Auteur: karim NASR
Date: 15/11/2025

Objet du document:

A pour but d'expliquer quelques programmes pour approfondir QT.

Historique:

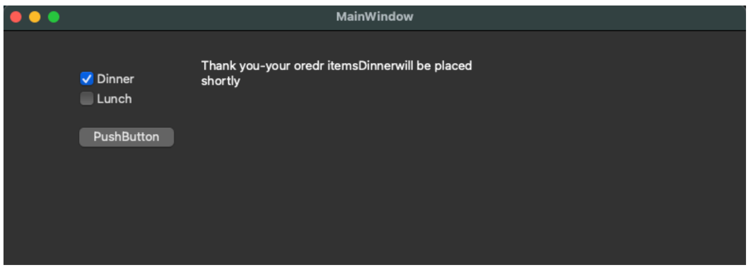
18/09/024 KNA ajout de QtThreeWidgetTutorial.Exemple sur les checkbox.
18/09/024 KNA, évolution de la première version, ajout de LCD et de la somme de la commande.
20/09/024 KNA, ajout de nouveaux projet DisplayWindows et ReadingWritingTextFile.
12/10/2024 KNA ajout exemple sur FontDialog-utilisateur change la police de caractere dans fenêtre, InputDialogMultiline, ToolBar.
Debut de manipulations avec les exemples du cours dans documentation « Qt-Widgets-Layouts »: QvBoxLayout.Ajout de QtWidgetGroupBox/QtWidgetSliderGroup/QtCheckedListWidget/QtLCDnumber.
11/11/2024 KNA ajout exemple sur la gestion d'une base de donnees contacts par SQLite (nom du projet GestionContacts)
13/11/2024 KNA Modification de la GestionContact par ajout de la gestion categorie avec une comboBox.
28/10/2025 KNA ajout TP horloge digitale compte a rebours
15/11/2025 KNA ajout Qt-QsqlRelationalModel (projet Pychart),QDataWidgetMapper extrait livre Qt framework,Qt-CreationFormulaire, utilisation QFONT avec formulaire

QtThreeWidgetTutorial:

Contient un exemple basique sur les checkbox dans QtCreator.
Ici le choix est fait entre Dinner et lunch et qhand appui sur « Push button » on affiche dans le label les differents choix faits.

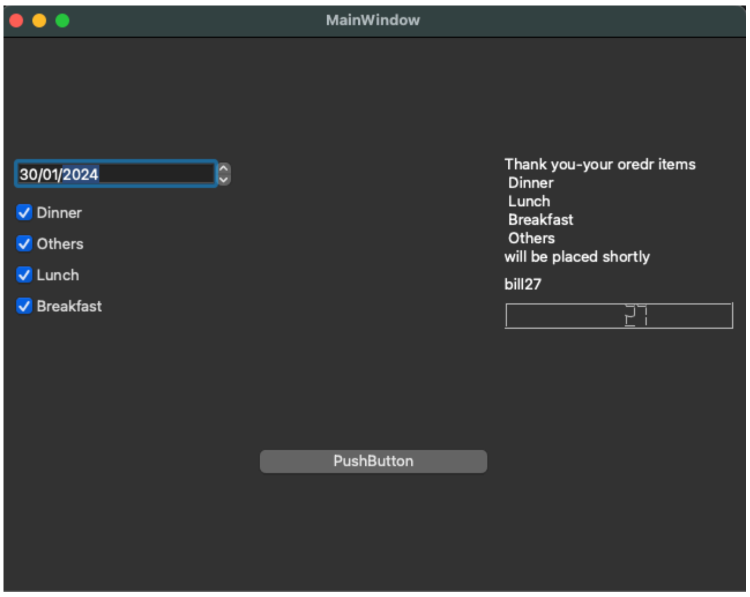
Extrait youtube
QCheckBox | How to use QCheckBox in Qt5 | (Qt C++ Tutorial #16)

MacDigia



Thank you-your oredr itemsDinnerwill be placed shortly

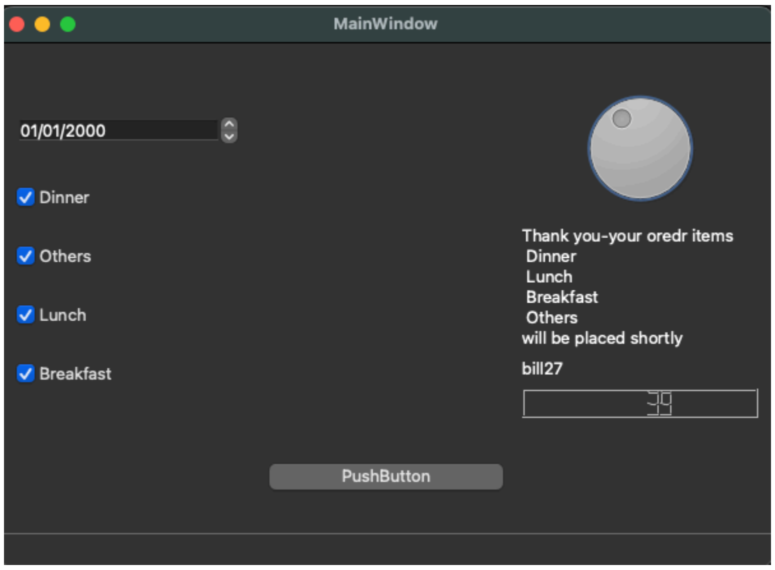
Ajout dans la deuxième version d'un label d 'affichage de la somme ainsi d'un LCD d'affichage de la valeur entière de la somme.



Thank you-your oredr items
Dinner
Lunch
Breakfast
Others
will be placed shortly

bill27

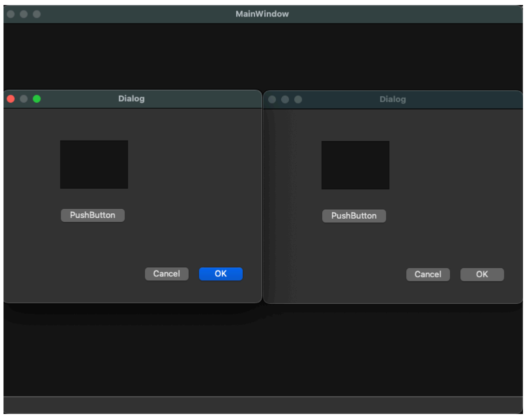
Ajout d'un curseur circulaire dont la valeur est affichée sur le LCD lorsque celui change.
On affiche également toujours la somme de la commande sur le LCD.



Thank you-your oredr items
Dinner
Lunch
Breakfast
Others
will be placed shortly

bill27

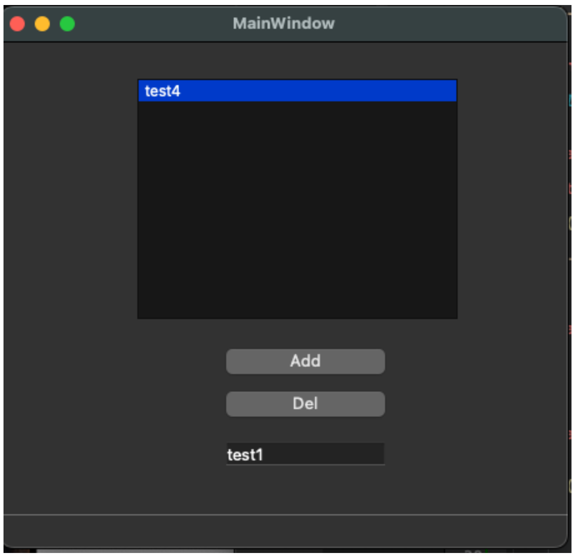
Le dossier displayingWindows est extrait de youtibe VoidRealms
Creation d'un projet nommé DisplayingWindows sous Qt
pour la création de boites de dialogues extrait de youtube VoidRealms
// creation d'un nouveau formulaire de dialog dans formulaire et le fichier //dialog.h
// modification de mainwindow.h pour intégrer Dialog *mDialog;



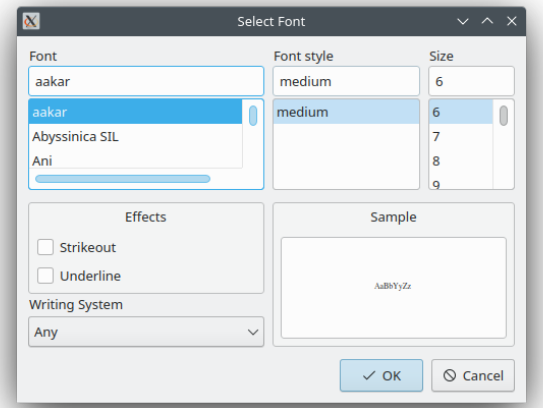
Le dossier ReadingWritingTextFile est extrait de YouTube
On ecrit et lit dans un fichier « test.txt » avec OStream, si on change le nom du fichier alors message « file not found »

Qt 6 - Episode 19 - Read and writing text files
VoidRealms

Le dossier QlistWidgetExemple est extrait de YouTube de LearnQT (How to add items and delete selected items with QListWidget)
Montre les fonctions de list widget , on peut ajouter ou supprimer
Des items saisies dans lineEdit avec boutons Add/Del



Le dossier QtFontdialog est extrait du site https://github.com/gammasoft71/Examples_Qt/tree/master
Montre une fenêtre ou utilisateur change au choix la police
Output



Font: aakar
aakar
Abyssinica SIL
Ani

Font style: medium
medium

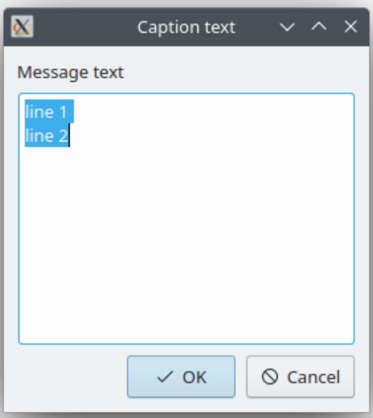
Size: 6
6
7
8
9

Effects
Strikeout
Underline

Writing System: Any

Sample
AaBbYyZz

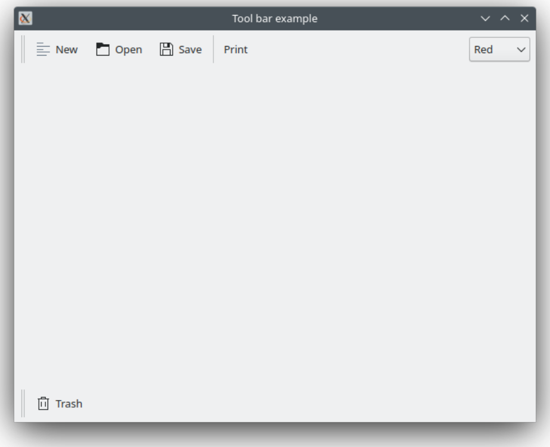
Le dossier Qt-InputDialogMultiline est extrait du même site



Message text

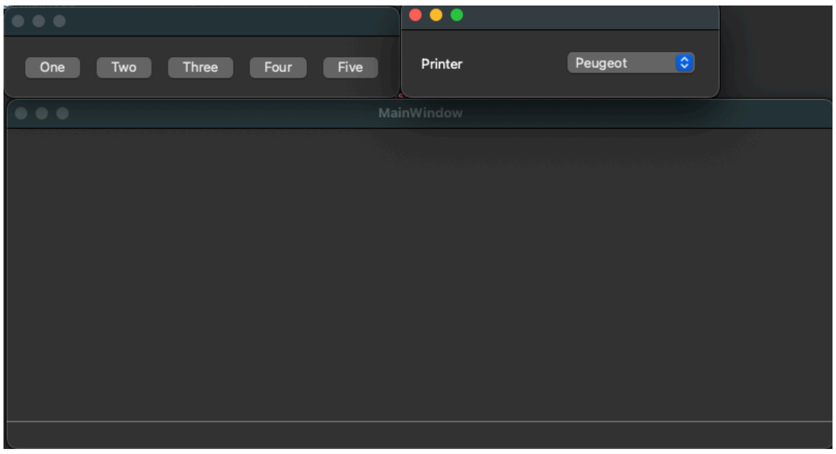
line 1
line 2

Le dossier Qt-ToolBar est extrait du même site



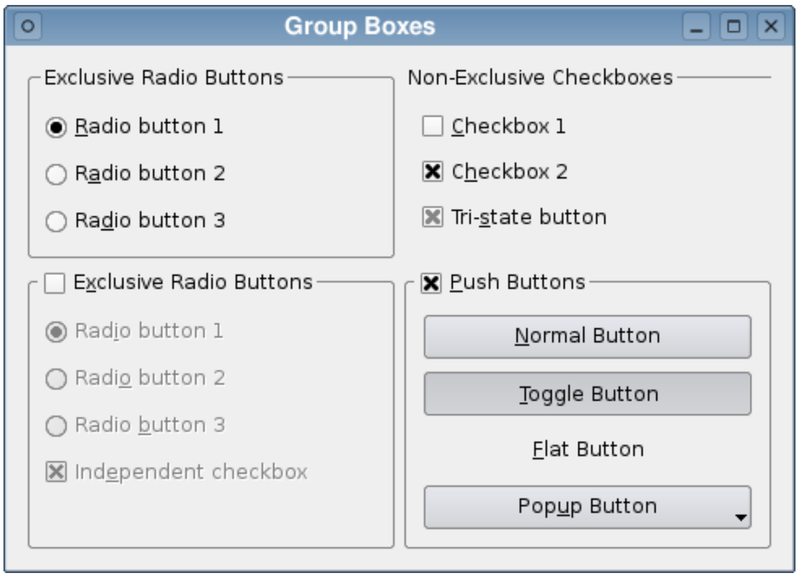
Les exemples suivants sont extrait du document Qt-Widgets-Layouts.dans repertoire documentation.

-QtVboxLayout (contient une ComboBox) , le tout configuré dans le main du programme principal.



Exemple sur Widget Group boxes (extrait de https://stuff.mit.edu/afs/athena/software/texmaker_v5.0.2/qt57/doc/qtwidgets/qtwidgets-widgets-groupbox-example.html)

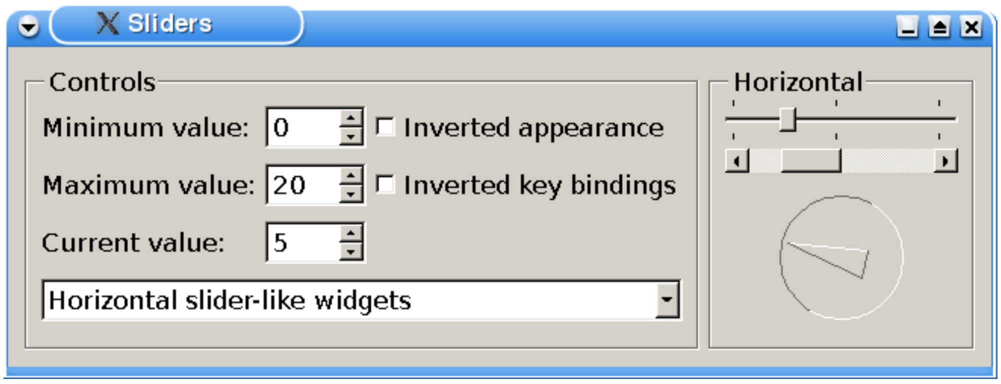
Group boxes are usually used to organize check boxes and radio buttons into exclusive groups.



The Group Boxes example consists of a single `Window` class that is used to show four group boxes: an exclusive radio button group, a non-exclusive checkbox group, an exclusive radio button group with an enabling checkbox, and a group box with normal push buttons.

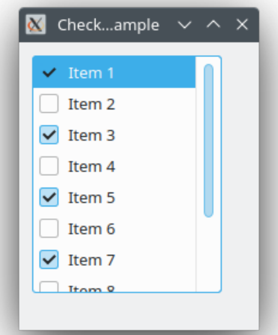
Exemple de Qt-sliderGroup

The example also demonstrates how signals and slots can be used to synchronize the behavior of two or more widgets.



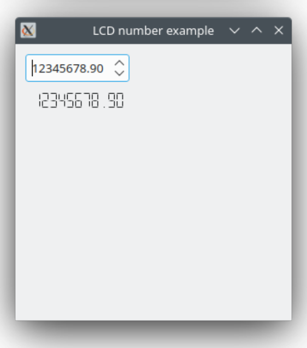
Minimum value: 0
Maximum value: 20
Current value: 5

QtChecked List Wiget est extrait de https://github.com/gammasoft71/Examples_Qt/blob/master/Qt.Widgets/Controls/CheckedListWidget/README.md



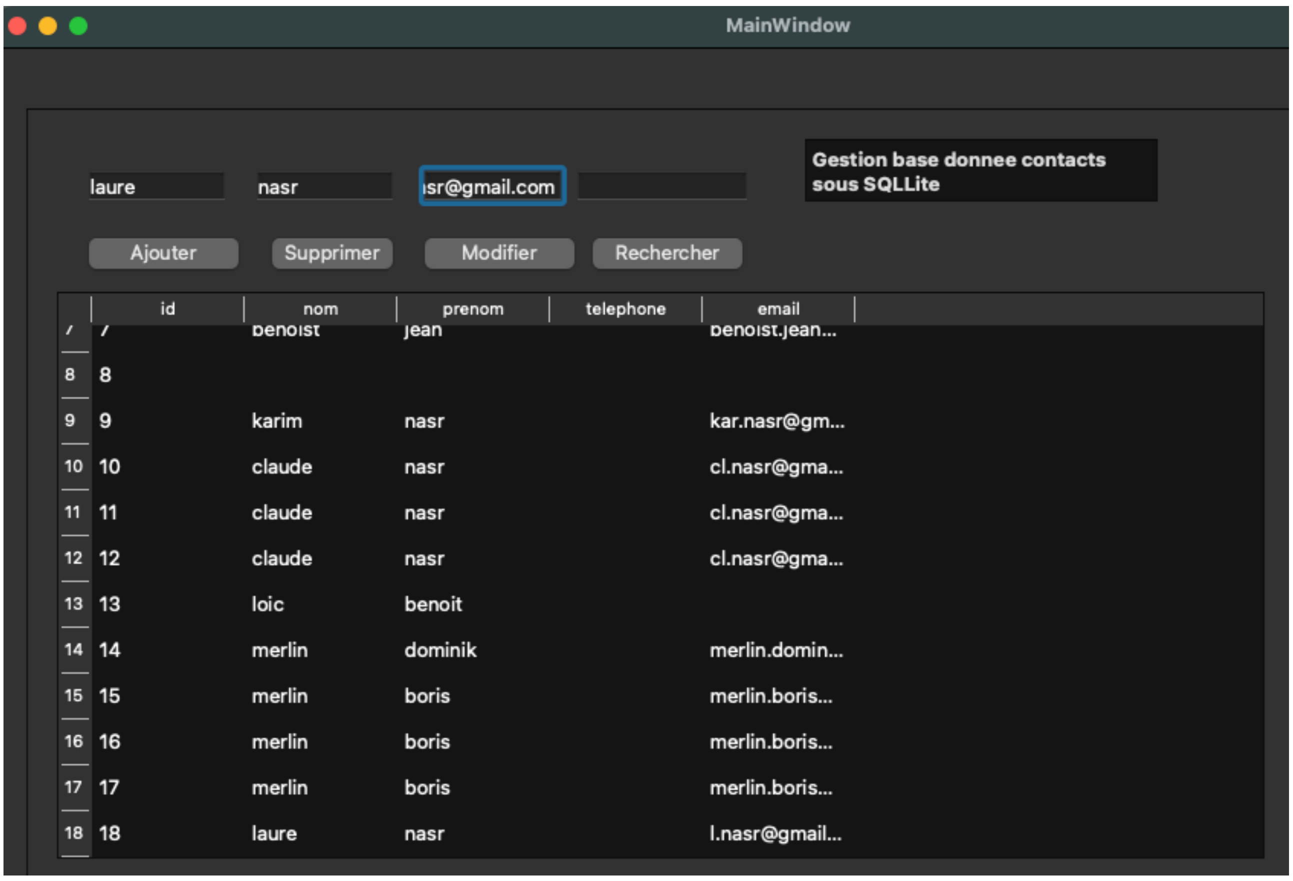
- [✓] Item 1
- [ ] Item 2
- [✓] Item 3
- [ ] Item 4
- [✓] Item 5
- [ ] Item 6
- [✓] Item 7
- [ ] Item 8

QtLCDnumber



12345678.90

12345678.90

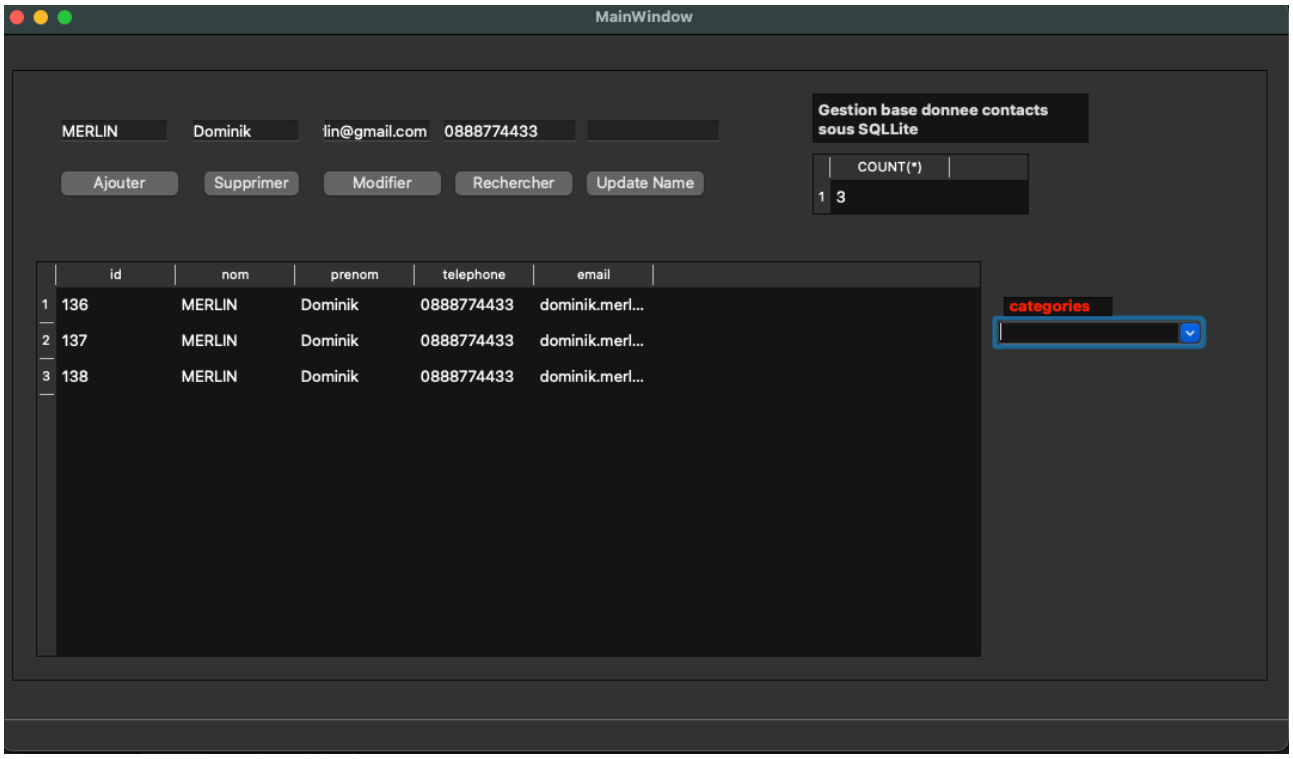


Gestion base donnee contacts sous SQLLite

laure | nasr | sr@gmail.com |

	id	nom	prenom	telephone	email
7	7	benoist	jean		benoist.jean...
8	8				
9	9	karim	nasr		kar.nasr@gm...
10	10	claude	nasr		cl.nasr@gma...
11	11	claude	nasr		cl.nasr@gma...
12	12	claude	nasr		cl.nasr@gma...
13	13	loic	benoit		
14	14	merlin	dominik		merlin.domin...
15	15	merlin	boris		merlin.boris...
16	16	merlin	boris		merlin.boris...
17	17	merlin	boris		merlin.boris...
18	18	laure	nasr		l.nasr@gmail...

Le projet GestionContact est une IHM pour afficher une base de donnee de type SQLite utilisation de widget SQLite, utilisation à l'aide de GEMINI.

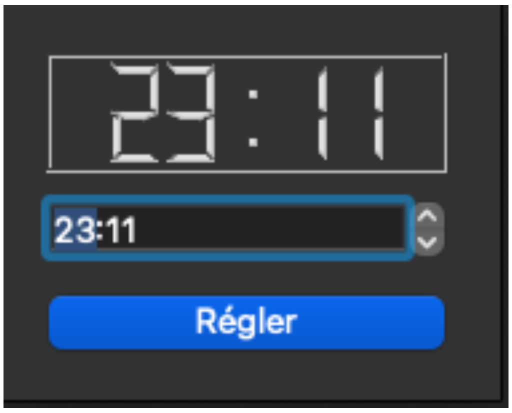


	COUNT(*)
1	3

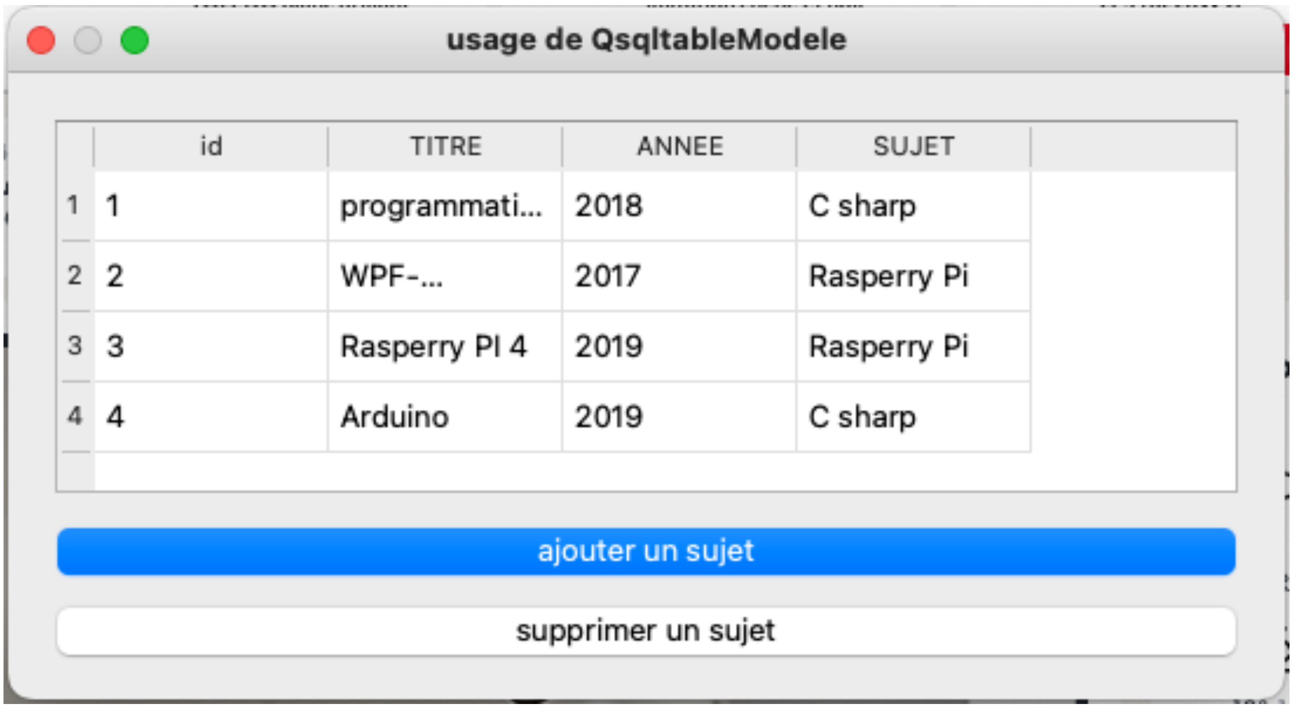
	id	nom	prenom	telephone	email
1	136	MERLIN	Dominik	0888774433	dominik.merl...
2	137	MERLIN	Dominik	0888774433	dominik.merl...
3	138	MERLIN	Dominik	0888774433	dominik.merl...

La deuxieme version integre la rubrique categorie intégrée à l aide d'une comboBox.

Exemple de Qt-horloge digitale (fichier PDF de TP joint)



Exemple QsqlTableModele , fait avec pychart , extrait cours Qt sur livre Framework PyQt (benoit prieur)



	id	TITRE	ANNEE	SUJET
1	1	programmati...	2018	C sharp
2	2	WPF-...	2017	Rasperry Pi
3	3	Rasperry PI 4	2019	Rasperry Pi
4	4	Arduino	2019	C sharp

Usage de la classe QsqlRelationaltablemodel (chapitre 8)

/Users/nasr/PycharmProjects/PySide/main.py-Qt-QsqlRelationalModel

```
import os
import PyQt6

from PyQt6 import QtSql
from PyQt6 import QtCore
from PyQt6 import QtGui

from PyQt6.QtSql import *
from PyQt6.QtCore import *
from PyQt6.QtGui import *
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import *
#from PyQt6 import QtWidget
```

```python
from PySide6.QtWidgets import QApplication,QLabel, QWidget,
QBoxLayout,QVBoxLayout,QTableView,QDialog, QPushButton

from PySide6.QtGui import QGuiApplication
from PySide6.QtCore import QCoreApplication,Qt
from PySide6.QtSql import
QSqlDatabase,QSqlRelationalTableModel,QSqlRelation,QSqlRelati
onalDelegate
from PySide6.QtSql import QSqlTableModel


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

        self.disposition = QVBoxLayout()
        self.clickbouton = QPushButton("Click",
clicked=self.creationDB)
        self.disposition.addWidget(self.clickbouton)
        self.setLayout(self.disposition)
        self.execute()

        #QtGui.QWindow.__init__(self,parent)
        self.resize(30, 30)
        #self.setFont(QtGui.QFont("Verdana"))
        self.setWindowTitle("Bases de données")
```

```
  #      self.clickbouton =
QPushButton("Click",clicked=self.creationDB)
  #     self.disposition.addWidget(self.clickbouton)
        self.setLayout(self.disposition)
  #      self.execute()

    def creationDB(self):


        self.db = QtSql.QSqlDatabase.addDatabase('QSQLITE')
        self.db.setDatabaseName('Baselivres.db')
        print ("Creation base ok")
        if not self.db.open():
            print ("la Db ne peut pas s'ouvrir.")
            return False

        query = QtSql.QSqlQuery()

        print(query.exec("SELECT COUNT(*) FROM PERSONNE"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM')
VALUES('Prieur','Benoit')")
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM')
VALUES('Mocq','François')")
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM')
VALUES('Lacaze','Sarah')")
```

```
            print(query.exec("SELECT COUNT(*) FROM 
PERSONNE"))
            while query.next():
                print(query.value(0))
```

```
        print(query.exec("SELECT COUNT(*) FROM SUJET"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO SUJET('SUJET') 
VALUES('Csharp')")
            query.exec("INSERT INTO SUJET('SUJET') 
VALUES('Rasperry pi')")
            query.exec("INSERT INTO SUJET('SUJET') 
VALUES('Scratch')")
```

```
            print(query.exec("SELECT COUNT(*) FROM SUJET"))
            while query.next():
                print(query.value(0))
```

```
        print(query.exec("SELECT COUNT(*) FROM LIVRE"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO 
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('programmation 
en C-preparation aux certifications MCSA-examen 
70-483',2018,1,1)")
```

```
            query.exec("INSERT INTO 
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-developper 
des applications structurees',2017,1,2)")
            query.exec("INSERT INTO 
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('Rasperry PI 
4',2019,2,3)")
            query.exec("INSERT INTO 
LIVRE('TITRE','ANNEE','sujet_id','id') 
VALUES('Arduino',2019,1,4)")
```

```
            print(query.exec("SELECT COUNT(*) FROM LIVRE"))
            while query.next():
                print(query.value(0))
```

```
        query = QtSql.QSqlQuery()
        query.exec("UPDATE SUJET SET SUJET = 'MICROSOFT C#' 
WHERE id =2")
```

```
        query.exec("SELECT COUNT(*) FROM SUJET")
        while query.next():
            print(query.value(0))
```

```
        query.exec("SELECT SUJET FROM SUJET WHERE id = 2")
        while query.next():
            print(query.value(0))
```

```
        self.db.commit()
        self.db.close()
```

```
    #Creation des requêtes SQL
        creationTableLivre = """create table LIVRE(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            TITRE TEXT NOT NULL,
            ANNEE int,
            sujet_id INTEGER REFERENCES SUJET(id)
            );"""

        creationTableSujet = """create table SUJET(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            SUJET TEXT NOT NULL
            );"""
        creationTablePersonne = """create table PERSONNE(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            NOM TEXT NOT NULL,
            PRENOM TEXT NOT NULL
        );"""

    #Execution des requetes SQL
        self.db.open()
        query = QtSql.QSqlQuery()
        #self.db.close()

        if query.exec(creationTableLivre):
            print("creation table LIVRE ok")

        if query.exec(creationTableSujet):
            print("creation table SUJET ok")

        if query.exec(creationTablePersonne):
            print("creation table Personne ok")

        self.db.close()



    def execute (self):
        self.resize(250, 300)
        self.move(50, 500)
        self.setWindowTitle("chapitre 8 - insertion de
donnees")
        self.show()

def joinDB():
        app = QtCore.QCoreApplication(sys.argv)

        db = QtSql.QSqlDatabase.addDatabase('QSQLITE')
        db.setDatabaseName('Baselivres.db')
        print("connexion base ok")

        if not db.open():
            print("la db ne peut s'ouvrir.")
            return False

        query = QtSql.QSqlQuery()
        #query.exec(
        #    "INSERT INTO 
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('programmation 
en C-preparation aux certifications MCSA-examen 
70-483',2018,1,1)")
        #query.exec(
        #    "INSERT INTO 
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-developper 
des applications structurees',2017,1,2)")
        #query.exec("INSERT INTO 
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('Rasperry PI 
4',2019,2,3)")
        #query.exec("INSERT INTO 
LIVRE('TITRE','ANNEE','sujet_id','id') 
VALUES('Arduino',2019,1,4)")

        #query.exec("""
        #    SELECT PERSONNE.PRENOM, PERSONNE.NOM
        #    FROM PERSONNE, LIVRE, SUJET
        #    WHERE PERSONNE.id = LIVRE.id
        #    AND SUJET.id = LIVRE.id
        #    AND SUJET.SUJET = 'Arduino'
        #    AND LIVRE.ANNEE = 2019
        #""")

        #affiche jointure
        #connexion
        #base ok
```

```
#Benoit Prieur
#François Mocq
#Sarah Lacaze

sujet = 'Scratch'
annee = 2019

requete="""
    SELECT PERSONNE.PRENOM, PERSONNE.NOM
    FROM PERSONNE, LIVRE, SUJET
    WHERE PERSONNE.id = LIVRE.id
    AND SUJET.id = LIVRE.id
        """

query.prepare(requete)
query.bindValue(":sujet",sujet)
query.bindValue(":annee",annee)

if query.exec():
    while query.next():
        print(query.value(0),query.value(1))
else:
    print("erreur dans l execution de la
requete:",query.lastError().text())

query.exec("""
    SELECT PERSONNE.PRENOM, PERSONNE.NOM
```

```
            FROM PERSONNE, LIVRE, SUJET
            WHERE PERSONNE.id = LIVRE.id
            AND SUJET.id = LIVRE.id
              """)

        while query.next():
            print(query.value(0),query.value(1))

def print_hi(name):
    # Use a breakpoint in the code line below to debug your
script.
    print(f'Hi, {name}')  # Press ⌘F8 to toggle the
breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    # affiche jointure
    # connexion
    # base ok
    # Benoit Prieur
    # François Mocq
    # Sarah Lacaze
    #joinDB()
    #Application.instance()
    app = QApplication(sys.argv)
    base = QtSql.QSqlDatabase.addDatabase('QSQLITE')

    base.setDatabaseName('Baselivres.db')


    #definition du modele

    #modele = QtCore.QAbstractItemModel

    modele = QSqlRelationalTableModel()

    modele.setTable('LIVRE')


modele.setEditStrategy(QSqlRelationalTableModel.EditStrategy.
OnFieldChange)


    modele.setRelation(3,QSqlRelation("SUJET","id","SUJET"))

    modele.setRelation(4, QSqlRelation("PERSONNE", "id",
"PRENOM"))


    #modification immediate

    #modele.select()

    modele.setHeaderData(3,Qt.Horizontal,"SUJET")

    modele.setHeaderData(3, Qt.Horizontal, "PRENOM")

    #modele.setHeaderData(3,
QtCore.Qt.Orientation.Horizontal, "SUJET")


#modele.setHeaderData(4,QtCore.Qt.Orientation.Horizontal,"PRE
NOM")

    modele.select()


    #creation de la vue et association au modele

    vue = QTableView()
    vue.setModel(modele)

    vue.setItemDelegate(QSqlRelationalDelegate(vue))

    #dialogue = QDialog()

    #disposition = QVBoxLayout()

    #disposition.addWidget(vue)

    #fenetre = QtWidgets.QWidget()

    #fenetre.QTableView().setModel(QtSql.QSqlTableModel())

    #vue.setItemDelegate(QSqlRelationalDelegate(vue))

    #creation de la boite de dialogue

    dialogue = QDialog()

    disposition = QVBoxLayout()

    disposition.addWidget(vue)

    #bouton d ajout

    bouton_ajout = QPushButton("ajouter un livre")

bouton_ajout.clicked.connect(lambda :modele.insertRows(modele
.rowCount(),1))

    disposition.addWidget(bouton_ajout)

    #bouton de suppression

    bouton_suppression = QPushButton("supprimer un livre")

    bouton_suppression.clicked.connect(lambda :
modele.removeRow(vue.currentIndex().row()))
```

```
    disposition.addWidget(bouton_suppression)

    #finalisation de l interface
    dialogue.setLayout(disposition)
    dialogue.setWindowTitle("usage de QsqRelationalModele")
    dialogue.show()


    #app = QtCore.QCoreApplication(sys.argv)

    #app = QtCore.QCoreApplication(sys.argv)
    #app = QtGui.QGuiApplication(sys.argv)
    #app = QtCore.QCoreApplication(sys.argv)

    #app= QtCore.QCoreApplication.arguments()


    # ...
    sys.exit(app.exec())



import sys
import os
import PyQt6

from PyQt6 import QtSql
from PyQt6 import QtCore
from PyQt6 import QtGui

from PyQt6.QtSql import *
from PyQt6.QtCore import *
from PyQt6.QtGui import *
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import *
#from PyQt6 import QtWidget


from PySide6.QtWidgets import QApplication,QLabel, QWidget,
QBoxLayout,QVBoxLayout,QTableView,QDialog, QPushButton

from PySide6.QtGui import QGuiApplication
from PySide6.QtCore import QCoreApplication,Qt
from PySide6.QtSql import
QSqlDatabase,QSqlRelationalTableModel,QSqlRelation,QSqlRelati
onalDelegate
from PySide6.QtSql import QSqlTableModel


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

        self.disposition = QVBoxLayout()
        self.clickbouton = QPushButton("Click",
clicked=self.creationDB)

        self.disposition.addWidget(self.clickbouton)

        self.setLayout(self.disposition)

        self.execute()
```

```
        #QtGui.QWindow.__init__(self,parent)

        self.resize(30, 30)

        #self.setFont(QtGui.QFont("Verdana"))

        self.setWindowTitle("Bases de données")
```

```
  #      self.clickbouton =
QPushButton("Click",clicked=self.creationDB)

  #     self.disposition.addWidget(self.clickbouton)

        self.setLayout(self.disposition)

  #      self.execute()
```

```
    def creationDB(self):
```

```
        self.db = QtSql.QSqlDatabase.addDatabase('QSQLITE')

        self.db.setDatabaseName('Baselivres.db')

        print ("Creation base ok")

        if not self.db.open():

            print ("la Db ne peut pas s'ouvrir.")

            return False
        query = QtSql.QSqlQuery()
```

```
        print(query.exec("SELECT COUNT(*) FROM PERSONNE"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM')
VALUES('Prieur','Benoit')")
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM')
VALUES('Mocq','François')")
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM')
VALUES('Lacaze','Sarah')")
```

```
            print(query.exec("SELECT COUNT(*) FROM
PERSONNE"))
            while query.next():
                print(query.value(0))
```

```
        print(query.exec("SELECT COUNT(*) FROM SUJET"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO SUJET('SUJET')
VALUES('Csharp')")
            query.exec("INSERT INTO SUJET('SUJET')
VALUES('Rasperry pi')")
            query.exec("INSERT INTO SUJET('SUJET')
VALUES('Scratch')")
```

```
            print(query.exec("SELECT COUNT(*) FROM SUJET"))
            while query.next():
                print(query.value(0))
```

```
        print(query.exec("SELECT COUNT(*) FROM LIVRE"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('programmation
en C-preparation aux certifications MCSA-examen
70-483',2018,1,1)")
            query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-developper
des applications structurees',2017,1,2)")
            query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('Rasperry PI
4',2019,2,3)")
            query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id')
VALUES('Arduino',2019,1,4)")
```

```
            print(query.exec("SELECT COUNT(*) FROM LIVRE"))
            while query.next():
                print(query.value(0))
```

```
        query = QtSql.QSqlQuery()
        query.exec("UPDATE SUJET SET SUJET = 'MICROSOFT C#'
WHERE id =2")
```

```
        query.exec("SELECT COUNT(*) FROM SUJET")
        while query.next():
            print(query.value(0))
        query.exec("SELECT SUJET FROM SUJET WHERE id = 2")
        while query.next():
            print(query.value(0))
```

```
        self.db.commit()
        self.db.close()
    #Creation des requêtes SQL
        creationTableLivre = """create table LIVRE(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            TITRE TEXT NOT NULL,
            ANNEE int,
            sujet_id INTEGER REFERENCES SUJET(id)
            );"""
```

```
        creationTableSujet = """create table SUJET(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            SUJET TEXT NOT NULL
            );"""
        creationTablePersonne = """create table PERSONNE(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            NOM TEXT NOT NULL,
            PRENOM TEXT NOT NULL
        );"""
```

```
    #Execution des requetes SQL
```

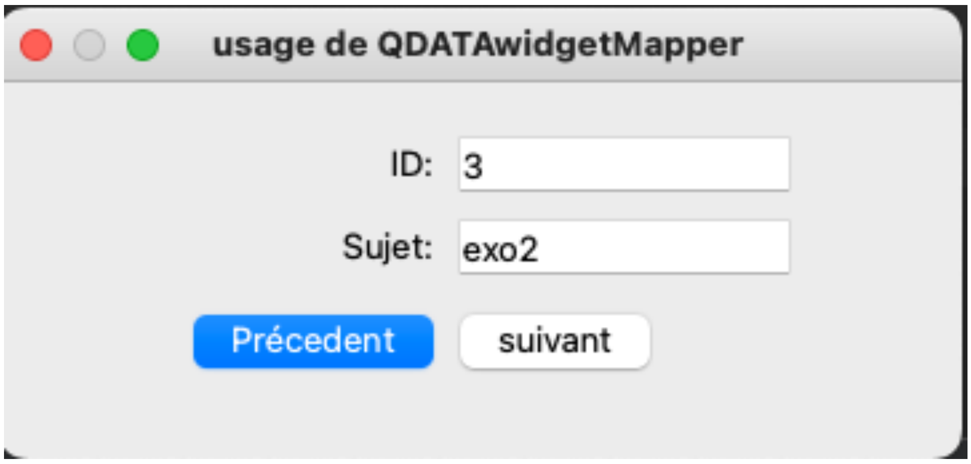
```
        self.db.open()
        query = QtSql.QSqlQuery()
        #self.db.close()

        if query.exec(creationTableLivre):
            print("creation table LIVRE ok")

        if query.exec(creationTableSujet):
            print("creation table SUJET ok")

        if query.exec(creationTablePersonne):
            print("creation table Personne ok")

        self.db.close()



    def execute (self):
        self.resize(250, 300)
        self.move(50, 500)
        self.setWindowTitle("chapitre 8 - insertion de 
donnees")
        self.show()

def joinDB():
        app = QtCore.QCoreApplication(sys.argv)

        db = QtSql.QSqlDatabase.addDatabase('QSQLITE')
        db.setDatabaseName('Baselivres.db')

        print("connexion base ok")

        if not db.open():

            print("la db ne peut s'ouvrir.")

            return False

        query = QtSql.QSqlQuery()

        #query.exec(

        #    "INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('programmation
en C-preparation aux certifications MCSA-examen
70-483',2018,1,1)")

        #query.exec(

        #    "INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-developper
des applications structurees',2017,1,2)")

        #query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('Rasperry PI
4',2019,2,3)")

        #query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id')
VALUES('Arduino',2019,1,4)")

        #query.exec("""

        #    SELECT PERSONNE.PRENOM, PERSONNE.NOM

        #    FROM PERSONNE, LIVRE, SUJET

        #    WHERE PERSONNE.id = LIVRE.id
        #    AND SUJET.id = LIVRE.id
        #    AND SUJET.SUJET = 'Arduino'
        #    AND LIVRE.ANNEE = 2019
        #""")

        #affiche jointure
        #connexion
        #base ok
        #Benoit Prieur
        #François Mocq
        #Sarah Lacaze

        sujet = 'Scratch'
        annee = 2019

        requete="""
            SELECT PERSONNE.PRENOM, PERSONNE.NOM
            FROM PERSONNE, LIVRE, SUJET
            WHERE PERSONNE.id = LIVRE.id
            AND SUJET.id = LIVRE.id
                """

        query.prepare(requete)
        query.bindValue(":sujet",sujet)
        query.bindValue(":annee",annee)

        if query.exec():
```

```python
            while query.next():
                print(query.value(0),query.value(1))
        else:
            print("erreur dans l execution de la 
requete:",query.lastError().text())

        query.exec("""
            SELECT PERSONNE.PRENOM, PERSONNE.NOM
            FROM PERSONNE, LIVRE, SUJET
            WHERE PERSONNE.id = LIVRE.id
            AND SUJET.id = LIVRE.id
              """)

        while query.next():
            print(query.value(0),query.value(1))

def print_hi(name):
    # Use a breakpoint in the code line below to debug your 
script.
    print(f'Hi, {name}')  # Press ⌘F8 to toggle the 
breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    # affiche jointure
    # connexion
    # base ok
    # Benoit Prieur
    # François Mocq
    # Sarah Lacaze
    #joinDB()
    #Application.instance()
    app = QApplication(sys.argv)

    base = QtSql.QSqlDatabase.addDatabase('QSQLITE')
    base.setDatabaseName('Baselivres.db')

    #definition du modele
    #modele = QtCore.QAbstractItemModel
    modele = QSqlRelationalTableModel()
    modele.setTable('LIVRE')

modele.setEditStrategy(QSqlRelationalTableModel.EditStrategy.
OnFieldChange)

    modele.setRelation(3,QSqlRelation("SUJET","id","SUJET"))
    modele.setRelation(4, QSqlRelation("PERSONNE", "id",
"PRENOM"))

    #modification immediate
    #modele.select()
    modele.setHeaderData(3,Qt.Horizontal,"SUJET")
    modele.setHeaderData(3, Qt.Horizontal, "PRENOM")
    #modele.setHeaderData(3,
QtCore.Qt.Orientation.Horizontal, "SUJET")
```

```
#modele.setHeaderData(4,QtCore.Qt.Orientation.Horizontal,"PRE
NOM")
    modele.select()

    #creation de la vue et association au modele
    vue = QTableView()
    vue.setModel(modele)
    vue.setItemDelegate(QSqlRelationalDelegate(vue))

    #dialogue = QDialog()
    #disposition = QVBoxLayout()
    #disposition.addWidget(vue)

    #fenetre = QtWidgets.QWidget()

    #fenetre.QTableView().setModel(QtSql.QSqlTableModel())
    #vue.setItemDelegate(QSqlRelationalDelegate(vue))

    #creation de la boite de dialogue
    dialogue = QDialog()
    disposition = QVBoxLayout()
    disposition.addWidget(vue)

    #bouton d ajout
    bouton_ajout = QPushButton("ajouter un livre")
```

```
bouton_ajout.clicked.connect(lambda :modele.insertRows(modele
.rowCount(),1))
    disposition.addWidget(bouton_ajout)

    #bouton de suppression
    bouton_suppression = QPushButton("supprimer un livre")
    bouton_suppression.clicked.connect(lambda :
modele.removeRow(vue.currentIndex().row()))
    disposition.addWidget(bouton_suppression)

    #finalisation de l interface
    dialogue.setLayout(disposition)
    dialogue.setWindowTitle("usage de QsqRelationalModele")
    dialogue.show()


    #app = QtCore.QCoreApplication(sys.argv)

    #app = QtCore.QCoreApplication(sys.argv)
    #app = QtGui.QGuiApplication(sys.argv)
    #app = QtCore.QCoreApplication(sys.argv)

    #app= QtCore.QCoreApplication.arguments()


    # ...
    sys.exit(app.exec())
```

Exemple QDATAWidgetMapper , fait avec pychart , extrait cours Qt sur livre Framework PyQt (benoit prieur)



```
import sys
import os
import PyQt6

from PyQt6 import QtSql
from PyQt6 import QtCore
from PyQt6 import QtGui

from PyQt6.QtSql import *
from PyQt6.QtCore import *
from PyQt6.QtGui import *
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import *
#from PyQt6 import QtWidget

from PySide6.QtWidgets import QApplication,QLabel, QWidget,
QBoxLayout,QVBoxLayout,QTableView,QDialog,QLineEdit,
QPushButton,QFormLayout,QDataWidgetMapper

from PySide6.QtGui import QGuiApplication
from PySide6.QtCore import QCoreApplication,Qt
```

```python
from PySide6.QtSql import
QSqlDatabase,QSqlRelationalTableModel,QSqlRelation,QSqlRelationalDelegate

from PySide6.QtSql import QSqlTableModel



class FenetreSimple(QWidget):

    def __init__(self):

        super().__init__()


        self.disposition = QVBoxLayout()

        self.clickbouton = QPushButton("Click", clicked=self.creationDB)

        self.disposition.addWidget(self.clickbouton)

        self.setLayout(self.disposition)

        self.execute()


        #QtGui.QWindow.__init__(self,parent)

        self.resize(30, 30)

        #self.setFont(QtGui.QFont("Verdana"))

        self.setWindowTitle("Bases de données")


  #      self.clickbouton = QPushButton("Click",clicked=self.creationDB)êêEE

  #     self.disposition.addWidget(self.clickbouton)

        self.setLayout(self.disposition)

  #      self.execute()


    def creationDB(self):



        self.db = QtSql.QSqlDatabase.addDatabase('QSQLITE')

        self.db.setDatabaseName('Baselivres.db')

        print ("Creation base ok")

        if not self.db.open():

            print ("la Db ne peut pas s'ouvrir.")

            return False
        query = QtSql.QSqlQuery()

        print(query.exec("SELECT COUNT(*) FROM PERSONNE"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM') VALUES('Prieur','Benoit')")
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM') VALUES('Mocq','François')")
            query.exec("INSERT INTO PERSONNE('NOM','PRENOM') VALUES('Lacaze','Sarah')")

            print(query.exec("SELECT COUNT(*) FROM PERSONNE"))
            while query.next():
                print(query.value(0))

        print(query.exec("SELECT COUNT(*) FROM SUJET"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO SUJET('SUJET') VALUES('Csharp')")
            query.exec("INSERT INTO SUJET('SUJET') VALUES('Rasperry pi')")
            query.exec("INSERT INTO SUJET('SUJET') VALUES('Scratch')")

            print(query.exec("SELECT COUNT(*) FROM SUJET"))
            while query.next():
                print(query.value(0))

        print(query.exec("SELECT COUNT(*) FROM LIVRE"))
        while query.next():
            print(query.value(0))
            query.exec("INSERT INTO LIVRE('TITRE','ANNEE','sujet_id','id') 
VALUES('programmation en C-preparation aux certifications MCSA-examen 70-483',2018,1,1)")
            query.exec("INSERT INTO LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-
developper des applications structurees',2017,1,2)")
            query.exec("INSERT INTO LIVRE('TITRE','ANNEE','sujet_id','id') 
VALUES('Rasperry PI 4',2019,2,3)")
            query.exec("INSERT INTO LIVRE('TITRE','ANNEE','sujet_id','id') 
VALUES('Arduino',2019,1,4)")
```

```
            print(query.exec("SELECT COUNT(*) FROM LIVRE"))
            while query.next():
                print(query.value(0))
```

```
        query = QtSql.QSqlQuery()
        query.exec("UPDATE SUJET SET SUJET = 'MICROSOFT C#' WHERE id =2")
```

```
        query.exec("SELECT COUNT(*) FROM SUJET")
        while query.next():
            print(query.value(0))
```

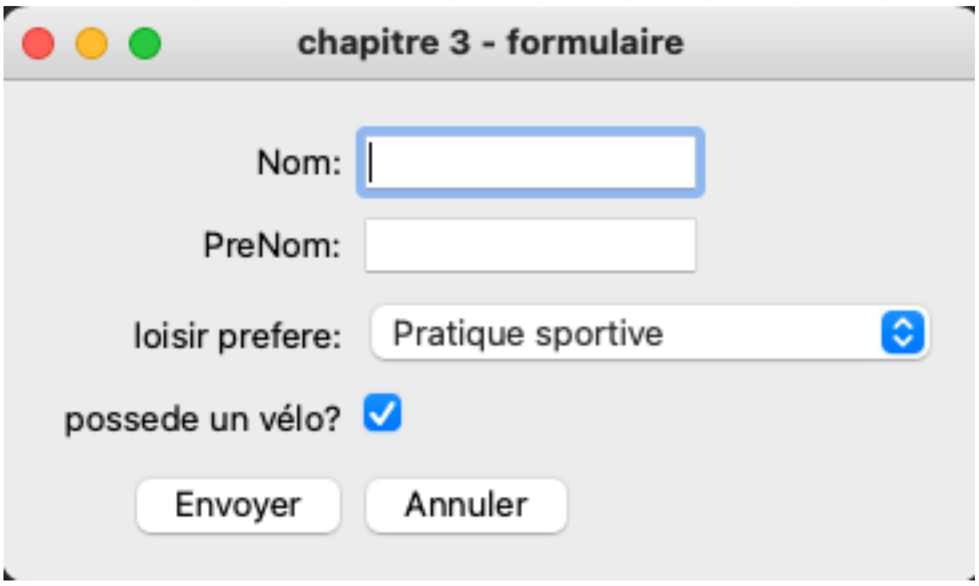
```
        query.exec("SELECT SUJET FROM SUJET WHERE id = 2")
        while query.next():
            print(query.value(0))
```

```
        self.db.commit()
        self.db.close()
    #Creation des requêtes SQL
        creationTableLivre = """create table LIVRE(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            TITRE TEXT NOT NULL,
            ANNEE int,
            sujet_id INTEGER REFERENCES SUJET(id)
            );"""
```

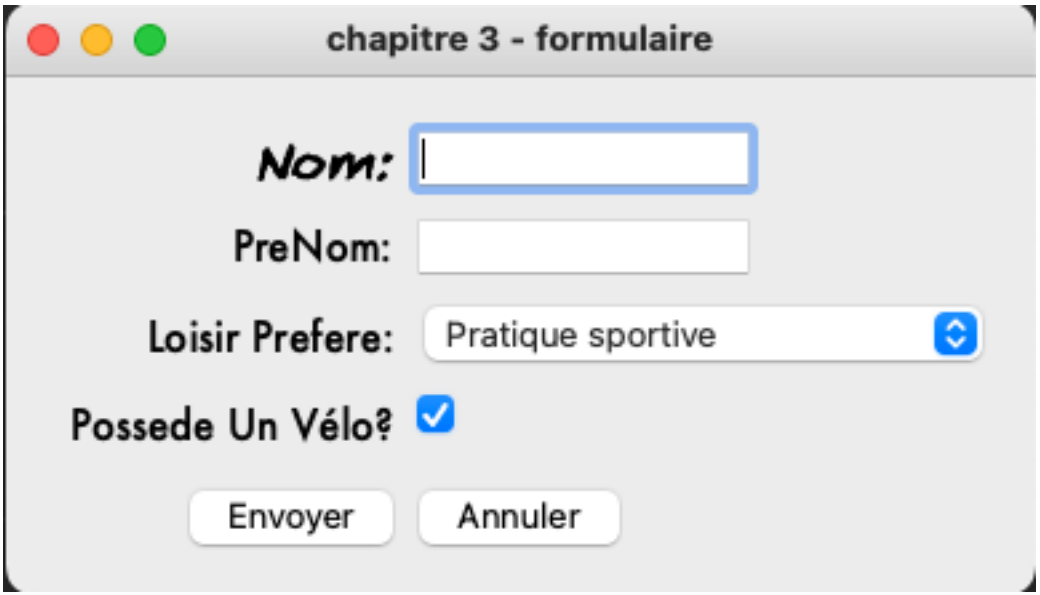
```
        creationTableSujet = """create table SUJET(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            SUJET TEXT NOT NULL
            );"""
        creationTablePersonne = """create table PERSONNE(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            NOM TEXT NOT NULL,
            PRENOM TEXT NOT NULL
            );"""

    #Execution des requetes SQL
        self.db.open()
        query = QtSql.QSqlQuery()
        #self.db.close()

        if query.exec(creationTableLivre):
            print("creation table LIVRE ok")

        if query.exec(creationTableSujet):
            print("creation table SUJET ok")

        if query.exec(creationTablePersonne):
            print("creation table Personne ok")

        self.db.close()


    def execute (self):
        self.resize(250, 300)
        self.move(50, 500)
        self.setWindowTitle("chapitre 8 - insertion de donnees")
        self.show()

def prec():
    print("Bouton précédent")
    mapping.toPrevious()
    id.repaint()
    sujet.repaint()

def suiv():
    print("Bouton suivant")
    mapping.toNext()
```

```python
    id.repaint()

    sujet.repaint()

def joinDB():

        app = QtCore.QCoreApplication(sys.argv)

        db = QtSql.QSqlDatabase.addDatabase('QSQLITE')

        db.setDatabaseName('Baselivres.db')

        print("connexion base ok")

        if not db.open():

            print("la db ne peut s'ouvrir.")

            return False

        query = QtSql.QSqlQuery()



        sujet = 'Scratch'

        annee = 2019

        requete="""

            SELECT PERSONNE.PRENOM, PERSONNE.NOM

            FROM PERSONNE, LIVRE, SUJET

            WHERE PERSONNE.id = LIVRE.id

            AND SUJET.id = LIVRE.id

                """

        query.prepare(requete)

        query.bindValue(":sujet",sujet)

        query.bindValue(":annee",annee)

        if query.exec():

            while query.next():
                print(query.value(0),query.value(1))
        else:
            print("erreur dans l execution de la requete:",query.lastError().text())

        query.exec("""
            SELECT PERSONNE.PRENOM, PERSONNE.NOM
            FROM PERSONNE, LIVRE, SUJET
            WHERE PERSONNE.id = LIVRE.id
            AND SUJET.id = LIVRE.id
              """)

        while query.next():
            print(query.value(0),query.value(1))

def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press ⌘F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    # affiche jointure
    # connexion
    # base ok
    # Benoit Prieur
    # François Mocq
    # Sarah Lacaze
    #joinDB()
    #Application.instance()
    app = QApplication(sys.argv)

    # creation de la boite de dialogue
    dialogue = QDialog()
    disposition = QFormLayout()
    #champ d'édition
    id = QLineEdit()
    id.setReadOnly(True)
    sujet = QLineEdit()

    disposition.addRow("ID:",id)
    disposition.addRow("Sujet:",sujet)

    precedentBouton = QPushButton("Précedent")
    suivantBouton = QPushButton("suivant")
    disposition.addRow(precedentBouton,suivantBouton)

    dialogue.setLayout(disposition)
    dialogue.setWindowTitle("usage de QDATAwidgetMapper")

    base = QtSql.QSqlDatabase.addDatabase('QSQLITE')
    base.setDatabaseName('Baselivres.db')

    #definition du modele
    #modele = QtCore.QAbstractItemModel
    modele = QSqlTableModel()
    modele.setTable('SUJET')
    modele.setEditStrategy(QSqlTableModel.EditStrategy.OnFieldChange)

    #modification immediate
    modele.select()

    mapping = QDataWidgetMapper()
    mapping.setModel(modele)
    mapping.addMapping(id, 0)
    mapping.addMapping(sujet, 1)
    mapping.toFirst()
    precedentBouton.clicked.connect(prec)

    suivantBouton.clicked.connect(suiv)


    modele.select()


    dialogue.show()


    # ...

    sys.exit(app.exec())
```

Exemple QTCreationFormulaire , fait avec pychart , extrait cours Qt sur livre Framework PyQt (benoit prieur)



```
  Identique au cas précédent avec QFONT et nouvelle police
```


chapitre 3 - formulaire
Nom:
PreNom:
Loisir Prefere:
Pratique sportive
Possede Un Vélo?
Envoyer
Annuler