramientas_activas WHERE herramienta_id=?",
(hid,))
            cur.execute("DELETE FROM historial_selecciones WHERE herramienta_id=?",
(hid,))

            cur.execute("DELETE FROM portaherramienta_herramientas WHERE
herramienta_id=?", (hid,))
            cur.execute("DELETE FROM herramienta_materiales WHERE herramienta_id=?",
(hid,))
            cur.execute("DELETE FROM herramienta_material_piezas WHERE
herramienta_id=?", (hid,))

            cur.execute("DELETE FROM herramientas WHERE id=?", (hid,))
            con.commit()
            con.close()
        except Exception as e:
            try:
                if con is not None:
                    con.rollback()
                    con.close()
            except Exception:
                pass
            return error(self, f"No se pudo eliminar:\n{e}")
        self.refresh()

    def export_excel(self):
        path, _ = QFileDialog.getSaveFileName(
            self,
            "Guardar base de herramientas",
            "base_herramientas.xlsx",
            "Excel (*.xlsx)"
        )
        if not path:
            return

        try:
            wb = Workbook()
            ws = wb.active
            ws.title = "herramientas"

            headers = [
                "codigo", "nombre", "maquina", "nivel", "operacion", "materiales",
"materiales_pieza", "stock", "ubicacion", "imagen",
                "ap_min", "ap_max",
                "fn_1", "vc1_min", "vc1_max",
                "fn_2", "vc2_min", "vc2_max",
                "fn_3", "vc3_min", "vc3_max"
```

```
        ]
        ws.append(headers)

        rows = fetchall("""
            SELECT
                h.id, h.codigo, h.nombre,
                mq.nombre, nv.nombre, op.nombre,
                h.stock, COALESCE(h.ubicacion,''), COALESCE(h.imagen,''),
                COALESCE(h.ap_min,0), COALESCE(h.ap_max,0),
                COALESCE(h.fn_1, COALESCE(h.fn_min,0)),
                COALESCE(h.vc1_min, COALESCE(h.vc_min,0)), COALESCE(h.vc1_max,
COALESCE(h.vc_max,0)),
                COALESCE(h.fn_2, CASE WHEN COALESCE(h.fn_min,0)>0 AND
COALESCE(h.fn_max,0)>0 THEN
(COALESCE(h.fn_min,0)+COALESCE(h.fn_max,0))/2.0 ELSE 0 END),
                COALESCE(h.vc2_min, COALESCE(h.vc_min,0)), COALESCE(h.vc2_max,
COALESCE(h.vc_max,0)),
                COALESCE(h.fn_3, COALESCE(h.fn_max,0)),
                COALESCE(h.vc3_min, COALESCE(h.vc_min,0)), COALESCE(h.vc3_max,
COALESCE(h.vc_max,0))
            FROM herramientas h
            JOIN maquinas mq ON mq.id=h.maquina_id
            JOIN niveles nv ON nv.id=h.nivel_id
            JOIN operaciones op ON op.id=h.operacion_id
            ORDER BY h.id ASC
        """)

        for (_hid, codigo, nombre, maq, niv, oper, stock, ubic, imagen,
             apmin, apmax, fn1, vc1min, vc1max, fn2, vc2min, vc2max, fn3, vc3min,
vc3max) in rows:
            mats = fetchall("""
                SELECT m.nombre
                FROM herramienta_materiales hm
                JOIN materiales m ON m.id = hm.material_id
                WHERE hm.herramienta_id=?
                ORDER BY m.nombre
            """, (_hid,))
            mats_txt = ",".join([x[0] for x in mats])
            mats_pieza = fetchall("""
                SELECT mp.nombre
                FROM herramienta_material_piezas hmp
                JOIN material_piezas mp ON mp.id = hmp.material_pieza_id
                WHERE hmp.herramienta_id=?
                ORDER BY mp.nombre
            """, (_hid,))
            mats_pieza_txt = ",".join([x[0] for x in mats_pieza])
```

```
            ws.append([
                codigo, nombre, maq, niv, oper, mats_txt, mats_pieza_txt, int(stock), ubic,
imagen,
                float(apmin), float(apmax),
                float(fn1), float(vc1min), float(vc1max),
                float(fn2), float(vc2min), float(vc2max),
                float(fn3), float(vc3min), float(vc3max)
            ])

        wb.save(path)
        info(self, f"Excel exportado correctamente:\n{path}")

    except Exception as e:
        error(self, f"No se pudo exportar:\n{e}")

  def import_excel(self):
    path, _ = QFileDialog.getOpenFileName(
        self,
        "Seleccionar Excel de herramientas",
        "",
        "Excel (*.xlsx)"
    )
    if not path:
        return

    try:
        wb = load_workbook(path)
        ws = wb.active

        header_row = [str(c.value).strip().lower() if c.value is not None else "" for c in
ws[1]]
        col = {name: idx for idx, name in enumerate(header_row)}

        required = ["codigo", "nombre", "maquina", "nivel", "operacion", "materiales",
"stock", "ubicacion", "ap_min", "ap_max"]
        new_cut_cols = ["fn_1", "vc1_min", "vc1_max", "fn_2", "vc2_min", "vc2_max",
"fn_3", "vc3_min", "vc3_max"]
        old_cut_cols = ["fn_min", "fn_max", "vc_min", "vc_max"]
        has_new_cut = all(k in col for k in new_cut_cols)
        has_old_cut = all(k in col for k in old_cut_cols)
        required += new_cut_cols if has_new_cut else old_cut_cols
        missing = [r for r in required if r not in col]
        if missing:
            return error(self, "Faltan columnas en el Excel:\n- " + "\n- ".join(missing))

        maq_map = {n.strip().lower(): i for i, n in fetchall("SELECT id, nombre FROM
maquinas")}
```

```
        niv_map = {n.strip().lower(): i for i, n in fetchall("SELECT id, nombre FROM
niveles")}
        mat_map = {n.strip().upper(): i for i, n in fetchall("SELECT id, nombre FROM
materiales")}
        mp_rows = fetchall("SELECT id, material_padre_id, nombre FROM
material_piezas")

        ops_rows = fetchall("SELECT id, maquina_id, nombre FROM operaciones")
        op_map = {(maquina_id, name.strip().lower()): oid for oid, maquina_id, name in
ops_rows}

        created = 0
        updated = 0
        errors_list = []

        def to_float(v, label):
            try:
                if v is None or str(v).strip() == "":
                    raise ValueError(f"{label} vacío")
                return float(v)
            except Exception:
                raise ValueError(f"{label} inválido: '{v}'")

        for r in range(2, ws.max_row + 1):
            row_vals = [ws.cell(row=r, column=c + 1).value for c in range(len(header_row))]
            if all(v is None or str(v).strip() == "" for v in row_vals):
                continue

            codigo = str(row_vals[col["codigo"]]).strip() if row_vals[col["codigo"]] is not
None else ""
            nombre = str(row_vals[col["nombre"]]).strip() if row_vals[col["nombre"]] is not
None else ""
            maquina = str(row_vals[col["maquina"]]).strip() if row_vals[col["maquina"]] is
not None else ""
            nivel = str(row_vals[col["nivel"]]).strip() if row_vals[col["nivel"]] is not None
else ""
            operacion = str(row_vals[col["operacion"]]).strip() if row_vals[col["operacion"]]
is not None else ""
            materiales = str(row_vals[col["materiales"]]).strip() if
row_vals[col["materiales"]] is not None else ""
            materiales_pieza = ""
            if "materiales_pieza" in col:
                mp_val = row_vals[col["materiales_pieza"]]
                materiales_pieza = str(mp_val).strip() if mp_val is not None else ""
            stock_raw = row_vals[col["stock"]]
            ubicacion = str(row_vals[col["ubicacion"]]).strip() if row_vals[col["ubicacion"]]
is not None else ""
```

```
            imagen = ""
            if "imagen" in col:
                imagen_val = row_vals[col["imagen"]]
                imagen = str(imagen_val).strip() if imagen_val is not None else ""

            apmin = row_vals[col["ap_min"]]
            apmax = row_vals[col["ap_max"]]
            if has_new_cut:
                fn1 = row_vals[col["fn_1"]]
                vc1min = row_vals[col["vc1_min"]]
                vc1max = row_vals[col["vc1_max"]]
                fn2 = row_vals[col["fn_2"]]
                vc2min = row_vals[col["vc2_min"]]
                vc2max = row_vals[col["vc2_max"]]
                fn3 = row_vals[col["fn_3"]]
                vc3min = row_vals[col["vc3_min"]]
                vc3max = row_vals[col["vc3_max"]]
            else:
                fnmin = row_vals[col["fn_min"]]
                fnmax = row_vals[col["fn_max"]]
                vcmin = row_vals[col["vc_min"]]
                vcmax = row_vals[col["vc_max"]]

            try:
                if not codigo:
                    raise ValueError("codigo vacío")
                if not nombre:
                    raise ValueError("nombre vacío")

                maq_id = maq_map.get(maquina.lower())
                if maq_id is None:
                    raise ValueError(f"maquina inválida: '{maquina}'")

                niv_id = niv_map.get(nivel.lower())
                if niv_id is None:
                    raise ValueError(f"nivel inválido: '{nivel}'")

                op_id = op_map.get((maq_id, operacion.lower()))
                if op_id is None:
                    raise ValueError(f"operacion inválida para esa maquina: '{operacion}'")

                try:
                    stock = int(stock_raw) if stock_raw is not None and str(stock_raw).strip() !=
"" else 0
                except Exception:
                    raise ValueError(f"stock inválido: '{stock_raw}'")
```

```
            mats_list = [m.strip().upper() for m in materiales.split(",") if m.strip()]
            if not mats_list:
                raise ValueError("materiales vacío")

            mat_ids = []
            for m in mats_list:
                mid = mat_map.get(m)
                if mid is None:
                    raise ValueError(f'material inválido: '{m}'")
                mat_ids.append(mid)

            mat_pieza_ids = []
            mats_pieza_list = [mp.strip() for mp in materiales_pieza.split(",") if mp and
mp.strip()]
            for mp_name in mats_pieza_list:
                candidates = [
                    pid for (pid, padre_id, pname) in mp_rows
                    if padre_id in mat_ids and str(pname or "").strip().lower() ==
mp_name.lower()
                ]
                if not candidates:
                    raise ValueError(f'material de pieza inválido para los materiales elegidos:
'{mp_name}'")
                if len(candidates) > 1:
                    raise ValueError(f'material de pieza ambiguo: '{mp_name}' (repite en
varias familias)")
                mat_pieza_ids.append(candidates[0])

            apmin_f = to_float(apmin, "ap_min")
            apmax_f = to_float(apmax, "ap_max")
            if has_new_cut:
                fn1_f = to_float(fn1, "fn_1")
                vc1min_f = to_float(vc1min, "vc1_min")
                vc1max_f = to_float(vc1max, "vc1_max")
                fn2_f = to_float(fn2, "fn_2")
                vc2min_f = to_float(vc2min, "vc2_min")
                vc2max_f = to_float(vc2max, "vc2_max")
                fn3_f = to_float(fn3, "fn_3")
                vc3min_f = to_float(vc3min, "vc3_min")
                vc3max_f = to_float(vc3max, "vc3_max")
            else:
                fnmin_f = to_float(fnmin, "fn_min")
                fnmax_f = to_float(fnmax, "fn_max")
                vcmin_f = to_float(vcmin, "vc_min")
                vcmax_f = to_float(vcmax, "vc_max")
                fn1_f = fnmin_f
                fn2_f = (fnmin_f + fnmax_f) / 2.0
```

```
                fn3_f = fnmax_f
                vc1min_f = vc2min_f = vc3min_f = vcmin_f
                vc1max_f = vc2max_f = vc3max_f = vcmax_f

            if apmin_f <= 0 or apmax_f <= 0:
                raise ValueError("Ap debe ser > 0 (min y max)")
            if apmax_f < apmin_f:
                raise ValueError("Ap_max no puede ser menor que Ap_min")
            if not (fn1_f > 0 and fn2_f > 0 and fn3_f > 0):
                raise ValueError("fn_1, fn_2 y fn_3 deben ser > 0")
            if not (fn1_f < fn2_f < fn3_f):
                raise ValueError("Los avances deben ser crecientes: fn_1 < fn_2 < fn_3")
            for idx, a, b in [("vc1", vc1min_f, vc1max_f), ("vc2", vc2min_f, vc2max_f),
("vc3", vc3min_f, vc3max_f)]:
                if a <= 0 or b <= 0:
                    raise ValueError(f"{idx}_min y {idx}_max deben ser > 0")
                if b < a:
                    raise ValueError(f"{idx}_max no puede ser menor que {idx}_min")

            fnmin_f = min(fn1_f, fn2_f, fn3_f)
            fnmax_f = max(fn1_f, fn2_f, fn3_f)
            vcmin_f = min(vc1min_f, vc2min_f, vc3min_f)
            vcmax_f = max(vc1max_f, vc2max_f, vc3max_f)

            existing = fetchall("SELECT id, COALESCE(stock_usado,0) FROM
herramientas WHERE codigo=? LIMIT 1", (codigo,))
            if existing:
                hid, stock_usado_db = existing[0]
                if stock < int(stock_usado_db):
                    raise ValueError(
                        f"stock inválido: no puede ser menor al stock previo uso actual
({int(stock_usado_db)})"
                    )
                stock_nuevo = stock - int(stock_usado_db)
                execute("""
                    UPDATE herramientas
                    SET nombre=?, maquina_id=?, nivel_id=?, operacion_id=?,
                        stock=?, stock_nuevo=?, ubicacion=?,
                        imagen=?, ap_min=?, ap_max=?, fn_min=?, fn_max=?, vc_min=?,
vc_max=?,
                        fn_1=?, fn_2=?, fn_3=?, vc1_min=?, vc1_max=?, vc2_min=?,
vc2_max=?, vc3_min=?, vc3_max=?
                    WHERE id=?
                """, (nombre, maq_id, niv_id, op_id, stock, stock_nuevo, ubicacion,
                      imagen, apmin_f, apmax_f, fnmin_f, fnmax_f, vcmin_f, vcmax_f,
                      fn1_f, fn2_f, fn3_f, vc1min_f, vc1max_f, vc2min_f, vc2max_f,
vc3min_f, vc3max_f,
```

```
                    hid))

                execute("DELETE FROM herramienta_materiales WHERE
herramienta_id=?", (hid,))
                for mid in mat_ids:
                    execute("INSERT OR IGNORE INTO herramienta_materiales
(herramienta_id, material_id) VALUES (?, ?)",
                            (hid, mid))
                execute("DELETE FROM herramienta_material_piezas WHERE
herramienta_id=?", (hid,))
                for mpid in mat_pieza_ids:
                    execute(
                        "INSERT OR IGNORE INTO herramienta_material_piezas
(herramienta_id, material_pieza_id) VALUES (?, ?)",
                        (hid, mpid)
                    )

                updated += 1
            else:
                next_hid = get_next_herramienta_id()
                hid = insert_and_get_id("""
                    INSERT INTO herramientas
                    (id, codigo, nombre, maquina_id, nivel_id, operacion_id, stock,
stock_nuevo, stock_usado, ubicacion,
                     imagen, ap_min, ap_max, fn_min, fn_max, vc_min, vc_max,
                     fn_1, fn_2, fn_3, vc1_min, vc1_max, vc2_min, vc2_max, vc3_min,
vc3_max)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?, 0, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
                """, (next_hid, codigo, nombre, maq_id, niv_id, op_id, stock, stock,
ubicacion,
                      imagen, apmin_f, apmax_f, fnmin_f, fnmax_f, vcmin_f, vcmax_f,
                      fn1_f, fn2_f, fn3_f, vc1min_f, vc1max_f, vc2min_f, vc2max_f,
vc3min_f, vc3max_f))

                for mid in mat_ids:
                    execute("INSERT OR IGNORE INTO herramienta_materiales
(herramienta_id, material_id) VALUES (?, ?)",
                            (hid, mid))
                for mpid in mat_pieza_ids:
                    execute(
                        "INSERT OR IGNORE INTO herramienta_material_piezas
(herramienta_id, material_pieza_id) VALUES (?, ?)",
                        (hid, mpid)
                    )

                created += 1
```

```
            except Exception as ex:
                errors_list.append(f"Fila {r}: {ex}")

        resumen = f"Importación finalizada\n\nCreadas: {created}\nActualizadas:
{updated}\nErrores: {len(errors_list)}"
        if errors_list:
            show = "\n".join(errors_list[:10])
            resumen += f"\n\nPrimeros errores:\n{show}"
            if len(errors_list) > 10:
                resumen += f"\n... y {len(errors_list)-10} más."
        info(self, resumen)

        self.refresh()

    except Exception as e:
        error(self, f"No se pudo importar:\n{e}")


class CatalogCard(QFrame):
    def __init__(self, short_code: str, title: str, desc: str, meta: str, action):
        super().__init__()
        self.setObjectName("catalogDashboardCard")

        lay = QVBoxLayout(self)
        lay.setContentsMargins(20, 20, 20, 20)
        lay.setSpacing(11)

        self.icon_lbl = QLabel(short_code)
        self.icon_lbl.setObjectName("catalogCardIcon")
        self.icon_lbl.setAlignment(Qt.AlignCenter)

        self.title_lbl = QLabel(title)
        self.title_lbl.setObjectName("catalogCardTitle")
        self.title_lbl.setAlignment(Qt.AlignCenter)

        self.desc_lbl = QLabel(desc)
        self.desc_lbl.setObjectName("catalogCardDesc")
        self.desc_lbl.setWordWrap(True)
        self.desc_lbl.setAlignment(Qt.AlignCenter)

        self.meta_lbl = QLabel(meta)
        self.meta_lbl.setObjectName("catalogCardMeta")
        self.meta_lbl.setAlignment(Qt.AlignCenter)

        self.btn = QPushButton("Administrar")
        self.btn.setObjectName("cardAction")
        self.btn.clicked.connect(action)
```

```
        lay.addWidget(self.icon_lbl, 0, Qt.AlignHCenter)
        lay.addSpacing(6)
        lay.addWidget(self.title_lbl)
        lay.addWidget(self.desc_lbl)
        lay.addWidget(self.meta_lbl)
        lay.addStretch(1)
        lay.addWidget(self.btn)

    def set_meta(self, text: str):
        self.meta_lbl.setText(text)


class CatalogDialog(QDialog):
    def __init__(self, title: str, page: QWidget, parent=None):
        super().__init__(parent)
        self.setWindowTitle(title)
        self.setModal(True)
        self.resize(1080, 700)

        header = QFrame()
        header.setObjectName("catalogDialogHeader")
        header_lay = QVBoxLayout(header)
        header_lay.setContentsMargins(16, 12, 16, 12)
        header_lay.setSpacing(4)

        breadcrumb = QLabel(f"CATÁLOGOS / {title.upper()}")
        breadcrumb.setObjectName("catalogBreadcrumb")
        breadcrumb.setAlignment(Qt.AlignCenter)

        title_lbl = QLabel(f"Administrar {title}")
        title_lbl.setObjectName("h2")
        title_lbl.setAlignment(Qt.AlignCenter)

        sub_lbl = QLabel("Crea, edita o elimina registros de este catálogo.")
        sub_lbl.setObjectName("muted")
        sub_lbl.setAlignment(Qt.AlignCenter)

        header_lay.addWidget(breadcrumb, 0, Qt.AlignCenter)
        header_lay.addWidget(title_lbl, 0, Qt.AlignCenter)
        header_lay.addWidget(sub_lbl, 0, Qt.AlignCenter)

        lay = QVBoxLayout(self)
        lay.setContentsMargins(18, 18, 18, 18)
        lay.setSpacing(14)
        lay.addWidget(header)
        lay.addWidget(make_card(page), 1)
```

```
class CatalogosContainer(QWidget):
   def __init__(self):
      super().__init__()
      self.header = make_module_header(
         "Catálogos",
         "",
         ""
      )

      self.bg = QWidget()
      self.bg.setObjectName("catalogBg")

      self.grid = QGridLayout()
      self.grid.setContentsMargins(28, 28, 28, 28)
      self.grid.setHorizontalSpacing(18)
      self.grid.setVerticalSpacing(18)

      self.card_maquinas = CatalogCard(
         "MQ", "Máquinas", "Tipos de máquina para procesos de mecanizado", "",
self.open_maquinas
      )
      self.card_materiales = CatalogCard(
         "MT", "Materiales de herramienta", "Familias base de la herramienta (P, M, K, N,
S, H)", "", self.open_materiales
      )
      self.card_materiales_pieza = CatalogCard(
         "MP", "Materiales de pieza", "Materiales específicos de la pieza a mecanizar", "",
self.open_materiales_pieza
      )
      self.card_operaciones = CatalogCard(
         "OP", "Operaciones", "Operaciones por cada tipo de máquina", "",
self.open_operaciones
      )
      self.card_herramientas = CatalogCard(
         "HR", "Herramientas", "Base completa de herramientas con parámetros", "",
self.open_herramientas
      )
      self.card_portaherramientas = CatalogCard(
         "PH", "Portaherramientas", "Portaherramientas vinculados a una o más
herramientas", "", self.open_portaherramientas
      )
      self.card_usuarios = CatalogCard(
         "US", "Usuarios", "Responsables de las selecciones registradas", "",
self.open_usuarios
      )
```

```
        cards = [
            self.card_maquinas, self.card_materiales, self.card_materiales_pieza,
            self.card_operaciones, self.card_herramientas,
            self.card_portaherramientas, self.card_usuarios
        ]
        positions = [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0)]

        for card, (r, c) in zip(cards, positions):
            self.grid.addWidget(card, r, c)

        self.grid.setColumnStretch(0, 1)
        self.grid.setColumnStretch(1, 1)
        self.grid.setColumnStretch(2, 1)

        self.bg.setLayout(self.grid)

        grid_scroll = QScrollArea()
        grid_scroll.setWidgetResizable(True)
        grid_scroll.setFrameShape(QFrame.NoFrame)
        grid_scroll.setWidget(self.bg)

        lay = QVBoxLayout(self)
        lay.setSpacing(14)
        lay.addWidget(self.header)
        lay.addWidget(grid_scroll, 1)

        self.refresh_counts()

    def _count(self, table: str) -> int:
        row = fetchall(f"SELECT COUNT(*) FROM {table}")
        return int(row[0][0]) if row else 0

    def _plural(self, n: int, singular: str, plural: str = None) -> str:
        plural = plural or singular + "s"
        return f"{n} {singular if n == 1 else plural}"

    def refresh_counts(self):
        self.card_maquinas.set_meta(self._plural(self._count("maquinas"), "registrada",
"registradas"))
        self.card_materiales.set_meta(self._plural(self._count("materiales"), "tipo", "tipos"))
        self.card_materiales_pieza.set_meta(self._plural(self._count("material_piezas"),
"tipo", "tipos"))
        self.card_operaciones.set_meta(self._plural(self._count("operaciones"), "tipo",
"tipos"))
        self.card_herramientas.set_meta(self._plural(self._count("herramientas"), "creada",
"creadas"))
```

```
        self.card_portaherramientas.set_meta(self._plural(self._count("portaherramientas"),
"registrado", "registrados"))
        self.card_usuarios.set_meta(self._plural(self._count("usuarios"), "activo", "activos"))

    def _open_dialog(self, title: str, page: QWidget):
        dlg = CatalogDialog(title, page, self)
        dlg.exec()
        self.refresh_counts()

    def open_maquinas(self):
        self._open_dialog("Máquinas", SimpleCatalogPage("Máquinas", "maquinas", "Ej:
Torno / Fresadora"))

    def open_materiales(self):
        self._open_dialog(
            "Materiales de herramienta",
            SimpleCatalogPage("Materiales de herramienta", "materiales", "Ej: P / M / H / K /
N / S")
        )

    def open_materiales_pieza(self):
        self._open_dialog("Materiales de pieza", MaterialesPiezaPage())

    def open_operaciones(self):
        self._open_dialog("Operaciones", OperacionesPage())

    def open_herramientas(self):
        self._open_dialog("Herramientas", HerramientasPage())

    def open_portaherramientas(self):
        self._open_dialog("Portaherramientas", PortaHerramientasPage())

    def open_usuarios(self):
        self._open_dialog("Usuarios", SimpleCatalogPage("Usuarios", "usuarios", "Ej: Juan
Pérez / Turno Noche"))

class StepChoiceCard(QFrame):
    clickedData = Signal(object)

    def __init__(
        self,
        key,
        icon: str,
        title: str,
        desc: str,
        accent: str = "#7EA4FF",
        image_path: str = "",
```

```
        image_canvas_mode: str = "",
        show_text: bool = True
    ):
        super().__init__()
        self.key = key
        self._emit_queued = False
        self.setObjectName("stepCard")
        self.setCursor(Qt.PointingHandCursor)
        self.setFocusPolicy(Qt.NoFocus)
        self.setMinimumHeight(315 if not show_text else 520)
        if not show_text:
            self.setMaximumHeight(315)

        layout = QVBoxLayout(self)
        if show_text:
            layout.setContentsMargins(22, 22, 22, 22)
            layout.setSpacing(12)
        else:
            layout.setContentsMargins(16, 14, 16, 14)
            layout.setSpacing(8)

        icon_lbl = QLabel()
        icon_lbl.setAlignment(Qt.AlignCenter)
        icon_lbl.setAttribute(Qt.WA_TransparentForMouseEvents, True)
        if image_canvas_mode == "material_340":
            icon_lbl.setFixedSize(230, 230)
        if image_path and os.path.isfile(image_path):
            pix = QPixmap(image_path)
            if not pix.isNull():
                if image_canvas_mode == "material_340":
                    canvas = QPixmap(230, 230)
                    canvas.fill(Qt.transparent)
                    scaled = pix.scaled(230, 230, Qt.KeepAspectRatio, Qt.SmoothTransformation)
                    x = (230 - scaled.width()) // 2
                    y = (230 - scaled.height()) // 2
                    painter = QPainter(canvas)
                    painter.setRenderHint(QPainter.SmoothPixmapTransform, True)
                    painter.drawPixmap(x, y, scaled)
                    painter.end()
                    icon_lbl.setPixmap(canvas)
                else:
                    icon_lbl.setPixmap(pix.scaled(340, 340, Qt.KeepAspectRatio,
Qt.SmoothTransformation))
            else:
                icon_lbl.setObjectName("stepCardIcon")
                icon_lbl.setStyleSheet(f"color: {accent};")
                icon_lbl.setText(icon)
```

```
        else:
            icon_lbl.setObjectName("stepCardIcon")
            icon_lbl.setStyleSheet(f"color: {accent};")
            icon_lbl.setText(icon)

        title_lbl = QLabel(title)
        title_lbl.setObjectName("stepCardTitle")
        title_lbl.setAlignment(Qt.AlignCenter)
        title_lbl.setAttribute(Qt.WA_TransparentForMouseEvents, True)
        title_lbl.setVisible(show_text)

        desc_lbl = QLabel(desc)
        desc_lbl.setObjectName("stepCardDesc")
        desc_lbl.setWordWrap(True)
        desc_lbl.setAlignment(Qt.AlignCenter)
        desc_lbl.setAttribute(Qt.WA_TransparentForMouseEvents, True)
        desc_lbl.setVisible(show_text)

        self.action_btn = QLabel("Seleccionar")
        self.action_btn.setObjectName("stepCardAction")
        self.action_btn.setAlignment(Qt.AlignCenter)
        self.action_btn.setFixedWidth(220)
        self.action_btn.setFixedHeight(42)
        self.action_btn.setAttribute(Qt.WA_TransparentForMouseEvents, True)

        if show_text:
            layout.addStretch(1)
            layout.addWidget(icon_lbl)
            layout.addWidget(title_lbl)
            layout.addWidget(desc_lbl)
            layout.addStretch(1)
            layout.addWidget(self.action_btn, 0, Qt.AlignHCenter)
        else:
            layout.addWidget(icon_lbl, 0, Qt.AlignHCenter | Qt.AlignTop)
            layout.addSpacing(0)
            layout.addWidget(self.action_btn, 0, Qt.AlignHCenter)

    def _emit_clicked(self):
        if self._emit_queued:
            return
        self._emit_queued = True
        QTimer.singleShot(20, self._emit_clicked_now)

    def _emit_clicked_now(self):
        self._emit_queued = False
        self.clickedData.emit(self.key)
```

```
    def mouseReleaseEvent(self, event):
        if event.button() == Qt.LeftButton:
            self._emit_clicked()
            event.accept()
            return
        super().mouseReleaseEvent(event)


class StockBadge(QLabel):
    def __init__(self, stock: int):
        super().__init__(str(stock))
        self.setAlignment(Qt.AlignCenter)
        self.setMinimumWidth(74)
        self.setMinimumHeight(30)
        self.setStyleSheet(self.styleSheet() + "font-size: 15px; font-weight: 900;")
        if stock <= 0:
            self.setObjectName("stockZero")
        elif stock <= 2:
            self.setObjectName("stockLow")
        else:
            self.setObjectName("stockGood")


class TableroPage(QWidget):
    MACHINE_VISUALS = {
        "torno": ("T", "Procesos de torneado y mecanizado cilíndrico", "#5D7BFF"),
        "fresadora": ("F", "Procesos de fresado y mecanizado de superficies", "#7CB4FF"),
    }

    OPERATION_VISUALS = {
        "cilindrado": ("C", "Mecanizado longitudinal exterior o interior", "#76A8FF"),
        "refrentado": ("R", "Generación de caras planas", "#8AB3FF"),
        "ranurado": ("RA", "Formación de ranuras y canales", "#69CCFF"),
        "roscado": ("RO", "Generación de roscas", "#6BE0C1"),
        "planeado": ("P", "Planeado de superficies", "#7FAEFF"),
        "taladrado": ("TA", "Perforación axial", "#77C7FF"),
        "mandrinado": ("M", "Ajuste interior y precisión", "#88B6FF"),
    }

    MATERIAL_VISUALS = {
        "P": ("P", "Aceros", "#4F8FFF"),
        "M": ("M", "Inoxidables", "#FFC14D"),
        "K": ("K", "Fundición", "#E25D78"),
        "N": ("N", "Aluminio y no ferrosos", "#47D27A"),
        "S": ("S", "Superaleaciones", "#FF9547"),
        "H": ("H", "Aceros duros", "#B4BDC9"),
    }
```

```
def __init__(self):
    super().__init__()
    self._machine_select_lock = False

    self.selected_maquina_id = None
    self.selected_maquina_name = ""
    self.selected_operacion_id = None
    self.selected_operacion_name = ""
    self.selected_material_id = None
    self.selected_material_name = ""
    self.selected_material_pieza_ids = []
    self.selected_material_pieza_names = []
    self.selected_material_pieza_id = None
    self.selected_material_pieza_name = ""
    self.selected_material_pieza_iso_min = None
    self.selected_material_pieza_iso_max = None
    self.selected_material_pieza_hb_min = None
    self.selected_material_pieza_hb_max = None

    self.module_header = make_module_header(
        "Selección de herramienta",
        "",
        ""
    )

    self.btn_reset = QPushButton("Limpiar selección")
    self.btn_reset.setObjectName("ghost")
    self.btn_reset.clicked.connect(self.reset_flow)

    self.progress_row = QHBoxLayout()
    self.progress_row.setSpacing(4)

    self.summary_row = QHBoxLayout()
    self.summary_row.setSpacing(6)

    self.stack = QStackedWidget()

    self.page_maquina = self._build_choice_page(
        "Paso 1 · Selecciona la máquina",
        "Elige el tipo de máquina sobre la que se realizará la operación."
    )

    self.page_operacion = self._build_choice_page(
        "Paso 2 · Selecciona la operación",
        "Solo se muestran operaciones válidas para la máquina seleccionada."
    )
```

```
        self.page_material = self._build_choice_page(
            "Paso 3 · Selecciona la familia de material",
            "Primero elige la familia (P/M/K/N/S/H). Luego se abrirá una ventana para
seleccionar el material específico de la pieza."
        )

        self.page_parametros = self._build_params_page()
        self.page_resultados = self._build_results_page()

        self.stack.addWidget(self.page_maquina["root"])
        self.stack.addWidget(self.page_operacion["root"])
        self.stack.addWidget(self.page_material["root"])
        self.stack.addWidget(self.page_parametros)
        self.stack.addWidget(self.page_resultados)

        top_header = QWidget()
        top_header_lay = QHBoxLayout(top_header)
        top_header_lay.setContentsMargins(0, 0, 0, 0)
        top_header_lay.setSpacing(8)
        top_header_lay.addWidget(self.module_header, 1)

        progress_wrap = QWidget()
        progress_wrap.setLayout(self.progress_row)

        summary_wrap = QWidget()
        summary_wrap.setLayout(self.summary_row)
        self.summary_wrap = summary_wrap

        reset_row = QHBoxLayout()
        reset_row.setContentsMargins(0, 0, 0, 0)
        reset_row.addStretch(1)
        reset_row.addWidget(self.btn_reset)
        reset_wrap = QWidget()
        reset_wrap.setLayout(reset_row)

        content = QWidget()
        content.setObjectName("heroPanel")
        content_lay = QVBoxLayout(content)
        content_lay.setContentsMargins(12, 10, 12, 12)
        content_lay.setSpacing(8)
        content_lay.addWidget(top_header)
        content_lay.addWidget(progress_wrap)
        content_lay.addWidget(summary_wrap)
        content_lay.addWidget(self.stack, 1)
        content_lay.addWidget(reset_wrap)
```

```
    lay = QVBoxLayout(self)
    lay.setContentsMargins(0, 0, 0, 0)
    lay.addWidget(content, 1)

    self.refresh()


def _build_choice_page(self, title: str, subtitle: str):
    title_lbl = QLabel(title)
    title_lbl.setObjectName("stepTitle")

    sub_lbl = QLabel(subtitle)
    sub_lbl.setObjectName("stepSub")
    sub_lbl.setWordWrap(True)

    grid = QGridLayout()
    grid.setSpacing(18)

    grid_widget = QWidget()
    grid_widget.setLayout(grid)

    scroll = QScrollArea()
    scroll.setWidgetResizable(True)
    scroll.setFrameShape(QFrame.NoFrame)
    scroll.setWidget(grid_widget)

    btn_back = QPushButton("← Volver")
    btn_back.setObjectName("ghost")

    top = QHBoxLayout()
    top.addWidget(title_lbl)
    top.addStretch(1)
    top.addWidget(btn_back)

    root = QWidget()
    lay = QVBoxLayout(root)
    lay.setContentsMargins(6, 6, 6, 6)
    lay.setSpacing(14)
    lay.addLayout(top)
    lay.addWidget(sub_lbl)
    lay.addWidget(scroll, 1)

    return {
        "root": root,
        "grid": grid,
        "btn_back": btn_back,
        "title": title_lbl,
```

```
            "subtitle": sub_lbl
        }

    def _build_params_page(self):
        title_lbl = QLabel("Paso 4 · Ingresa parámetros opcionales")
        title_lbl.setObjectName("stepTitle")

        sub_lbl = QLabel("Estos valores refinan la búsqueda. Si no deseas filtrar por uno,
déjalo en 0.")
        sub_lbl.setObjectName("stepSub")
        sub_lbl.setWordWrap(True)

        self.in_ap = DualDecimalSpinBox()
        self.in_ap.setRange(0.0, 500.0)
        self.in_ap.setDecimals(3)
        self.in_ap.setSingleStep(0.1)
        self.in_ap.setSpecialValueText("")
        self.in_ap.setValue(0.0)

        self.in_fn = DualDecimalSpinBox()
        self.in_fn.setRange(0.0, 50.0)
        self.in_fn.setDecimals(4)
        self.in_fn.setSingleStep(0.01)
        self.in_fn.setSpecialValueText("")
        self.in_fn.setValue(0.0)

        self.in_vc = DualDecimalSpinBox()
        self.in_vc.setRange(0.0, 10000.0)
        self.in_vc.setDecimals(2)
        self.in_vc.setSingleStep(1.0)
        self.in_vc.setSpecialValueText("")
        self.in_vc.setValue(0.0)

        form = QFormLayout()
        form.addRow("Ap (mm):", self.in_ap)
        form.addRow("Fn (mm/rev):", self.in_fn)
        form.addRow("Vc (m/min):", self.in_vc)

        form_widget = QWidget()
        form_widget.setLayout(form)

        self.params_summary = QLabel("")
        self.params_summary.setObjectName("muted")
        self.params_summary.setWordWrap(True)

        btn_back = QPushButton("← Volver")
        btn_back.setObjectName("ghost")
```

```
        btn_back.clicked.connect(lambda: self.stack.setCurrentIndex(2))

        self.btn_buscar = QPushButton("Buscar herramientas compatibles")
        self.btn_buscar.setObjectName("primary")
        self.btn_buscar.clicked.connect(self.buscar)

        row = QHBoxLayout()
        row.addWidget(btn_back)
        row.addStretch(1)
        row.addWidget(self.btn_buscar)

        root = QWidget()
        lay = QVBoxLayout(root)
        lay.setContentsMargins(6, 6, 6, 6)
        lay.setSpacing(14)
        lay.addWidget(title_lbl)
        lay.addWidget(sub_lbl)
        lay.addWidget(make_glass_card(form_widget))
        lay.addWidget(self.params_summary)
        lay.addStretch(1)
        lay.addLayout(row)
        return root

    def _build_results_page(self):
        root = QWidget()
        lay = QVBoxLayout(root)
        lay.setContentsMargins(6, 6, 6, 6)
        lay.setSpacing(14)

        title_lbl = QLabel("Paso 5 · Resultados compatibles")
        title_lbl.setObjectName("stepTitle")

        self.results_info = QLabel("Resultados encontrados: 0")
        self.results_info.setObjectName("stepSub")

        self.table = QTableWidget(0, 17)
        self.table.setHorizontalHeaderLabels([
            "ID", "Código", "Nombre", "Máquina", "Operación", "Stock",
            "Ap mín", "Ap máx",
            "Fn1", "Vc1 mín", "Vc1 máx",
            "Fn2", "Vc2 mín", "Vc2 máx",
            "Fn3", "Vc3 mín", "Vc3 máx"
        ])
        self.table.setSelectionBehavior(QAbstractItemView.SelectRows)
        self.table.setSelectionMode(QAbstractItemView.SingleSelection)
        self.table.setEditTriggers(QAbstractItemView.NoEditTriggers)
        self.table.verticalHeader().setVisible(False)
```

```
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Interactive)
        self.table.horizontalHeader().setStretchLastSection(False)
        self.table.horizontalHeader().setMinimumSectionSize(78)
        self.table.setHorizontalScrollMode(QAbstractItemView.ScrollPerPixel)
        self.table.setHorizontalScrollBarPolicy(Qt.ScrollBarAsNeeded)
        apply_table_polish(self.table)
        self._apply_results_column_widths()

        btn_back = QPushButton("← Volver a parámetros")
        btn_back.setObjectName("ghost")
        btn_back.clicked.connect(lambda: self.stack.setCurrentIndex(3))

        self.btn_seleccionar = QPushButton("Seleccionar herramienta")
        self.btn_seleccionar.setObjectName("primary")
        self.btn_seleccionar.setFocusPolicy(Qt.NoFocus)
        self.btn_seleccionar.setAutoDefault(False)
        self.btn_seleccionar.setDefault(False)
        self.btn_seleccionar.clicked.connect(self.seleccionar_herramienta)

        row = QHBoxLayout()
        row.addWidget(btn_back)
        row.addStretch(1)
        row.addWidget(self.btn_seleccionar)

        lay.addWidget(title_lbl)
        lay.addWidget(self.results_info)
        lay.addWidget(self.table, 1)
        lay.addLayout(row)

        return root

    def _apply_results_column_widths(self):
        widths = {
            0: 60,
            1: 120,
            2: 170,
            3: 120,
            4: 130,
            5: 90,
            6: 90,
            7: 90,
            8: 85,
            9: 90,
            10: 90,
            11: 85,
            12: 90,
            13: 90,
```

```
            14: 85,
            15: 90,
            16: 90,
        }
        for c, w in widths.items():
            self.table.setColumnWidth(c, w)

    def _clear_layout_grid(self, grid: QGridLayout):
        while grid.count():
            item = grid.takeAt(0)
            widget = item.widget()
            if widget is not None:
                widget.hide()
                widget.setParent(None)
                widget.deleteLater()

    def _set_progress(self, active_step: int):
        while self.progress_row.count():
            item = self.progress_row.takeAt(0)
            widget = item.widget()
            if widget:
                widget.deleteLater()

        labels = [
            "Máquina",
            "Operación",
            "Material",
            "Parámetros",
            "Resultado"
        ]

        for idx, text in enumerate(labels, start=1):
            lbl = QLabel(f"{idx}. {text}")
            lbl.setObjectName("progressNodeActive" if idx == active_step else
"progressNode")
            self.progress_row.addWidget(lbl)

            if idx < len(labels):
                arrow = QLabel(">")
                arrow.setObjectName("progressArrow")
                self.progress_row.addWidget(arrow)

        self.progress_row.addStretch(1)

    def _selected_material_pieza_label(self) -> str:
        if self.selected_material_pieza_names and len(self.selected_material_pieza_names) >
1:
```

```
            return ", ".join(self.selected_material_pieza_names)
        if not self.selected_material_pieza_name:
            return self.selected_material_name
        details = []
        if (
            self.selected_material_name
            and self.selected_material_pieza_iso_min is not None
            and self.selected_material_pieza_iso_max is not None
            and int(self.selected_material_pieza_iso_min) > 0
            and int(self.selected_material_pieza_iso_max) > 0
        ):
            pref = str(self.selected_material_name).strip().split(" ")[0].upper()[:1]
            pref = pref if pref in ("P", "M", "K", "N", "S", "H") else "ISO"
            details.append(f"{pref}{int(self.selected_material_pieza_iso_min)}-
{pref}{int(self.selected_material_pieza_iso_max)}")
        if (
            self.selected_material_pieza_hb_min is not None
            and self.selected_material_pieza_hb_max is not None
            and float(self.selected_material_pieza_hb_min) > 0
            and float(self.selected_material_pieza_hb_max) > 0
        ):
            hb_min_val = float(self.selected_material_pieza_hb_min)
            hb_max_val = float(self.selected_material_pieza_hb_max)
            hb_min_txt = f"{int(hb_min_val)}" if hb_min_val.is_integer() else
f"{hb_min_val:.1f}"
            hb_max_txt = f"{int(hb_max_val)}" if hb_max_val.is_integer() else
f"{hb_max_val:.1f}"
            details.append(f"HB {hb_min_txt}-{hb_max_txt}")
        return f"{self.selected_material_pieza_name} ({' | '.join(details)})" if details else
self.selected_material_pieza_name

    def _set_summary(self):
        while self.summary_row.count():
            item = self.summary_row.takeAt(0)
            widget = item.widget()
            if widget:
                widget.deleteLater()

        material_value = self._selected_material_pieza_label()

        selections = [
            ("Máquina", self.selected_maquina_name),
            ("Operación", self.selected_operacion_name),
            ("Material", material_value),
        ]

        chips_count = 0
```

```
        for label, value in selections:
            if value:
                chip = QLabel(f"{label}: {value}")
                chip.setObjectName("summaryChip")
                self.summary_row.addWidget(chip)
                chips_count += 1

        self.summary_row.addStretch(1)
        if hasattr(self, "summary_wrap"):
            self.summary_wrap.setVisible(chips_count > 0)

    def _build_card(self, key, icon, title, desc, callback, accent, image_path="",
image_canvas_mode="", show_text=True):
        card = StepChoiceCard(
            key,
            icon,
            title,
            desc,
            accent,
            image_path=image_path,
            image_canvas_mode=image_canvas_mode,
            show_text=show_text
        )
        card.clickedData.connect(callback)
        return card

    def _machine_visual(self, name: str):
        return self.MACHINE_VISUALS.get(name.strip().lower(), ("M", "Máquina
disponible en catálogo", "#7EA4FF"))

    def _operation_visual(self, name: str):
        return self.OPERATION_VISUALS.get(name.strip().lower(), ("O", "Operación
disponible", "#7EA4FF"))

    def _material_visual(self, name: str):
        return self.MATERIAL_VISUALS.get(name.strip().upper(), ("MAT", "Material
disponible", "#7EA4FF"))

    def refresh(self):
        self.selected_maquina_id = None
        self.selected_maquina_name = ""
        self.selected_operacion_id = None
        self.selected_operacion_name = ""
        self.selected_material_id = None
        self.selected_material_name = ""
        self.selected_material_pieza_ids = []
        self.selected_material_pieza_names = []
```

```
        self.selected_material_pieza_id = None
        self.selected_material_pieza_name = ""
        self.selected_material_pieza_iso_min = None
        self.selected_material_pieza_iso_max = None
        self.selected_material_pieza_hb_min = None
        self.selected_material_pieza_hb_max = None

        self.in_ap.setValue(0.0)
        self.in_fn.setValue(0.0)
        self.in_vc.setValue(0.0)

        self._load_machine_cards()
        self._set_progress(1)
        self._set_summary()
        self.stack.setCurrentIndex(0)

    def reset_flow(self):
        self.selected_maquina_id = None
        self.selected_maquina_name = ""
        self.selected_operacion_id = None
        self.selected_operacion_name = ""
        self.selected_material_id = None
        self.selected_material_name = ""
        self.selected_material_pieza_ids = []
        self.selected_material_pieza_names = []
        self.selected_material_pieza_id = None
        self.selected_material_pieza_name = ""
        self.selected_material_pieza_iso_min = None
        self.selected_material_pieza_iso_max = None
        self.selected_material_pieza_hb_min = None
        self.selected_material_pieza_hb_max = None

        self.in_ap.setValue(0.0)
        self.in_fn.setValue(0.0)
        self.in_vc.setValue(0.0)

        self.table.setRowCount(0)
        self.results_info.setText("Resultados encontrados: 0")
        self.params_summary.setText("")

        self._load_machine_cards()
        self._set_progress(1)
        self._set_summary()
        self.stack.setCurrentIndex(0)

    def _safe_reconnect(self, button, callback):
        try:
```

```
            button.clicked.disconnect()
        except Exception:
            pass
        button.clicked.connect(callback)

    def _load_machine_cards(self):
        grid = self.page_maquina["grid"]
        self._clear_layout_grid(grid)

        rows = fetchall("SELECT id, nombre, COALESCE(imagen,") FROM maquinas
ORDER BY nombre")
        for idx, (mid, nombre, img) in enumerate(rows):
            icon, desc, accent = self._machine_visual(nombre)
            card = self._build_card(
                (mid, nombre), icon, nombre, desc, self._on_machine_selected, accent,
image_path=img
            )
            grid.addWidget(card, idx // 2, idx % 2)

        self.page_maquina["btn_back"].hide()

    def _load_operation_cards(self):
        grid = self.page_operacion["grid"]
        self._clear_layout_grid(grid)

        rows = fetchall(
            "SELECT id, nombre, COALESCE(imagen,") FROM operaciones WHERE
maquina_id=? ORDER BY nombre",
            (self.selected_maquina_id,)
        )

        for idx, (oid, nombre, img) in enumerate(rows):
            icon, desc, accent = self._operation_visual(nombre)
            card = self._build_card(
                (oid, nombre), icon, nombre, desc, self._on_operation_selected, accent,
image_path=img
            )
            grid.addWidget(card, idx // 2, idx % 2)

        self.page_operacion["btn_back"].show()
        self._safe_reconnect(self.page_operacion["btn_back"], lambda: self._go_to_step(0))

    def _load_material_cards(self):
        grid = self.page_material["grid"]
        self._clear_layout_grid(grid)

        rows = fetchall("""
```

```
            SELECT m.id, m.nombre, COALESCE(m.imagen,'')
            FROM materiales m
            ORDER BY m.nombre
        """)

        if not rows:
            msg = QLabel("No hay familias de material creadas. Ve a Catálogos > Materiales de herramienta.")
            msg.setObjectName("muted")
            msg.setWordWrap(True)
            grid.addWidget(msg, 0, 0, 1, 2)
            self.page_material["btn_back"].show()
            self._safe_reconnect(self.page_material["btn_back"], lambda: self._go_to_step(1))
            return

        for idx, (mat_padre_id, mat_padre_nombre, img) in enumerate(rows):
            icon, desc, accent = self._material_visual(mat_padre_nombre)
            card = self._build_card(
                (mat_padre_id, mat_padre_nombre),
                icon,
                mat_padre_nombre,
                "Selecciona la familia para elegir luego el material de pieza",
                self._on_material_family_selected,
                accent,
                image_path=img,
                image_canvas_mode="material_340",
                show_text=False
            )
            grid.addWidget(card, idx // 3, idx % 3)

        self.page_material["btn_back"].show()
        self._safe_reconnect(self.page_material["btn_back"], lambda: self._go_to_step(1))

    def _on_machine_selected(self, data):
        if self._machine_select_lock:
            return
        self._machine_select_lock = True
        QTimer.singleShot(220, self._unlock_machine_select)

        self.selected_maquina_id, self.selected_maquina_name = data
        self.selected_operacion_id = None
        self.selected_operacion_name = ""
        self.selected_material_id = None
        self.selected_material_name = ""
        self.selected_material_pieza_ids = []
        self.selected_material_pieza_names = []
        self.selected_material_pieza_id = None
```

```
        self.selected_material_pieza_name = ""
        self.selected_material_pieza_iso_min = None
        self.selected_material_pieza_iso_max = None
        self.selected_material_pieza_hb_min = None
        self.selected_material_pieza_hb_max = None
        QTimer.singleShot(30, self._go_to_operation_step_after_machine)

    def _unlock_machine_select(self):
        self._machine_select_lock = False

    def _go_to_operation_step_after_machine(self):
        self._load_operation_cards()
        self._set_progress(2)
        self._set_summary()
        self.stack.setCurrentIndex(1)

    def _on_operation_selected(self, data):
        self.selected_operacion_id, self.selected_operacion_name = data
        self.selected_material_id = None
        self.selected_material_name = ""
        self.selected_material_pieza_ids = []
        self.selected_material_pieza_names = []
        self.selected_material_pieza_id = None
        self.selected_material_pieza_name = ""
        self.selected_material_pieza_iso_min = None
        self.selected_material_pieza_iso_max = None
        self.selected_material_pieza_hb_min = None
        self.selected_material_pieza_hb_max = None
        self._load_material_cards()
        self._set_progress(3)
        self._set_summary()
        self.stack.setCurrentIndex(2)

    def _on_material_family_selected(self, data):
        mat_padre_id, mat_padre_nombre = data
        dlg = SelectMaterialPiezaDialog(mat_padre_id, mat_padre_nombre, self)
        if dlg.exec() != QDialog.Accepted:
            return

        selected = dlg.selected_data()
        if not selected:
            return

        pieza_ids = [x[0] for x in selected]
        pieza_names = [f"[{x[1]}] {x[2]}" if x[1] else x[2] for x in selected]
        pieza_id, pieza_codigo, pieza_nombre, pieza_iso_min, pieza_iso_max, pieza_hb_min,
pieza_hb_max = selected[0]
```

```
        self.selected_material_id = mat_padre_id
        self.selected_material_name = mat_padre_nombre
        self.selected_material_pieza_ids = pieza_ids
        self.selected_material_pieza_names = pieza_names
        self.selected_material_pieza_id = pieza_id
        self.selected_material_pieza_name = f"[{pieza_codigo}] {pieza_nombre}" if
pieza_codigo else pieza_nombre
        self.selected_material_pieza_iso_min = pieza_iso_min
        self.selected_material_pieza_iso_max = pieza_iso_max
        self.selected_material_pieza_hb_min = pieza_hb_min
        self.selected_material_pieza_hb_max = pieza_hb_max
        self._set_progress(4)
        self._set_summary()
        self._update_params_summary()
        self.stack.setCurrentIndex(3)

    def _go_to_step(self, idx: int):
        self.stack.setCurrentIndex(idx)
        self._set_progress(idx + 1)

    def _update_params_summary(self):
        material_txt = self._selected_material_pieza_label()
        self.params_summary.setText(
            f"Selección actual: Máquina={self.selected_maquina_name} | "
            f"Operación={self.selected_operacion_name} | "
            f"Material={material_txt}"
        )

    def buscar(self):
        if None in (
            self.selected_maquina_id,
            self.selected_operacion_id,
            self.selected_material_id
        ):
            return info(self, "Completa todos los pasos antes de buscar.")
        if not self.selected_material_pieza_ids:
            return info(self, "Selecciona al menos un material de pieza.")

        self.table.setRowCount(0)

        ap = float(self.in_ap.value())
        fn = float(self.in_fn.value())
        vc = float(self.in_vc.value())

        mp_placeholders = ",".join("?" for _ in self.selected_material_pieza_ids)
        rows = fetchall(f"""
            SELECT h.id, h.codigo, h.nombre, mq.nombre, op.nombre, h.stock,
```

```
            COALESCE(hpp.material_pieza_id, 0),
            COALESCE(h.ap_min,0), COALESCE(h.ap_max,0),
            COALESCE(hpp.fn_1, COALESCE(h.fn_1, COALESCE(h.fn_min,0))),
            COALESCE(hpp.fn_2, COALESCE(h.fn_2, CASE WHEN
COALESCE(h.fn_min,0)>0 AND COALESCE(h.fn_max,0)>0 THEN
(COALESCE(h.fn_min,0)+COALESCE(h.fn_max,0))/2.0 ELSE 0 END)),
            COALESCE(hpp.fn_3, COALESCE(h.fn_3, COALESCE(h.fn_max,0))),
            COALESCE(hpp.vc1_min, COALESCE(h.vc1_min,
COALESCE(h.vc_min,0))), COALESCE(hpp.vc1_max, COALESCE(h.vc1_max,
COALESCE(h.vc_max,0))),
            COALESCE(hpp.vc2_min, COALESCE(h.vc2_min,
COALESCE(h.vc_min,0))), COALESCE(hpp.vc2_max, COALESCE(h.vc2_max,
COALESCE(h.vc_max,0))),
            COALESCE(hpp.vc3_min, COALESCE(h.vc3_min,
COALESCE(h.vc_min,0))), COALESCE(hpp.vc3_max, COALESCE(h.vc3_max,
COALESCE(h.vc_max,0)))
        FROM herramientas h
        JOIN maquinas mq ON mq.id=h.maquina_id
        JOIN operaciones op ON op.id=h.operacion_id
        LEFT JOIN herramienta_material_pieza_parametros hpp
            ON hpp.herramienta_id=h.id AND hpp.material_pieza_id IN
({mp_placeholders})
        WHERE h.maquina_id=?
          AND (
            h.operacion_id=?
            OR EXISTS (
              SELECT 1
              FROM herramienta_operaciones ho
              WHERE ho.herramienta_id=h.id AND ho.operacion_id=?
            )
          )
          AND EXISTS (
            SELECT 1
            FROM herramienta_materiales hm
            WHERE hm.herramienta_id=h.id AND hm.material_id=?
          )
          AND (
            EXISTS (
              SELECT 1
              FROM herramienta_material_piezas hmp
              WHERE hmp.herramienta_id=h.id AND hmp.material_pieza_id IN
({mp_placeholders})
            )
            OR NOT EXISTS (
              SELECT 1
              FROM herramienta_material_piezas hmp2
              WHERE hmp2.herramienta_id=h.id
```

```
                )
            )
             AND ( ? <= 0 OR ( (h.ap_min=0 AND h.ap_max=0) OR (? BETWEEN h.ap_min
AND h.ap_max) ) )
            ORDER BY h.stock DESC, h.nombre ASC
        """, (
            *self.selected_material_pieza_ids,
            self.selected_maquina_id,
            self.selected_operacion_id,
            self.selected_operacion_id,
            self.selected_material_id,
            *self.selected_material_pieza_ids,
            ap, ap,
        ))
        filtered = []
        tol = 1e-6
        for r in rows:
            (hid, codigo, nombre, maq, oper, stock, _mpid, apmin, apmax,
             fn1, fn2, fn3, vc1min, vc1max, vc2min, vc2max, vc3min, vc3max) = r

            lanes = [
                (float(fn1 or 0), float(vc1min or 0), float(vc1max or 0)),
                (float(fn2 or 0), float(vc2min or 0), float(vc2max or 0)),
                (float(fn3 or 0), float(vc3min or 0), float(vc3max or 0)),
            ]

            valid_fn = sorted([x[0] for x in lanes if x[0] > 0])
            valid_vc_lanes = [(f, vmin, vmax) for f, vmin, vmax in lanes if f > 0 and vmin > 0
and vmax > 0]

            compatible = False
            if fn <= 0 and vc <= 0:
                compatible = True
            elif fn <= 0 and vc > 0:
                compatible = any((vmin > 0 and vmax > 0 and vmin - tol <= vc <= vmax + tol)
for _, vmin, vmax in lanes)
            else:
                if not valid_fn:
                    compatible = False
                elif vc <= 0:
                    direct_match = any(abs(fn - f) <= tol for f in valid_fn)
                    under_max = fn <= (valid_fn[-1] + tol)
                    compatible = direct_match or under_max
                else:
                    lane_pair_match = any(
                        (fn <= (f + tol)) and ((vmin - tol) <= vc <= (vmax + tol))
                        for f, vmin, vmax in valid_vc_lanes
```

```
                )
                if lane_pair_match:
                    compatible = True
                else:
                    fn_under_max = fn <= (valid_fn[-1] + tol)
                    vc_in_any = any((vmin - tol) <= vc <= (vmax + tol) for _, vmin, vmax in
valid_vc_lanes)
                    compatible = fn_under_max and vc_in_any
            if compatible:
                filtered.append(r)

        unique_filtered = []
        seen_hids = set()
        for r in filtered:
            hid = r[0]
            if hid in seen_hids:
                continue
            seen_hids.add(hid)
            unique_filtered.append(r)

        self.results_info.setText(f"Resultados encontrados: {len(unique_filtered)}
herramienta(s) compatibles")

        if not unique_filtered:
            self._set_progress(5)
            self.stack.setCurrentIndex(4)
            return info(self, "No se encontraron herramientas compatibles.")

        for r in unique_filtered:
            (hid, codigo, nombre, maq, oper, stock, _mpid,
             apmin, apmax, fn1, fn2, fn3, vc1min, vc1max, vc2min, vc2max, vc3min, vc3max)
= r

            def fmt_num(v, dec=2):
                vv = float(v or 0.0)
                if vv <= 0:
                    return "-"
                return f"{vv:.{dec}f}".rstrip("0").rstrip(".")

            row = self.table.rowCount()
            self.table.insertRow(row)

            values = [
                hid, codigo, nombre, maq, oper, stock,
                fmt_num(apmin, 3), fmt_num(apmax, 3),
                fmt_num(fn1, 4), fmt_num(vc1min, 2), fmt_num(vc1max, 2),
                fmt_num(fn2, 4), fmt_num(vc2min, 2), fmt_num(vc2max, 2),
```

```
                fmt_num(fn3, 4), fmt_num(vc3min, 2), fmt_num(vc3max, 2),
            ]
            for c, v in enumerate(values):
                if c == 5:
                    self.table.setCellWidget(row, c, StockBadge(int(stock)))
                else:
                    it = QTableWidgetItem(str(v))
                    if c == 0:
                        it.setTextAlignment(Qt.AlignCenter)
                    self.table.setItem(row, c, it)
            self.table.setRowHeight(row, 42)

        self._apply_results_column_widths()
        self._set_progress(5)
        self.stack.setCurrentIndex(4)

    def _selected_row_tool(self):
        r = self.table.currentRow()
        if r < 0:
            return None
        tool_id = int(self.table.item(r, 0).text())
        codigo = self.table.item(r, 1).text()

        stock_widget = self.table.cellWidget(r, 5)
        if stock_widget:
            stock = int(stock_widget.text())
        else:
            stock = int(self.table.item(r, 5).text())

        return tool_id, codigo, stock

    def seleccionar_herramienta(self):
        sel = self._selected_row_tool()
        if not sel:
            return info(self, "Selecciona una herramienta de la tabla.")

        tool_id, codigo, stock = sel
        if stock <= 0:
            return info(self, "La herramienta seleccionada actualmente no tiene existencias.")

        usuarios = fetchall("SELECT nombre FROM usuarios ORDER BY nombre")
        if not usuarios:
            return info(self, "No hay usuarios creados. Ve a Catálogos > Usuarios y agrega al
menos uno.")

        lista = [u[0] for u in usuarios]
```

```
        selected_mp_for_detail = self.selected_material_pieza_ids[0] if
self.selected_material_pieza_ids else self.selected_material_pieza_id
        detalle = fetchall("""
            SELECT h.codigo, h.nombre, mq.nombre, op.nombre, h.nivel_id, nv.nombre,
                h.stock, COALESCE(h.stock_nuevo,0), COALESCE(h.stock_usado,0),
                COALESCE(h.ubicacion,''), COALESCE(h.imagen,''),
                COALESCE(h.ap_min,0), COALESCE(h.ap_max,0),
                COALESCE(hpp.fn_1, COALESCE(h.fn_1, COALESCE(h.fn_min,0))),
                COALESCE(hpp.fn_2, COALESCE(h.fn_2, CASE WHEN
COALESCE(h.fn_min,0)>0 AND COALESCE(h.fn_max,0)>0 THEN
(COALESCE(h.fn_min,0)+COALESCE(h.fn_max,0))/2.0 ELSE 0 END)),
                COALESCE(hpp.fn_3, COALESCE(h.fn_3, COALESCE(h.fn_max,0))),
                COALESCE(hpp.vc1_min, COALESCE(h.vc1_min,
COALESCE(h.vc_min,0))), COALESCE(hpp.vc1_max, COALESCE(h.vc1_max,
COALESCE(h.vc_max,0))),
                COALESCE(hpp.vc2_min, COALESCE(h.vc2_min,
COALESCE(h.vc_min,0))), COALESCE(hpp.vc2_max, COALESCE(h.vc2_max,
COALESCE(h.vc_max,0))),
                COALESCE(hpp.vc3_min, COALESCE(h.vc3_min,
COALESCE(h.vc_min,0))), COALESCE(hpp.vc3_max, COALESCE(h.vc3_max,
COALESCE(h.vc_max,0)))
            FROM herramientas h
            JOIN maquinas mq ON mq.id=h.maquina_id
            JOIN operaciones op ON op.id=h.operacion_id
            JOIN niveles nv ON nv.id=h.nivel_id
            LEFT JOIN herramienta_material_pieza_parametros hpp
                ON hpp.herramienta_id=h.id AND hpp.material_pieza_id=?
            WHERE h.id=?
            LIMIT 1
        """, (selected_mp_for_detail, tool_id))
        if not detalle:
            return error(self, "La herramienta ya no existe en la base de datos.")

        mats = fetchall("""
            SELECT m.nombre
            FROM herramienta_materiales hm
            JOIN materiales m ON m.id=hm.material_id
            WHERE hm.herramienta_id=?
            ORDER BY m.nombre
        """, (tool_id,))
        portas = fetchall("""
            SELECT p.nombre, COALESCE(p.imagen,'')
            FROM portaherramienta_herramientas ph
            JOIN portaherramientas p ON p.id = ph.portaherramienta_id
            WHERE ph.herramienta_id=?
            ORDER BY p.nombre
        """, (tool_id,))
```

```
        (d_codigo, d_nombre, d_maquina, d_operacion, d_nivel_id, d_nivel, d_stock,
d_stock_nuevo, d_stock_usado, d_ubic, d_img,
         d_ap_min, d_ap_max, d_fn_1, d_fn_2, d_fn_3, d_vc1_min, d_vc1_max, d_vc2_min,
d_vc2_max, d_vc3_min, d_vc3_max) = detalle[0]
        mats_txt = ", ".join([m[0] for m in mats]) if mats else "-"
        porta_nombre = str(portas[0][0]) if portas else "-"
        porta_imagen = str(portas[0][1]) if portas else ""

        dlg = ConfirmToolSelectionDialog(
            {
                "codigo": str(d_codigo),
                "nombre": str(d_nombre),
                "maquina": str(d_maquina),
                "operacion": str(d_operacion),
                "nivel": str(d_nivel),
                "stock": int(d_stock or 0),
                "stock_nuevo": int(d_stock_nuevo or 0),
                "stock_usado": int(d_stock_usado or 0),
                "ubicacion": str(d_ubic or "-"),
                "imagen": str(d_img or ""),
                "materiales": mats_txt,
                "portaherramientas": porta_nombre,
                "porta_nombre": porta_nombre,
                "porta_imagen": porta_imagen,
                "ap_min": float(d_ap_min or 0),
                "ap_max": float(d_ap_max or 0),
                "fn_1": float(d_fn_1 or 0),
                "fn_2": float(d_fn_2 or 0),
                "fn_3": float(d_fn_3 or 0),
                "vc1_min": float(d_vc1_min or 0),
                "vc1_max": float(d_vc1_max or 0),
                "vc2_min": float(d_vc2_min or 0),
                "vc2_max": float(d_vc2_max or 0),
                "vc3_min": float(d_vc3_min or 0),
                "vc3_max": float(d_vc3_max or 0),
            },
            lista,
            self
        )
        if dlg.exec() != QDialog.Accepted:
            return

        usuario = dlg.selected_usuario()
        if not usuario:
            return info(self, "Debes seleccionar un usuario responsable.")
        tipo_salida = dlg.selected_tipo_salida()
```

```
        if tipo_salida not in ("NUEVA", "DESGASTE"):
            tipo_salida = "NUEVA"

        try:
            con = sqlite3.connect(DB_PATH)
            con.execute("PRAGMA foreign_keys = ON;")
            cur = con.cursor()

            cur.execute("SELECT stock, COALESCE(stock_nuevo,0),
COALESCE(stock_usado,0) FROM herramientas WHERE id=? LIMIT 1", (tool_id,))
            row = cur.fetchone()
            if not row:
                con.close()
                return error(self, "La herramienta ya no existe en la base de datos.")

            stock_real = int(row[0])
            stock_nuevo_real = int(row[1])
            stock_usado_real = int(row[2])
            if stock_real <= 0:
                con.close()
                return info(self, "La herramienta seleccionada actualmente no tiene existencias.")

            if tipo_salida == "NUEVA" and stock_nuevo_real <= 0:
                con.close()
                return info(self, "No hay stock nuevo disponible para esta herramienta.")
            if tipo_salida == "DESGASTE" and stock_usado_real <= 0:
                con.close()
                return info(self, "No hay stock de previo uso disponible para esta herramienta.")

            fecha_local = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            if tipo_salida == "NUEVA":
                cur.execute("UPDATE herramientas SET stock_nuevo = stock_nuevo - 1
WHERE id=?", (tool_id,))
            else:
                cur.execute("UPDATE herramientas SET stock_usado = stock_usado - 1
WHERE id=?", (tool_id,))
            cur.execute("UPDATE herramientas SET stock = COALESCE(stock_nuevo,0) +
COALESCE(stock_usado,0) WHERE id=?", (tool_id,))
            cur.execute("""
                INSERT INTO historial_selecciones
                (fecha_hora, herramienta_id, maquina_id, operacion_id, nivel_id, material_id,
material_pieza, usuario, tipo_salida, cantidad)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, 1)
            """, (
                fecha_local,
                tool_id,
```

```
                self.selected_maquina_id,
                self.selected_operacion_id,
                int(d_nivel_id),
                self.selected_material_id,
                self._selected_material_pieza_label() or None,
                usuario,
                tipo_salida
            ))

            cur.execute("""
                INSERT INTO herramientas_activas
                (fecha_salida, estado, tipo_salida, herramienta_id, maquina_id, operacion_id,
nivel_id, material_id, material_pieza, usuario)
                VALUES (?, 'EN_USO', ?, ?, ?, ?, ?, ?, ?, ?)
            """, (
                fecha_local,
                tipo_salida,
                tool_id,
                self.selected_maquina_id,
                self.selected_operacion_id,
                int(d_nivel_id),
                self.selected_material_id,
                self._selected_material_pieza_label() or None,
                usuario
            ))

            con.commit()
            con.close()

        except Exception as e:
            return error(self, f"No se pudo registrar la selección:\n{e}")

        info(self, f"Se ha seleccionado la herramienta {codigo}.")
        mw = self.window()
        if mw and hasattr(mw, "refresh_active_alert"):
            mw.refresh_active_alert()
        self.buscar()

class HistorialSeleccionesPage(QWidget):
    def __init__(self):
        super().__init__()

        title = QLabel("Historial de selecciones")
        title.setObjectName("h2")

        self.min_date = QDate(1900, 1, 1)
        self.date_from = BlankDateEdit(self.min_date)
```

```
        self.date_to = BlankDateEdit(self.min_date)

        self.btn_filter = QPushButton("Filtrar")
        self.btn_filter.setObjectName("primary")
        self.btn_filter.clicked.connect(self.refresh)

        self.btn_clear = QPushButton("Limpiar filtro")
        self.btn_clear.setObjectName("primary")
        self.btn_clear.clicked.connect(self.clear_filter)

        self.btn_refresh = QPushButton("Actualizar")
        self.btn_refresh.setObjectName("primary")
        self.btn_refresh.clicked.connect(self.refresh)

        self.btn_export = QPushButton("Descargar Excel")
        self.btn_export.setObjectName("primary")
        self.btn_export.clicked.connect(self.export_excel)

        self.table = QTableWidget(0, 9)
        self.table.setHorizontalHeaderLabels([
            "Fecha/Hora", "Herramienta", "Máquina", "Operación", "Nivel", "Material",
"Usuario", "Tipo", "Cantidad"
        ])
        self.table.setSelectionBehavior(QAbstractItemView.SelectRows)
        self.table.setEditTriggers(QAbstractItemView.NoEditTriggers)
        self.table.verticalHeader().setVisible(False)
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
        apply_table_polish(self.table)

        top = QHBoxLayout()
        top.addWidget(title)
        top.addStretch(1)
        top.addWidget(QLabel("Desde:"))
        top.addWidget(self.date_from)
        top.addWidget(QLabel("Hasta:"))
        top.addWidget(self.date_to)
        top.addWidget(self.btn_filter)
        top.addWidget(self.btn_clear)
        top.addSpacing(12)
        top.addWidget(self.btn_refresh)
        top.addWidget(self.btn_export)

        wrap = QWidget()
        wlay = QVBoxLayout(wrap)
        wlay.setContentsMargins(0, 0, 0, 0)
        wlay.addLayout(top)
        wlay.addWidget(self.table, 1)
```

```
    lay = QVBoxLayout(self)
    lay.addWidget(make_card(wrap), 1)

    self._apply_column_widths()
    self.refresh()

def _apply_column_widths(self):
    widths = {
        0: 95,
        1: 170,
        2: 330,
        3: 150,
        4: 210,
        5: 130,
        6: 140,
        7: 170,
        8: 160,
        9: 150,
        10: 110,
    }
    for col, width in widths.items():
        self.table.setColumnWidth(col, width)

def _get_filter_range(self):
    d_from = self.date_from.date()
    d_to = self.date_to.date()

    if d_from == self.min_date and d_to == self.min_date:
        return None, None

    if d_from != self.min_date and d_to == self.min_date:
        start = f"{d_from.toString('yyyy-MM-dd')} 00:00:00"
        end = "9999-12-31 23:59:59"
        return start, end

    if d_from == self.min_date and d_to != self.min_date:
        start = "0001-01-01 00:00:00"
        end = f"{d_to.toString('yyyy-MM-dd')} 23:59:59"
        return start, end

    if d_from > d_to:
        return "INVALID", "INVALID"

    start = f"{d_from.toString('yyyy-MM-dd')} 00:00:00"
    end = f"{d_to.toString('yyyy-MM-dd')} 23:59:59"
    return start, end
```

```
    def _query_rows(self):
        start, end = self._get_filter_range()

        base_sql = """
            SELECT hs.fecha_hora,
                   (h.codigo || ' - ' || h.nombre) AS herramienta,
                   mq.nombre AS maquina,
                   op.nombre AS operacion,
                   nv.nombre AS nivel,
                   COALESCE(NULLIF(hs.material_pieza,''), m.nombre) AS material,
                   hs.usuario,
                   hs.tipo_salida,
                   hs.cantidad
            FROM historial_selecciones hs
            JOIN herramientas h ON h.id = hs.herramienta_id
            JOIN maquinas mq ON mq.id = hs.maquina_id
            JOIN operaciones op ON op.id = hs.operacion_id
            JOIN niveles nv ON nv.id = hs.nivel_id
            JOIN materiales m ON m.id = hs.material_id
        """

        if start == "INVALID":
            return []

        if start is None and end is None:
            return fetchall(base_sql + " ORDER BY hs.id DESC")

        return fetchall(base_sql + " WHERE hs.fecha_hora BETWEEN ? AND ? ORDER BY
hs.id DESC", (start, end))

    def refresh(self):
        start, _ = self._get_filter_range()
        if start == "INVALID":
            return info(self, "La fecha 'Desde' no puede ser mayor que la fecha 'Hasta'.")

        rows = self._query_rows()

        self.table.setRowCount(0)
        for row_data in rows:
            r = self.table.rowCount()
            self.table.insertRow(r)
            for c, v in enumerate(row_data):
                it = QTableWidgetItem(str(v))
                if c == 8:
                    it.setTextAlignment(Qt.AlignCenter)
                self.table.setItem(r, c, it)
```

```
            self.table.setRowHeight(r, 36)
        self._apply_column_widths()

    def clear_filter(self):
        self.date_from.setDate(self.min_date)
        self.date_to.setDate(self.min_date)
        if self.date_from.lineEdit():
            self.date_from.lineEdit().setText("")
        if self.date_to.lineEdit():
            self.date_to.lineEdit().setText("")
        self.refresh()

    def export_excel(self):
        start, _ = self._get_filter_range()
        if start == "INVALID":
            return info(self, "La fecha 'Desde' no puede ser mayor que la fecha 'Hasta'.")

        path, _ = QFileDialog.getSaveFileName(
            self,
            "Guardar reporte - Historial de selecciones",
            "reporte_historial_selecciones.xlsx",
            "Excel (*.xlsx)"
        )
        if not path:
            return

        try:
            rows = self._query_rows()

            wb = Workbook()
            ws = wb.active
            ws.title = "historial_selecciones"
            headers = ["fecha_hora", "herramienta", "maquina", "operacion", "nivel",
"material", "usuario", "tipo_salida", "cantidad"]
            ws.append(headers)

            for r in rows:
                ws.append(list(r))

            wb.save(path)
            info(self, f"Reporte exportado correctamente:\n{path}")

        except Exception as e:
            error(self, f"No se pudo exportar el reporte:\n{e}")


class HerramientasActivasPage(QWidget):
```

```
def __init__(self):
    super().__init__()
    module_header = make_module_header(
        "Herramientas activas",
        "",
        ""
    )

    self.btn_refresh = QPushButton("Actualizar")
    self.btn_refresh.setObjectName("primary")
    self.btn_refresh.clicked.connect(self.refresh)

    self.btn_return = QPushButton("Devolver a stock")
    self.btn_return.setObjectName("primary")
    self.btn_return.clicked.connect(self.return_to_stock)

    self.btn_discard = QPushButton("Desechar herramienta")
    self.btn_discard.setObjectName("danger")
    self.btn_discard.clicked.connect(self.discard_tool)

    self.table = QTableWidget(0, 11)
    self.table.setHorizontalHeaderLabels([
        "ID Activa", "Fecha salida", "Herramienta", "Máquina", "Operación",
        "Nivel", "Material", "Usuario", "Tipo", "Estado", "Stock actual"
    ])
    self.table.setSelectionBehavior(QAbstractItemView.SelectRows)
    self.table.setSelectionMode(QAbstractItemView.SingleSelection)
    self.table.setEditTriggers(QAbstractItemView.NoEditTriggers)
    self.table.verticalHeader().setVisible(False)
    self.table.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
    apply_table_polish(self.table)

    top_title = QLabel("Herramientas en uso")
    top_title.setObjectName("h3")

    top = QHBoxLayout()
    top.addWidget(top_title)
    top.addStretch(1)
    top.addWidget(self.btn_refresh)
    top.addWidget(self.btn_return)
    top.addWidget(self.btn_discard)

    wrap = QWidget()
    wlay = QVBoxLayout(wrap)
    wlay.setContentsMargins(0, 0, 0, 0)
    wlay.addLayout(top)
    wlay.addWidget(self.table, 1)
```

```
        lay = QVBoxLayout(self)
        lay.addWidget(module_header)
        lay.addWidget(make_card(wrap), 1)

        self._apply_column_widths()
        self.refresh()

    def _apply_column_widths(self):
        header = self.table.horizontalHeader()
        header.setStretchLastSection(False)
        header.setSectionResizeMode(0, QHeaderView.ResizeToContents)
        for col in range(1, 10):
            header.setSectionResizeMode(col, QHeaderView.Stretch)
        header.setSectionResizeMode(10, QHeaderView.ResizeToContents)

    def _selected_activa(self):
        row = self.table.currentRow()
        if row < 0:
            return None
        activa_id = int(self.table.item(row, 0).text())
        return fetchall("""
            SELECT id, herramienta_id, estado
            FROM herramientas_activas
            WHERE id=?
            LIMIT 1
        """, (activa_id,))

    def refresh(self):
        rows = fetchall("""
            SELECT ha.id,
                   ha.fecha_salida,
                   (h.codigo || ' - ' || h.nombre) AS herramienta,
                   mq.nombre,
                   op.nombre,
                   nv.nombre,
                   COALESCE(NULLIF(ha.material_pieza,''), m.nombre),
                   ha.usuario,
                   ha.tipo_salida,
                   ha.estado,
                   h.stock
            FROM herramientas_activas ha
            JOIN herramientas h ON h.id = ha.herramienta_id
            JOIN maquinas mq ON mq.id = ha.maquina_id
            JOIN operaciones op ON op.id = ha.operacion_id
            JOIN niveles nv ON nv.id = ha.nivel_id
            JOIN materiales m ON m.id = ha.material_id
```

```
            WHERE ha.estado = 'EN_USO'
            ORDER BY ha.id DESC
        """)

        self.table.setRowCount(0)
        for row_data in rows:
            r = self.table.rowCount()
            self.table.insertRow(r)
            for c, v in enumerate(row_data):
                it = QTableWidgetItem(str(v))
                if c in (0, 10):
                    it.setTextAlignment(Qt.AlignCenter)
                self.table.setItem(r, c, it)
            self.table.setRowHeight(r, 36)
        self._apply_column_widths()

    def return_to_stock(self):
        sel = self._selected_activa()
        if not sel:
            return info(self, "Selecciona una herramienta activa.")

        activa_id, herramienta_id, estado = sel[0]
        if estado != "EN_USO":
            return info(self, "Esta herramienta ya no está activa.")

        if not confirm(self, "¿Confirmas devolver esta herramienta a stock como 'Herramienta
con desgaste'?"):
            return

        try:
            con = sqlite3.connect(DB_PATH)
            con.execute("PRAGMA foreign_keys = ON;")
            cur = con.cursor()
            now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            cur.execute("""
                UPDATE herramientas_activas
                SET estado='CERRADA_DEVUELTA',
                    motivo_cierre=?,
                    fecha_cierre=?,
                    devuelta_a_stock=1
                WHERE id=? AND estado='EN_USO'
            """, ("Herramienta con desgaste", now, activa_id))

            if cur.rowcount == 0:
                con.close()
                return info(self, "La herramienta ya fue gestionada previamente.")
```

```
            cur.execute("UPDATE herramientas SET stock_usado =
COALESCE(stock_usado,0) + 1 WHERE id=?", (herramienta_id,))
            cur.execute("UPDATE herramientas SET stock = COALESCE(stock_nuevo,0) +
COALESCE(stock_usado,0) WHERE id=?", (herramienta_id,))
            con.commit()
            con.close()

        except Exception as e:
            return error(self, f"No se pudo devolver a stock:\n{e}")

        info(self, "Herramienta devuelta a stock correctamente.")
        self.refresh()
        mw = self.window()
        if mw and hasattr(mw, "refresh_active_alert"):
            mw.refresh_active_alert()

    def discard_tool(self):
        sel = self._selected_activa()
        if not sel:
            return info(self, "Selecciona una herramienta activa.")

        activa_id, _herramienta_id, estado = sel[0]
        if estado != "EN_USO":
            return info(self, "Esta herramienta ya no está activa.")

        if not confirm(self, "¿Confirmas desechar esta herramienta? Esta acción no la
devuelve a stock."):
            return

        try:
            con = sqlite3.connect(DB_PATH)
            con.execute("PRAGMA foreign_keys = ON;")
            cur = con.cursor()
            now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            cur.execute("""
                UPDATE herramientas_activas
                SET estado='DESECHADA',
                    motivo_cierre='Herramienta desechada',
                    fecha_cierre=?,
                    devuelta_a_stock=0
                WHERE id=? AND estado='EN_USO'
            """, (now, activa_id))

            if cur.rowcount == 0:
                con.close()
```

```
                return info(self, "La herramienta ya fue gestionada previamente.")

            con.commit()
            con.close()

        except Exception as e:
            return error(self, f"No se pudo desechar la herramienta:\n{e}")

        info(self, "Herramienta marcada como desechada.")
        self.refresh()
        mw = self.window()
        if mw and hasattr(mw, "refresh_active_alert"):
            mw.refresh_active_alert()


class TopHerramientasUsadasPage(QWidget):
    def __init__(self):
        super().__init__()

        title = QLabel("Top herramientas más usadas")
        title.setObjectName("h2")

        self.min_date = QDate(1900, 1, 1)
        self.date_from = BlankDateEdit(self.min_date)
        self.date_to = BlankDateEdit(self.min_date)

        self.btn_refresh = QPushButton("Actualizar")
        self.btn_refresh.setObjectName("primary")
        self.btn_refresh.clicked.connect(self.refresh)

        self.btn_clear = QPushButton("Limpiar filtro")
        self.btn_clear.setObjectName("primary")
        self.btn_clear.clicked.connect(self.clear_filter)

        self.lbl_info = QLabel("Mostrando top 10 herramientas por cantidad de selecciones.")
        self.lbl_info.setObjectName("muted")

        top = QHBoxLayout()
        top.addWidget(title)
        top.addStretch(1)
        top.addWidget(QLabel("Desde:"))
        top.addWidget(self.date_from)
        top.addWidget(QLabel("Hasta:"))
        top.addWidget(self.date_to)
        top.addWidget(self.btn_clear)
        top.addWidget(self.btn_refresh)
```

```
        self.chart_container = QVBoxLayout()
        self.chart_container.setContentsMargins(0, 0, 0, 0)
        self.chart_container.setSpacing(8)

        chart_wrap = QWidget()
        chart_wrap.setLayout(self.chart_container)

        self.table = QTableWidget(0, 3)
        self.table.setHorizontalHeaderLabels(["Posición", "Herramienta", "Selecciones"])
        self.table.setSelectionBehavior(QAbstractItemView.SelectRows)
        self.table.setSelectionMode(QAbstractItemView.SingleSelection)
        self.table.setEditTriggers(QAbstractItemView.NoEditTriggers)
        self.table.verticalHeader().setVisible(False)
        self.table.horizontalHeader().setSectionResizeMode(0,
QHeaderView.ResizeToContents)
        self.table.horizontalHeader().setSectionResizeMode(1, QHeaderView.Stretch)
        self.table.horizontalHeader().setSectionResizeMode(2,
QHeaderView.ResizeToContents)
        self.table.horizontalHeader().setStretchLastSection(False)
        self.table.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
        self.table.setMinimumHeight(180)
        self.table.setMaximumHeight(230)
        self.table.horizontalHeader().setDefaultAlignment(Qt.AlignCenter)
        self.table.horizontalHeader().setMinimumHeight(38)
        self.table.setStyleSheet(self.table.styleSheet() + """
            QHeaderView::section {
                font-size: 16px;
                font-weight: 950;
                padding: 8px;
            }
            QTableWidget::item {
                font-size: 14px;
                padding: 7px;
            }
        """)
        apply_table_polish(self.table)

        content = QVBoxLayout(self)
        content.addLayout(top)
        content.addWidget(self.lbl_info)
        content.addWidget(make_glass_card(chart_wrap))
        content.addWidget(self.table, 1)
        content.setStretch(2, 5)
        content.setStretch(3, 3)

        self.refresh()
```

```
    def _get_filter_range(self):
        d_from = self.date_from.date()
        d_to = self.date_to.date()

        if d_from == self.min_date and d_to == self.min_date:
            return None, None
        if d_from != self.min_date and d_to == self.min_date:
            return f"{d_from.toString('yyyy-MM-dd')} 00:00:00", "9999-12-31 23:59:59"
        if d_from == self.min_date and d_to != self.min_date:
            return "0001-01-01 00:00:00", f"{d_to.toString('yyyy-MM-dd')} 23:59:59"
        if d_from > d_to:
            return "INVALID", "INVALID"
        return (
            f"{d_from.toString('yyyy-MM-dd')} 00:00:00",
            f"{d_to.toString('yyyy-MM-dd')} 23:59:59"
        )

    def _query_rows(self):
        start, end = self._get_filter_range()
        if start == "INVALID":
            return []

        base_sql = """
            SELECT (h.codigo || ' - ' || h.nombre) AS herramienta,
                   COUNT(*) AS total
            FROM historial_selecciones hs
            JOIN herramientas h ON h.id = hs.herramienta_id
        """
        if start is None and end is None:
            sql = base_sql + " GROUP BY hs.herramienta_id ORDER BY total DESC,
herramienta ASC LIMIT 10"
            return fetchall(sql)

        sql = base_sql + """
            WHERE hs.fecha_hora BETWEEN ? AND ?
            GROUP BY hs.herramienta_id
            ORDER BY total DESC, herramienta ASC
            LIMIT 10
        """
        return fetchall(sql, (start, end))

    def _clear_chart(self):
        while self.chart_container.count():
            item = self.chart_container.takeAt(0)
            w = item.widget()
            if w:
                w.deleteLater()
```

```
    def _build_chart(self, rows):
        self._clear_chart()
        if not rows:
            self.chart_container.addWidget(QLabel("No hay datos para graficar en el rango
seleccionado."))
            return

        if not HAS_QT_CHARTS:
            self.chart_container.addWidget(QLabel("QtCharts no está disponible en este
entorno."))
            return

        labels = []
        values = []
        for herramienta, total in rows:
            label = str(herramienta)
            if len(label) > 34:
                label = label[:31] + "..."
            labels.append(label)
            values.append(int(total))

        bar_set = QBarSet("Selecciones")
        for v in values:
            bar_set.append(v)
        bar_set.setColor(Qt.cyan)
        bar_set.setBorderColor(QColor("#9FFBFF"))

        series = QBarSeries()
        series.append(bar_set)
        series.setLabelsFormat("@value")
        series.setLabelsVisible(True)
        series.setBarWidth(0.62)

        chart = QChart()
        chart.addSeries(series)
        chart.setTitle("Top 10 herramientas más usadas")
        chart.setAnimationOptions(QChart.SeriesAnimations)
        chart.setBackgroundVisible(False)
        chart.legend().setVisible(False)
        chart.setPlotAreaBackgroundVisible(True)
        chart.setTitleBrush(QColor("#EAF0FF"))
        chart.setTitleFont(QFont("Segoe UI", 14, QFont.Bold))

        axis_x = QBarCategoryAxis()
        axis_x.append(labels)
        axis_x.setLabelsAngle(-28)
```

```
        axis_x.setLabelsBrush(QColor("#EAF0FF"))
        axis_x.setLabelsFont(QFont("Segoe UI", 10, QFont.DemiBold))
        axis_x.setGridLineVisible(False)
        axis_x.setLinePenColor(QColor("#AFC2E8"))
        chart.addAxis(axis_x, Qt.AlignBottom)
        series.attachAxis(axis_x)

        axis_y = QValueAxis()
        max_y = max(values) if values else 1
        upper = max_y + max(1, int(math.ceil(max_y * 0.15)))
        step = max(1, int(math.ceil(upper / 8)))
        upper = int(math.ceil(upper / step) * step)
        axis_y.setRange(0, upper)
        axis_y.setTickType(QValueAxis.TicksDynamic)
        axis_y.setTickAnchor(0)
        axis_y.setTickInterval(step)
        axis_y.setMinorTickCount(0)
        axis_y.setLabelFormat("%d")
        axis_y.setLabelsBrush(QColor("#EAF0FF"))
        axis_y.setLabelsFont(QFont("Segoe UI", 11, QFont.Bold))
        axis_y.setGridLineColor(QColor("#5D6F95"))
        axis_y.setLinePenColor(QColor("#AFC2E8"))
        chart.addAxis(axis_y, Qt.AlignLeft)
        series.attachAxis(axis_y)

        view = QChartView(chart)
        view.setRenderHint(QPainter.Antialiasing)
        view.setMinimumHeight(360)
        self.chart_container.addWidget(view)

    def _fill_table(self, rows):
        self.table.setRowCount(0)
        for idx, (herramienta, total) in enumerate(rows, start=1):
            r = self.table.rowCount()
            self.table.insertRow(r)
            pos_txt = f"{idx}°"
            vals = [pos_txt, str(herramienta), int(total)]
            for c, v in enumerate(vals):
                it = QTableWidgetItem(str(v))
                it.setTextAlignment(Qt.AlignCenter)
                self.table.setItem(r, c, it)
            self.table.setRowHeight(r, 36)

    def refresh(self):
        start, _ = self._get_filter_range()
        if start == "INVALID":
            return info(self, "La fecha 'Desde' no puede ser mayor que la fecha 'Hasta'.")
```

```
        rows = self._query_rows()
        self.lbl_info.setText(f"Mostrando top {len(rows)} herramienta(s) por cantidad de
selecciones.")
        self._build_chart(rows)
        self._fill_table(rows)

    def clear_filter(self):
        self.date_from.setDate(self.min_date)
        self.date_to.setDate(self.min_date)
        if self.date_from.lineEdit():
            self.date_from.lineEdit().setText("")
        if self.date_to.lineEdit():
            self.date_to.lineEdit().setText("")
        self.refresh()


class ReportesPage(QWidget):
    def __init__(self):
        super().__init__()
        header = make_module_header(
            "Informes",
            "",
            ""
        )

        tabs = QTabWidget()
        self.hist = HistorialSeleccionesPage()
        self.top_uso = TopHerramientasUsadasPage()
        tabs.addTab(self.hist, "Historial de selecciones")
        tabs.addTab(self.top_uso, "Top herramientas más usadas")

        lay = QVBoxLayout(self)
        lay.addWidget(header)
        lay.addWidget(make_card(tabs), 1)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Sistema de Herramientas")
        self.resize(1340, 820)

        self.stack = QStackedWidget()
        self.page_tab = TableroPage()
        self.page_cat = CatalogosContainer()
        self.page_rep = ReportesPage()
        self.page_act = HerramientasActivasPage()
```

```
self.stack.addWidget(self.page_tab)
self.stack.addWidget(self.page_cat)
self.stack.addWidget(self.page_rep)
self.stack.addWidget(self.page_act)

self.stack.setStyleSheet("background: transparent;")

side = QVBoxLayout()
side.setContentsMargins(16, 16, 16, 16)
side.setSpacing(10)

logo_top = QLabel("Sistema de Herramientas")
logo_top.setObjectName("sideBrandTitle")

logo_bottom = QLabel("Inventario y selección de herramientas")
logo_bottom.setObjectName("sideBrandSub")

logo_badge = QLabel("Nombre del software: pendiente")
logo_badge.setObjectName("sideBrandPending")

brand_text = QVBoxLayout()
brand_text.setSpacing(3)
brand_text.addWidget(logo_top)
brand_text.addWidget(logo_bottom)
brand_text.addWidget(logo_badge, 0, Qt.AlignLeft)

brand_row = QVBoxLayout()
brand_row.setContentsMargins(14, 12, 14, 12)
brand_row.setSpacing(2)
brand_row.addLayout(brand_text)

brand_wrap = QFrame()
brand_wrap.setObjectName("sideBrandCard")
brand_wrap.setLayout(brand_row)

nav_label = QLabel("NAVEGACIÓN")
nav_label.setObjectName("sideSection")

self.btn_tab = QPushButton("Selección de herramienta")
self.btn_cat = QPushButton("Catálogos")
self.btn_rep = QPushButton("Informes")
self.btn_act = QPushButton("Herramientas activas")

for b in (self.btn_tab, self.btn_cat, self.btn_rep, self.btn_act):
    b.setCheckable(True)
    b.setObjectName("navBtn")
```

```
self.nav_group = QButtonGroup(self)
self.nav_group.setExclusive(True)
self.nav_group.addButton(self.btn_tab, 0)
self.nav_group.addButton(self.btn_cat, 1)
self.nav_group.addButton(self.btn_rep, 2)
self.nav_group.addButton(self.btn_act, 3)

self.btn_tab.clicked.connect(lambda: self.goto_tablero())
self.btn_cat.clicked.connect(lambda: self.goto_catalogos())
self.btn_rep.clicked.connect(lambda: self.goto_reportes())
self.btn_act.clicked.connect(lambda: self.goto_activas())

side.addWidget(brand_wrap)
side.addSpacing(6)
side.addWidget(nav_label)
side.addWidget(self.btn_cat)
side.addWidget(self.btn_tab)
side.addWidget(self.btn_act)
side.addWidget(self.btn_rep)

self.lbl_active_alert = QLabel("")
self.lbl_active_alert.setObjectName("sideAlert")
self.lbl_active_alert.setWordWrap(True)
self.lbl_active_alert.hide()
side.addWidget(self.lbl_active_alert)
side.addStretch(1)

version = QLabel("Versión 1.4.3")
version.setObjectName("sideVersion")
version.setAlignment(Qt.AlignCenter)
side.addWidget(version)

side_w = QFrame()
side_w.setObjectName("sidePanel")
side_w.setLayout(side)
side_w.setFixedWidth(380)

content_wrap = QWidget()
content_wrap_lay = QVBoxLayout(content_wrap)
content_wrap_lay.setContentsMargins(0, 0, 0, 0)
content_wrap_lay.addWidget(self.stack, 1)

root = QWidget()
root.setObjectName("mainBg")
main = QHBoxLayout(root)
main.setContentsMargins(14, 14, 14, 14)
main.setSpacing(14)
```

```
        main.addWidget(side_w)
        main.addWidget(content_wrap, 1)

        self.setCentralWidget(root)

        self.refresh_active_alert()
        self.stack.setCurrentIndex(1)
        self.btn_cat.setChecked(True)

    def refresh_active_alert(self):
        try:
            row = fetchall("SELECT COUNT(*) FROM herramientas_activas WHERE
estado='EN_USO'")
            active_count = int(row[0][0]) if row else 0
        except Exception:
            active_count = 0

        if active_count > 0:
            txt = f"Hay {active_count} herramienta(s) activa(s) en uso."
            self.lbl_active_alert.setText(txt)
            self.lbl_active_alert.show()
            self.btn_act.setText(f"Herramientas activas ({active_count})")
        else:
            self.lbl_active_alert.hide()
            self.lbl_active_alert.setText("")
            self.btn_act.setText("Herramientas activas")

    def goto_catalogos(self):
        try:
            self.page_cat.refresh_counts()
        except Exception:
            pass
        self.refresh_active_alert()
        self.stack.setCurrentIndex(1)
        self.nav_group.button(1).setChecked(True)

    def goto_tablero(self):
        self.refresh_active_alert()
        self.stack.setCurrentIndex(0)
        self.nav_group.button(0).setChecked(True)

    def goto_reportes(self):
        try:
            self.page_rep.hist.refresh()
            self.page_rep.top_uso.refresh()
        except Exception:
            pass
```

```
        self.refresh_active_alert()
        self.stack.setCurrentIndex(2)
        self.nav_group.button(2).setChecked(True)

    def goto_activas(self):
        try:
            self.page_act.refresh()
        except Exception:
            pass
        self.refresh_active_alert()
        self.stack.setCurrentIndex(3)
        self.nav_group.button(3).setChecked(True)

def main():
    init_db()
    app = QApplication(sys.argv)
    dark_theme(app)
    w = MainWindow()
    w.show()
    sys.exit(app.exec())


if __name__ == "__main__":
    main()
```

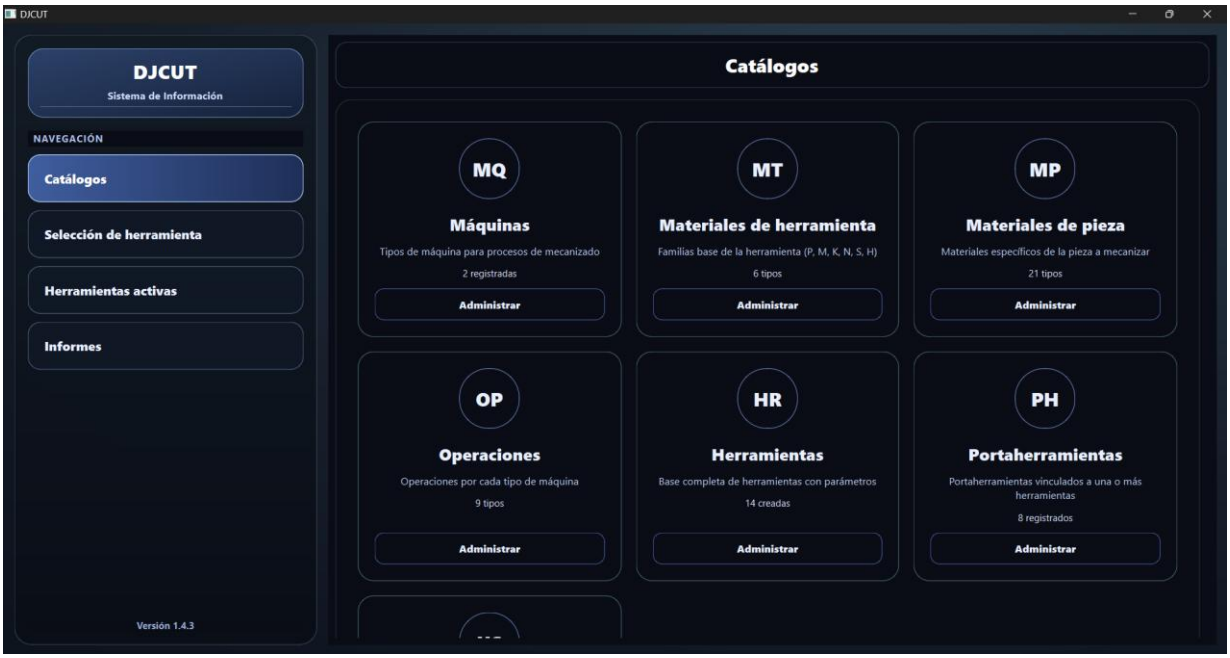
Apéndice D. *Manual*

Manual Sistema de Información DJCUT

DJCUT fue desarrollado para facilitar la selección y gestión de herramientas de torno y fresadora en los procesos de mecanizado. Permite organizar la información de máquinas, materiales, operaciones y herramientas en un solo lugar, logrando un acceso rápido, claro y ordenado a datos de herramientas. A continuación, se presentan las instrucciones para el uso del sistema de información DJCUT, el cual se divide en cinco partes principales.

- Sección 1 (Catálogos)

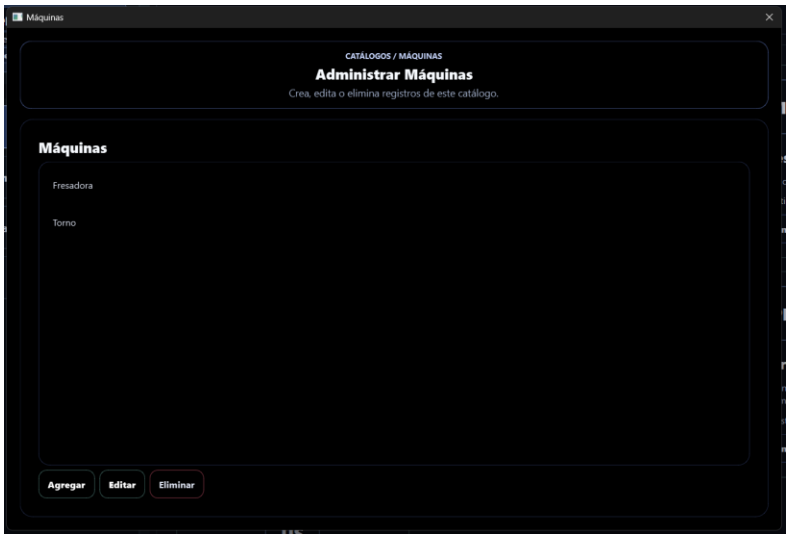
La primera es la sección de catálogos, donde se realiza toda la configuración del sistema; aquí se pueden crear, editar y administrar los elementos necesarios, como máquinas, materiales, operaciones, herramientas, portaherramientas y usuarios, es decir, es donde se define toda la información que el sistema utilizará como base de datos.



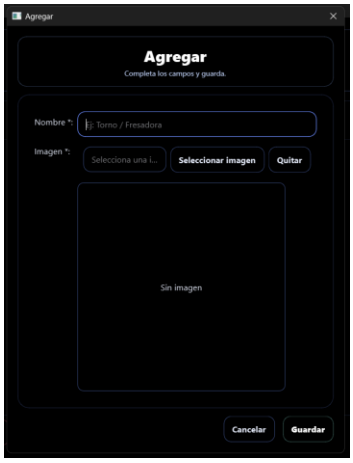
Como se puede observar, cada uno de los módulos cuenta con un botón de "Administrar", el cual permite acceder a un menú adicional. Este menú varía según el módulo seleccionado, mostrando las opciones correspondientes a la sección en la que se encuentre el usuario.

1. Máquinas

Haciendo clic en el botón de “Administrar” de máquinas, se desplegará el siguiente menú, donde se podrán visualizar las máquinas creadas actualmente.

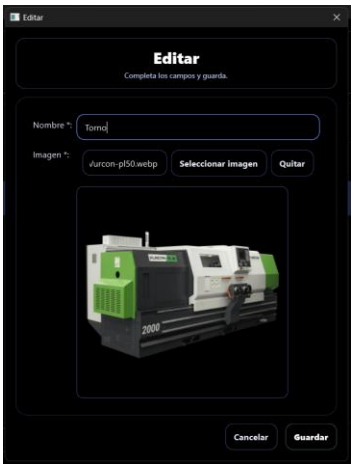


Para crear una máquina, se debe hacer clic en el botón “Agregar”, el cual llevará al siguiente menú.



En este apartado se deben diligenciar los datos correspondientes para crear la máquina. Una vez completada la información, se debe hacer clic en “Guardar” y la máquina quedará registrada en el menú de máquinas.

Para editar una máquina existente, se debe hacer clic primero en la máquina que se desea modificar y luego en el botón “Editar”. Esto desplegará el siguiente menú.

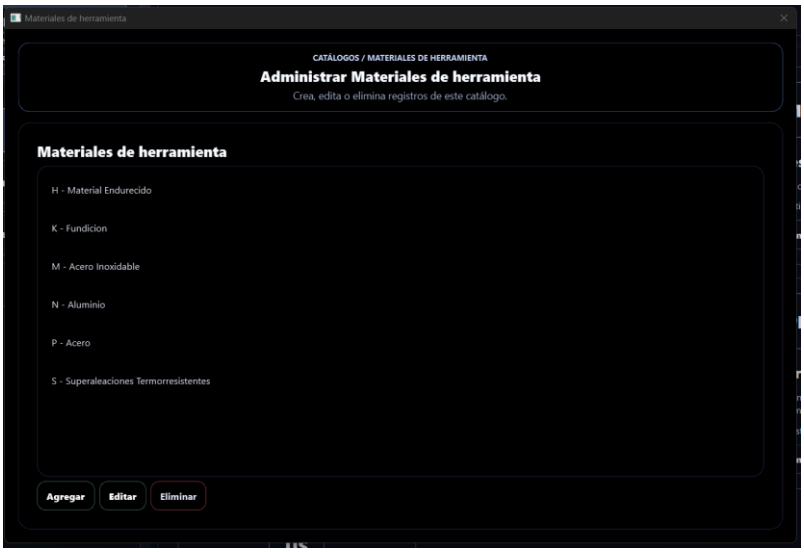


Se realizan los cambios deseados y, una vez verificados, se debe hacer clic en “Guardar”. Los cambios se reflejarán automáticamente en el menú de máquinas.

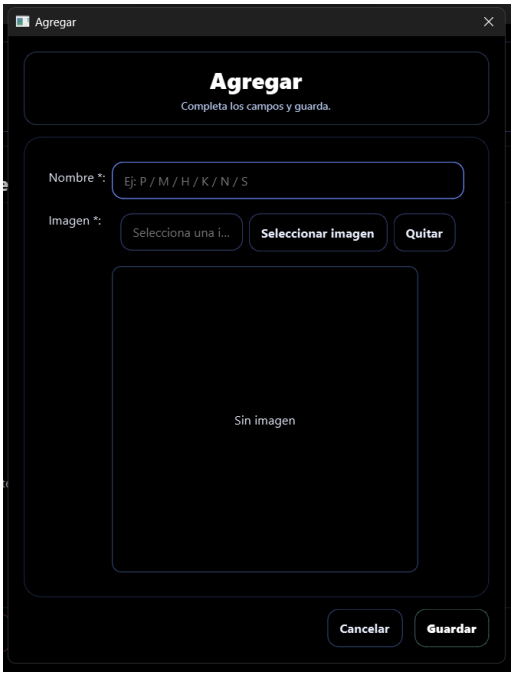
Para eliminar una máquina existente, se debe hacer clic en la máquina que se desea eliminar y luego en el botón “Eliminar”. Esto hará que la máquina desaparezca automáticamente del sistema de información.

2. Materiales de herramienta

Haciendo clic en el botón de “Administrar” de materiales de herramienta, se desplegará el siguiente menú, donde se podrán visualizar los materiales de herramienta creados actualmente.



Para crear un material de herramienta, se debe hacer clic en el botón “Agregar”, el cual llevará al siguiente menú.



En este apartado se deben diligenciar los datos correspondientes para crear el material. Una vez completada la información, se debe hacer clic en “Guardar” y el material quedará registrado en el menú de materiales de herramienta.

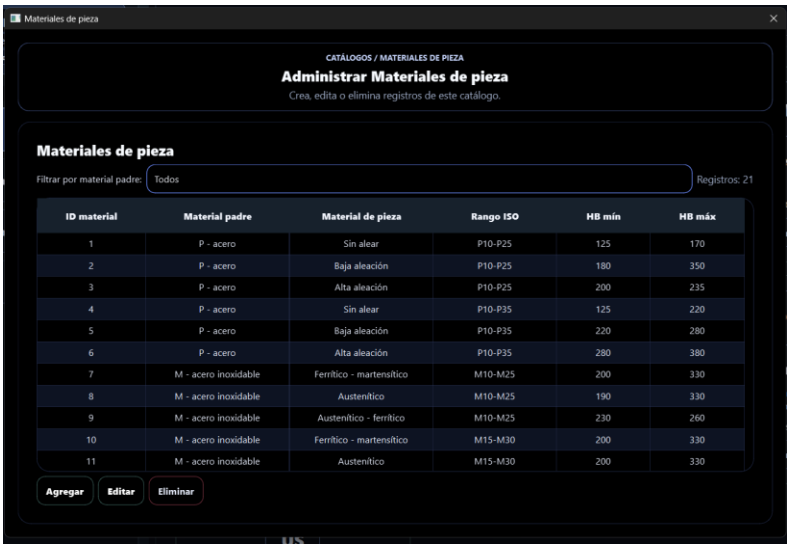
Para editar el material existente, se debe hacer clic primero en el material que se desea modificar y luego en el botón “Editar”. Esto desplegará el siguiente menú.

Se realizan los cambios deseados y, una vez verificados, se debe hacer clic en “Guardar”. Los cambios se reflejarán automáticamente en el menú de materiales de herramienta.

Para eliminar un material de herramienta existente, se debe hacer clic en el material que se desea eliminar y luego en el botón “Eliminar”. Esto hará que el material desaparezca automáticamente del sistema de información.

3. Materiales de pieza

Haciendo clic en el botón de “Administrar” de materiales de pieza, se desplegará el siguiente menú, donde se podrán visualizar los materiales de pieza creados actualmente.



Para crear un material de pieza, se debe hacer clic en el botón “Agregar”, el cual llevará al siguiente menú.

En este apartado se deben diligenciar los datos correspondientes para crear el material de la pieza a mecanizar. Es importante tener en cuenta que el material padre corresponde a los materiales de herramienta previamente creados; por lo tanto, se debe asignar uno, ya que este definirá su familia. Una vez completada la información, se debe hacer clic en “Guardar” y el material quedará registrado en el menú de materiales de pieza.

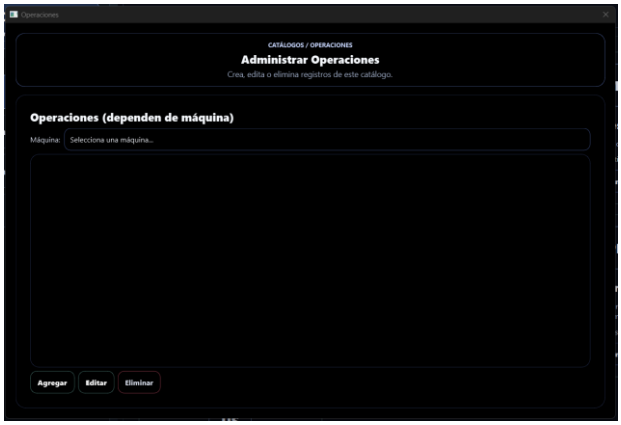
Para editar el material de pieza existente, se debe hacer clic primero en el material que se desea modificar y luego en el botón “Editar”. Esto desplegará el siguiente menú.

Se realizan los cambios deseados y, una vez verificados, se debe hacer clic en “Guardar”. Los cambios se reflejarán automáticamente en el menú de materiales de pieza.

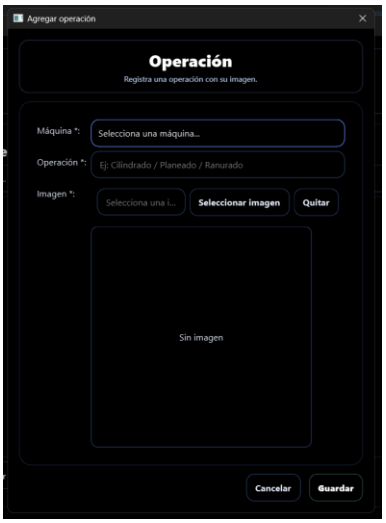
Para eliminar un material de herramienta existente, se debe hacer clic en el material que se desea eliminar y luego en el botón “Eliminar”. Esto hará que el material desaparezca automáticamente del sistema de información.

4. Operaciones

Haciendo clic en el botón de “Administrar” de operaciones, se desplegará el siguiente menú, donde se podrán visualizar las operaciones por máquinas creadas actualmente.

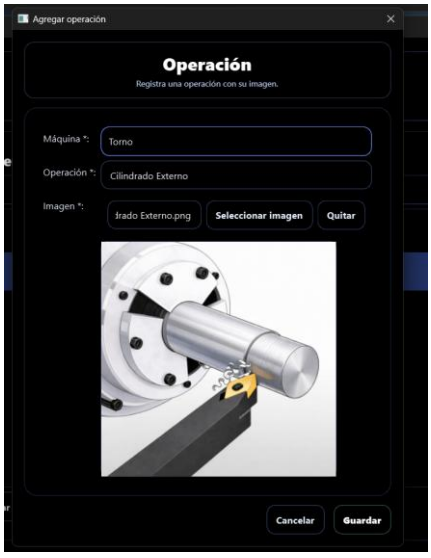


Para crear una operación, se debe hacer clic en el botón “Agregar”, el cual llevará al siguiente menú.



En este apartado se deben diligenciar los datos correspondientes para crear la operación de la máquina. Es importante tener en cuenta que la operación se le debe asignar una máquina ya creada. Una vez completada la información, se debe hacer clic en “Guardar” y la operación quedará registrada en el menú de operaciones.

Para editar una operación existente, se debe seleccionar la máquina y luego clic en la operación que se desea modificar y luego en el botón “Editar”. Esto desplegará el siguiente menú.

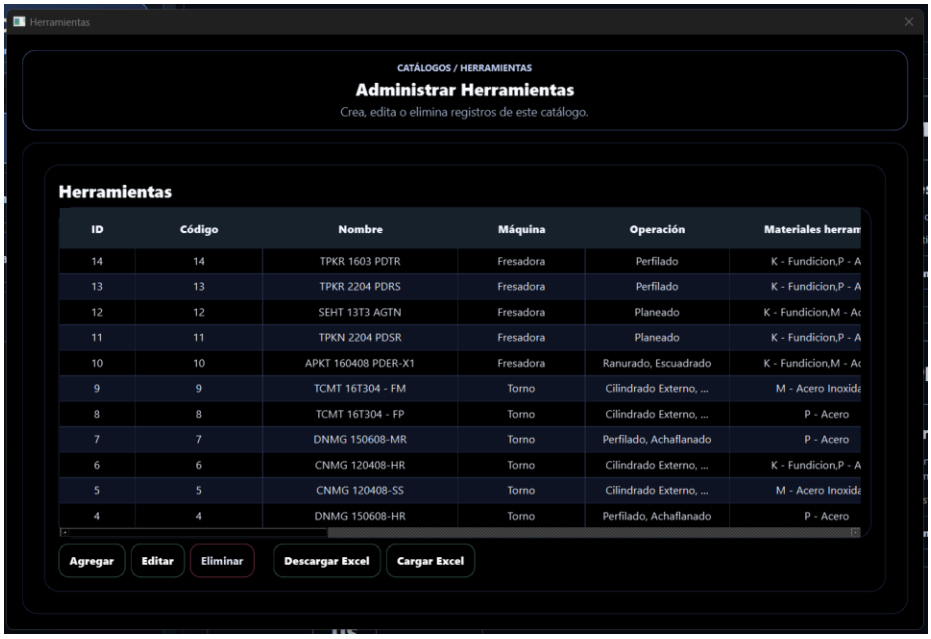


Se realizan los cambios deseados y, una vez verificados, se debe hacer clic en “Guardar”. Los cambios se reflejarán automáticamente en el menú de operaciones.

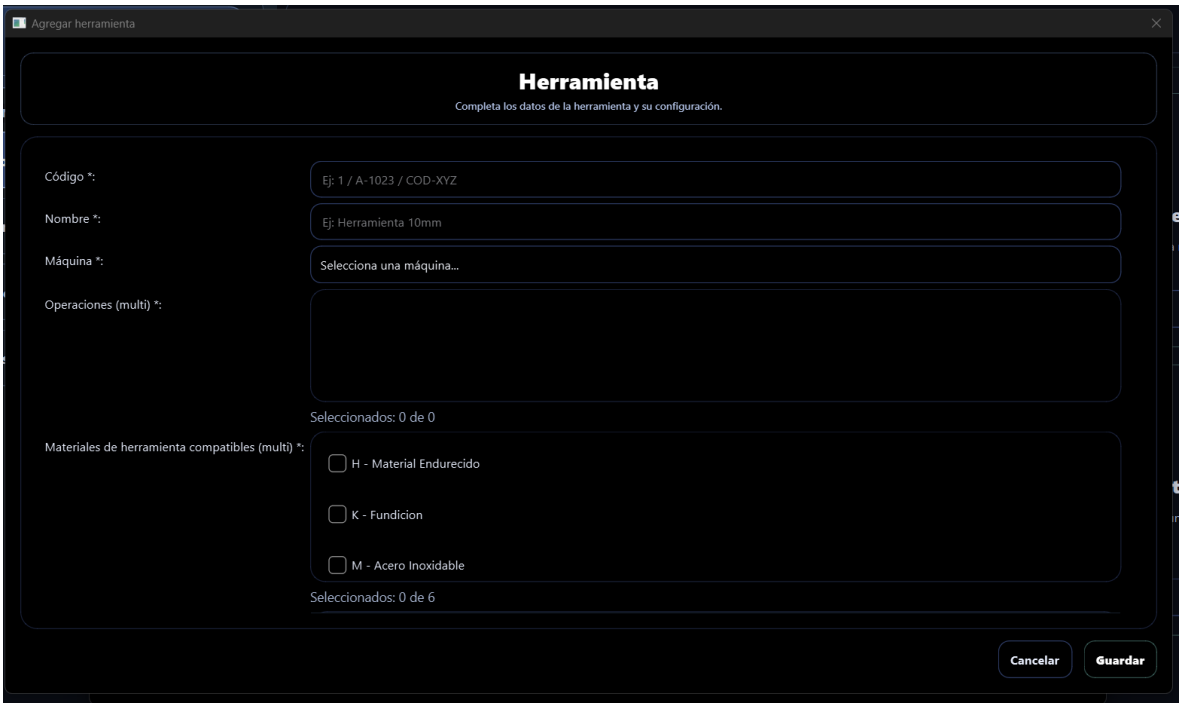
Para eliminar la operación existente, se debe seleccionar la máquina y luego hacer clic en la operación que se desea eliminar y luego en el botón “Eliminar”. Esto hará que la operación desaparezca automáticamente del sistema de información.

5. Herramientas

Haciendo clic en el botón de “Administrar” de herramientas, se desplegará el siguiente menú, donde se podrán visualizar las herramientas creadas actualmente.

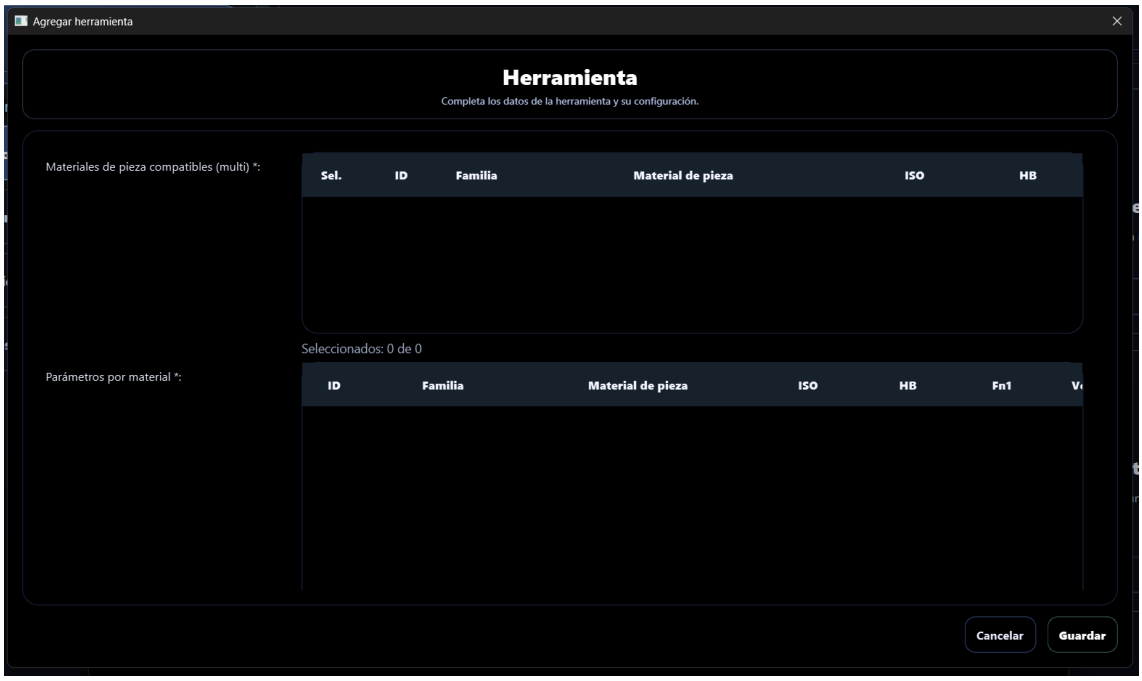


Para crear una herramienta, se debe hacer clic en el botón “Agregar”, el cual llevará al siguiente menú.

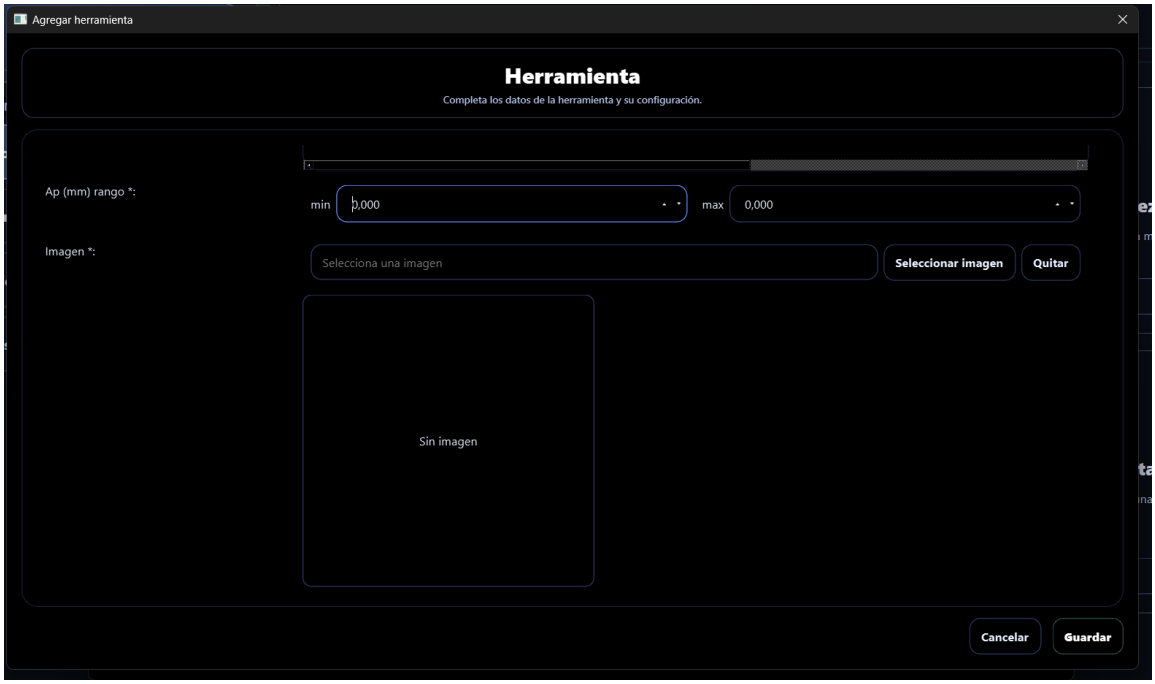


En este apartado se deben diligenciar los datos correspondientes. Es importante tener en cuenta que el código y el nombre deben ser asignados por el usuario. En el campo de máquina se mostrará una lista desplegable con las máquinas creadas actualmente; esta selección es clave, ya que de ella dependen las operaciones que se podrán asignar a la herramienta que se está creando. En el campo de materiales, se podrán seleccionar uno o varios de los que aparecen en el listado desplegable.

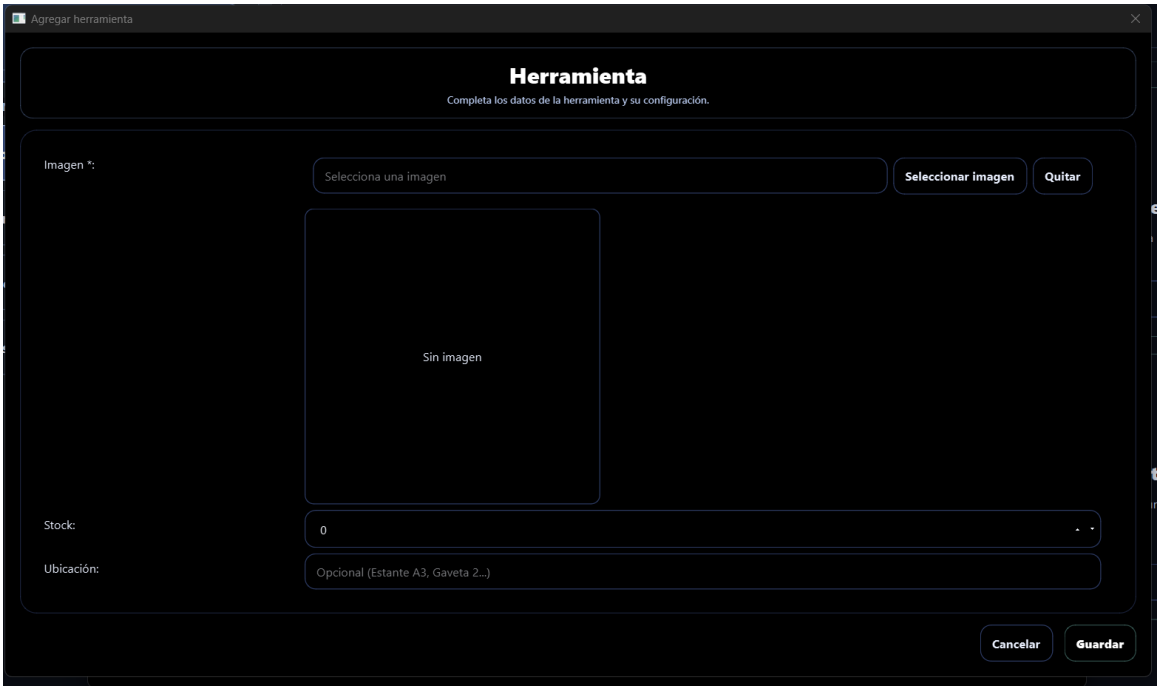
Una vez seleccionado el material se habilitarán las dos tablas de la siguiente imagen, en la primera tabla se desplegará una serie de materiales de pieza que se pueden asignar a la herramienta; en esta sección es posible seleccionar varios. Posteriormente, en la segunda tabla aparecerán los materiales de pieza seleccionados y será necesario diligenciar los valores de avance y velocidad de corte, ya que estos son específicos para cada herramienta. El sistema está configurado para permitir el ingreso de hasta tres valores de avance y tres de velocidad de corte, cada uno con su respectivo mínimo y máximo; sin embargo, únicamente es obligatorio registrar al menos un valor de avance y uno de velocidad de corte con sus rangos mínimo y máximo.



Una vez diligenciada la información en cada una de las tablas, se debe completar el valor de Ap y cargar la imagen correspondiente a la herramienta que se desea crear.

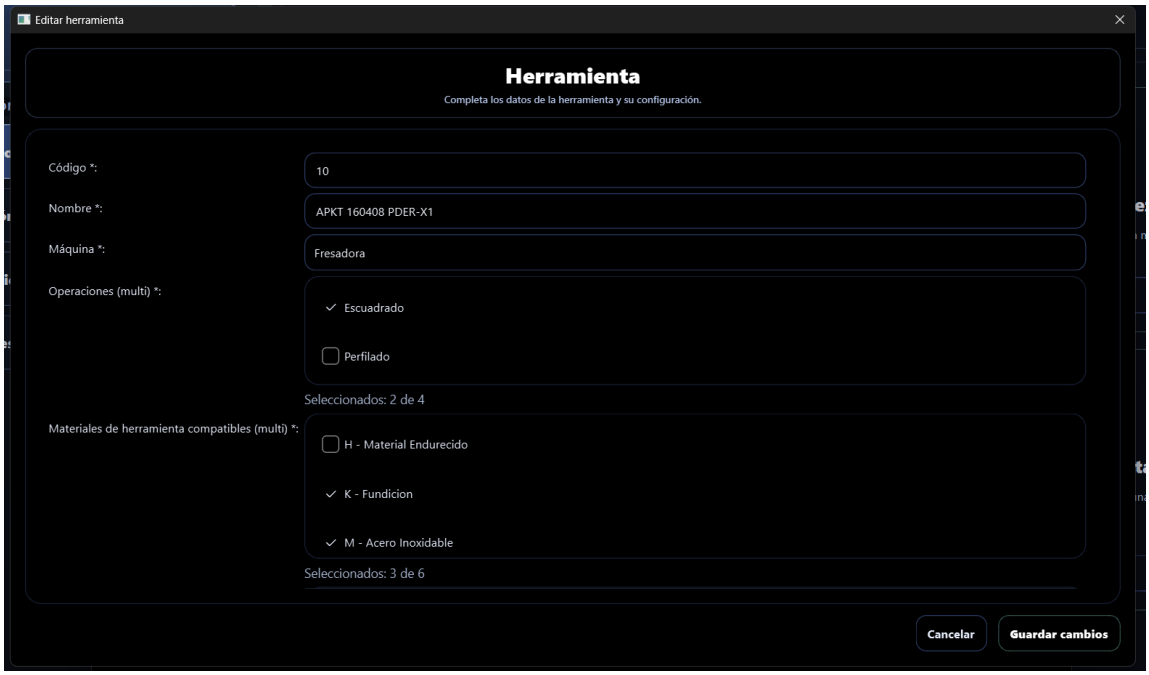


En el campo de stock es importante ingresar un valor diferente de cero, ya que, aunque la herramienta se puede crear con un stock en cero, no será posible seleccionarla. Esto se debe a que el sistema la considerará como una herramienta no disponible.



Una vez completada la información, se debe hacer clic en “Guardar” y la herramienta quedará registrada en el menú de herramientas.

Para editar una operación existente, se debe dar clic en la herramienta y luego clic en el botón “Editar”. Esto desplegará el siguiente menú.

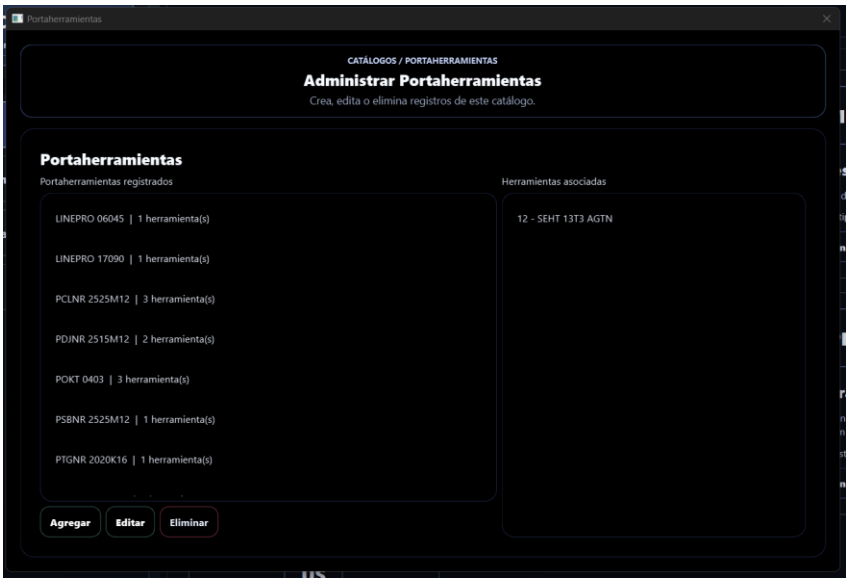


Se realizan los cambios deseados y, una vez verificados, se debe hacer clic en "Guardar". Los cambios se reflejarán automáticamente en el menú de herramientas.

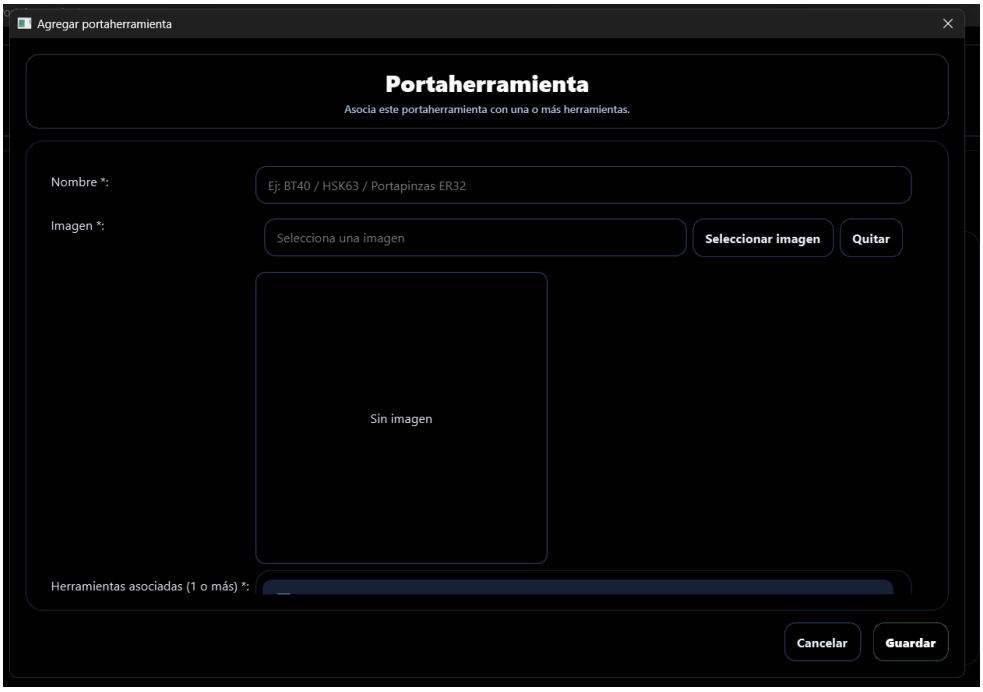
Para eliminar una herramienta existente, se debe seleccionar la herramienta y luego hacer clic luego en el botón "Eliminar". Esto hará que la operación desaparezca automáticamente del sistema de información.

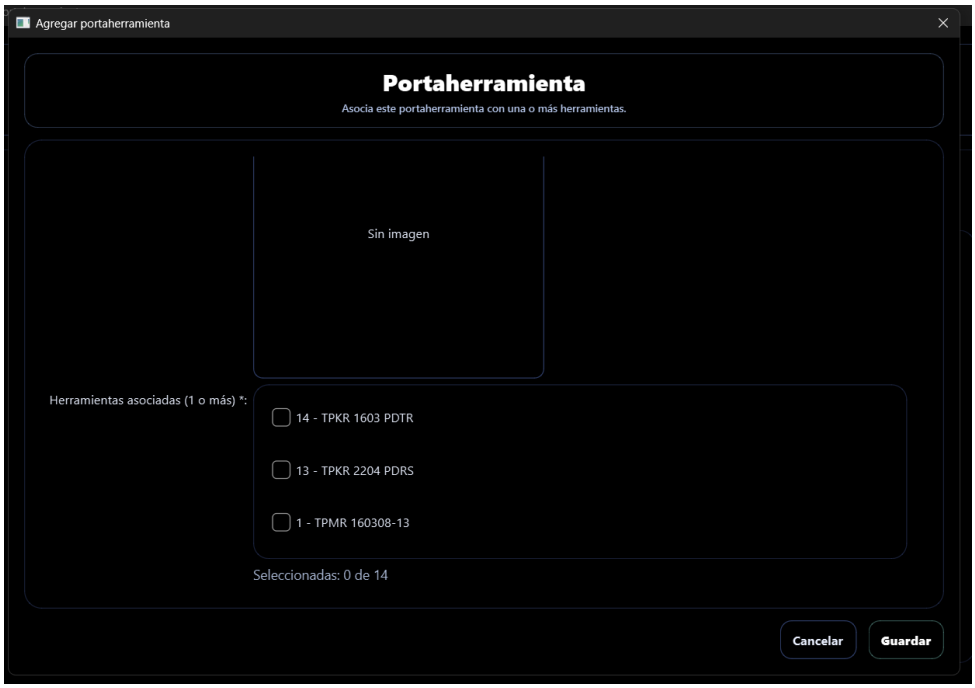
6. Portaherramientas

Haciendo clic en el botón de "Administrar" de portaherramientas, se desplegará el siguiente menú, donde se podrán visualizar los portaherramientas creados actualmente.



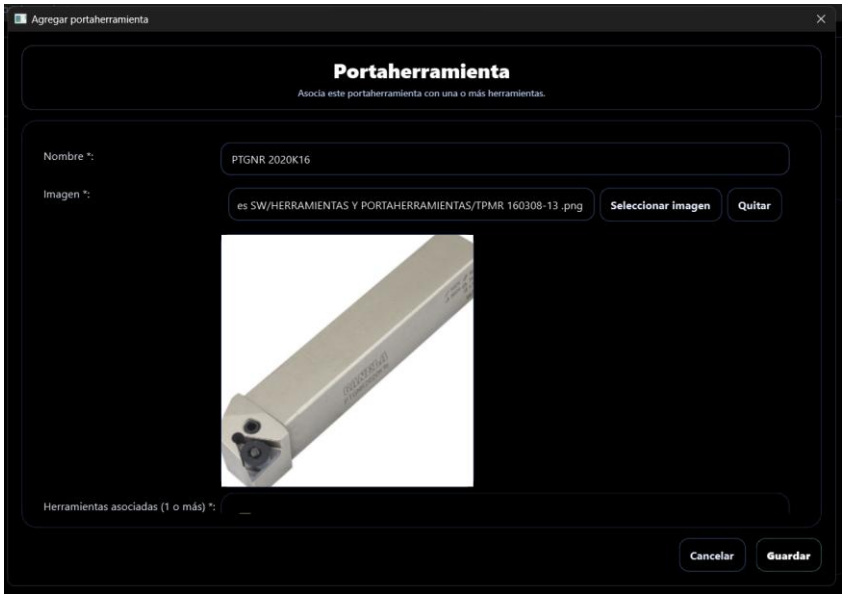
Para crear un portaherramientas, se debe hacer clic en el botón “Agregar”, el cual llevará al siguiente menú.





En este apartado se deben diligenciar los datos correspondientes para crear el portaherramientas. Es importante tener en cuenta que al portaherramientas se le debe asignar una herramienta ya creada. Una vez completada la información, se debe hacer clic en “Guardar” y la operación quedará registrada en el menú de portaherramientas.

Para editar un portaherramientas existente, se debe seleccionar el portaherramientas que se desea modificar y luego en el botón “Editar”. Esto desplegará el siguiente menú.

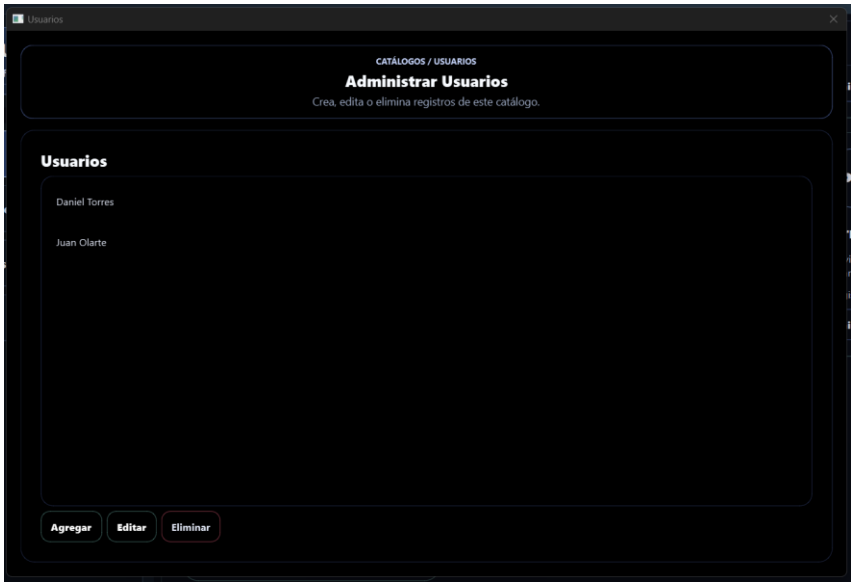


Se realizan los cambios deseados y, una vez verificados, se debe hacer clic en “Guardar”. Los cambios se reflejarán automáticamente en el menú de portaherramientas.

Para eliminar un portaherramientas, se debe seleccionar el portaherramientas que se desea eliminar y luego en el botón “Eliminar”. Esto hará que la operación desaparezca automáticamente del sistema de información.

7. Usuarios

Haciendo clic en el botón de “Administrar” de operaciones, se desplegará el siguiente menú, donde se podrán visualizar los usuarios creados actualmente.



Para crear un usuario, se debe hacer clic en el botón “Agregar”, el cual llevará al siguiente menú.

En este apartado se deben diligenciar los datos correspondientes para crear el usuario. Una vez completada la información, se debe hacer clic en “Guardar” y el usuario quedará registrado en el menú de usuarios.

Para editar un usuario existente, se debe seleccionar el nombre del usuario y luego clic en el botón “Editar”. Esto desplegará el siguiente menú.

Se realiza el cambio deseado y, una vez verificado, se debe hacer clic en “Guardar”. Los cambios se reflejarán automáticamente en el menú de usuarios.

Para eliminar un usuario existente, se debe seleccionar el usuario y luego hacer clic en el botón “Eliminar”. Esto hará que el usuario desaparezca automáticamente del sistema de información.

- Sección 2 (Selección de herramienta)

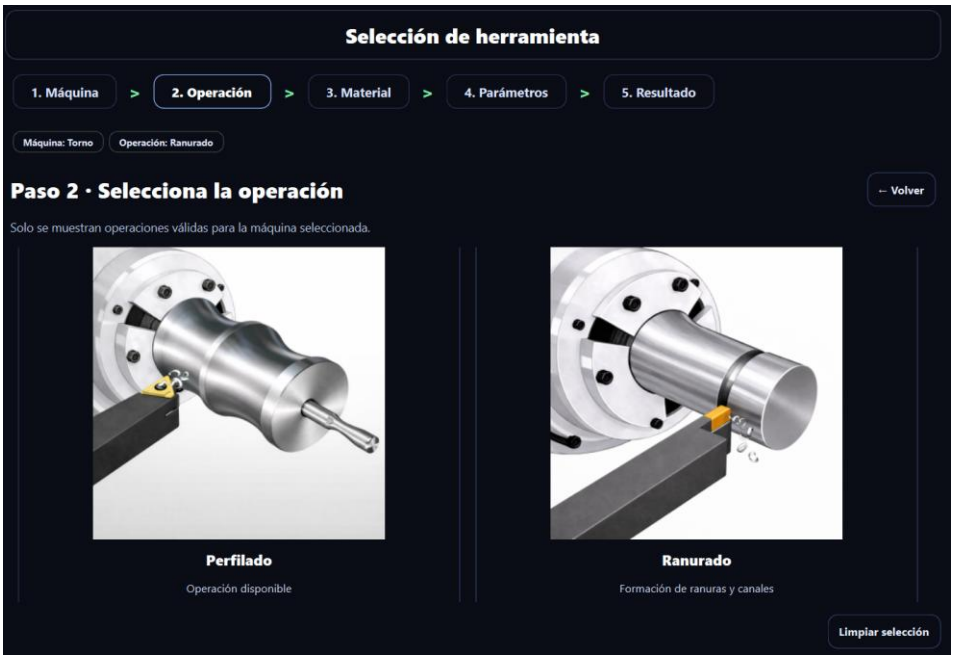
La segunda sección es la de selección de herramienta, en la cual el sistema guía al usuario de forma intuitiva para elegir la herramienta adecuada; es importante tener en cuenta que esta parte depende de los catálogos, por lo que, si una herramienta no ha sido creada previamente o no tiene stock, no podrá ser seleccionada.

El sistema de información guiará al usuario en cinco pasos para realizar la selección de la herramienta. En este proceso, se seguirá un orden cronológico que incluye la máquina en la que se va a mecanizar, la operación a realizar, el tipo de material de la pieza, los parámetros de corte y, finalmente, el resultado de la selección.

La selección se realiza simplemente haciendo clic sobre cada una de las opciones, lo que hace que el proceso sea fácil e intuitivo. Por lo cual a continuación se hará un ejemplo:

1. Torno

2. Ranurado



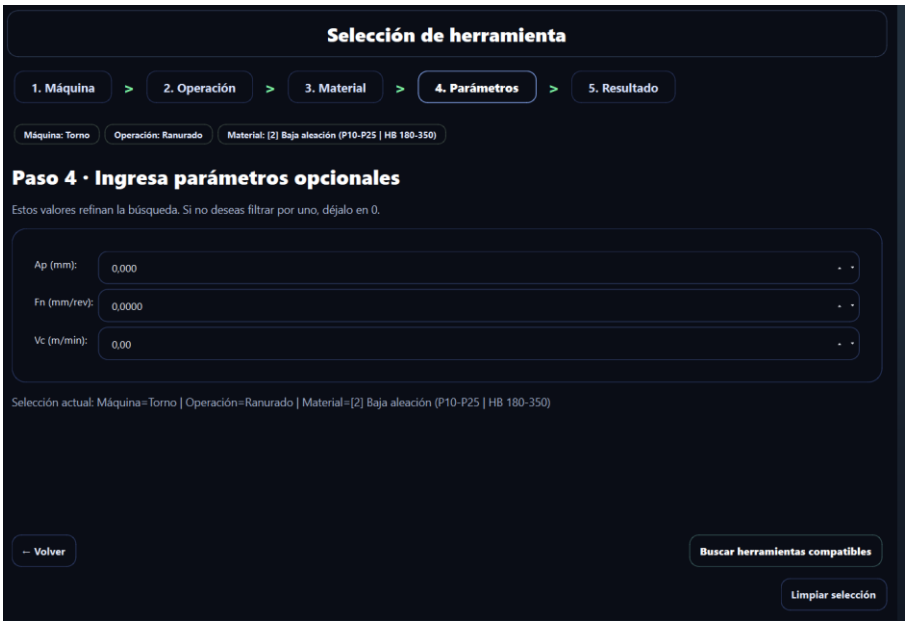
3. P – Acero

4. Acero de baja aleación

Se hace clic en el material y luego seleccionar

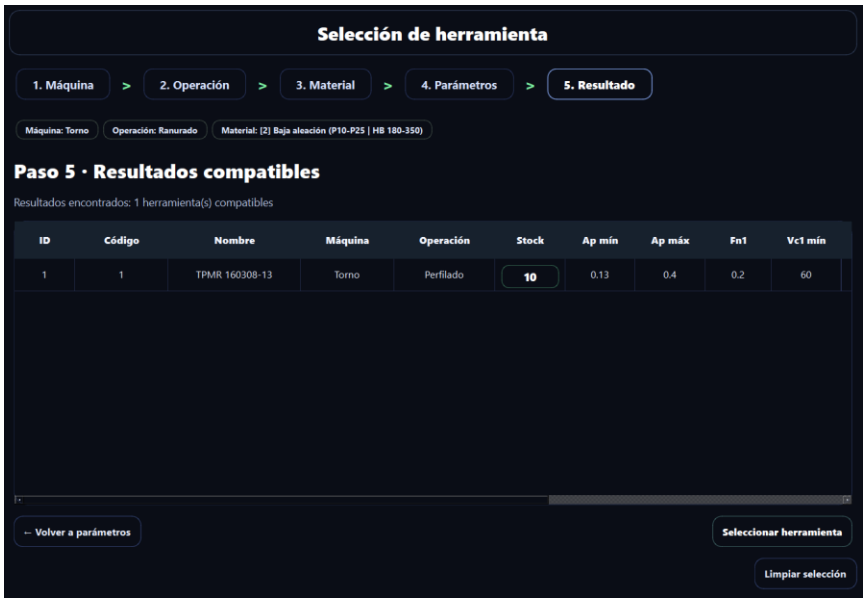
5. Parámetros

Los parámetros se pueden dejar vacíos o simplemente colocar los valores requeridos, en este caso se dejaron vacíos para ver varias opciones de herramienta.

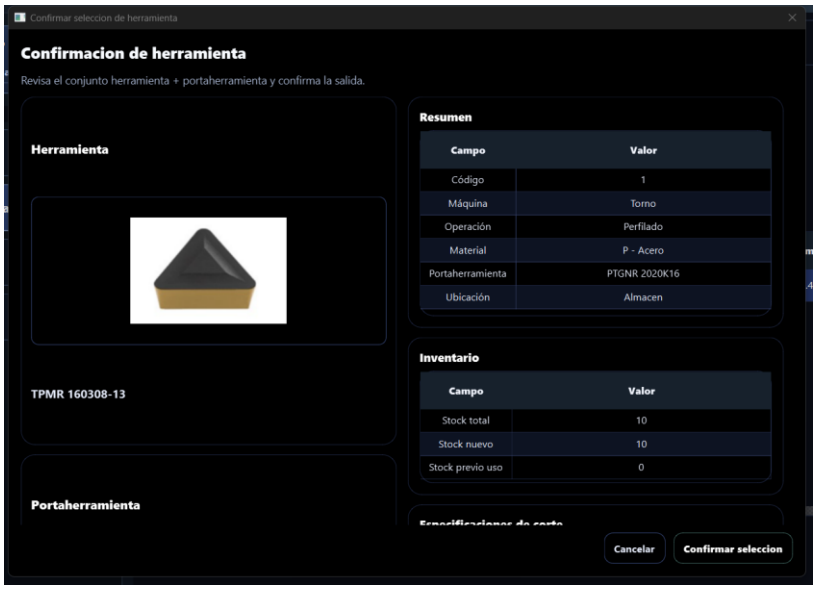


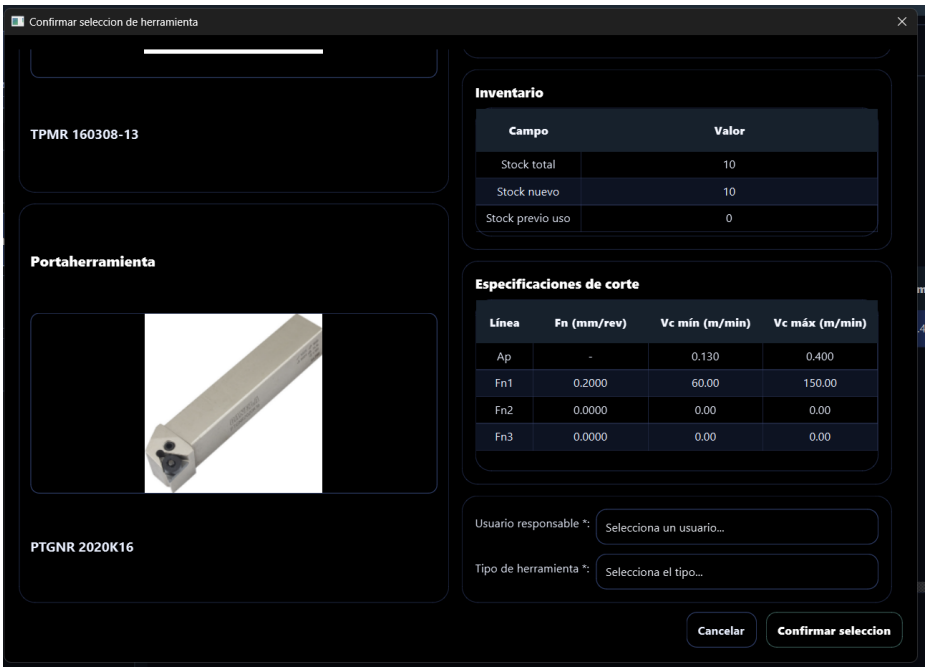
6. Selección de la herramienta

Se debe hacer clic en el botón “Buscar herramientas compatibles”, lo que desplegará un menú con las herramientas disponibles según la selección realizada.

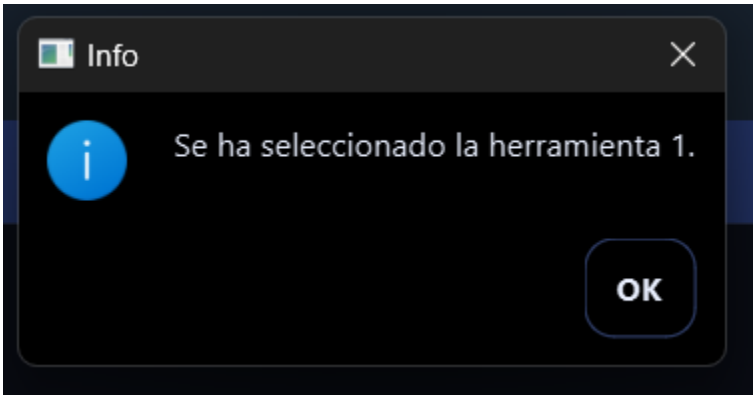


Se debe hacer clic sobre la herramienta deseada y luego en el botón “Seleccionar herramienta”. A continuación, se mostrará una ventana para confirmar la selección.



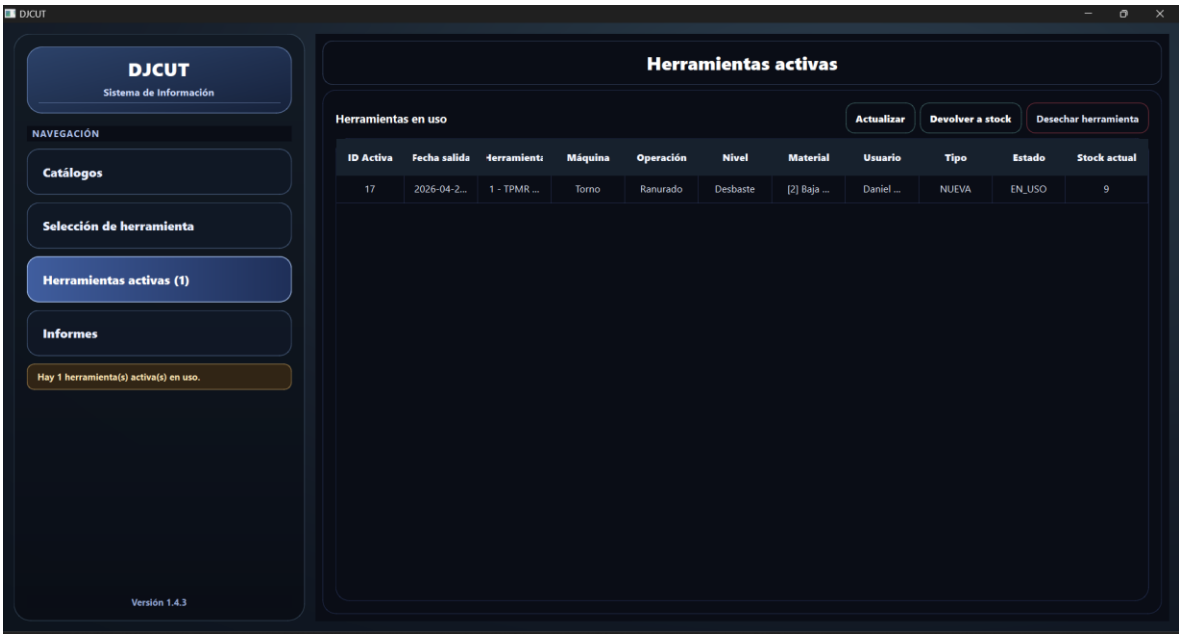


En esta ventana se mostrarán las especificaciones de la herramienta, la cantidad disponible y su respectivo portaherramientas. Es importante seleccionar el usuario que realiza la selección y el tipo de herramienta, ya que se puede elegir entre una herramienta nueva o una que ya haya tenido uso previo. Por último, se mostrará una notificación como confirmación de la selección.



- Sección 3 (herramientas activas)

La tercera es sección de herramientas activas, donde se visualiza las herramientas que se encuentran en uso.



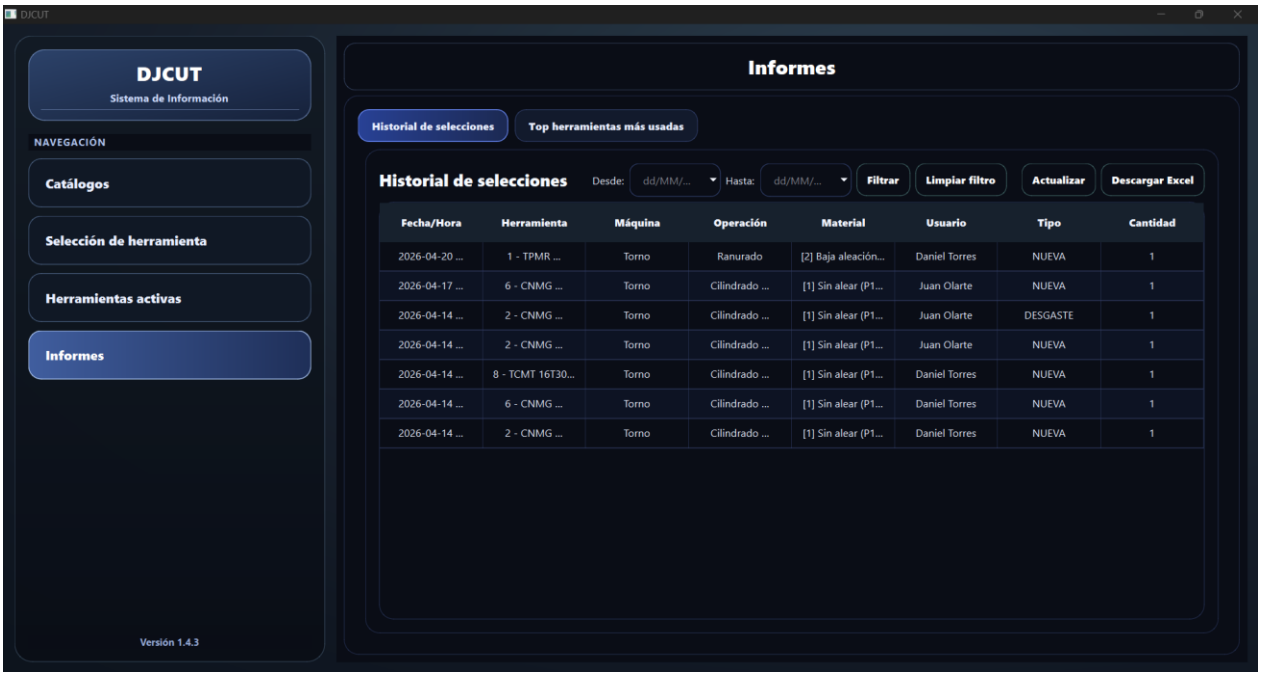
En este apartado se pueden realizar dos acciones importantes: devolver la herramienta al stock (cuando aún cuenta con vida útil) o desecharla (cuando presenta un desgaste excesivo). Para devolver una herramienta al stock, se debe hacer clic sobre la herramienta deseada y luego en el botón “Devolver a stock”. Por otro lado, si se desea desecharla, se debe seleccionar la herramienta y hacer clic en “Desechar herramienta”; al realizar esta acción, se descontará una unidad del stock disponible.

- Sección 4 (Informes)

En esta sección, se encuentra la sección de informes, donde se puede consultar el historial de selecciones realizadas y visualizar cuáles son las herramientas más utilizadas, permitiendo así llevar un seguimiento del uso del sistema.

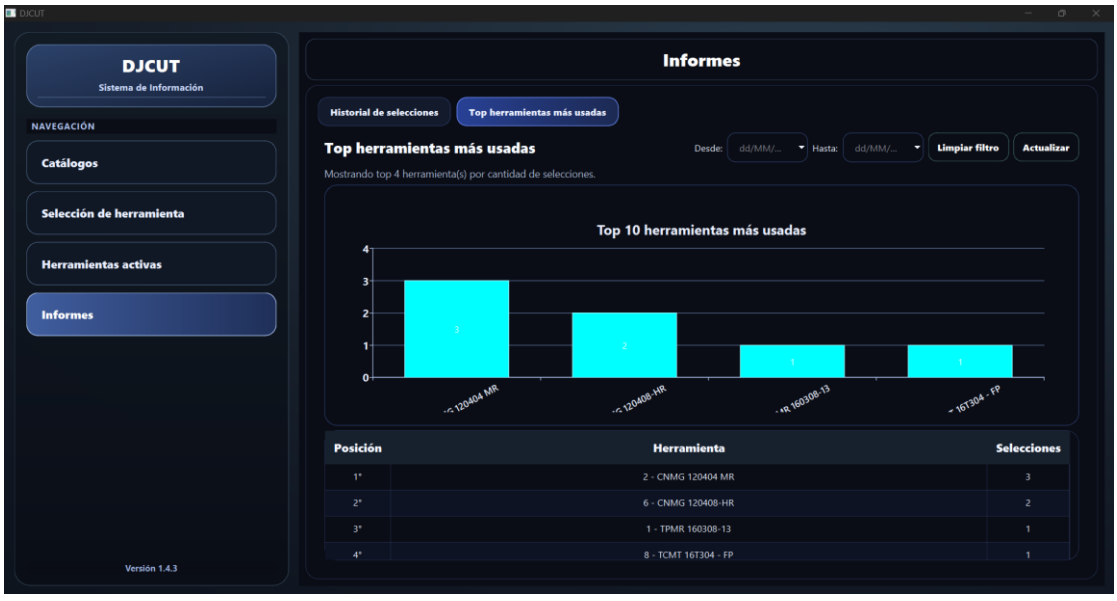
En el apartado de historial de selecciones se pueden visualizar las herramientas que han sido seleccionadas a lo largo del tiempo. Esta información incluye la fecha y hora, la herramienta utilizada, la operación para la cual fue seleccionada, el material con el que se empleó, el usuario que realizó la selección, si la herramienta era nueva o tenía desgaste, y la cantidad seleccionada en ese momento.

Además, se cuenta con un filtro por fechas, el cual permite consultar la información por días específicos o por intervalos de tiempo, como semanas o meses.



En la sección de herramientas más usadas se puede visualizar una gráfica con las herramientas que han tenido mayor uso a lo largo del tiempo. Esta información puede consultarse por días, semanas o meses, según se requiera.

Es importante tener en cuenta que la gráfica muestra un máximo de diez herramientas. En cuanto al número de selecciones por herramienta, la gráfica permite visualizar hasta un máximo de 20 registros.



- Sección 5 (Ayuda)

Por último, en este apartado se encuentra el manual del sistema de información DJCUT, el cual puede ser consultado en cualquier momento en formato PDF.