**TECHNICAL REPORT**

**PEMROGRAMAN DESKTOP**

**MODUL 6**



**Disusun Oleh :**

TGL. PRAKTIKUM : Jum’at, 18 Desember 2020

NAMA : Achmad Farid Alfa Waid

NRP : 190411100073

KELOMPOK : 2

DOSEN : Moh. Kautsar Sophan, S.Kom., M.MT.

TELAH DISETUJUI TANGGAL :

...........................................

ASISTEN PRAKTIKUM

Nadia Asri

(180411100063)

**LABORATORIUM MULTIMEDIA COMPUTING**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**BAB I**

**TUJUAN DAN DASAR TEORI**

1. **TUJUAN**

Pada praktikum kali ini bertujuan untuk memahami tentang cara menghubungkan ke database Mysql online, menampilkan data, mencari data, menambahkan data, merubah data, dan menghapus data pada table yang ada di database.

1. **DASAR TEORI**

Database berfungsi untuk menyimpan dan mengelompokkan suatu data berdasarkan identifikasi data yang sudah dibuat, meminimalisir terjadinya duplikasi data pada saat penyimpanan data, memudahkan pengguna dalam proses (akses, edit, tambah, searching, dan delete)

**BAB II**

**PEMBAHASAN**

1. **SOAL**
2. Buat koneksi database dengan rincian informasi berikut:
3. Host: kprikaryasehat.site
4. Username: kprikary\_kuliah
5. Password: unijoyo2020
6. Databasename: kprikary\_resto
7. Fokus di tabel : menu1
8. Buat form untuk:
9. Menampilkan isi menu1, dan mencari data
10. Menambahkan data di tabel tesebut
11. Merubah data
12. Menghapus data
13. **JAWABAN**
14. **modul 6.py**

import sys

from PyQt5 import \*

from PyQt5.QtWidgets import \*

from PyQt5.Qt import \*

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5 import QtSql

import sqlite3

import MySQLdb as mdb

import pymysql.cursors

class Mysql(QWidget):

#Membuat fungsi init untuk inisialisasi class Mahasiswa

def \_\_init\_\_(self):

#untuk mengembalikan semua atribut dan method yang ada

super().\_\_init\_\_()

#membuka database

self.OpenDatabase()

#memanggil fungsi Layout yang sudah dibuat agar ditampilkan hasilnya

self.Layout()

def OpenDatabase(self):

try:

self.db = mdb.connect('kprikaryasehat.site', 'kprikary\_kuliah', 'unijoyo2020', 'kprikary\_resto')

QMessageBox.about(self, 'Connection', 'Database Connected Successfully')

except mdb.Error as e:

QMessageBox.about(self, 'Connection', 'Failed To Connect Database')

def Layout(self):

#Membuat Grid Layout

grid = QGridLayout()

add\_data = QLabel("Tambah Data:")

grid.addWidget(add\_data,0,0)

menu = QLabel("Nama Menu:")

grid.addWidget(menu,2,0)

self.Menu = QLineEdit(self)

grid.addWidget(self.Menu,2,1,1,2)

self.Menu.setStyleSheet("background-color: #f6f6f6;")

keterangan = QLabel("Keterangan:")

grid.addWidget(keterangan,3,0)

self.Keterangan = QLineEdit(self)

grid.addWidget(self.Keterangan,3,1,1,2)

self.Keterangan.setStyleSheet("background-color: #f6f6f6;")

harga = QLabel("Harga:")

grid.addWidget(harga,4,0)

self.Harga = QLineEdit(self)

grid.addWidget(self.Harga,4,1,1,2)

self.Harga.setStyleSheet("background-color: #f6f6f6;")

satuan = QLabel("Satuan:")

grid.addWidget(satuan,5,0)

self.Satuan = QLineEdit(self)

grid.addWidget(self.Satuan,5,1,1,2)

self.Satuan.setStyleSheet("background-color: #f6f6f6;")

add\_button = QPushButton("Tambah Data")

grid.addWidget(add\_button,6,0,1,0)

add\_button.setStyleSheet("background-color: #ffc7c7;")

edit\_button = QPushButton("Edit Data")

grid.addWidget(edit\_button,7,0,1,1)

edit\_button.setStyleSheet("background-color: #ffc7c7;")

update\_button = QPushButton("Update Data")

grid.addWidget(update\_button,7,1,1,2)

update\_button.setStyleSheet("background-color: #ffc7c7;")

search = QLabel("Cari Data:")

grid.addWidget(search,8,0)

self.Cari = QLineEdit(self)

grid.addWidget(self.Cari,8,1)

self.Cari.setStyleSheet("background-color: #f6f6f6;")

search\_button = QPushButton("Cari")

grid.addWidget(search\_button,8,2)

search\_button.setStyleSheet("background-color: #ffc7c7;")

delete\_button = QPushButton("Delete Data")

grid.addWidget(delete\_button,9,0,1,0)

delete\_button.setStyleSheet("background-color: #ffc7c7;")

#Membuat widget table widget yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid

self.tablewidget = QTableWidget(self)

self.tablewidget.setObjectName("Data")

self.tablewidget.setStyleSheet("background-color: #f6f6f6;")

grid.addWidget(self.tampilData(),11,0,5,0)

#Ketika button di klik akan memanggi fungsi masing - masing

add\_button.clicked.connect(self.tambahData)

search\_button.clicked.connect(self.filterData)

edit\_button.clicked.connect(self.editData)

update\_button.clicked.connect(self.updateData)

delete\_button.clicked.connect(self.deleteData)

#Layout grid di jadikan layout utama

self.setLayout(grid)

#Fungsi menampilkan Data menu1 ke dalam tabelwidget

def tampilData(self):

# Membuat Cursor

cur = self.db.cursor()

# Mengeksekusi perintah sql

cur.execute("SELECT \* FROM menu1")

# Mengambil Semua Data

data = cur.fetchall()

# Menjadikan Data bentuk list

record = list(data)

# Membuat Baris Yang Akan Ditampilkan DI table

self.tablewidget.setRowCount(len(record)+1)

# Membuat Kolom yang akan ditampilkan di table

self.tablewidget.setColumnCount(8)

# Membuat Header

self.tablewidget.setItem(0, 0, QTableWidgetItem("ID Menu"))

self.tablewidget.setItem(0, 1, QTableWidgetItem("ID Menu Kat"))

self.tablewidget.setItem(0, 2, QTableWidgetItem("ID Resto"))

self.tablewidget.setItem(0, 3, QTableWidgetItem("Nama Menu"))

self.tablewidget.setItem(0, 4, QTableWidgetItem("Keterangan"))

self.tablewidget.setItem(0, 5, QTableWidgetItem("Gambar"))

self.tablewidget.setItem(0, 6, QTableWidgetItem("Harga"))

self.tablewidget.setItem(0, 7, QTableWidgetItem("Satuan"))

# Menampilkan Data Yang Diambil Dari SQL

for i in range(len(record)):

baris = i + 1

self.tablewidget.setItem(baris, 0, QTableWidgetItem(str(record[i][0])))

self.tablewidget.setItem(baris, 1, QTableWidgetItem(str(record[i][1])))

self.tablewidget.setItem(baris, 2, QTableWidgetItem(str(record[i][2])))

self.tablewidget.setItem(baris, 3, QTableWidgetItem(str(record[i][3])))

self.tablewidget.setItem(baris, 4, QTableWidgetItem(str(record[i][4])))

self.tablewidget.setItem(baris, 5, QTableWidgetItem(str(record[i][5])))

self.tablewidget.setItem(baris, 6, QTableWidgetItem(str(record[i][6])))

self.tablewidget.setItem(baris, 7, QTableWidgetItem(str(record[i][7])))

# Membuat Agar Table Strech

self.tablewidget.horizontalHeader().setStretchLastSection(True)

self.tablewidget.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

cur.close()

return self.tablewidget

#Fungsi tambah Data ke dalam tabel menu1

def tambahData(self):

# Mengambil Text inputan

menu = str(self.Menu.text())

keterangan = str(self.Keterangan.text())

harga = int(self.Harga.text())

satuan = str(self.Satuan.text())

# Mendefinisikan Cursor

cur = self.db.cursor()

id = self.db.cursor()

# Menjalankan Perintah SQL

id.execute("SELECT idmenu FROM menu1 ORDER BY idmenu DESC LIMIT 1")

# Mengambil Satu Data

idMenu = id.fetchone()

# Membuat Id Auto Increment

idMenu = idMenu[0] + 1

# Menuliskan Perintah SQL

sql = "INSERT INTO menu1(idmenu,idmenukat,idresto,namamenu,keterangan,filegambar,harga,satuan) VALUES ('%d','%d','%d','%s','%s','%s','%d','%s')" % (idMenu, 1, 5,menu,keterangan,'https://img-global.cpcdn.com/recipes/84fae0149dbe9168/751x532cq70/telor-balado-foto-resep-utama.jpg',harga,satuan)

# MenjalanKan Perintah Try

try:

# Eksekusi Perintah SQL

cur.execute(sql)

# Mengcommit agar perubahan Tersimpan

self.db.commit()

# Membuat Notifikasi Berhasil Tambah Data

QMessageBox.about(self, 'Berhasil', 'Berhasil Menambah Data')

# Mengubah Inputan Agar Menjadi Kosong

self.Menu.setText("")

self.Keterangan.setText("")

self.Harga.setText("")

self.Satuan.setText("")

# Menampilkan Data

self.tampilData()

id.close()

cur.close()

# Jika Terjadi Error

except:

# Merollback data

self.db.rollback()

print("Gagal")

#Fungsi edit Data tabel menu1

def editData(self):

# Mengambil Data Yang DIpilih

index = self.tablewidget.selectedIndexes()[0]

id = self.tablewidget.model().data(index)

# Membuat Cursor

cur = self.db.cursor()

# Perintah Mencari Data Bedasarkan ID

sql = "SELECT \* FROM menu1 WHERE idmenu = '%s' " % (id)

# Eksekusi Perintah SQL

cur.execute(sql)

# Mengambil Satu Data

data = cur.fetchone()

# Menyimpan Data Id Di variabel idEdit

self.idEdit = data[0]

# Menampilkan Data Ke Textbox

self.Menu.setText(data[3])

self.Keterangan.setText(data[4])

self.Harga.setText(str(data[6]))

self.Satuan.setText(data[7])

# Fungsi Update Data menu1

def updateData(self):

# Mengambil Data Dari Inputan

menu = str(self.Menu.text())

keterangan = str(self.Keterangan.text())

harga = int(self.Harga.text())

satuan = str(self.Satuan.text())

# Membuat Cursor

cur = self.db.cursor()

# Perintah SQL Update

sql = "UPDATE menu1 SET namamenu = '%s', keterangan = '%s', harga = '%d', satuan = '%s' WHERE idmenu = '%d'" % (menu,keterangan,harga,satuan,self.idEdit)

try:

# Eksekusi Perintah SQL

cur.execute(sql)

# Commit DB Agar Terjadi Perubahan DI database

self.db.commit()

# Membuat Message Box

QMessageBox.about(self, 'Berhasil', 'Berhasil Update Data')

self.Menu.setText("")

self.Keterangan.setText("")

self.Harga.setText("")

self.Satuan.setText("")

self.tampilData()

print("Berhasil")

except:

self.db.rollback()

print("Gagal")

# Fungsi Hapus Data pada table menu1

def deleteData(self):

# Mengambil Data Yang Di pilih

index = self.tablewidget.selectedIndexes()[0]

id = self.tablewidget.model().data(index)

# Membuat Cursor

cur = self.db.cursor()

# Perintah SQL Untuk Menghapus Data

sql = "DELETE FROM menu1 WHERE idmenu = '%s' " % (id)

try:

# Eksekusi Perintah SQL

cur.execute(sql)

# Commit Ke Database Agar Perubahan Tersimpan

self.db.commit()

# Menampilkan Data

self.tampilData()

print("Berhasil")

except:

# Merollback data

self.db.rollback()

print("Gagal")

#Fungsi serching Data yang ada di dalam tabel menu1

def filterData(self):

# Membuat Cursor

cur = self.db.cursor()

filter\_search = str(self.Cari.text())

# Mengeksekusi perintah sql

cur.execute("SELECT \* FROM menu1 WHERE namamenu LIKE '%"+str(filter\_search)+"%' OR keterangan LIKE '%"+str(filter\_search)+"%' OR harga LIKE '%"+str(filter\_search)+"%' OR satuan LIKE '%"+str(filter\_search)+"%'")

# Mengambil Semua Data

data = cur.fetchall()

# Menjadikan Data bentuk list

record = list(data)

# Membuat Baris Yang Akan Ditampilkan DI table

self.tablewidget.setRowCount(len(record)+1)

# Membuat Kolom yang akan ditampilkan di table

self.tablewidget.setColumnCount(8)

# Membuat Header

self.tablewidget.setItem(0, 0, QTableWidgetItem("ID Menu"))

self.tablewidget.setItem(0, 1, QTableWidgetItem("ID Menu Kat"))

self.tablewidget.setItem(0, 2, QTableWidgetItem("ID Resto"))

self.tablewidget.setItem(0, 3, QTableWidgetItem("Nama Menu"))

self.tablewidget.setItem(0, 4, QTableWidgetItem("Keterangan"))

self.tablewidget.setItem(0, 5, QTableWidgetItem("Gambar"))

self.tablewidget.setItem(0, 6, QTableWidgetItem("Harga"))

self.tablewidget.setItem(0, 7, QTableWidgetItem("Satuan"))

# Menampilkan Data Yang Diambil Dari SQL

for i in range(len(record)):

baris = i + 1

self.tablewidget.setItem(baris, 0, QTableWidgetItem(str(record[i][0])))

self.tablewidget.setItem(baris, 1, QTableWidgetItem(str(record[i][1])))

self.tablewidget.setItem(baris, 2, QTableWidgetItem(str(record[i][2])))

self.tablewidget.setItem(baris, 3, QTableWidgetItem(str(record[i][3])))

self.tablewidget.setItem(baris, 4, QTableWidgetItem(str(record[i][4])))

self.tablewidget.setItem(baris, 5, QTableWidgetItem(str(record[i][5])))

self.tablewidget.setItem(baris, 6, QTableWidgetItem(str(record[i][6])))

self.tablewidget.setItem(baris, 7, QTableWidgetItem(str(record[i][7])))

# Membuat Agar Table Strech

self.tablewidget.horizontalHeader().setStretchLastSection(True)

self.tablewidget.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

cur.close()

return self.tablewidget

if \_\_name\_\_ == '\_\_main\_\_':

#Inisisalisai pyqt

app = QApplication(sys.argv)

#mengatur style di window menjadi style fusion

app.setStyle("fusion")

#membuat variabel ex yang berisi class FormulaMath

ex = Mysql()

ex.setStyleSheet("background-color: #fcf1f1;")

ye = QPushButton()

#Menentukan ukuran window dan title untuk menampilkan

ex.setGeometry(100,100,800,600)

#membuat judul window

ex.setWindowTitle("Database Mysql in Pyqt5")

#menampilan isi dari variabel ex

ex.show()

#membuat system exit

sys.exit(app.exec\_())

1. **Penjelasan Kode Program**

* *class Mysql(QWidget):*

*#Membuat fungsi init untuk inisialisasi class Mahasiswa*

*def \_\_init\_\_(self):*

*#untuk mengembalikan semua atribut dan method yang ada*

*super().\_\_init\_\_()*

*#membuka database*

*self.OpenDatabase()*

*#memanggil fungsi Layout yang sudah dibuat agar ditampilkan hasilnya*

*self.Layout()*

Membuat sebuah class Mysql dan fungsi init untuk menampilan semua fungsi yang telah dibuat seperti fungsi OpenDatabase dan Layout()

* *def OpenDatabase(self):*

*try:*

*self.db = mdb.connect('kprikaryasehat.site', 'kprikary\_kuliah', 'unijoyo2020', 'kprikary\_resto')*

*QMessageBox.about(self, 'Connection', 'Database Connected Successfully')*

*except mdb.Error as e:*

*QMessageBox.about(self, 'Connection', 'Failed To Connect Database')*

Membuat fungsi OpenDatabase yang bersi variable db untuk menghubungkan database, jika berhasil masuk maka akan menampilkan Qessagebox berhasil, namun jika gagal akan menampilkan QMessageBox gagal.

* *def Layout(self):*

*#Membuat Grid Layout*

*grid = QGridLayout()*

*add\_data = QLabel("Tambah Data:")*

*grid.addWidget(add\_data,0,0)*

*menu = QLabel("Nama Menu:")*

*grid.addWidget(menu,2,0)*

*self.Menu = QLineEdit(self)*

*grid.addWidget(self.Menu,2,1,1,2)*

*self.Menu.setStyleSheet("background-color: #f6f6f6;")*

*keterangan = QLabel("Keterangan:")*

*grid.addWidget(keterangan,3,0)*

*self.Keterangan = QLineEdit(self)*

*grid.addWidget(self.Keterangan,3,1,1,2)*

*self.Keterangan.setStyleSheet("background-color: #f6f6f6;")*

*harga = QLabel("Harga:")*

*grid.addWidget(harga,4,0)*

*self.Harga = QLineEdit(self)*

*grid.addWidget(self.Harga,4,1,1,2)*

*self.Harga.setStyleSheet("background-color: #f6f6f6;")*

*satuan = QLabel("Satuan:")*

*grid.addWidget(satuan,5,0)*

*self.Satuan = QLineEdit(self)*

*grid.addWidget(self.Satuan,5,1,1,2)*

*self.Satuan.setStyleSheet("background-color: #f6f6f6;")*

*add\_button = QPushButton("Tambah Data")*

*grid.addWidget(add\_button,6,0,1,0)*

*add\_button.setStyleSheet("background-color: #ffc7c7;")*

*edit\_button = QPushButton("Edit Data")*

*grid.addWidget(edit\_button,7,0,1,1)*

*edit\_button.setStyleSheet("background-color: #ffc7c7;")*

*update\_button = QPushButton("Update Data")*

*grid.addWidget(update\_button,7,1,1,2)*

*update\_button.setStyleSheet("background-color: #ffc7c7;")*

*search = QLabel("Cari Data:")*

*grid.addWidget(search,8,0)*

*self.Cari = QLineEdit(self)*

*grid.addWidget(self.Cari,8,1)*

*self.Cari.setStyleSheet("background-color: #f6f6f6;")*

*search\_button = QPushButton("Cari")*

*grid.addWidget(search\_button,8,2)*

*search\_button.setStyleSheet("background-color: #ffc7c7;")*

*delete\_button = QPushButton("Delete Data")*

*grid.addWidget(delete\_button,9,0,1,0)*

*delete\_button.setStyleSheet("background-color: #ffc7c7;")*

*#Membuat widget table widget yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid*

*self.tablewidget = QTableWidget(self)*

*self.tablewidget.setObjectName("Data")*

*self.tablewidget.setStyleSheet("background-color: #f6f6f6;")*

*grid.addWidget(self.tampilData(),11,0,5,0)*

Membuat fungsi layout yang di dalamnya berisi beberapa widget. Dan membuat sebuah layout grid yang dimasukkan ke dalam variable grid. Membuat label, Line Edit, dan Button dengan identifikasi variable yang berbeda dan akan dimasukkan ke dalam layout Grid dengan syntax addWidget, dan mengatur posisi masing – masing widget tersebut agar rapi dan tidak bertumpukan. Dan yang terakhir Membuat widget tablewidget yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid.

* *#Ketika button di klik akan memanggi fungsi masing - masing*

*add\_button.clicked.connect(self.tambahData)*

*search\_button.clicked.connect(self.filterData)*

*edit\_button.clicked.connect(self.editData)*

*update\_button.clicked.connect(self.updateData)*

*delete\_button.clicked.connect(self.deleteData)*

*#Layout grid di jadikan layout utama*

*self.setLayout(grid)*

Membuat 5 signal dimana ketika masing – masing button di klik, akan memanggil fungsi yang sudah di set. Serta menjadikan layout grid menjadi layout utama pada window.

* #Fungsi menampilkan Data menu1 ke dalam tabelwidget

def tampilData(self):

# Membuat Cursor

cur = self.db.cursor()

# Mengeksekusi perintah sql

cur.execute("SELECT \* FROM menu1")

# Mengambil Semua Data

data = cur.fetchall()

# Menjadikan Data bentuk list

record = list(data)

# Membuat Baris Yang Akan Ditampilkan DI table

self.tablewidget.setRowCount(len(record)+1)

# Membuat Kolom yang akan ditampilkan di table

self.tablewidget.setColumnCount(8)

# Membuat Header

self.tablewidget.setItem(0, 0, QTableWidgetItem("ID Menu"))

self.tablewidget.setItem(0, 1, QTableWidgetItem("ID Menu Kat"))

self.tablewidget.setItem(0, 2, QTableWidgetItem("ID Resto"))

self.tablewidget.setItem(0, 3, QTableWidgetItem("Nama Menu"))

self.tablewidget.setItem(0, 4, QTableWidgetItem("Keterangan"))

self.tablewidget.setItem(0, 5, QTableWidgetItem("Gambar"))

self.tablewidget.setItem(0, 6, QTableWidgetItem("Harga"))

self.tablewidget.setItem(0, 7, QTableWidgetItem("Satuan"))

# Menampilkan Data Yang Diambil Dari SQL

for i in range(len(record)):

baris = i + 1

self.tablewidget.setItem(baris, 0, QTableWidgetItem(str(record[i][0])))

self.tablewidget.setItem(baris, 1, QTableWidgetItem(str(record[i][1])))

self.tablewidget.setItem(baris, 2, QTableWidgetItem(str(record[i][2])))

self.tablewidget.setItem(baris, 3, QTableWidgetItem(str(record[i][3])))

self.tablewidget.setItem(baris, 4, QTableWidgetItem(str(record[i][4])))

self.tablewidget.setItem(baris, 5, QTableWidgetItem(str(record[i][5])))

self.tablewidget.setItem(baris, 6, QTableWidgetItem(str(record[i][6])))

self.tablewidget.setItem(baris, 7, QTableWidgetItem(str(record[i][7])))

# Membuat Agar Table Strech

self.tablewidget.horizontalHeader().setStretchLastSection(True)

self.tablewidget.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

cur.close()

return self.tablewidget

Membuat fungsitampilData untuk menampilkan data yang ada di dalam database. Membuat variable cur yang didalamnya berisi syntax untuk db.cursor, dan kemudia mengeksekusi perintah sql dan dimasukkan ke dalam variable data, dan nantinya akan dilakukan perulangan untuk menampilkan data – data yang ada di dalam table, dan dimasukkan ke dalam tablewidget.

* #Fungsi tambah Data ke dalam tabel menu1

def tambahData(self):

# Mengambil Text inputan

menu = str(self.Menu.text())

keterangan = str(self.Keterangan.text())

harga = int(self.Harga.text())

satuan = str(self.Satuan.text())

# Mendefinisikan Cursor

cur = self.db.cursor()

id = self.db.cursor()

# Menjalankan Perintah SQL

id.execute("SELECT idmenu FROM menu1 ORDER BY idmenu DESC LIMIT 1")

# Mengambil Satu Data

idMenu = id.fetchone()

# Membuat Id Auto Increment

idMenu = idMenu[0] + 1

# Menuliskan Perintah SQL

sql = "INSERT INTO menu1(idmenu,idmenukat,idresto,namamenu,keterangan,filegambar,harga,satuan) VALUES ('%d','%d','%d','%s','%s','%s','%d','%s')" % (idMenu, 1, 5,menu,keterangan,'https://img-global.cpcdn.com/recipes/84fae0149dbe9168/751x532cq70/telor-balado-foto-resep-utama.jpg',harga,satuan)

# MenjalanKan Perintah Try

try:

# Eksekusi Perintah SQL

cur.execute(sql)

# Mengcommit agar perubahan Tersimpan

self.db.commit()

# Membuat Notifikasi Berhasil Tambah Data

QMessageBox.about(self, 'Berhasil', 'Berhasil Menambah Data')

# Mengubah Inputan Agar Menjadi Kosong

self.Menu.setText("")

self.Keterangan.setText("")

self.Harga.setText("")

self.Satuan.setText("")

# Menampilkan Data

self.tampilData()

id.close()

cur.close()

# Jika Terjadi Error

except:

# Merollback data

self.db.rollback()

print("Gagal")

Membuat fungsi tambahData untuk menambahkan data hasil inputan dari line edit kedalam database. Dan membuat suatu kondisi dimana jika query berhasil dieksekusi value dalam line edit akan dikosongkan Kembali kemudian menamilkan data record terbaru kedalam tablewidget dengan fungsi tampilData. Jika gagal makan akan mengeluarkan output Gagal.

* *#Fungsi edit Data tabel menu1*

*def editData(self):*

*# Mengambil Data Yang DIpilih*

*index = self.tablewidget.selectedIndexes()[0]*

*id = self.tablewidget.model().data(index)*

*# Membuat Cursor*

*cur = self.db.cursor()*

*# Perintah Mencari Data Bedasarkan ID*

*sql = "SELECT \* FROM menu1 WHERE idmenu = '%s' " % (id)*

*# Eksekusi Perintah SQL*

*cur.execute(sql)*

*# Mengambil Satu Data*

*data = cur.fetchone()*

*# Menyimpan Data Id Di variabel idEdit*

*self.idEdit = data[0]*

*# Menampilkan Data Ke Textbox*

*self.Menu.setText(data[3])*

*self.Keterangan.setText(data[4])*

*self.Harga.setText(str(data[6]))*

*self.Satuan.setText(data[7])*

*# Fungsi Update Data menu1*

*def updateData(self):*

*# Mengambil Data Dari Inputan*

*menu = str(self.Menu.text())*

*keterangan = str(self.Keterangan.text())*

*harga = int(self.Harga.text())*

*satuan = str(self.Satuan.text())*

*# Membuat Cursor*

*cur = self.db.cursor()*

*# Perintah SQL Update*

*sql = "UPDATE menu1 SET namamenu = '%s', keterangan = '%s', harga = '%d', satuan = '%s' WHERE idmenu = '%d'" % (menu,keterangan,harga,satuan,self.idEdit)*

*try:*

*# Eksekusi Perintah SQL*

*cur.execute(sql)*

*# Commit DB Agar Terjadi Perubahan DI database*

*self.db.commit()*

*# Membuat Message Box*

*QMessageBox.about(self, 'Berhasil', 'Berhasil Update Data')*

*self.Menu.setText("")*

*self.Keterangan.setText("")*

*self.Harga.setText("")*

*self.Satuan.setText("")*

*self.tampilData()*

*print("Berhasil")*

*except:*

*self.db.rollback()*

*print("Gagal")*

Membuat fungsi editdata dan update data untuk merubah data yang ada pada table menu 1, dengan cara memilih data record mana yang mau di update kemudian setelah dipilih data record tersebut akan ditampilkan di line edit, selanjutnya jika sudah diedit maka akan masuk ke fungsi updatedata, dengan membuat variable cur yang berisi db.cursor kemudian mengeksekusi perintah sql. Jika berhasil akan menampilkan QMessagebox berhasil dan mengosongkon lineedit Kembali, namun jika gagal akan megoutputkan gagal pada terminal.

* # Fungsi Hapus Data pada table menu1

*def deleteData(self):*

*# Mengambil Data Yang Di pilih*

*index = self.tablewidget.selectedIndexes()[0]*

*id = self.tablewidget.model().data(index)*

*# Membuat Cursor*

*cur = self.db.cursor()*

*# Perintah SQL Untuk Menghapus Data*

*sql = "DELETE FROM menu1 WHERE idmenu = '%s' " % (id)*

*try:*

*# Eksekusi Perintah SQL*

*cur.execute(sql)*

*# Commit Ke Database Agar Perubahan Tersimpan*

*self.db.commit()*

*# Menampilkan Data*

*self.tampilData()*

*print("Berhasil")*

*except:*

*# Merollback data*

*self.db.rollback()*

*print("Gagal")*

Pertama memilih data record mana yang mau dihapus, kemudian mengeksekusi perintah sql pada data record yang dipilih, jika berhasil akan menampilkan Berhasil pada terminal, namun jika gagal akan menampilkan Gagak pada terminal.

* *#Fungsi serching Data yang ada di dalam tabel menu1*

*def filterData(self):*

*# Membuat Cursor*

*cur = self.db.cursor()*

*filter\_search = str(self.Cari.text())*

*# Mengeksekusi perintah sql*

*cur.execute("SELECT \* FROM menu1 WHERE namamenu LIKE '%"+str(filter\_search)+"%' OR keterangan LIKE '%"+str(filter\_search)+"%' OR harga LIKE '%"+str(filter\_search)+"%' OR satuan LIKE '%"+str(filter\_search)+"%'")*

*# Mengambil Semua Data*

*data = cur.fetchall()*

*# Menjadikan Data bentuk list*

*record = list(data)*

*# Membuat Baris Yang Akan Ditampilkan DI table*

*self.tablewidget.setRowCount(len(record)+1)*

*# Membuat Kolom yang akan ditampilkan di table*

*self.tablewidget.setColumnCount(8)*

*# Membuat Header*

*self.tablewidget.setItem(0, 0, QTableWidgetItem("ID Menu"))*

*self.tablewidget.setItem(0, 1, QTableWidgetItem("ID Menu Kat"))*

*self.tablewidget.setItem(0, 2, QTableWidgetItem("ID Resto"))*

*self.tablewidget.setItem(0, 3, QTableWidgetItem("Nama Menu"))*

*self.tablewidget.setItem(0, 4, QTableWidgetItem("Keterangan"))*

*self.tablewidget.setItem(0, 5, QTableWidgetItem("Gambar"))*

*self.tablewidget.setItem(0, 6, QTableWidgetItem("Harga"))*

*self.tablewidget.setItem(0, 7, QTableWidgetItem("Satuan"))*

*# Menampilkan Data Yang Diambil Dari SQL*

*for i in range(len(record)):*

*baris = i + 1*

*self.tablewidget.setItem(baris, 0, QTableWidgetItem(str(record[i][0])))*

*self.tablewidget.setItem(baris, 1, QTableWidgetItem(str(record[i][1])))*

*self.tablewidget.setItem(baris, 2, QTableWidgetItem(str(record[i][2])))*

*self.tablewidget.setItem(baris, 3, QTableWidgetItem(str(record[i][3])))*

*self.tablewidget.setItem(baris, 4, QTableWidgetItem(str(record[i][4])))*

*self.tablewidget.setItem(baris, 5, QTableWidgetItem(str(record[i][5])))*

*self.tablewidget.setItem(baris, 6, QTableWidgetItem(str(record[i][6])))*

*self.tablewidget.setItem(baris, 7, QTableWidgetItem(str(record[i][7])))*

*# Membuat Agar Table Strech*

*self.tablewidget.horizontalHeader().setStretchLastSection(True)*

*self.tablewidget.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)*

*cur.close()*

*return self.tablewidget*

Membuat fungs filterData untuk mencari data record dengan cari menginputkan value ke dalam line edit, dimana value di dalam line edit akan diterima dan dimasukkan ke dalam variable filter\_search. Kemudian mmengeksekusi perintah sql dan akan menampilkan data pada tablewidget sesuai value variable filter\_search. Dan akan ditampilkan dengan syntax yang sama seperti fungsi tampildata.

* *if \_\_name\_\_ == '\_\_main\_\_':*

*#Inisisalisai pyqt*

*app = QApplication(sys.argv)*

*#mengatur style di window menjadi style fusion*

*app.setStyle("fusion")*

*#membuat variabel ex yang berisi class FormulaMath*

*ex = Mysql()*

*ex.setStyleSheet("background-color: #fcf1f1;")*

*ye = QPushButton()*

*#Menentukan ukuran window dan title untuk menampilkan*

*ex.setGeometry(100,100,800,600)*

*#membuat judul window*

*ex.setWindowTitle("Database Mysql in Pyqt5")*

*#menampilan isi dari variabel ex*

*ex.show()*

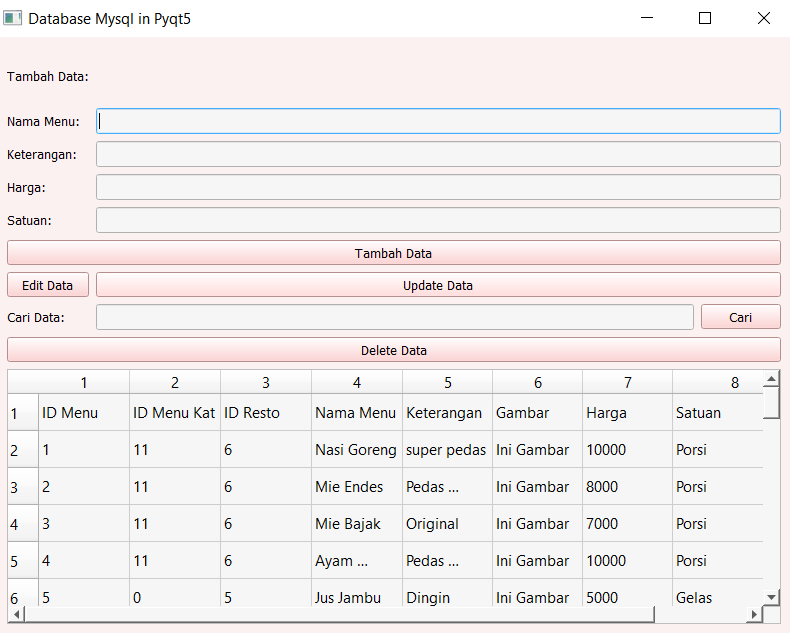
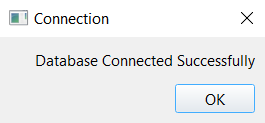
*#membuat system exit*

*sys.exit(app.exec\_())*

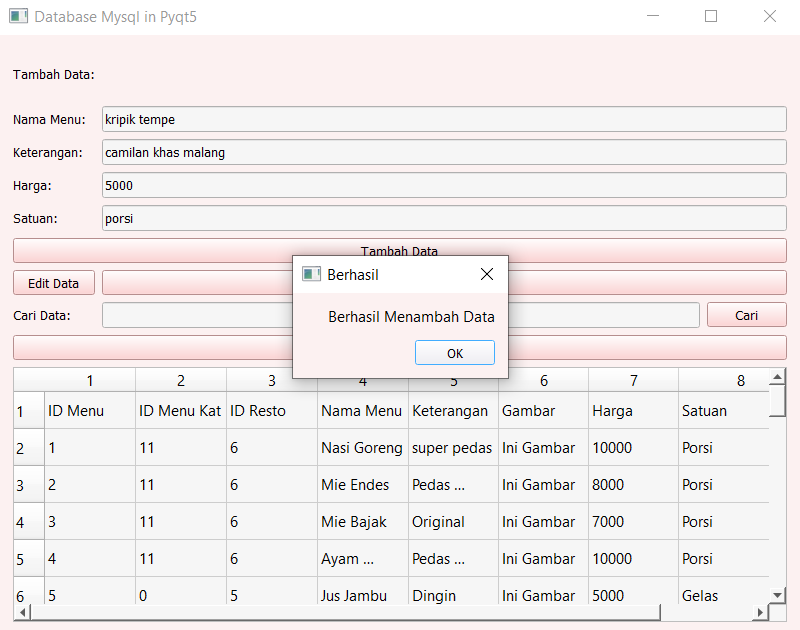
Mendeklarasikan QApplication di dalam variable app, memasukkan value yang ada di class Film ke dalam variable ex, dan memberikan style fusion ke dalam variable app. Lalu mengset ukuran geometry window dan memberikan title pada window dengan syntax setWindowTitle. Kemudian menampilkan variable tersebut dengan fungsi show(), dan membuat system exit.

1. **Hasil Running Program**

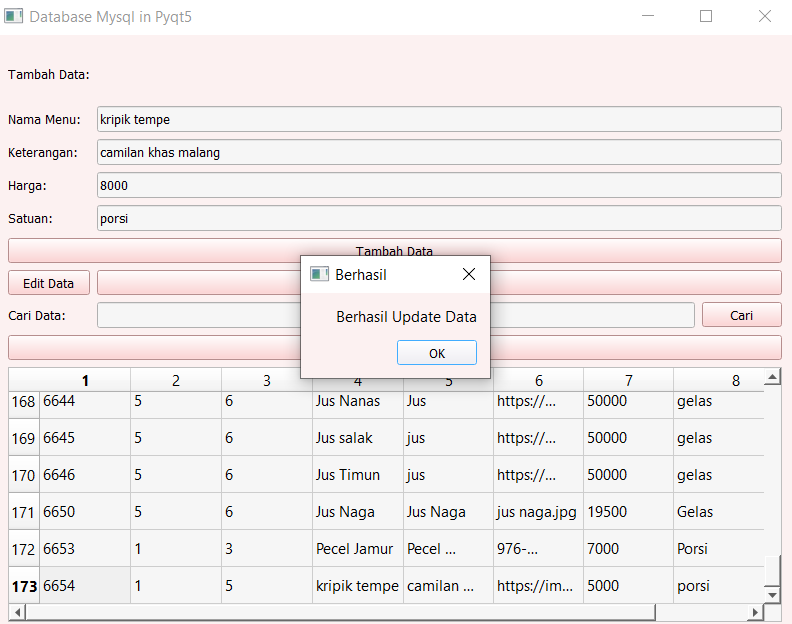
Tampilan awal



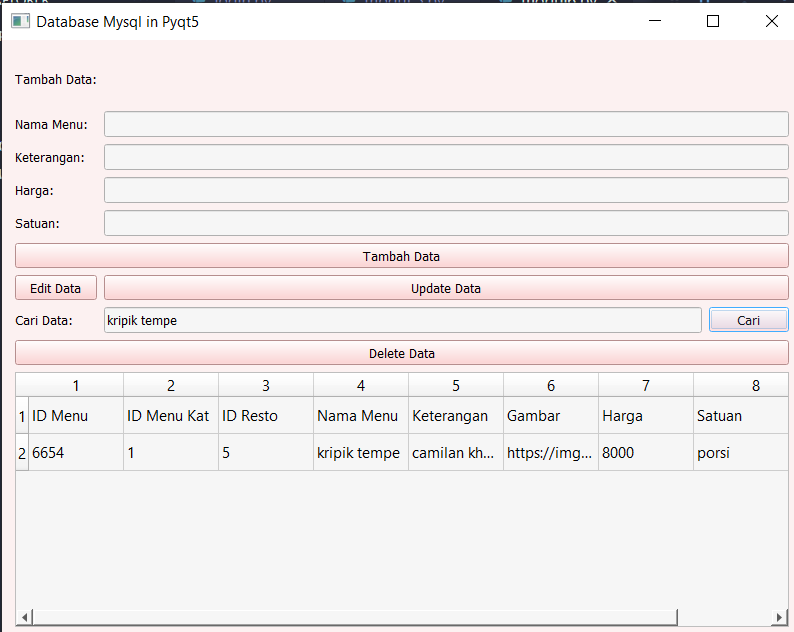
Untuk menambah data pada table database, yaitu dengan cara mengisi line edit dan menekan tombol tambah data.



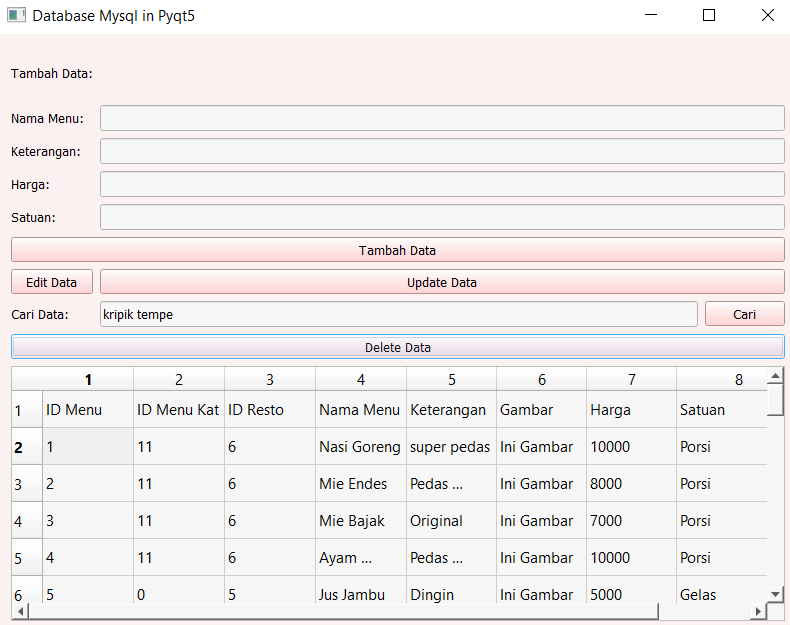
Untuk mengaupdate data, yaitu dengan cara memilih salah satu data records, dan kemudian merubah data yang diperlukan di line edit dan menekan tombol update data.



Selanjutnya untuk mencari data, kita masukkan keyword data yang dicari dan menekan tombol cari, lalu pada table widget akan menampilkan data sesuai keyword yang akan dicari.



Untuk menghapus data yaitu meimilih data record yang akan dihapus pada table widget dan menekan tombol delete data.



**BAB II**

**PENUTUP**

1. **Kesimpulan**
2. Kita bisa mendapatkan value dari inputan user menggunakan line edit.
3. Dengan adanya signal dan slot dapat membuat aplikasi yang di buat berfungsi lebih baik.
4. Database sangat membantu user untuk menyimpan beberapa data, dan data tersbut bisa diakses kapan saja
5. **Saran**

Banyak mencoba dan mengekplorasi widget yang lain agar lebih paham