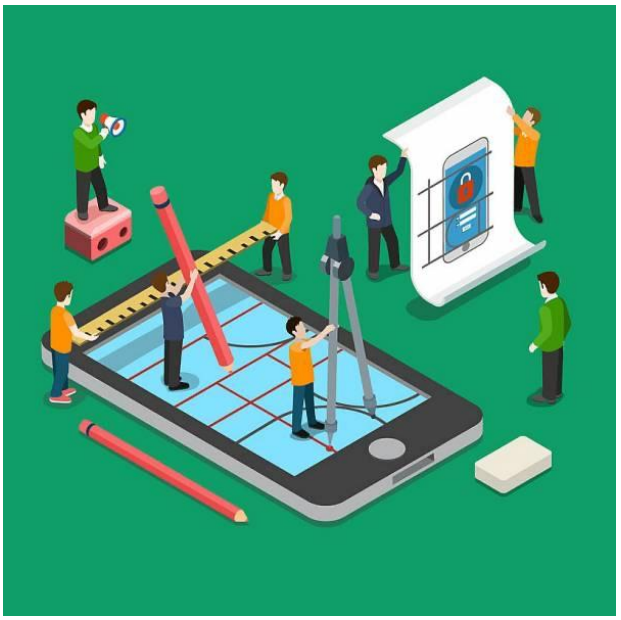
CIP de FP BATOI

DESENROTLLAMENT D'INTERFÍCIES
2N DAM

UD2: Creació i us de components



Creació de bases de dades i taules fent ús de PySide6 i SQLite

En aquesta part, anem a connectar a una base de dades que farem amb SQLite.

Per començar la pràctica, anem a introduir a SQLite.

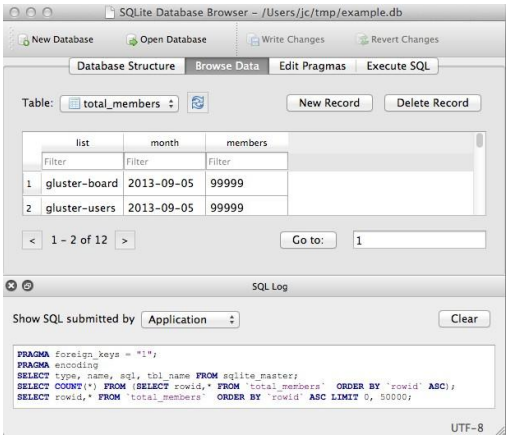
SQLite és un sistema gestor de bases de dades relacional sent aquest un motor de bases de dades molt lleuger que ens permetrà emmagatzemar informació per a una aplicació d'escriptori o per una app. SQLite fa ús del llenguatge SQL que ja coneixem dels cursos anteriors (per això no tindrem problemes a l'hora de fer un INSERT, SELECT o un UPDATE).

La pregunta és, per a què fem ús d'una base de dades amb SQLite i no fem ús de MySQL o MariaDB per exemple?

La resposta és que per la simplicitat que té en el seu ús, així com en la seua lleugeresa i la seua portabilitat. Les bases de dades son arxius senzills, portables i fàcils d'utilitzar, copiar o emmagatzemar. Aquestes bases de dades no consumeixen quasi recursos i ens serà útil per aplicacions que NO tenen un volum elevat de dades.
SQLite no requereix cap manteniment ni administració i és ideal per a entorns de dispositius IoT i aplicacions mòbils.
Si volem utilitzar un gran volum de dades, SQLite NO és l'opció adequada.
També cal destacar que SQL és de codi obert, gratuït, es de domini públic i per tant no tenim que preocupar-nos per les llicències.

DB BROWSER FOR SQLITE

Ara anem a preparar l'entorn. Per això farem ús de la ferramenta visual **DB Browser for SQLite** (**DB4S**) (https://sqlitebrowser.org/) que és de codi obert i multiplataforma i que ens permetrà crear, dissenyar i editar arxius de bases de dades que siguen compatibles amb SQLite.



En primera instància ens descarreguem la ferramenta DB Browser for SQLite i després començarem a programar i anirem veient com apareixen les noves taules i les dades i registres que anem inserint en la nostra base de dades creada amb SQLite.

Ara tornem al **Visual Studio Code** i creem un fitxer anomenat consultes.py on anem a practicar tot el necessari per connectar amb el servidor i crear una nova base de dades.

En primera instància farem els imports necessaris per poder treballar, concretament
import sqlite3 # Librería para la gestión de bases de datos SQLite
from PySide6.QtSql import QSqlDatabase, QSqlTableModel, QSqlQuery

En primera instància amb **addDatabase** indicarem que treballarem amb QSQLITE. Podeu mirar els diferents drivers per connectar altres SGBD (https://doc.qt.io/qt-6/sql-driver.html).

Supported Databases

The table below lists the drivers included with Qt:

Driver name	DBMS
QDB2	IBM DB2 (version 7.1 and above)
QMYSQL / MARIADB	MySQL or MariaDB (version 5.6 and above)
QOCI	Oracle Call Interface Driver (version 12.1 and above)
QODBC	Open Database Connectivity (ODBC) - Microsoft SQL Server and other ODBC-compliant databases
QPSQL	PostgreSQL (versions 7.3 and above)
QSQLITE	SQLite version 3

Després amb **setDatabaseName** indicarem el nom de la base de dades que anem a crear.

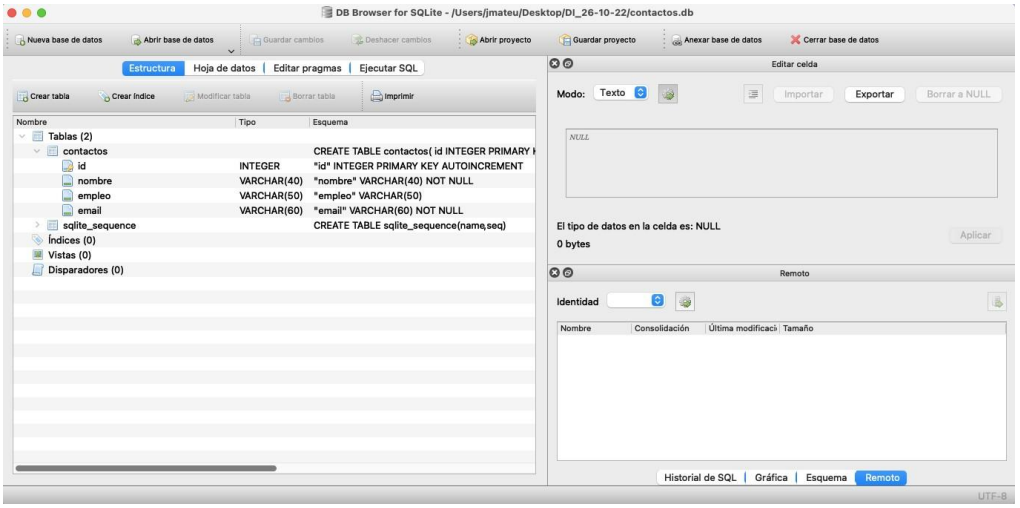
```
def _connectar_bd(self):
    """
    Abre (o crea) la BD SQLite en el archivo local SQLite/contactos.db
    y deja la conexión abierta en self.db para usarla en todo el programa.
    """
    self.db = QSqlDatabase.addDatabase("QSQLITE")
    self.db.setDatabaseName("SQLite/contactos.db")

    if not self.db.open():
        QMessageBox.critical(
            self,
            "Error BD",
            "No s'ha pogut obrir la BD SQLite/contactos.db"
        )
        sys.exit(1)
```

Ara anem a crear una taula de contactes i per això farem ús de QSqlQuery(), el qual ens permetrà fer ús de les sentències SQL que ja coneixem. Amb exec podem executar el codi i després anar al DB Browser por SQLite per comprovar si apareix la taula amb la seua estructura.

```
def _crear_taula_si_no_existe(self):
    """
    Crea la tabla 'personas' si aún no existe.
    """
    consulta = QSqlQuery()
    consulta.exec(
        """
        CREATE TABLE IF NOT EXISTS personas
        (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nombre  VARCHAR(40) NOT NULL,
            empleo  VARCHAR(50),
            email   VARCHAR(60) NOT NULL
        )
        """
    )
```

En la següent imatge podem comprovar com ja tenim la taula contactes creada amb els camps id, nom, ocupació i email.



Ara anem a afegir un nou registre i per això podem emprar el còdec que adjunte a continuació:

```python
def _taula_buida(self):
    """
    Devuelve True si la tabla 'personas' está vacía.
    """
    consulta = QSqlQuery("SELECT COUNT(*) FROM personas")
    if consulta.next(): #Se situa en el primer (y único) resultado
        total = consulta.value(0)
        return total == 0
    return True

def _insertar_demo_si_cal(self):
    """
    Inserta un registro de ejemplo SOLO si la tabla está vacía.
    Así los alumnos ven datos nada más ejecutar.
    """
    if not self._taula_buida():
        return

    nom = "Iván"
    ocupacio = "Professor"
    correo = "i.martos@profesor.com"

    consulta = QSqlQuery()
    consulta.prepare(
        "INSERT INTO personas (nombre, empleo, email) VALUES (?, ?, ?)"
    )
    consulta.addBindValue(nom)
    consulta.addBindValue(ocupacio)
    consulta.addBindValue(correo)

    if not consulta.exec():
        QMessageBox.warning(
            self,
            "Error inserció",
            "No s'ha pogut inserir el registre inicial."
        )
```

Podem comprovar amb DB Browser for SQLite si apareix el registre que acabem d'inserir.

A continuació, el que anem a fer es mostrar les dedes de la base de dades en una taula en la interfície, per això utilitzem el següent còdec:

```python
def _configurar_model_i_taula(self):
    """
    Crea el modelo QSqlTableModel vinculado a la tabla 'personas'
    y la QTableView para mostrar los datos.
    """
    self.model = QSqlTableModel(self, self.db)
    self.model.setTable("personas")

    # Estrategia OnFieldChange:
    # cualquier cambio en la celda se guarda directamente en la BD.
    self.model.setEditStrategy(QSqlTableModel.OnFieldChange)

    # Ejecuta una consulta SELECT para cargar los datos de la tabla personas en el modelo.
    self.model.select()

    # Nombres bonitos para las columnas visibles
    self.model.setHeaderData(1, Qt.Horizontal, "Nom")
    self.model.setHeaderData(2, Qt.Horizontal, "Emplec")
    self.model.setHeaderData(3, Qt.Horizontal, "Email")

    # Creamos la vista de tabla
    self.taula = QTableView()
    self.taula.setModel(self.model)

    # Ocultar la columna ID (columna 0)
    self.taula.setColumnHidden(0, True)

    # Ajustar tamaño de columnas
    self.taula.resizeColumnsToContents()
```

Ara anem a donar a la taula la funcionalitat de transmetre signals al seleccionar una fila de la taula amb la propietat selectionModel:

```python
def _conectar_seleccio_fila(self):
    """
    Conecta la señal de cambio de fila seleccionada.
    Cuando el usuario selecciona una fila en la tabla,
    leemos sus datos y los mostramos (por consola y en la barra de estado).
    """
    sel_model = self.taula.selectionModel() #emite señales cuando cambia la selección en la tabla
    sel_model.currentRowChanged.connect(self._fila_seleccionada_canvia) #emite señal cuando cambia la fila seleccionada
    #envia la fila actual seleccionada y la anterior seleccionada (current, previous)
```

Finalment generem una funció que mostre per consola i al status bar les dades de la fila seleccionada:

```python
def _fila_seleccionada_canvia(self, current, previous):
    """
    current = nueva fila seleccionada
    previous = fila seleccionada antes
    Leemos los datos de la fila actual del modelo y los mostramos.
    """
    if not current.isValid():
        print("Sense selecció")
        return

    fila = current.row()

    # Ojo: Aquí estamos leyendo directamente del modelo base (self.model),
    # no de la vista (self.taula), porque la vista puede tener filtros o
    # estar ordenada de forma diferente. Por eso usamos current.row() para obtener
    # la fila correcta en el modelo base.
    idx_nom = self.model.index(fila, 1)    # columna 'nombre'
    idx_emp = self.model.index(fila, 2)    # columna 'empleo'
    idx_cor = self.model.index(fila, 3)    # columna 'email'

    nom = self.model.data(idx_nom)
    emp = self.model.data(idx_emp)
    correo = self.model.data(idx_cor)

    print("Fila seleccionada:")
    print(" - Nom   :", nom)
    print(" - Emplec:", emp)
    print(" - Email :", correo)

    # Feedback visual rápido en la barra de estado
    self.statusBar().showMessage(f"Seleccionat: {nom} ({emp})")
```