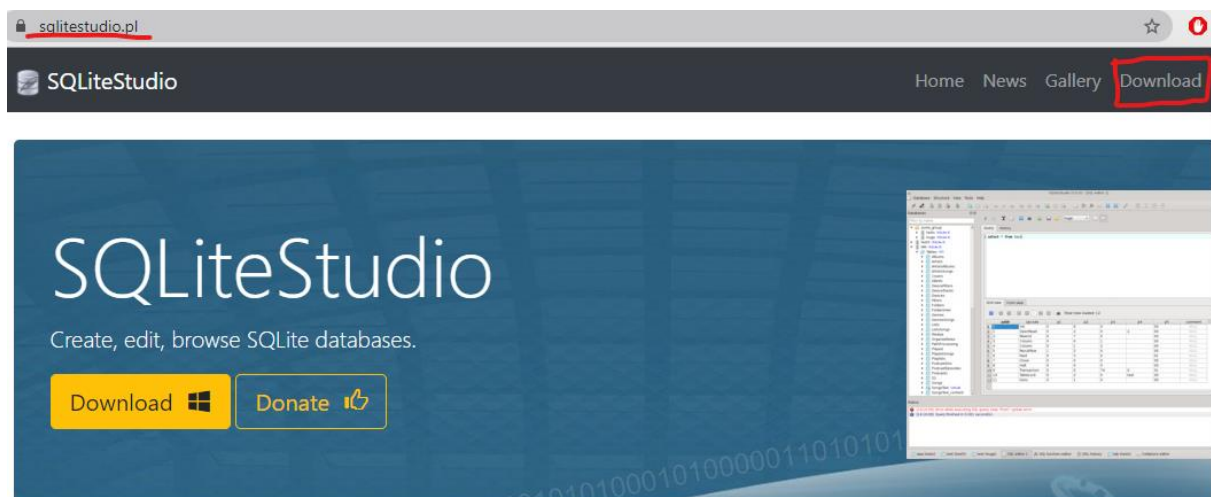







Unidade 2 – Graphical User Interfaces (GUI's)

Exemplo: Aplicação com Banco de Dados

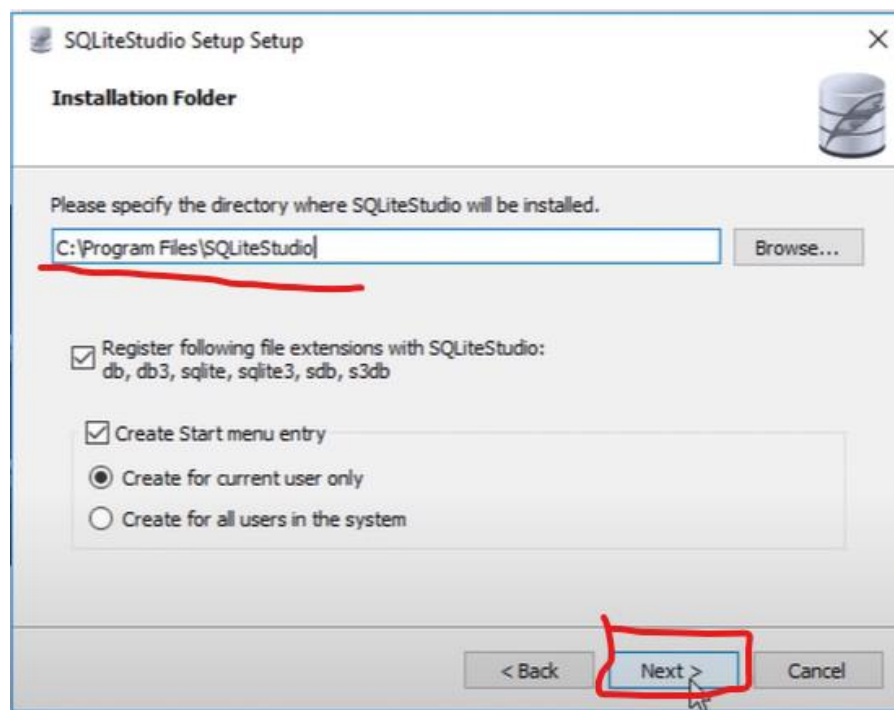
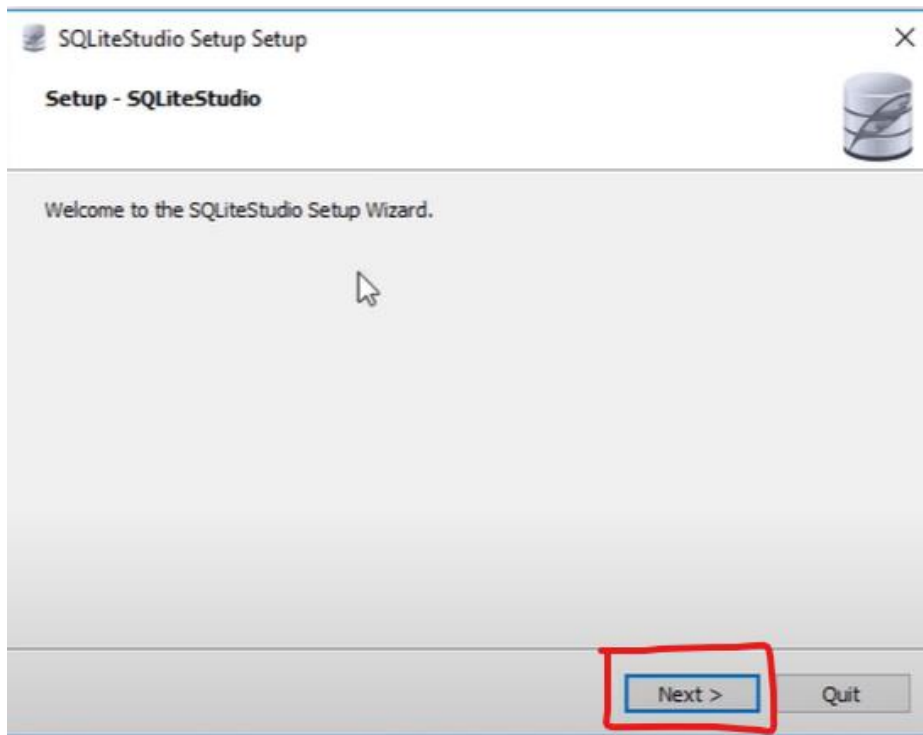
Nesse exemplo vamos usar um programa para modelagem das tabelas do banco: **SQLiteStudio**. Assim para instalação deve-se acessar o site: <https://sqlitestudio.pl/> ir em Download e baixar o arquivo.exe

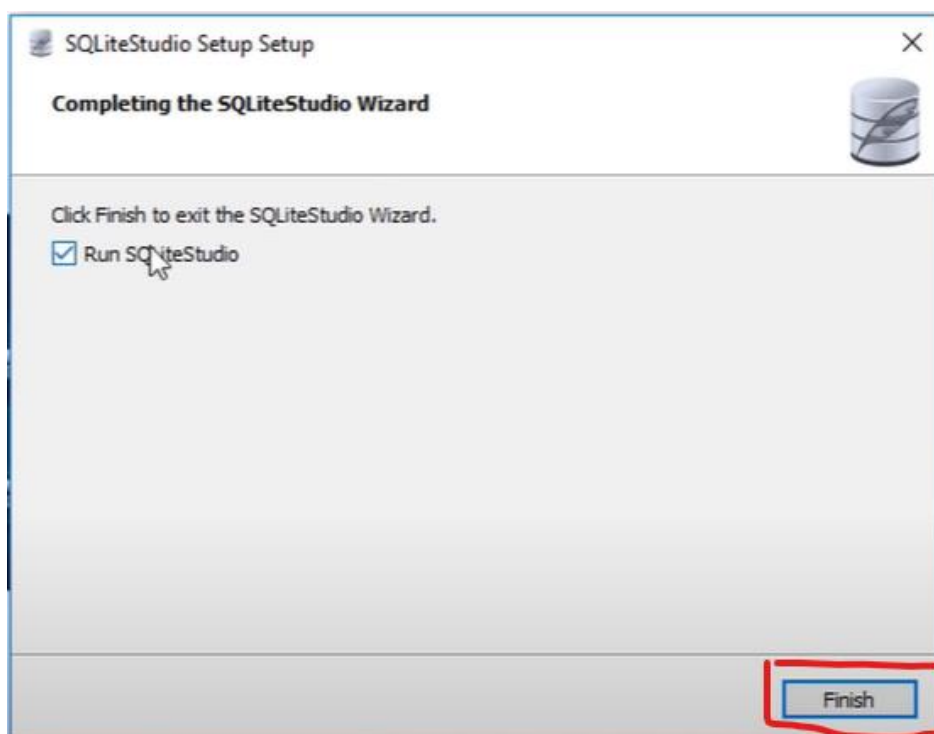
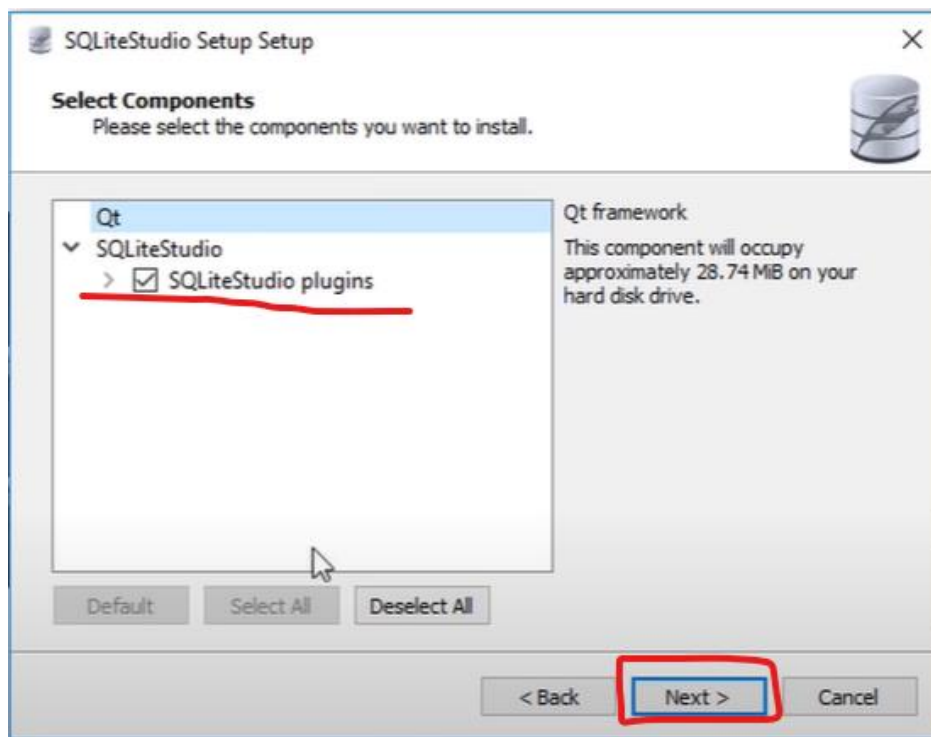


Após clicar em Download será possível selecionar o arquivo para a instalação

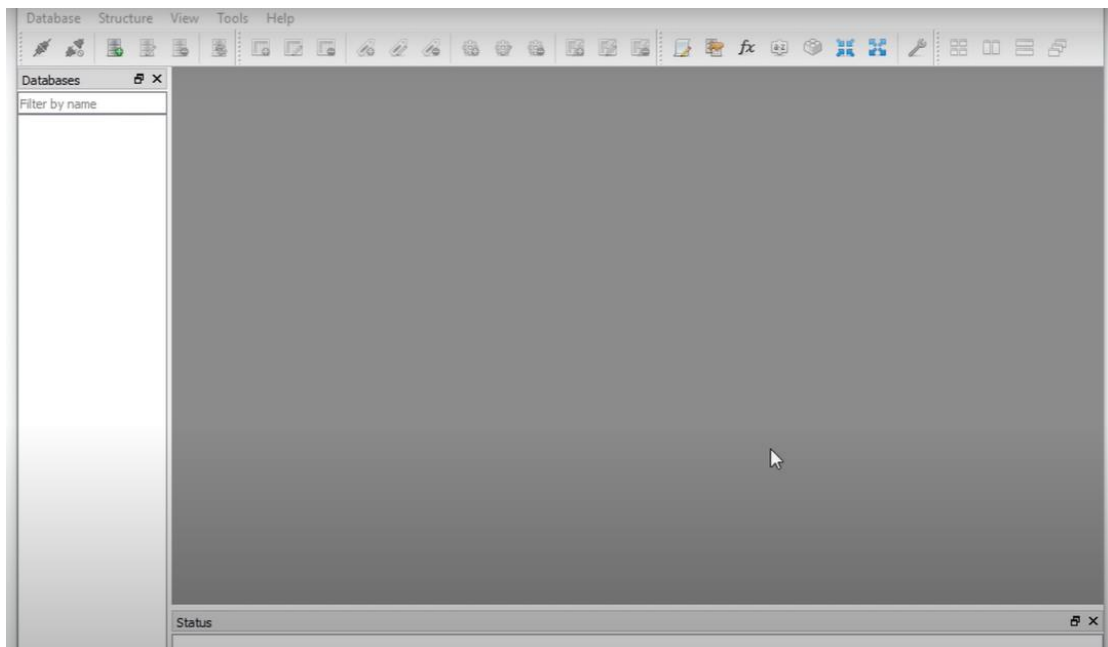
 InstallSQLiteStudio-3.2.1	46.1 MB
 InstallSQLiteStudio-3.2.1.dmg	25.5 MB
 InstallSQLiteStudio-3.2.1.exe	33.2 MB
 SQLiteStudio-3.2.1.dmg	30.9 MB
 sqlitestudio-3.2.1.tar.xz	30 MB
 SQLiteStudio-3.2.1.zip	27.5 MB
 Source code (zip)	
 Source code (tar.gz)	

Em seguida seguir os passos para instalação

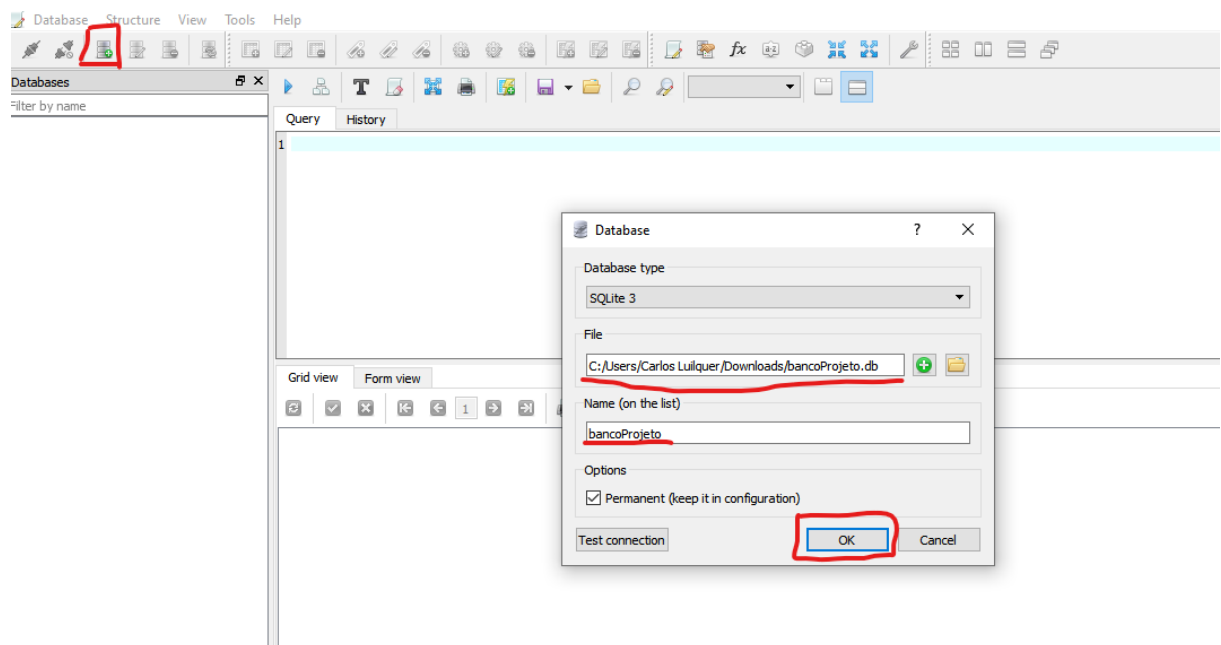




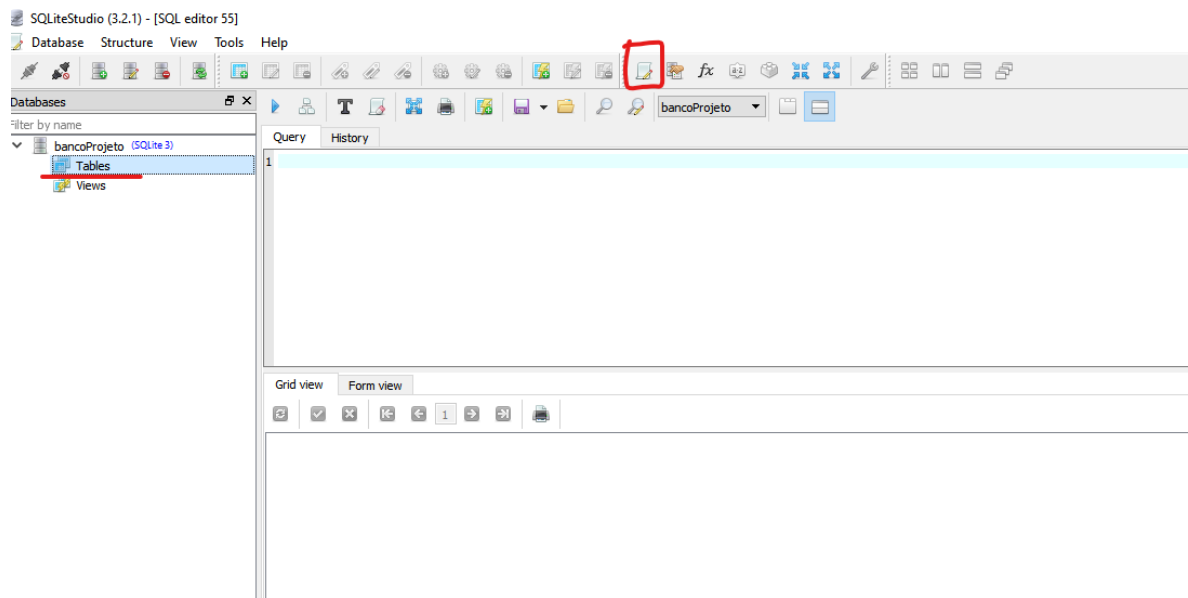
Por fim, será possível visualizar a seguinte janela:



Para criar um arquivo de banco de dados, deve-se clicar no botão na parte superior em seguida irá aparecer a janela **Database**, onde será possível selecionar o local e nomear o arquivo.

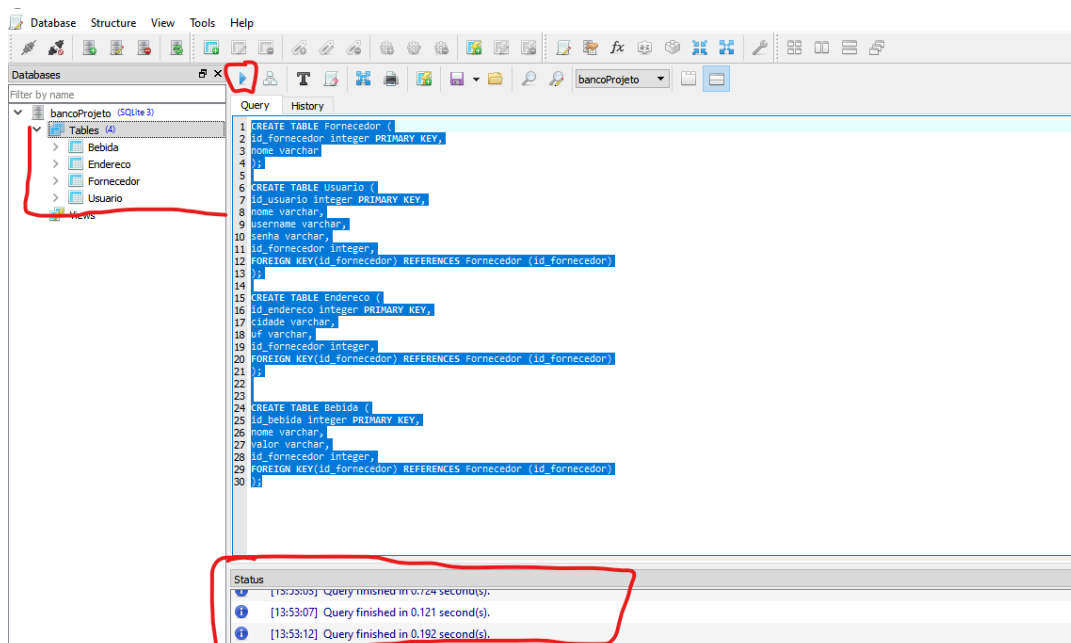


Depois de criar o banco de dados selecionar **Tables** e clicar na opção superior para iniciar o processo de modelagem.



Após isso, será possível inserir o seguinte código na linguagem SQL:

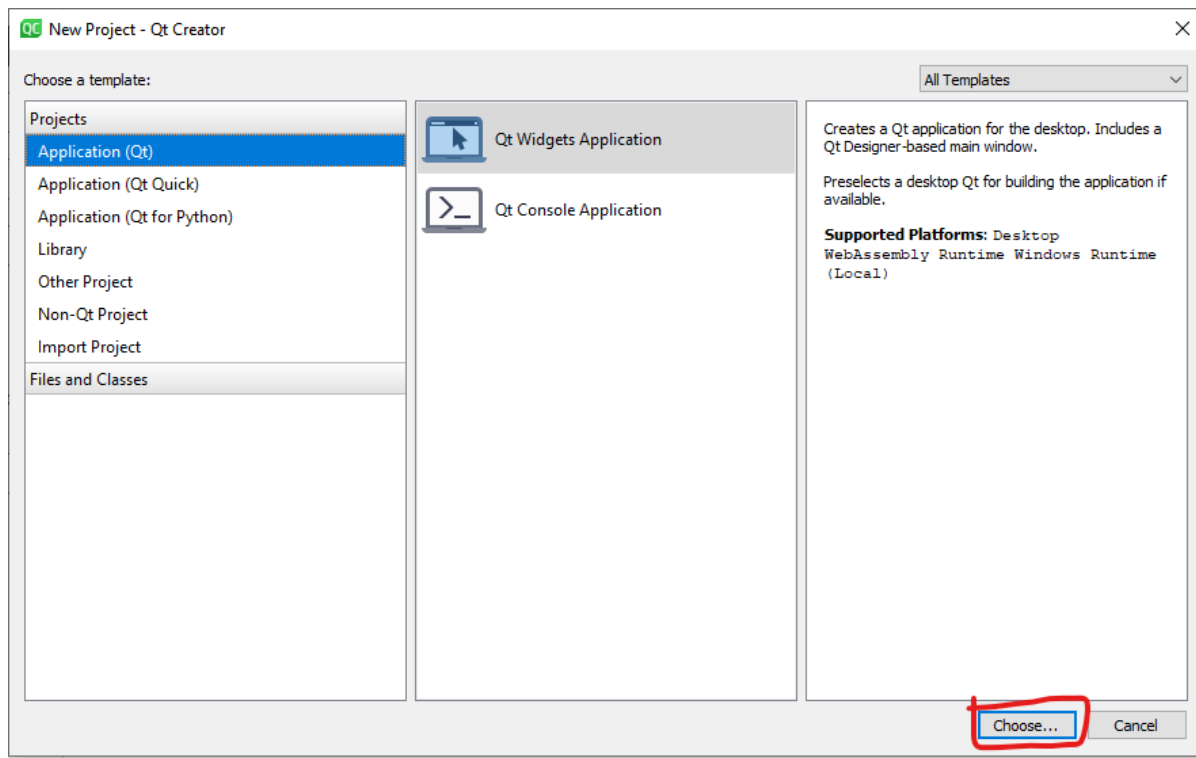
```
CREATE TABLE Fornecedor (  
id_fornecedor integer PRIMARY KEY AUTOINCREMENT,  
nome varchar  
);  
  
CREATE TABLE Usuario (  
id_usuario integer PRIMARY KEY AUTOINCREMENT,  
username varchar,  
senha varchar,  
id_fornecedor integer,  
FOREIGN KEY(id_fornecedor) REFERENCES Fornecedor (id_fornecedor)  
);  
  
CREATE TABLE Endereco (  
id_endereco integer PRIMARY KEY AUTOINCREMENT,  
cidade varchar,  
uf varchar,  
id_fornecedor integer,  
FOREIGN KEY(id_fornecedor) REFERENCES Fornecedor (id_fornecedor)  
);  
  
CREATE TABLE Bebida (  
id_bebida integer PRIMARY KEY AUTOINCREMENT,  
nome varchar,  
valor varchar,  
id_fornecedor integer,  
FOREIGN KEY(id_fornecedor) REFERENCES Fornecedor (id_fornecedor)  
);
```



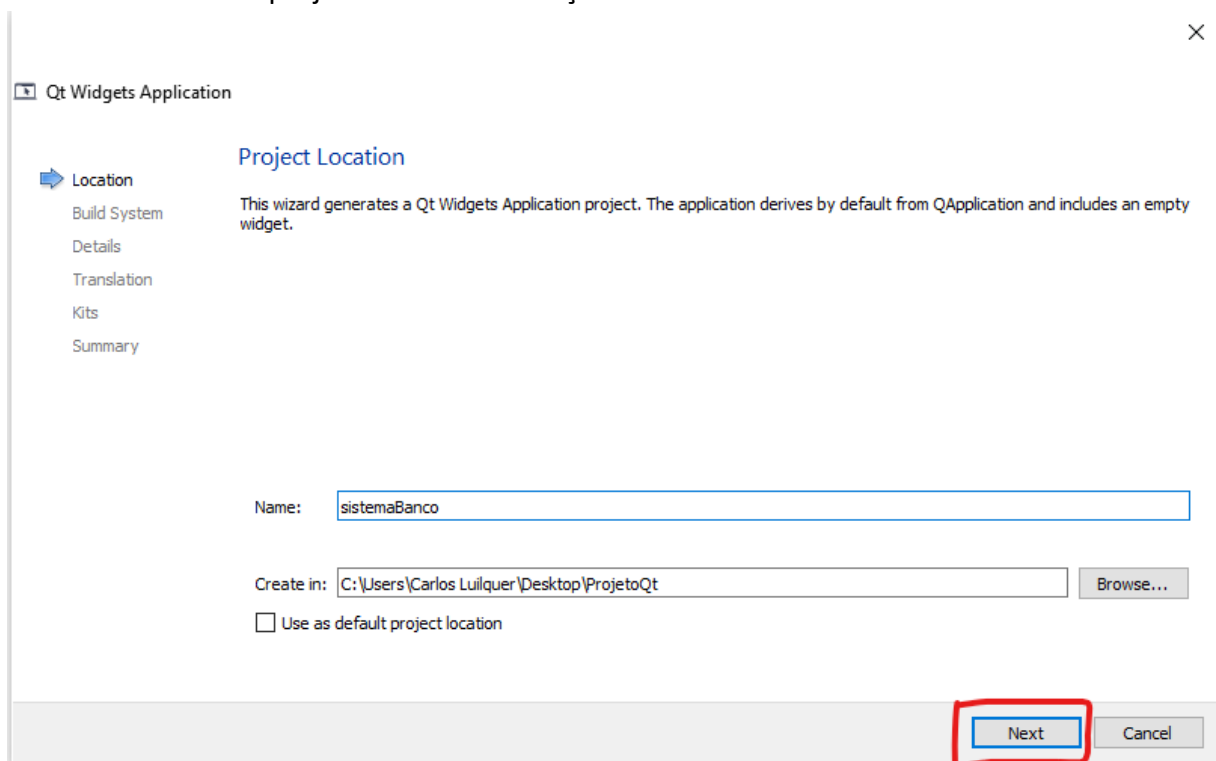
Depois que o código SQL for digitado como demonstrado na imagem acima, selecionar todas as linhas (**Ctrl + A**) e clicar na seta em azul (parte superior) para a criação das tabelas no banco de dados. Na aba Status será exibido as mensagens de êxito. Ao lado pode-se visualizar todas as tabelas criadas.

Após finalizar o processo de modelagem vamos criar um novo projeto junto com todos os arquivos necessários. Depois vamos usar o **Qt Designer** para modificar a parte da interface gráfica com usuário (GUI).

- 1) Criar um novo projeto do tipo **QT Widgets Application** no **QT Creator**:
File > New File or Project > Applications > Qt Widgets Application > Choose

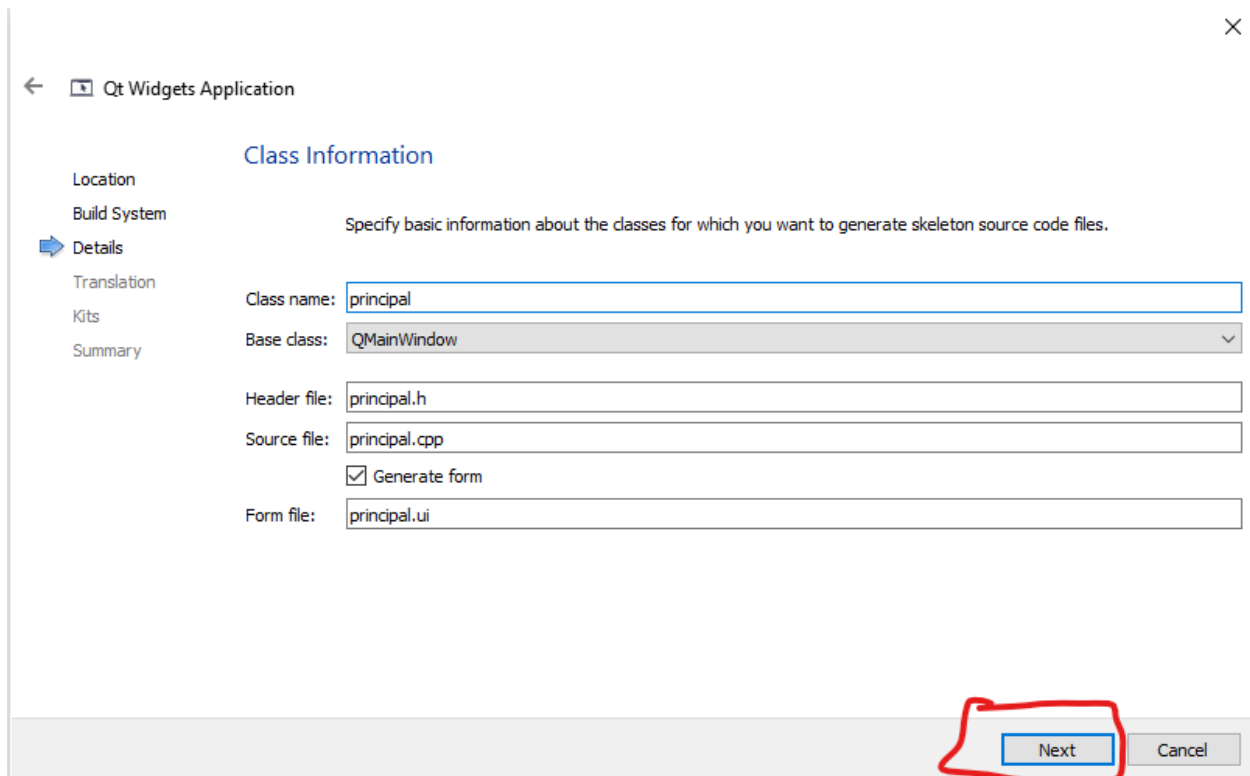


2) Escolher o Nome do projeto e a sua localização no HD:

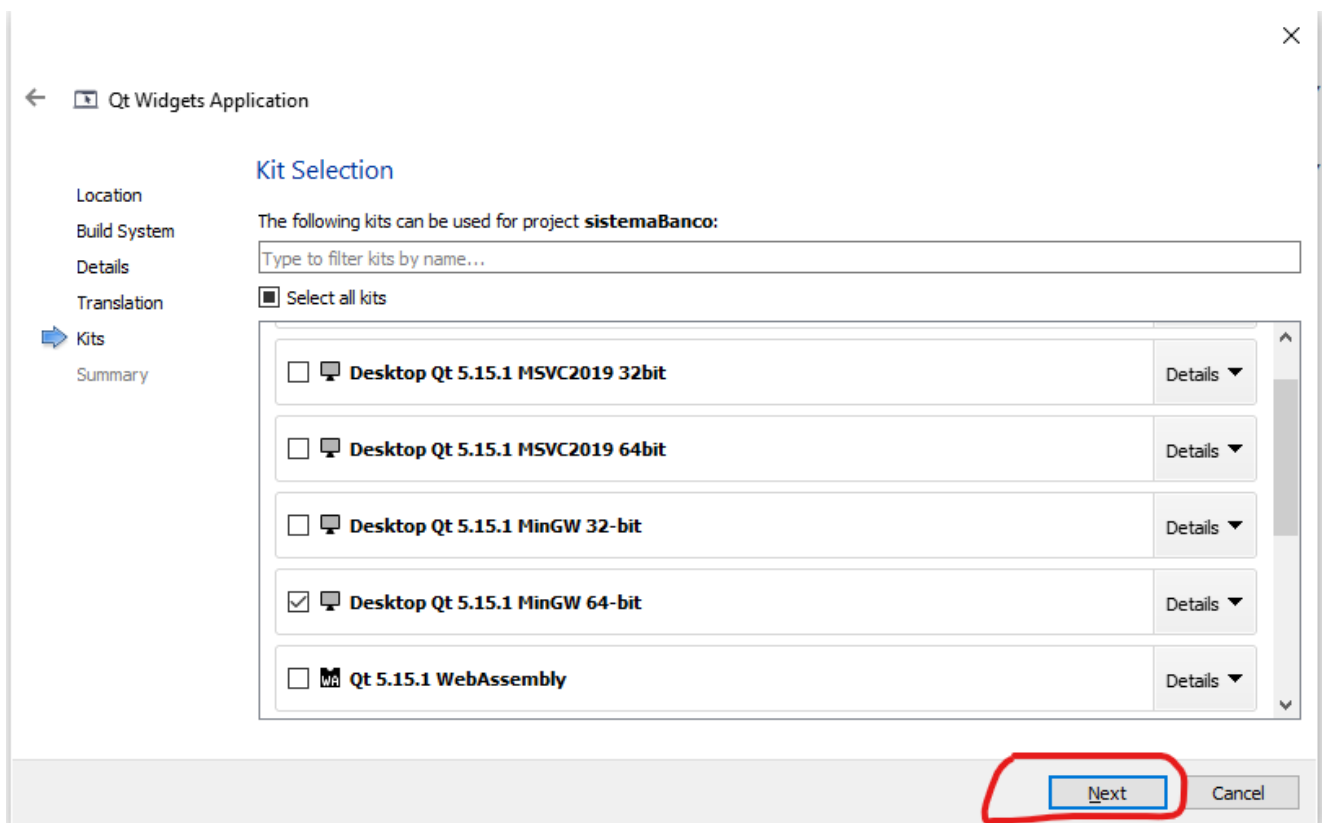


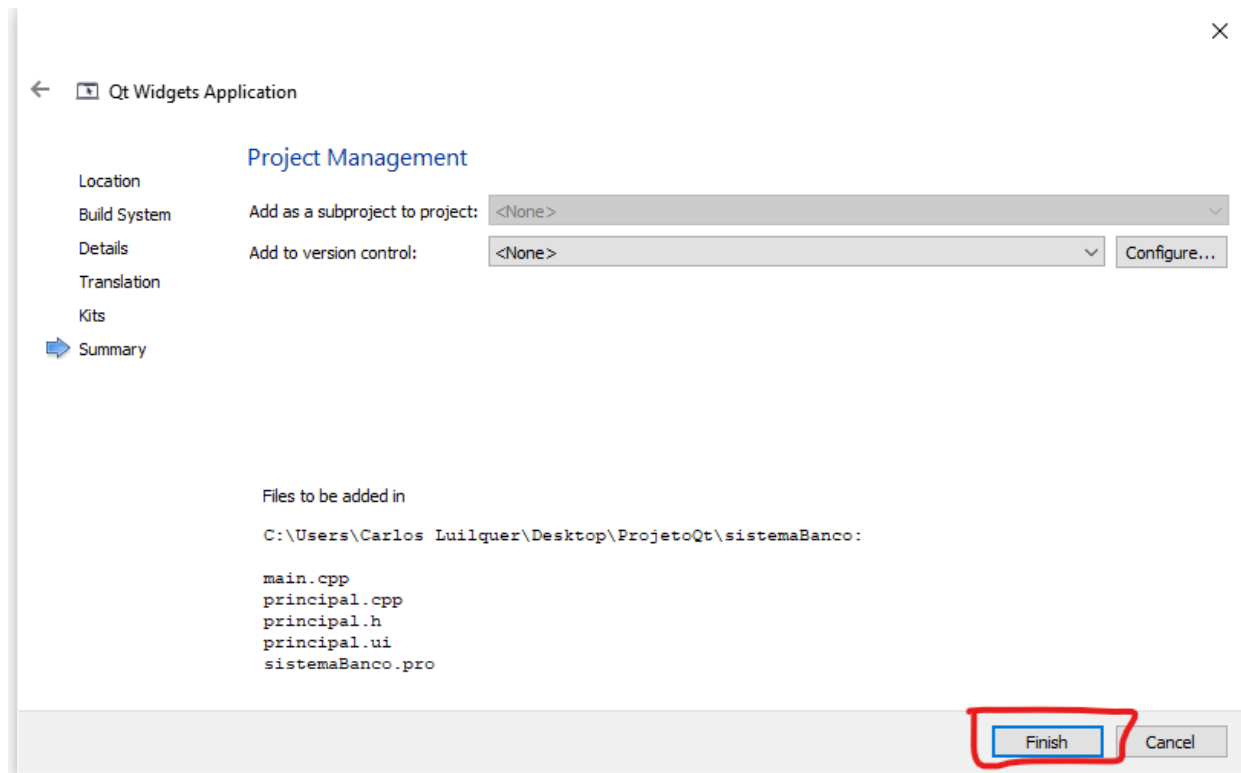
Seguir para os próximos 3 passos.

3) Modificar o nome da classe para **Principal** na janela **Class Information** dialog:

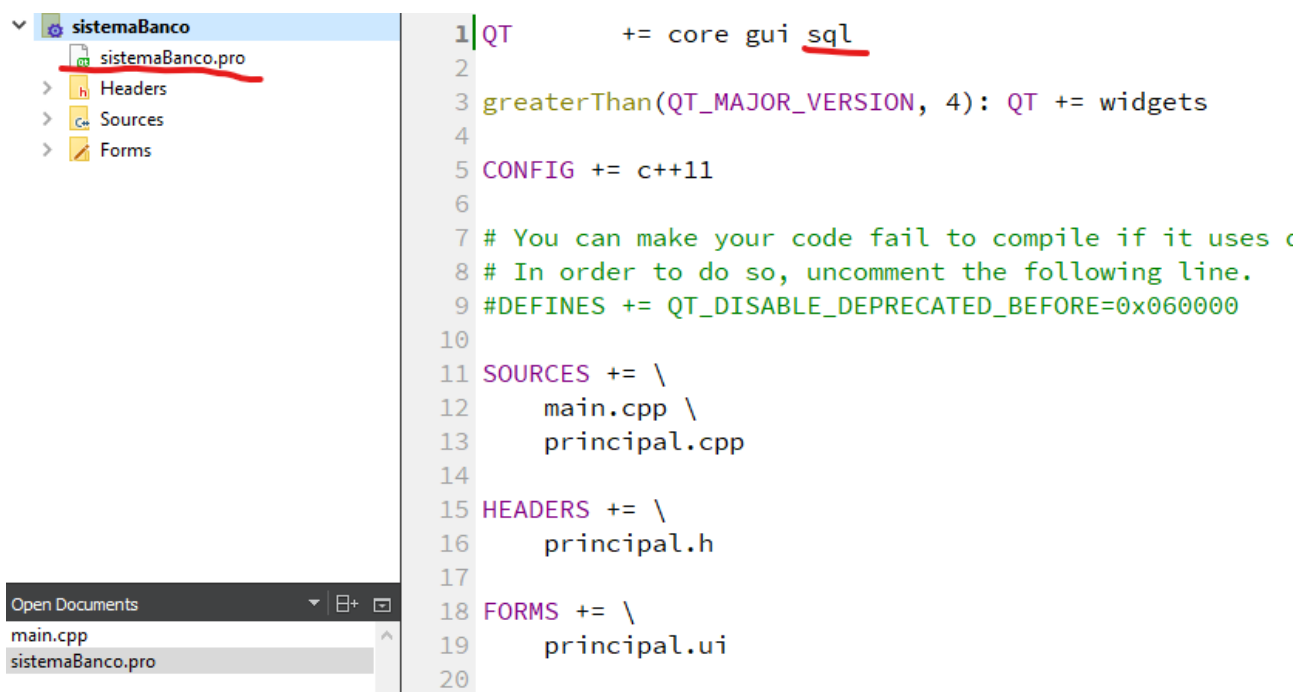


4) Selecionar a caixinha de acordo com o seu sistema operacional (32bit ou 64bit):

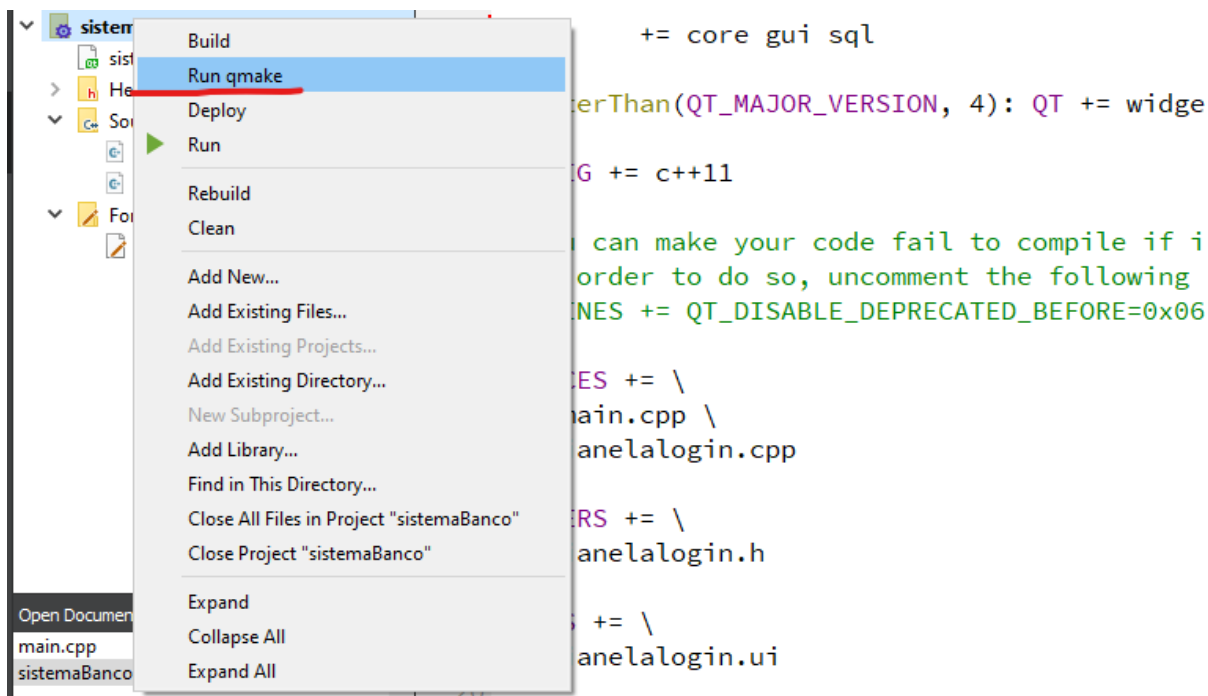




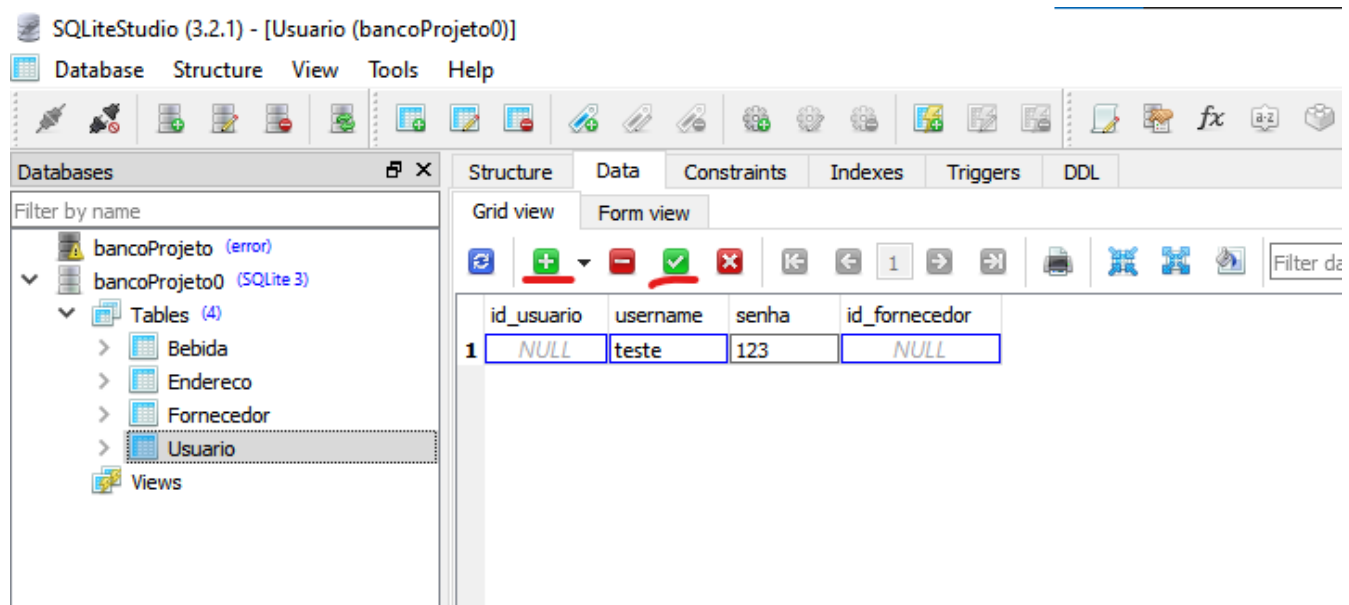
No **sistemaBanco.pro** inserir **sql** como demonstrado na imagem:



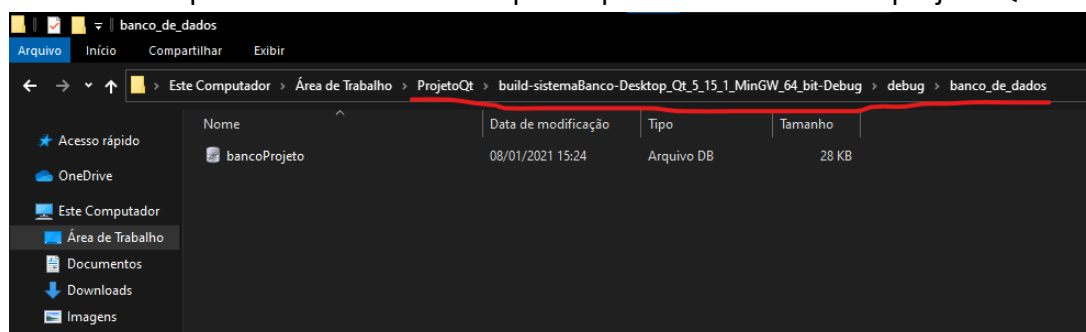
Em seguida clicar em **Run qmake**.



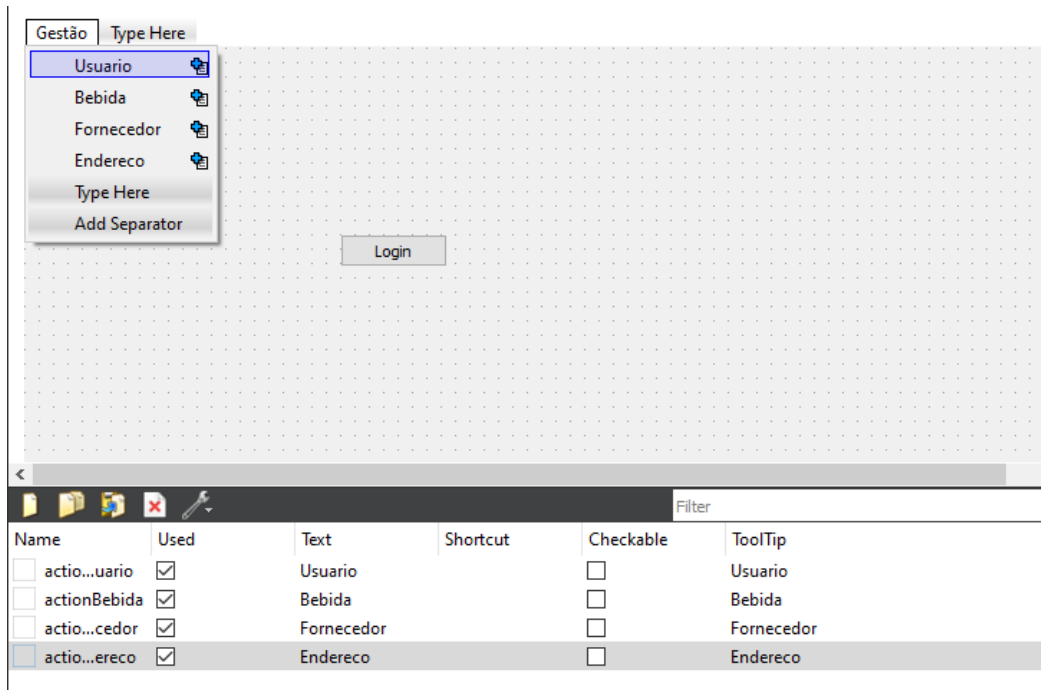
No **SQLiteStudio** clicar na tabela Usuário e inserir os seguintes atributos. Por fim clicar em **Commit**.



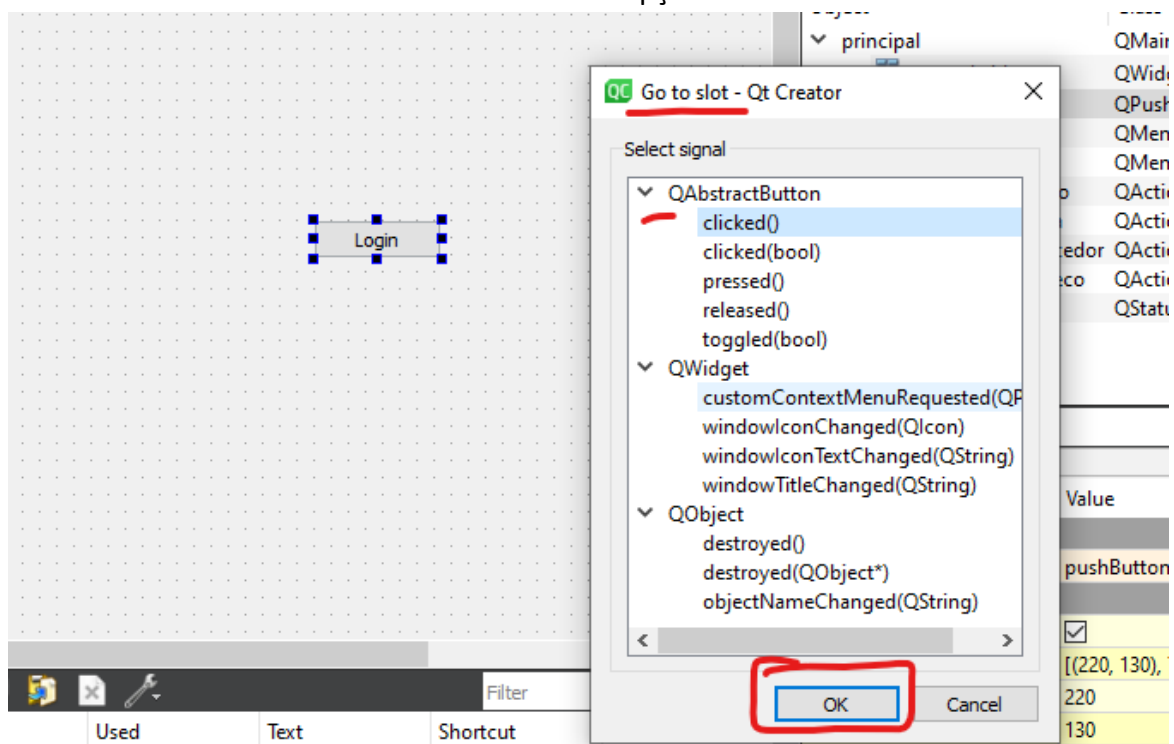
Após isso mover o arquivo do banco de dados para a pasta onde foi salvo o projeto Qt:



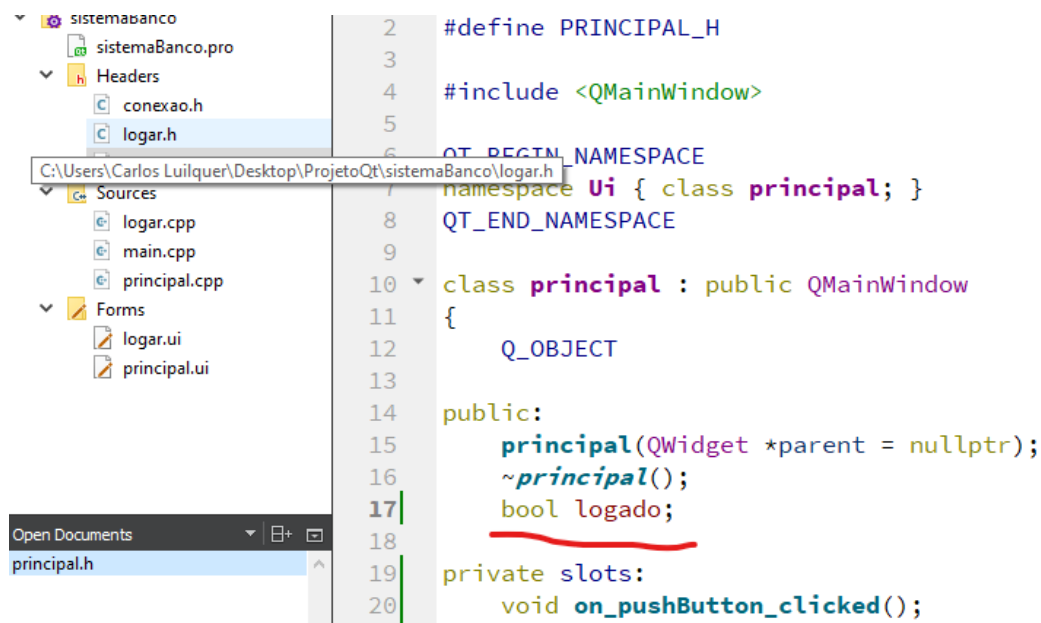
Na janela **principal** criar os seguintes componentes:



Clicar com o botão direito do mouse e seleccionar as opções:

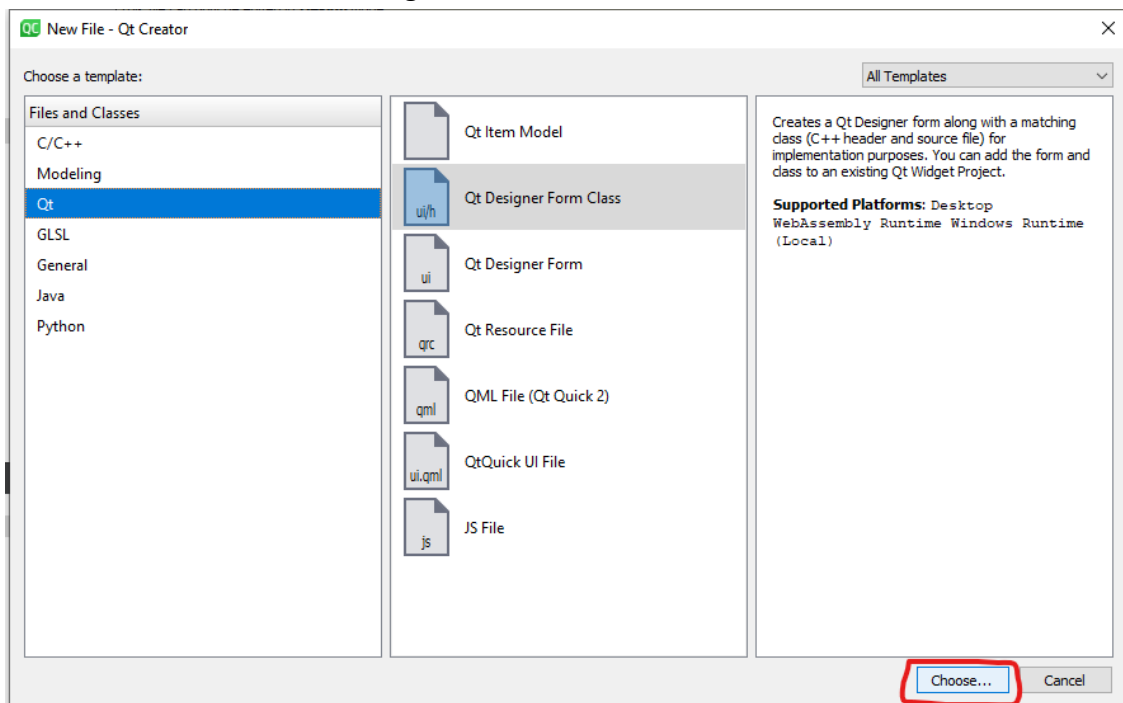


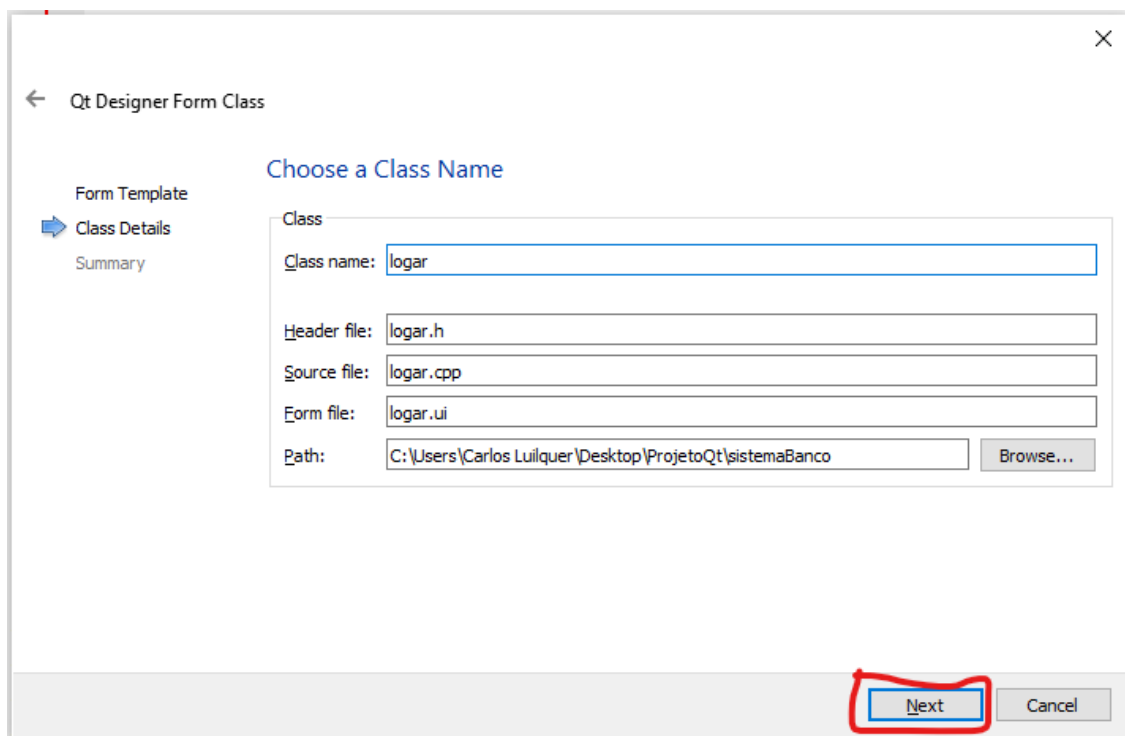
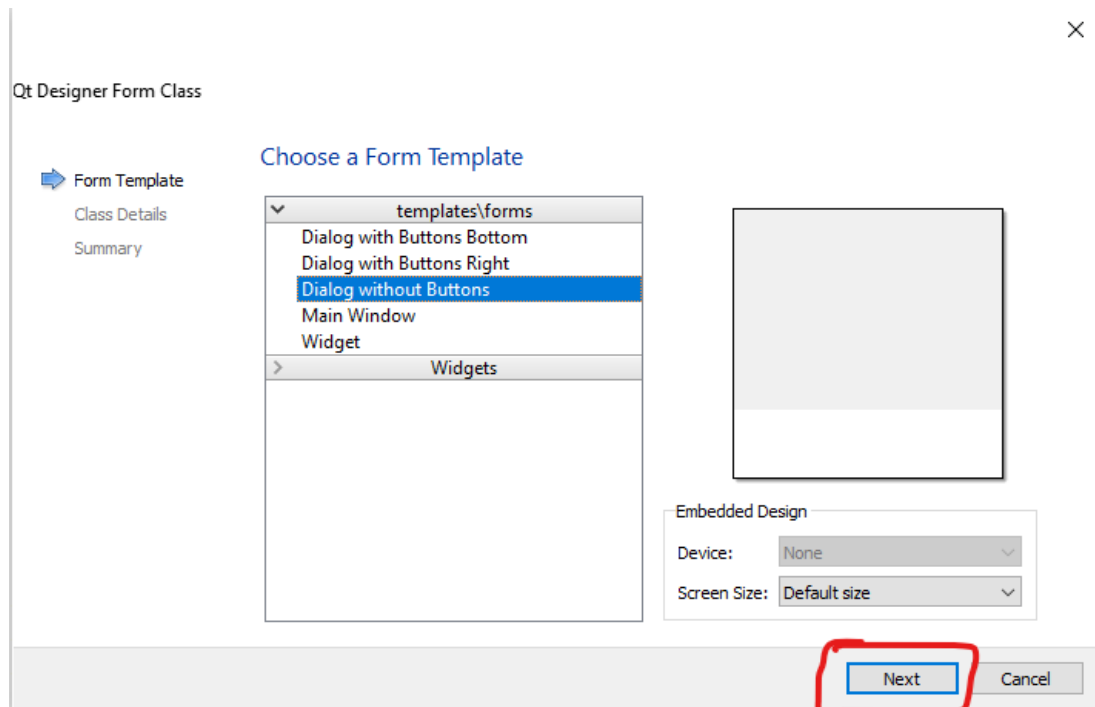
No arquivo principal.h atualizar como demonstrado na imagem:

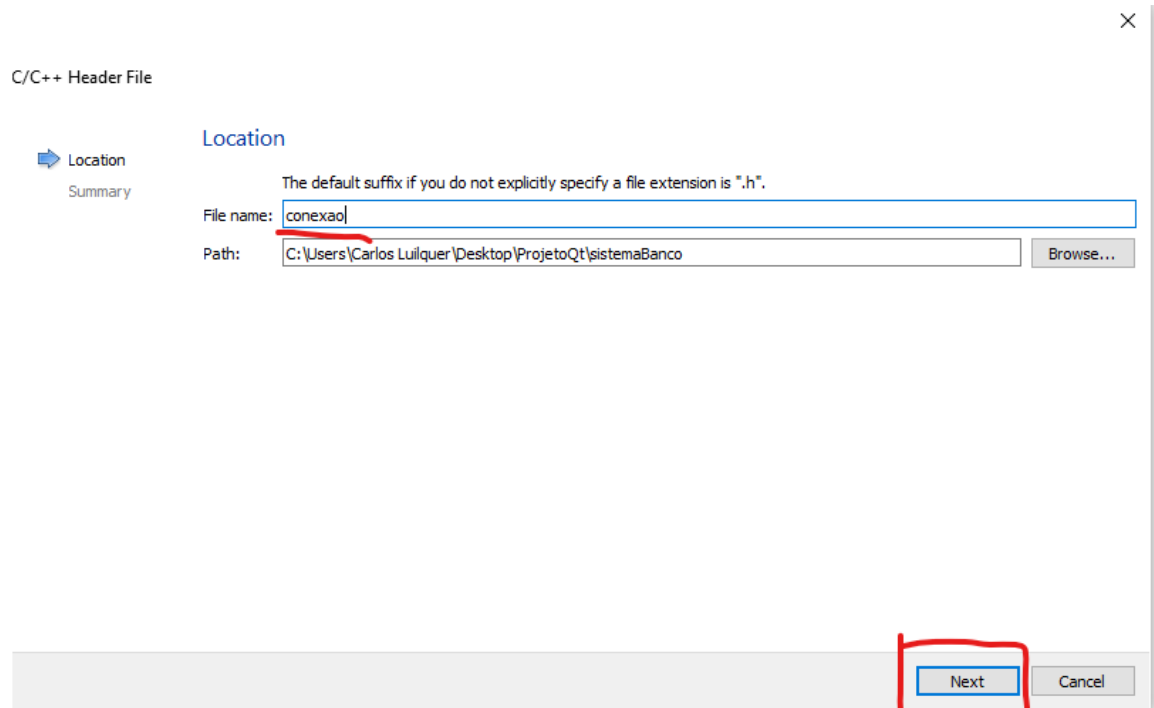
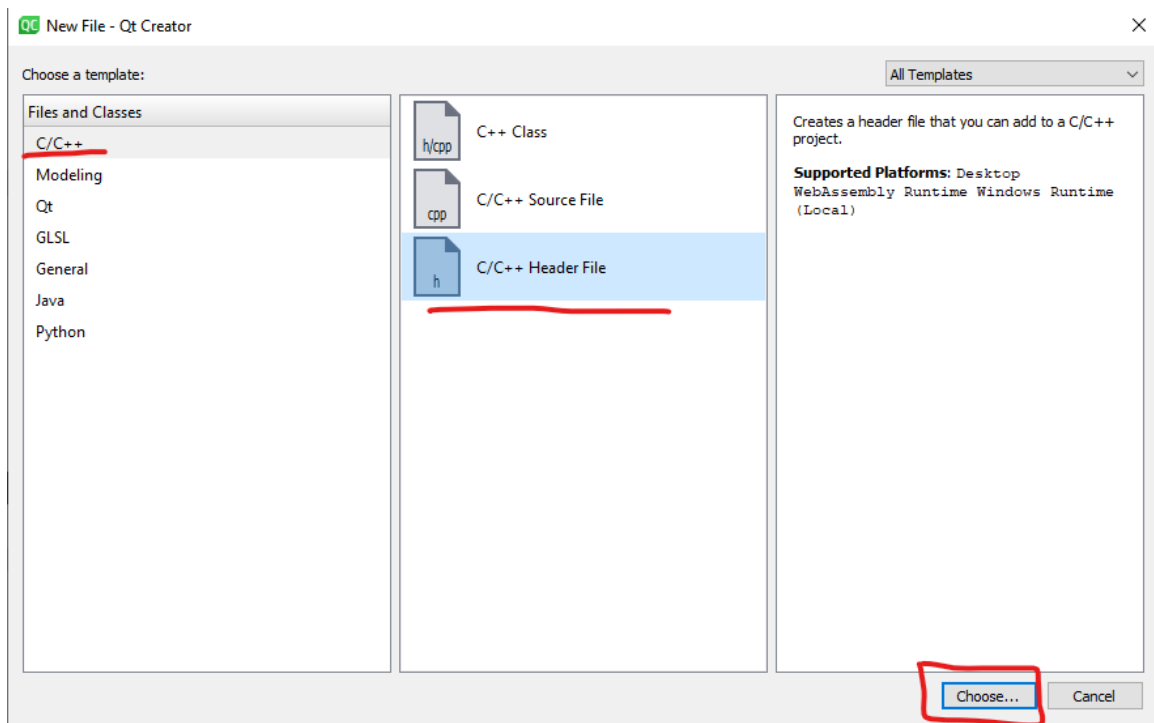


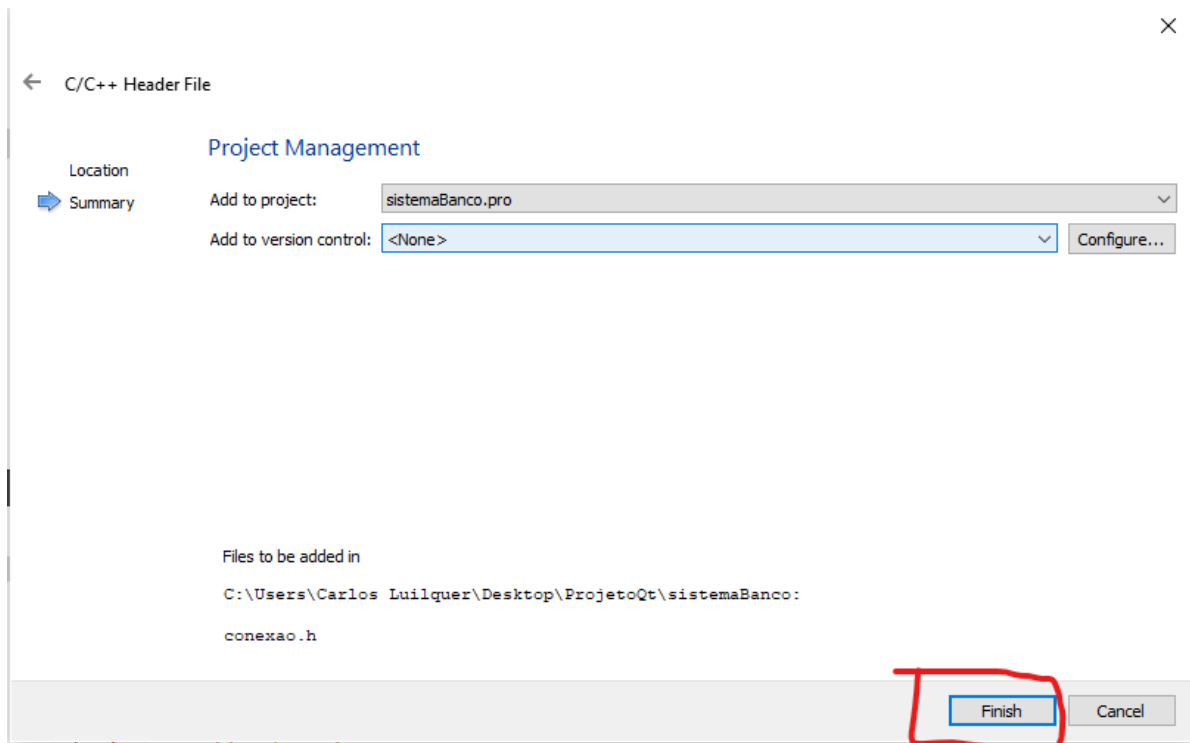
Seguir os passos abaixo para criar uma janela **login**:

- 1) Clicar com o botão direito do mouse no arquivo do **sistemaBanco** ir em **Add New**, escolher as opções como demonstrado nas imagens.









Inserir o seguinte código no arquivo.h

```
conexao.h
#include <QtSql>

class Conexao{
public:
    QSqlDatabase bancoDeDados;
    QString banco;

    //conexão com o banco de dados
    Conexao() {
        //endereço do arquivo banco de dados
        banco = "/Users/Carlos Luilquer/Desktop/Sistema de bebida/build-
ControlEstoque-Desktop_Qt_5_15_0_clang_64bit-Debug/db/bancoProjeto.db";

        //nome do programa para conexão
        bancoDeDados=QSqlDatabase::addDatabase("SQLITE");
    }

    void fechar() {
        bancoDeDados.close();
    }

    bool abrir() {
        bancoDeDados.setDatabaseName(banco);

        if(!bancoDeDados.open())
        {
            return false;
        }else{
            return true;
        }
    }
}
```



```

    }
}

bool aberto() {
    if(bancoDeDados.isOpen()) {

        return true;

    }else{

        return false;

    }
}

};

```

Inserir o seguinte código em `logar.h`

```

logar.h
#include "conexao.h"

namespace Ui {
class logar;
}

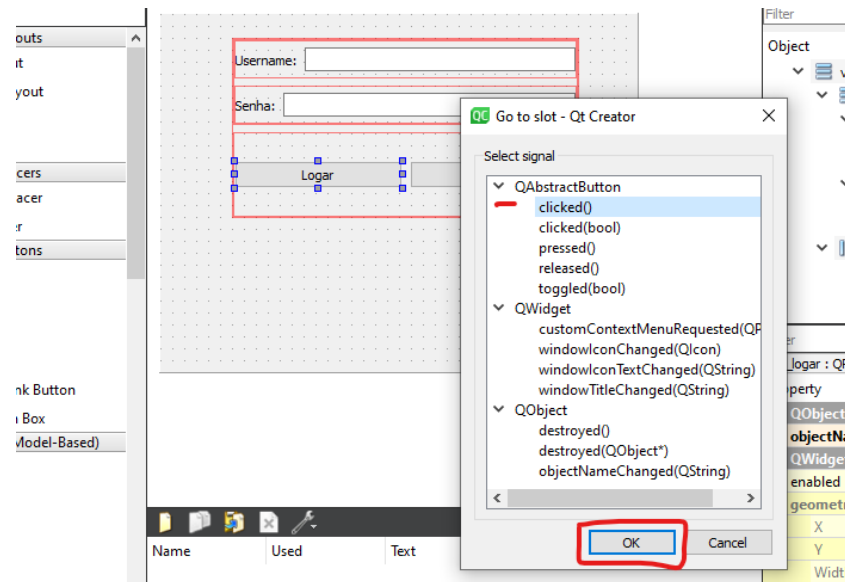
class logar : public QDialog
{
    Q_OBJECT

public:
    explicit logar(QWidget *parent = nullptr);
    ~logar();
    bool logado;
    Conexao con;
    QString nome, acesso;
    int id;

private:
    Ui::logar *ui;
};

```

Com o botão direito do mouse selecionar a opção abaixo:



Em seguida inserir o código:

logar.cpp

```
#include <QMessageBox>
#include "principal.h"

logar::logar(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::logar)
{
    ui->setupUi(this);
    logado=false;
}

logar::~logar()
{
    delete ui;
}

void logar::on_btn_logar_clicked()
{
    if(!con.abrir()){
        QMessageBox::warning(this, "Erro", "Erro ao abrir o banco de dados");
    }else{
        QString username, senha;
        username=ui->txt_username->text();
        senha=ui->txt_senha->text();
        QSqlQuery query;
        query.prepare("select * from Usuario where username='"+username+"' and
senha='"+senha+"'");

        if(query.exec()){
            query.first();
            if(query.value(1).toString()!=""){
                logado=true;
                nome=query.value(1).toString();
                id=query.value(0).toInt();
                con.fechar();
                close();
            }else{
                QMessageBox::warning(this, "Erro", "Usuario não encontrado");
            }
        }
    }
}
```

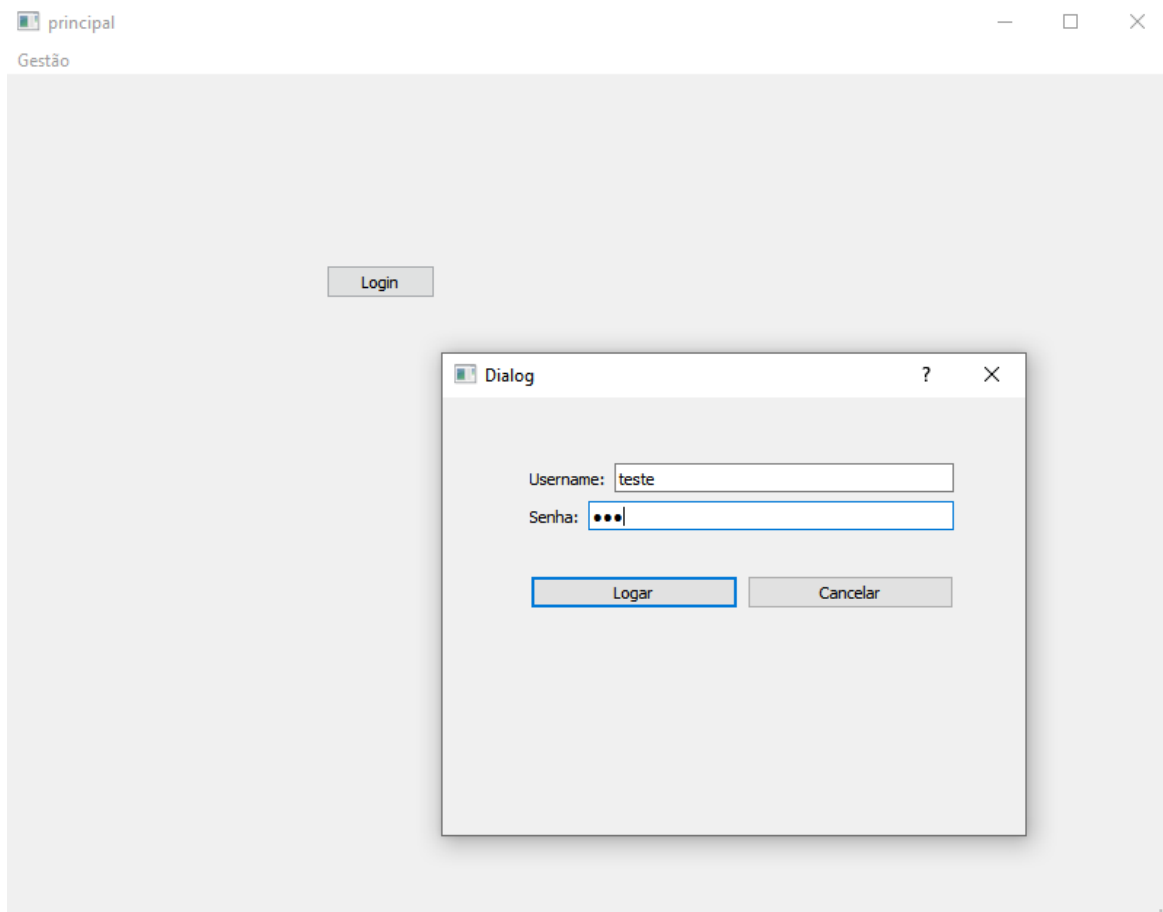
```

    }
    }else{
        QMessageBox::warning(this, "Erro", "Falha no login");
    }
}
//Login
con.fechar();
}

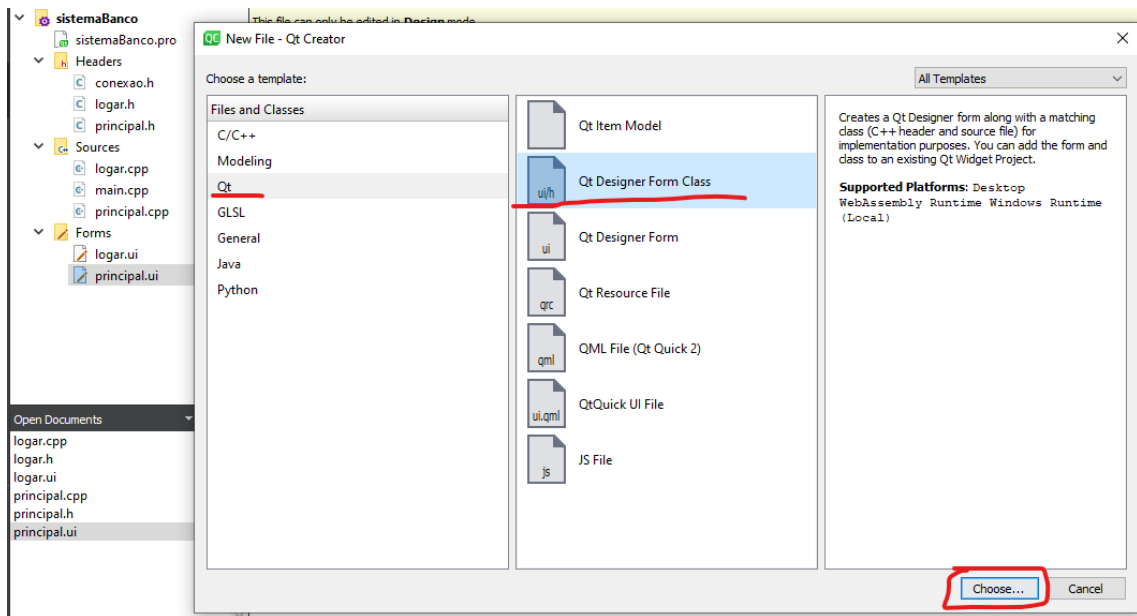
void logar::on_btn_cancelar_clicked()
{
    logado=false;
    close();
}

```

Após todos os processos o programa ficará com a seguinte cara:



Para cada tabela criar uma janela com os respectivos arquivos:



← Qt Designer Form Class

Form Template
➡ Class Details
Summary

Choose a Class Name

Class

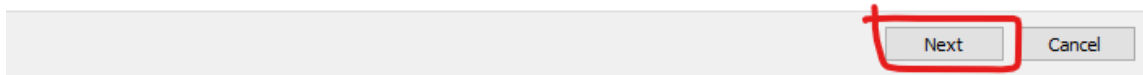
Class name: usuario

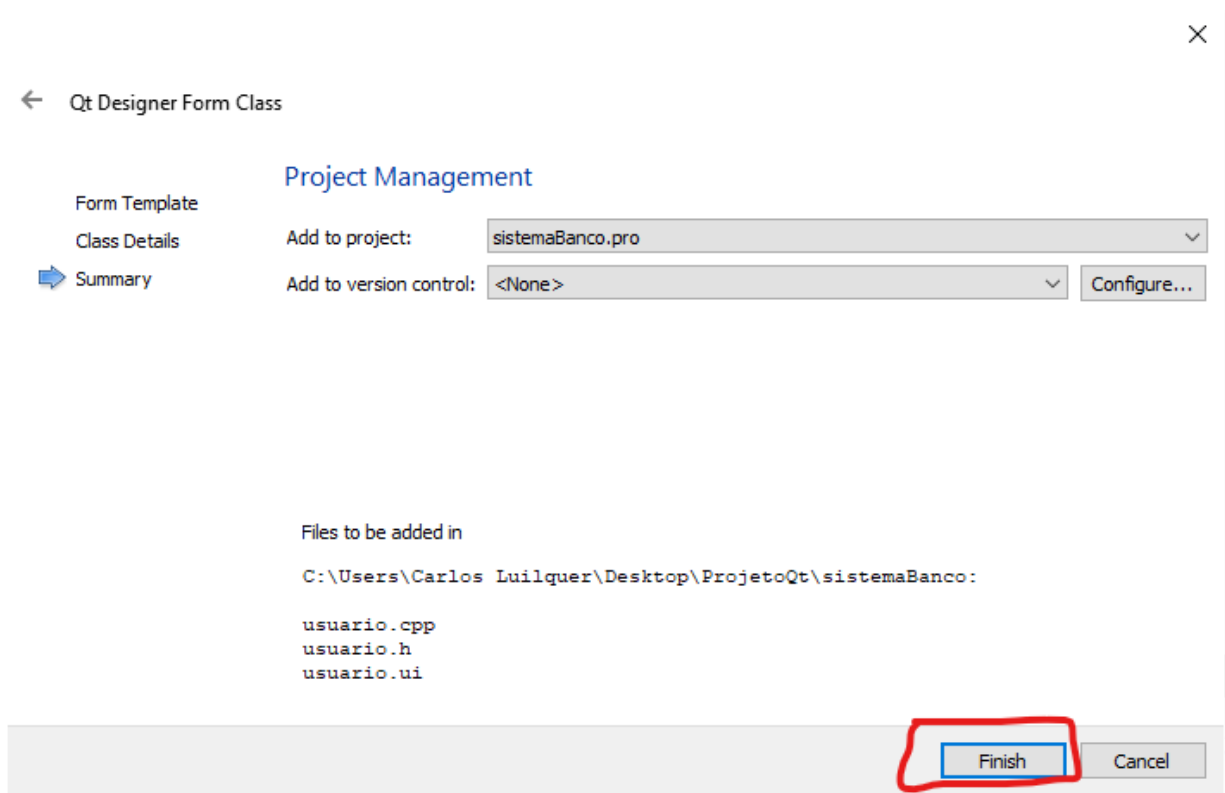
Header file: usuario.h

Source file: usuario.cpp

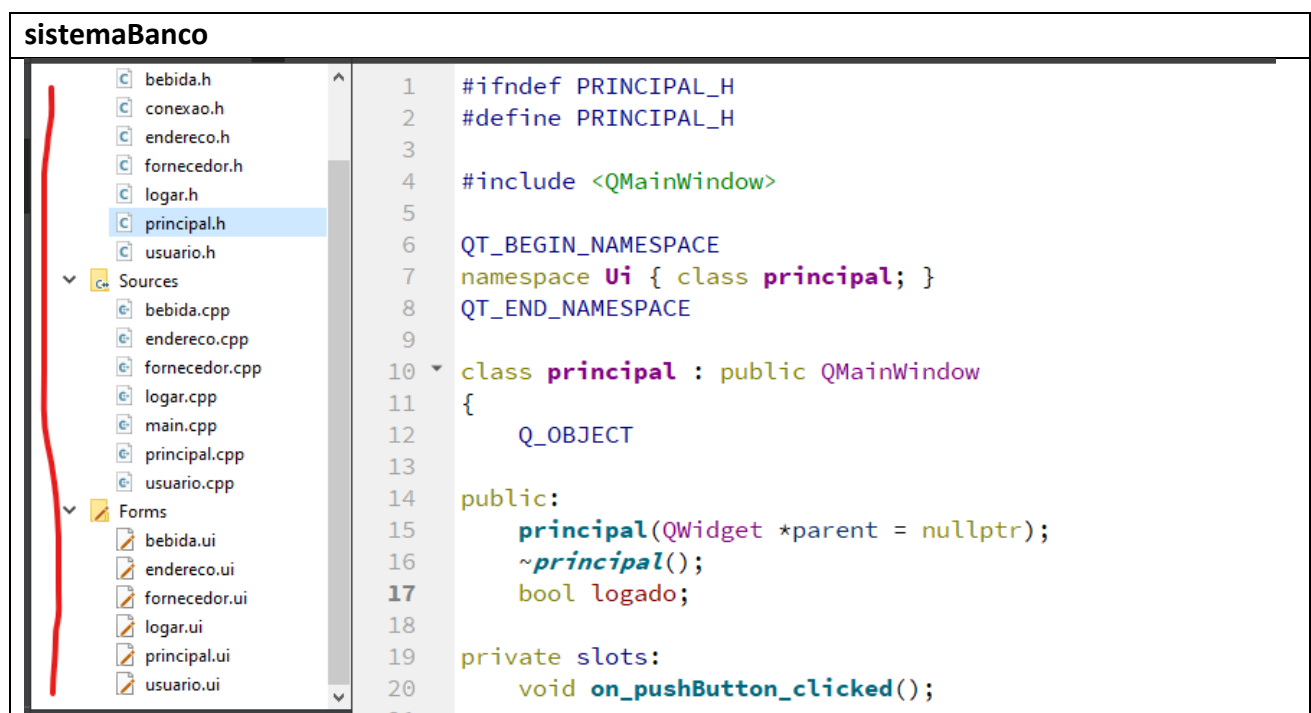
Form file: usuario.ui

Path: C:\Users\Carlos Luilquer\Desktop\ProjetoQt\sistemaBanco Browse...

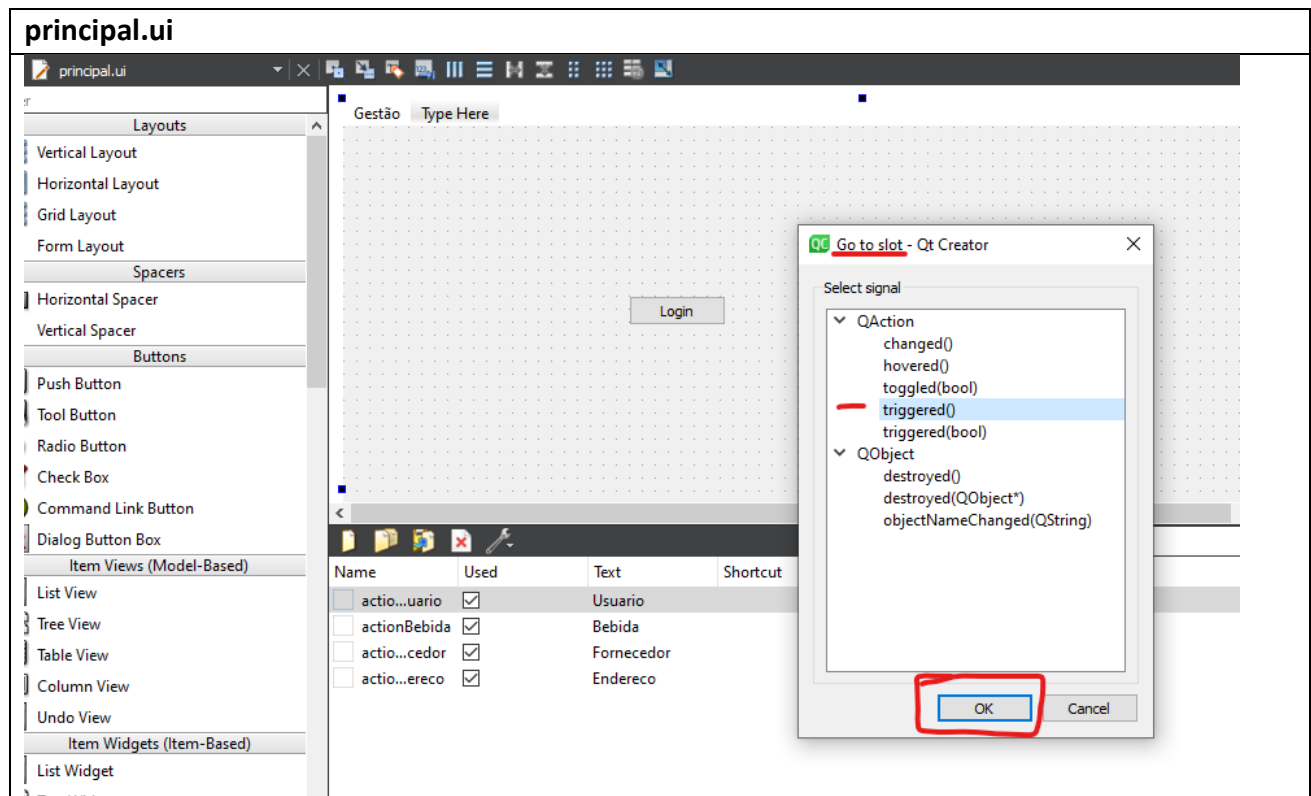




Depois de criar todos os arquivos correspondentes será possível visualizar cada um na aba ao lado, como demonstrado na imagem abaixo:



Em **principal.ui** selecionar a opção clicar com o botão direito e seguir as instruções para cada tabela.



Inserir o código no arquivo.cpp:

```
principal.cpp

#include "principal.h"
#include "ui_principal.h"
#include <QMessageBox>
#include "logar.h"
#include "usuario.h"
#include "bebida.h"
#include "fornecedor.h"
#include "endereco.h"

principal::principal(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::principal)
{
    ui->setupUi(this);
    logado=false;
}

principal::~principal()
{
    delete ui;
}

void principal::on_pushButton_clicked()
{
    if(!logado) {
        //chamar tela tela desbloqueio
    }
}
```

```

        logar f_logar;
        f_logar.exec();
        logado=true;

    }else{
        logado=false;
    }
}

//-----Usuario-----
void principal::on_actionUsuario_triggered()
{
    //verifica se existe Usuario logado
    if(logado){

        //cria um objeto do tipo
        usuario f_gestaousuario;
        //executa a janela
        f_gestaousuario.exec();

        //se for inserir dados que não constam no banco de dados
    }else{
        QMessageBox::information(this, "Login", "Não existe Usuario logado");
    }
}

//-----Bebida-----
void principal::on_actionBebida_triggered()
{
    if(logado){
        bebida f_gestaobebida;
        f_gestaobebida.exec();
    }else{
        QMessageBox::information(this, "Login", "Não existe Usuario logado");
    }
}

//-----Fornecedor-----
void principal::on_actionFornecedor_triggered()
{
    if(logado){

        fornecedor f_gestaofornecedor;
        f_gestaofornecedor.exec();

    }else{
        QMessageBox::information(this, "Login", "Não existe Usuario logado");
    }
}

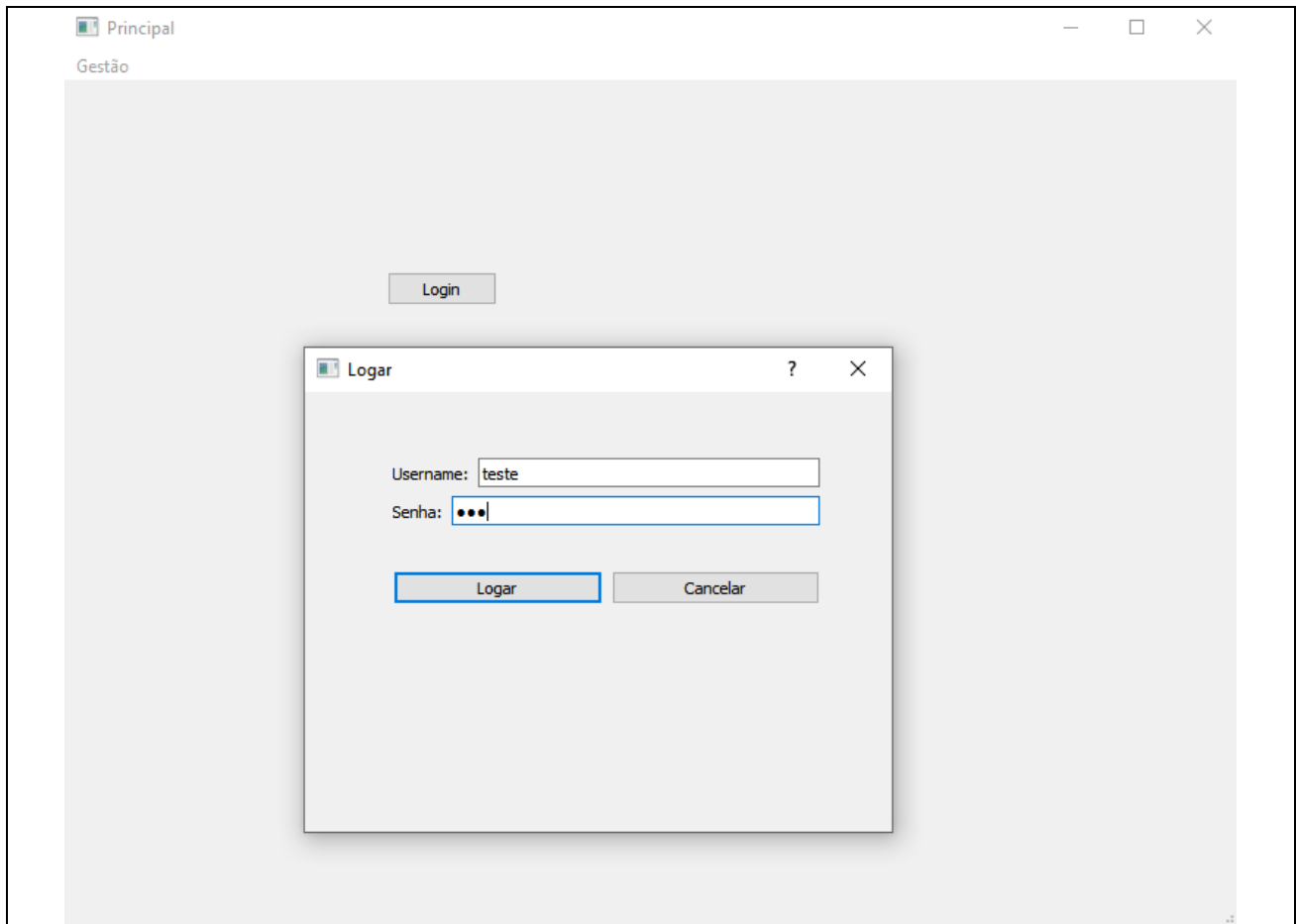
//-----Endereco-----
void principal::on_actionEndereco_triggered()
{
    if(logado){

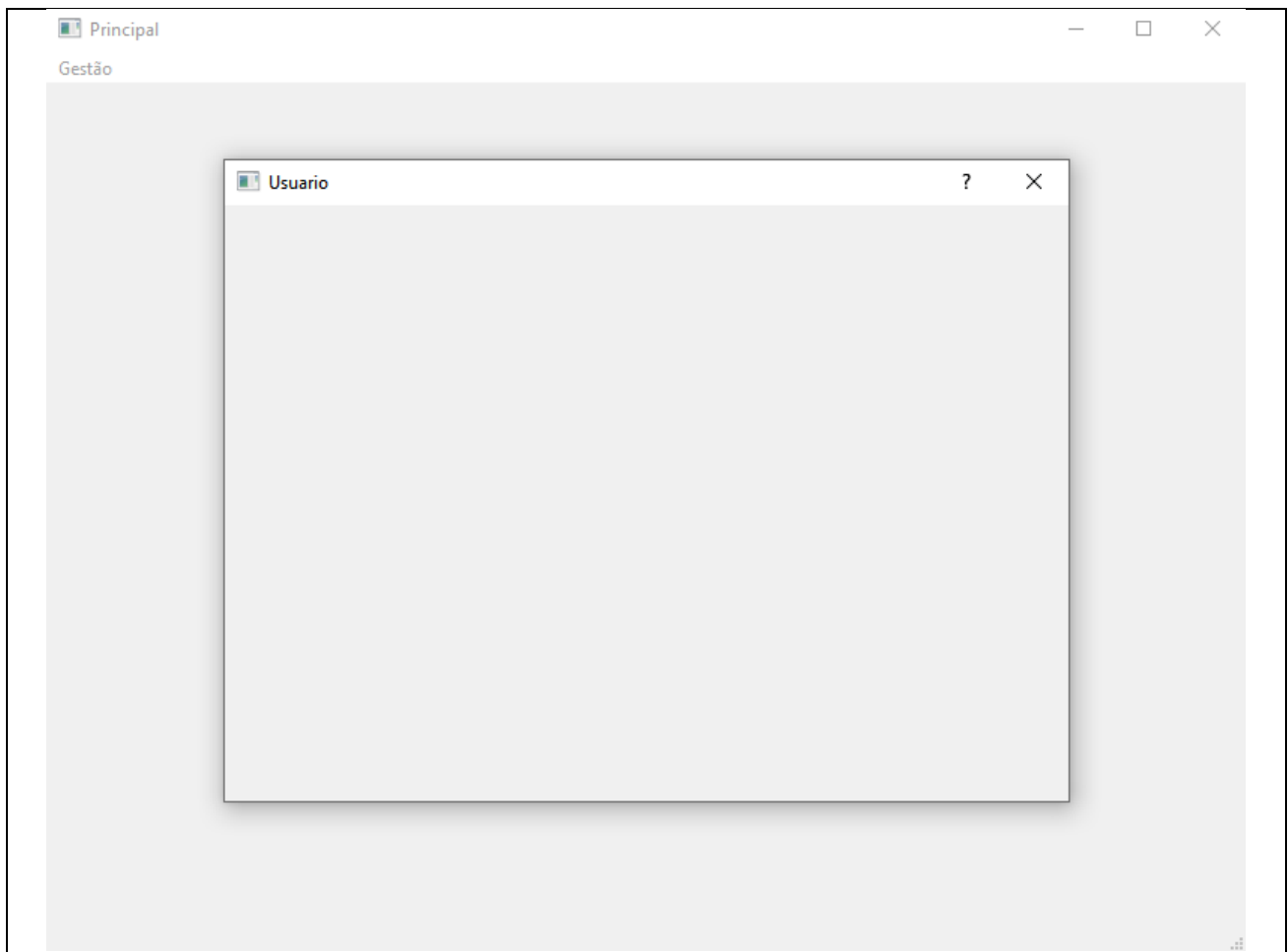
        Endereco f_gestaoEndereco;
        f_gestaoEndereco.exec();
    }
}

```

```
}else{  
    QMessageBox::information(this, "Login", "Não existe Usuario logado");  
}  
}
```

Após isso o programa fica com a seguinte aparência:





Em **Forms** configurar a interface para cada tabela como nas imagens abaixo e renomear cada campo respectivamente.

usuario.ui

Filter

Object	Class
	QLabel
	QLineEdit
▼	QVBoxLayout
	QLabel
	QLineEdit
▼	QHBoxLayout
▼	QVBoxLayout
	QLabel
	QLineEdit
	QVBoxLayout
	Spacer
	QHBoxLayout

Filter

txt_username : QLineEdit		
Property	Value	
▼	QObject	
	objectName	txt_username
▼	QWidget	
	enabled	<input checked="" type="checkbox"/>
▼	geometry	(11, 20), 215 x 201

bebida.ui

Filter

Object	Class
	QLineEdit
▼	QVBoxLayout
	QLabel
	QLineEdit
▼	QHBoxLayout
▼	QVBoxLayout
	QLabel
	QLineEdit
	QVBoxLayout
	Spacer
▼	QHBoxLayout
	QPushButton

Filter

txt_nome : QLineEdit		
Property	Value	
▼	QObject	
	objectName	txt_nome
▼	QWidget	
	enabled	<input checked="" type="checkbox"/>

fornecedor.ui

Object	Class
▼ fornecedor	QDialog
▼ ve..._6	QVBoxLayout
...	QHBoxLayout
...	QPushButton
...	QPushButton
...	QHBoxLayout
▼ ...	QHBoxLayout
...	QVBoxLayout
▼ ...	QVBoxLayout
...	QLabel
...	QLineEdit
...	QVBoxLayout

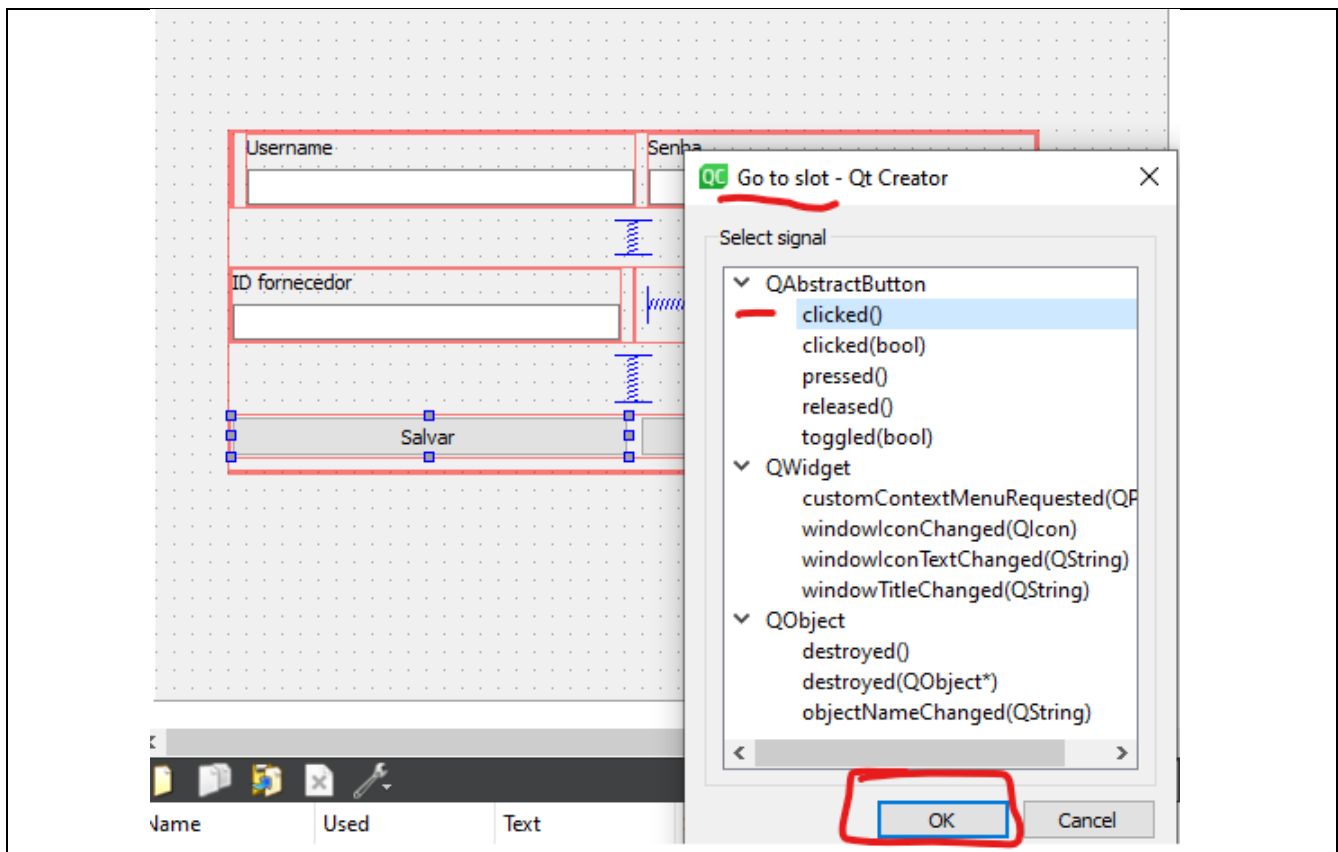
Filter	
btn_cancelar : QPushButton	
Property	Value
▼ QObject	
objectName	btn_cancelar
▼ QWidget	
enabled	<input checked="" type="checkbox"/>

endereco.ui

Object	Class
▼ ...	QVBoxLayout
...	QLabel
...	QLineEdit
▼ ...	QHBoxLayout
▼ ...	QVBoxLayout
...	QLabel
...	QLineEdit
...	QVBoxLayout
...	Spacer
▼ ...	QHBoxLayout
...	QPushButton
...	QPushButton

Filter	
btn_salvar : QPushButton	
Property	Value
▼ QObject	
objectName	btn_salvar
▼ QWidget	

Gerar o comando em todos os **Push Button** e selecionar **Go to slot > clicked()** como demonstrado abaixo:



Nos arquivos.h (**usuario.h**, **bebida.h**, **endereco.h**, **fornecedor.h**) fazer as seguintes operações:

usuario.h

```
#include <QDialog>
#include <QObject>
#include "conexao.h"

namespace Ui {
class usuario;
}

class usuario : public QDialog
{
    Q_OBJECT

public:
    explicit usuario(QWidget *parent = nullptr);
    ~usuario();
    Conexao con;

private slots:
    void on_btn_salvar_clicked();

    void on_btn_cancelar_clicked();

private:
    Ui::usuario *ui;
};
```

Em seguida atualizar os arquivos.cpp (**usuario.cpp**, **bebida.cpp**, **endereco.cpp**, **fornecedor.cpp**):

usuario.cpp

```
#include <QMessageBox>
#include <QtSql>

usuario::usuario(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::usuario)
{
    ui->setupUi(this);
    //abrir o banco de dados
    if(!con.aberto()){
        if(!con.abrir()){
            QMessageBox::warning(this, "ERROR", "Erro ao abrir banco de Dados");
        }
    }
}

usuario::~usuario()
{
    delete ui;
}

void usuario::on_btn_salvar_clicked()
{
    QString username=ui->txt_username->text();
    QString senha=ui->txt_senha->text();
    QString id=ui->txt_id->text();

    //INSERIR NA TABELA
    QSqlQuery query;
    query.prepare("insert into Usuario (username,senha,id_fornecedor) values"
                  "('" +username+"','"+senha+"','"+id+"')");

    if(!query.exec()){
        QMessageBox::critical(this, "ERRO", "Erro ao inserir no Banco de
Dados");
    }else{
        QMessageBox::information(this, "GRAVADO", "inserido no Banco de Dados");
        //limpar todos os campos e posicipn
        ui->txt_username->clear();
        ui->txt_senha->clear();
        ui->txt_id->clear();
        ui->txt_username->setFocus();
    }
}

void usuario::on_btn_cancelar_clicked()
{
    //limpar todos os campos e posicipn
    ui->txt_username->clear();
    ui->txt_senha->clear();
    ui->txt_id->clear();
    ui->txt_username->setFocus();
}
```

bebida.cpp

```
#include <QMessageBox>
#include <QtSql>

bebida::bebida(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::bebida)
{
    ui->setupUi(this);
    //abrir o banco de dados
    if(!con.aberto()){
        if(!con.abrir()){
            QMessageBox::warning(this, "ERROR", "Erro ao abrir banco de Dados");
        }
    }
}

bebida::~bebida()
{
    delete ui;
}

void bebida::on_btn_salvar_clicked()
{
    QString nome=ui->txt_nome->text();
    QString valor=ui->txt_valor->text();
    QString id=ui->txt_id->text();

    //INSERIR NA TABELA
    QSqlQuery query;
    query.prepare("insert into Bebida (nome,valor,id_fornecedor) values"
        "('"+nome+"','"+valor+"','"+id+"'");

    if(!query.exec()){
        QMessageBox::critical(this, "ERRO", "Erro ao inserir no Banco de
Dados");
    }else{
        QMessageBox::information(this, "GRAVADO", "inserido no Banco de Dados");
        //limpar todos os campos e posicipn
        ui->txt_nome->clear();
        ui->txt_valor->clear();
        ui->txt_id->clear();
        ui->txt_nome->setFocus();
    }
}

void bebida::on_btn_cancelar_clicked()
{
    //limpar todos os campos e posicipn
    ui->txt_nome->clear();
    ui->txt_valor->clear();
    ui->txt_id->clear();
    ui->txt_nome->setFocus();
}
```

endereco.cpp

```
#include <QMessageBox>
```

```

#include <QtSql>

Endereco::Endereco(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Endereco)
{
    ui->setupUi(this);
    //abrir o banco de dados
    if(!con.aberto()){
        if(!con.abrir()){
            QMessageBox::warning(this, "ERROR", "Erro ao abrir banco de Dados");
        }
    }
}

Endereco::~Endereco()
{
    delete ui;
}

void Endereco::on_btn_salvar_clicked()
{
    QString cidade=ui->txt_cidade->text();
    QString uf=ui->txt_uf->text();
    QString id=ui->txt_id->text();

    //INSERIR NA TABELA
    QSqlQuery query;
    query.prepare("insert into Endereco (cidade,uf,id_fornecedor) values"
        " ('"+cidade+"','"+uf+"','"+id+"')");

    if(!query.exec()){
        QMessageBox::critical(this, "ERRO", "Erro ao inserir no Banco de
Dados");
    }else{
        QMessageBox::information(this, "GRAVADO", "inserido no Banco de Dados");
        //limpar todos os campos e posicipn
        ui->txt_cidade->clear();
        ui->txt_uf->clear();
        ui->txt_id->clear();
        ui->txt_cidade->setFocus();
    }
}

void Endereco::on_btn_cancelar_clicked()
{
    //limpar todos os campos e posicipn
    ui->txt_cidade->clear();
    ui->txt_uf->clear();
    ui->txt_id->clear();
    ui->txt_cidade->setFocus();
}

```

fornecedor.cpp

```

#include <QMessageBox>
#include <QtSql>

```

```

fornecedor::fornecedor(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::fornecedor)
{
    ui->setupUi(this);
    //abrir o banco de dados
    if(!con.aberto()){
        if(!con.abrir()){
            QMessageBox::warning(this, "ERROR", "Erro ao abrir banco de Dados");
        }
    }
}

fornecedor::~fornecedor()
{
    delete ui;
}

void fornecedor::on_btn_salvar_clicked()
{
    QString nome=ui->txt_nome->text();

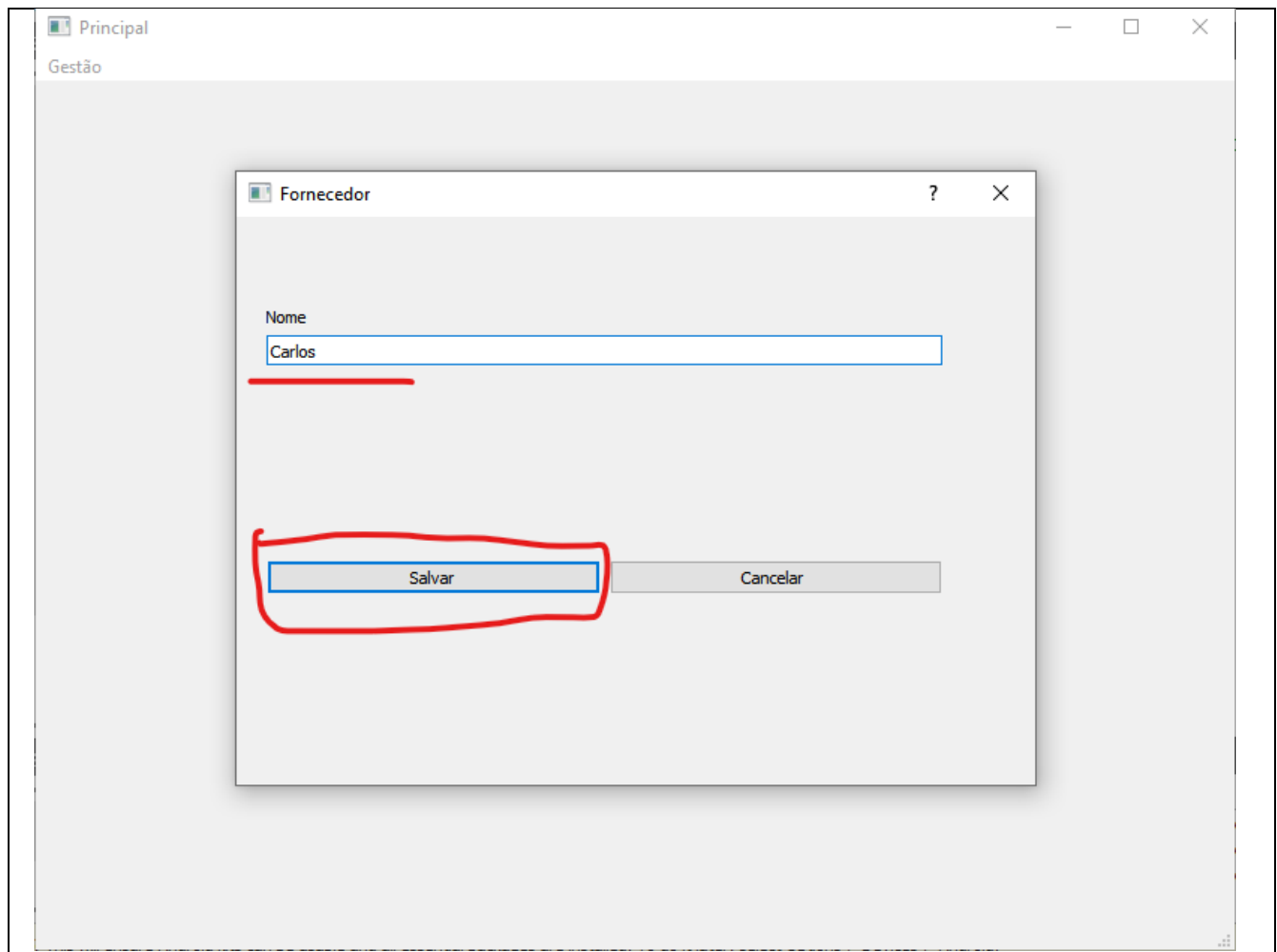
    //INSERIR NA TABELA
    QSqlQuery query;
    query.prepare("insert into Fornecedor (nome) values"
                  "('" + nome + "')");

    if(!query.exec()){
        QMessageBox::critical(this, "ERRO", "Erro ao inserir no Banco de
Dados");
    }else{
        QMessageBox::information(this, "GRAVADO", "inserido no Banco de Dados");
        //limpar todos os campos
        ui->txt_nome->clear();
        ui->txt_nome->setFocus();
    }
}

void fornecedor::on_btn_cancelar_clicked()
{
    //limpar todos os campos
    ui->txt_nome->clear();
    ui->txt_nome->setFocus();
}

```

Feito isso podemos compilar e executar o nosso projeto. Assim será possível inserir em cada tabela:



Verificando no **SQLiteStudio** percebe-se que foi salvo com sucesso:

