

## 6.2 Desenvolvimento do projecto Tarefas para VC++

Modelar e codificar um sistema informático que possibilite ao utilizador introduzir e remover tarefas. A introdução consiste em três dados:

Prioridade : Identifica o nível de prioridade para executar a tarefa;

Descrição : Uma breve descrição do que consiste a tarefa;

Data Execução : Data em que deverá ser executada a tarefa.

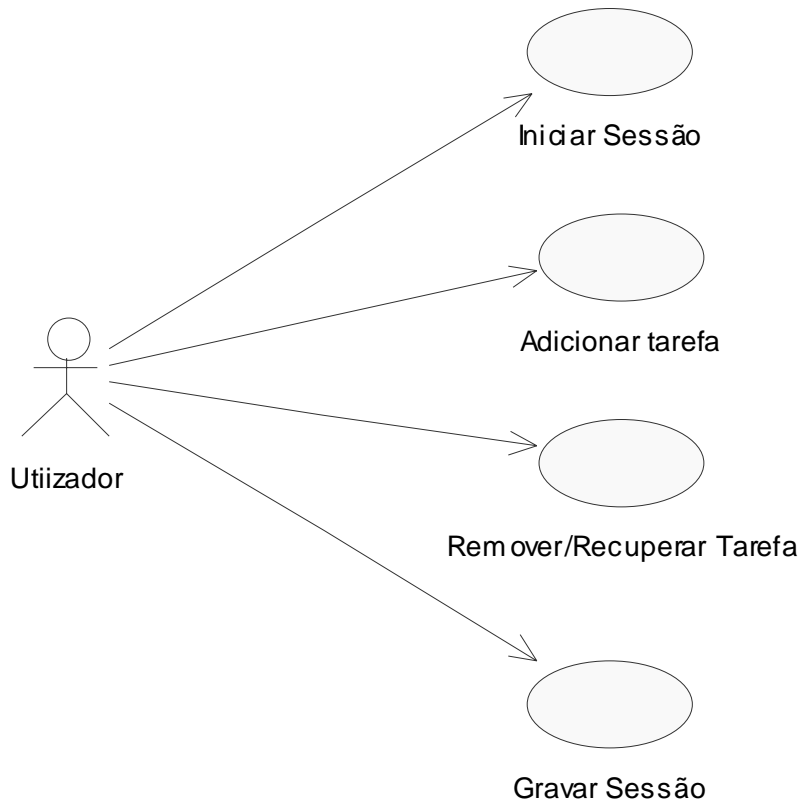
O sistema deverá sempre que seja criada uma tarefa, atribuir um identificador único e a data em que foi criada a tarefa.

A aplicação deverá permitir que só no fim da sessão o utilizador decida se o seu trabalho é gravado para posterior utilização ou não.

Durante a sessão de trabalho se o utilizador remover alguma tarefa deverá ter a possibilidade de a recuperar, sem que para isso tenha de cancelar todo o seu trabalho. Após a gravação da sessão deixa de ser possível recuperar as tarefas apagadas.

Ao iniciar a sessão deverão ser mostradas ao utilizador todas as tarefas existentes por ordem de entrada no sistema.

## 6.2.1 Definir os use cases, os actors e o protótipo



### Use Case Iniciar Sessão:

Este use case começa quando o utilizador entra no sistema e automaticamente é iniciada uma sessão de trabalho. São mostradas ao utilizador todas as tarefas existentes por ordem de criação. Por cada tarefa é mostrado:

Identificador;

Prioridade da tarefa;

Descrição da tarefa;

Data de execução.

### Use Case Adicionar Tarefa:

Este use case começa quando o utilizador decide introduzir uma nova tarefa. Para isso introduz a prioridade, a descrição da tarefa e a data de execução. Em seguida o utilizador selecciona o botão adicionar.

O sistema não faz qualquer validação aos dados introduzidos, mas atribui um identificador único e uma data de execução. Após a introdução da tarefa esta é visualizada juntamente com as restantes por ordem de entrada.

## Use Case Remover/Recuperar

Este use case começa quando o utilizador decide remover uma tarefa. O utilizador selecciona uma tarefa da lista e de seguida carrega no botão Remover/Recuperar, se a tarefa já estava removida então será recuperada, senão será removida.

Se não for seleccionada nenhuma tarefa então nada acontecerá.

## Use Case Gravar Sessão

Este use case começa quando o utilizador decide gravar todo o trabalho que fez durante a sessão.

Todo o seu trabalho será guardado numa base de dados para posterior utilização: As novas tarefas serão guardadas e as apagadas serão removidas fisicamente da base de dados.

## Actor Utilizador

O utilizador é a pessoa que pode executar todas as opções relacionadas com as tarefas, é o único perfil do sistema.

## Protótipo

ID	Prioridade	Descrição	Data Execução	Apagada
1	1	Executar tarefa X	22/01/2001	0
2	2	Executar tarefa Y	23/01/2001	0
3	6	Executar tarefa Z	23/01/2000	0

## 6.2.2 Definir as classes genéricas

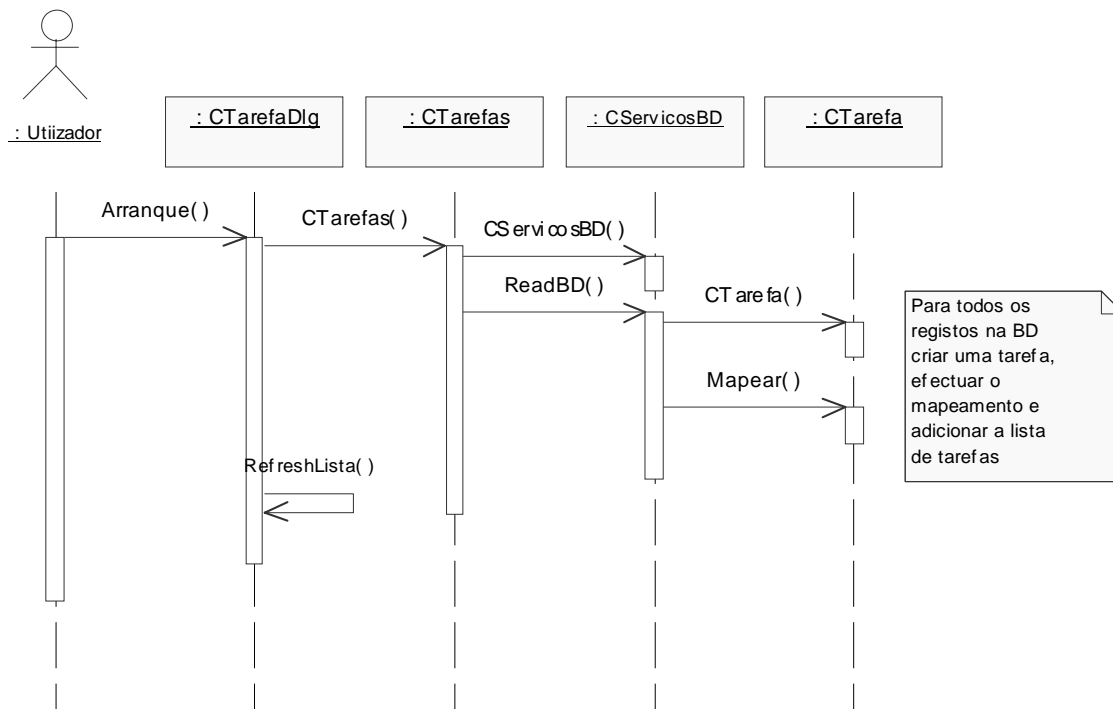
As classes genéricas são as CTarefa, CTarefas e CServicosBD, é necessário mais uma que é o form de interacção com o utilizador que vai ser feita no próximo ponto.

## 6.2.3 Associar um projecto Visual C++ a um modelo Rose

- a. Arrancar com o Visual C++ :
  - I. Criar projecto MFC AppWizard(exe) chamado Tarefa;
  - II. Seleccionar Dialog Based e Finish;
- b. Voltar ao Rose e efectuar reverse engineer, para isso:
  - I. Tools / Visual C++ / Update Model From Code;
  - II. Add Component;
  - III. Assign do projecto Tarefa ao modelo;
  - IV. Seleccionar o projecto Tarefa e finish;
- c. Arrastar a classe CTarefaDlg para junto das outras classes
- d. Assign das três classes anteriores ao projecto Visual C++:
  - I. Tools / Visual C++ / Component Assignment Tool;
  - II. Arrastar as 3 classes para cima do projecto Tarefa;
  - III. Seleccionar OK;
- e. Para cada uma das três classes:
  - I. Click com o botão direito do rato em cima da classe e seleccionar a opção Model Assistant;
  - II. No Model Assistant seleccionar o constructor, pois vai ser necessário nos diagramas de sequência;

## 6.2.4 Definir os diagramas de sequência

### Diagrama de sequência Iniciar Sessão

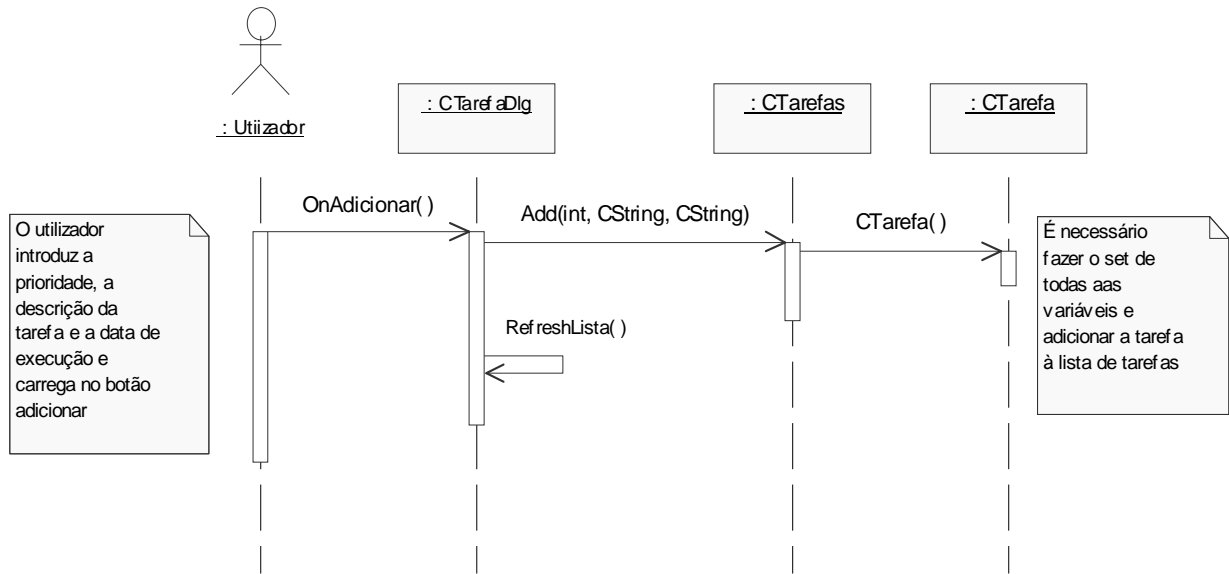


Neste diagrama foram definidas novas operações e associações:

Classe	Operações	Associações
CTarefaDlg	Arranque RefreshLista	CTarefas
CTarefas		CServicosBD
CServicosBD	ReadBD	CTarefa
CTarefa	Mapear	

Nota: O método Mapear tem como parâmetro Rs:\_RecordsetPtr

### Diagrama de sequência Adicionar Tarefa

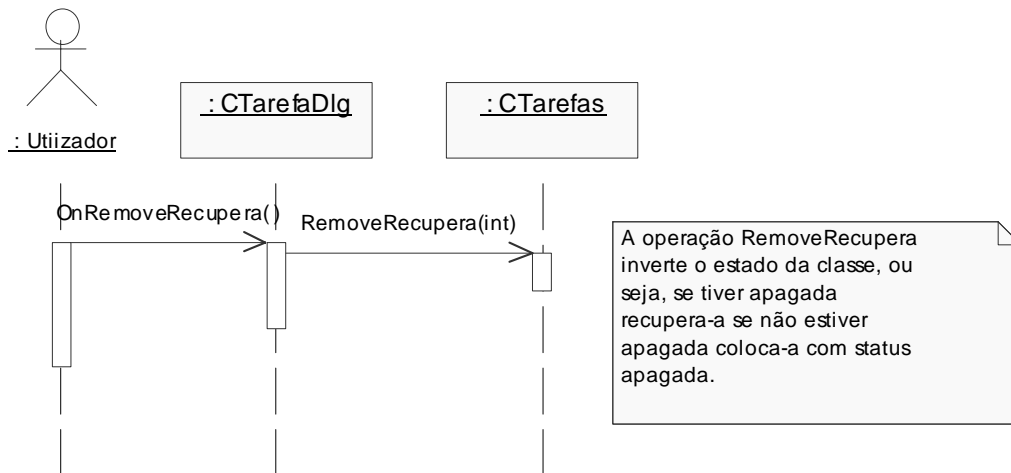


Neste diagrama foram definidas novas operações e associações:

Classe	Operações	Associações
CTarefaDlg	OnAdicionar	
CTarefas	Add	CTarefa

Nota: a classe Add retorna uma CTarefa

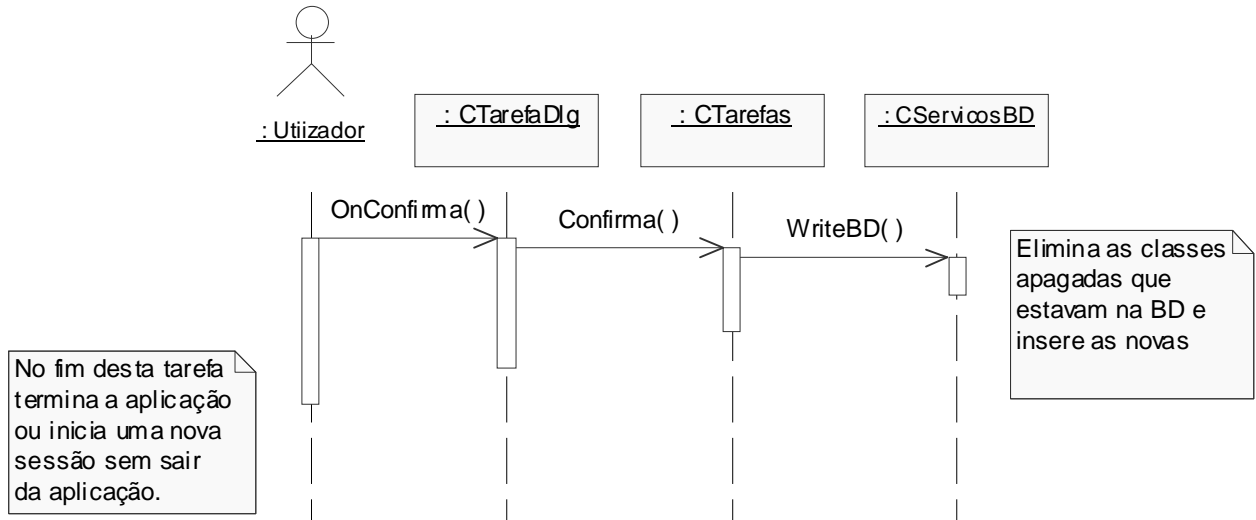
### Diagrama de sequência Remove/Recupera Tarefa



Neste diagrama foram definidas novas operações e associações:

Classe	Operações	Associações
CTarefaDlg	OnRemoveRecupera	
CTarefas	RemoveRecupera	

### Diagrama de sequência gravação da sessão

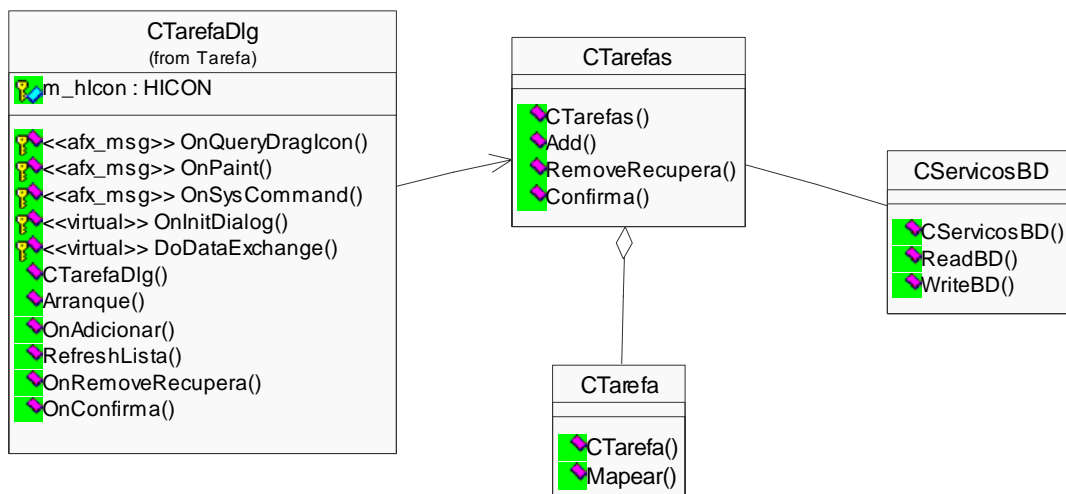


Neste diagrama foram definidas novas operações e associações:

Classe	Operações	Associações
CTarefaDlg	OnConfirma	
CTarefas	Confirma	
CServicosBD	WriteBD	

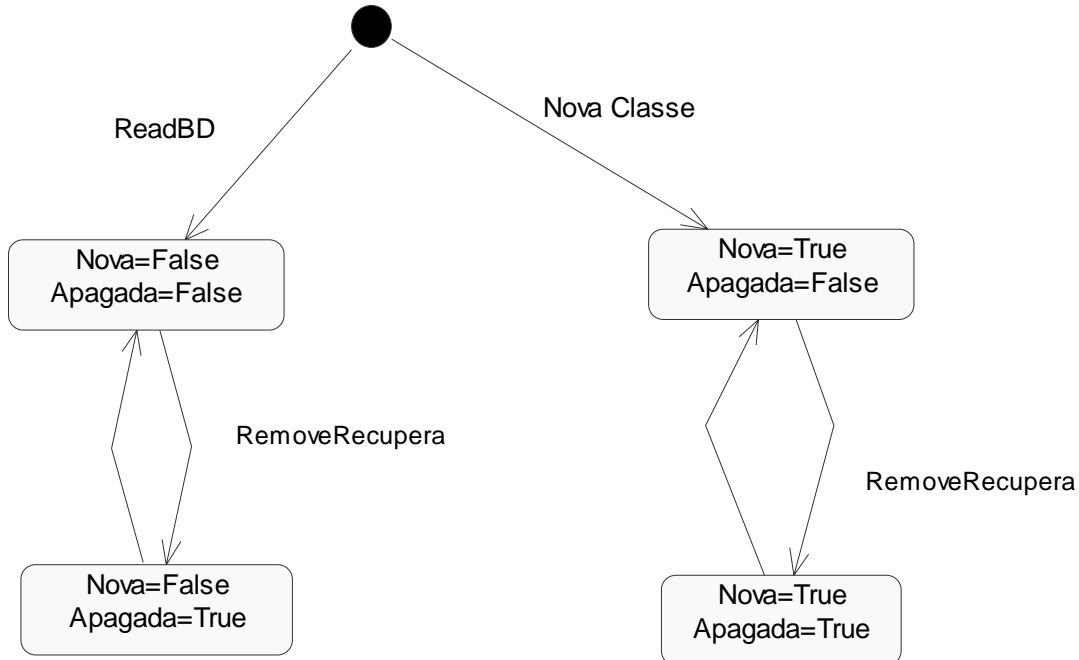
### 6.2.5 Definir as relações

Definir as relações a partir dos diagramas anteriores, colocá-las por referência, definir uma relação 1:N entre CTarefas e CTarefa, chamar-lhe theConjTarefas.



## 6.2.6 Definir diagramas de estado

### Diagrama de estados da classe CTarefa



## 6.2.7 Definir outros atributos e operações

### Classe CTarefa

Classe	Descrição
DataCriacao:CString	Atributo persistente
Descricao:CString	“
DataExecucao:CString	“
ID:long	“
Prioridade:long	“
Nova:BOOL	indica se a classe é nova, ou seja se ainda não existe na BD, dá suporte ao diagrama de estados
Apagada:BOOL	indica se a classe foi apagada, para mais tarde elimina-la da BD (se existir) ou para não a gravar (se nova).

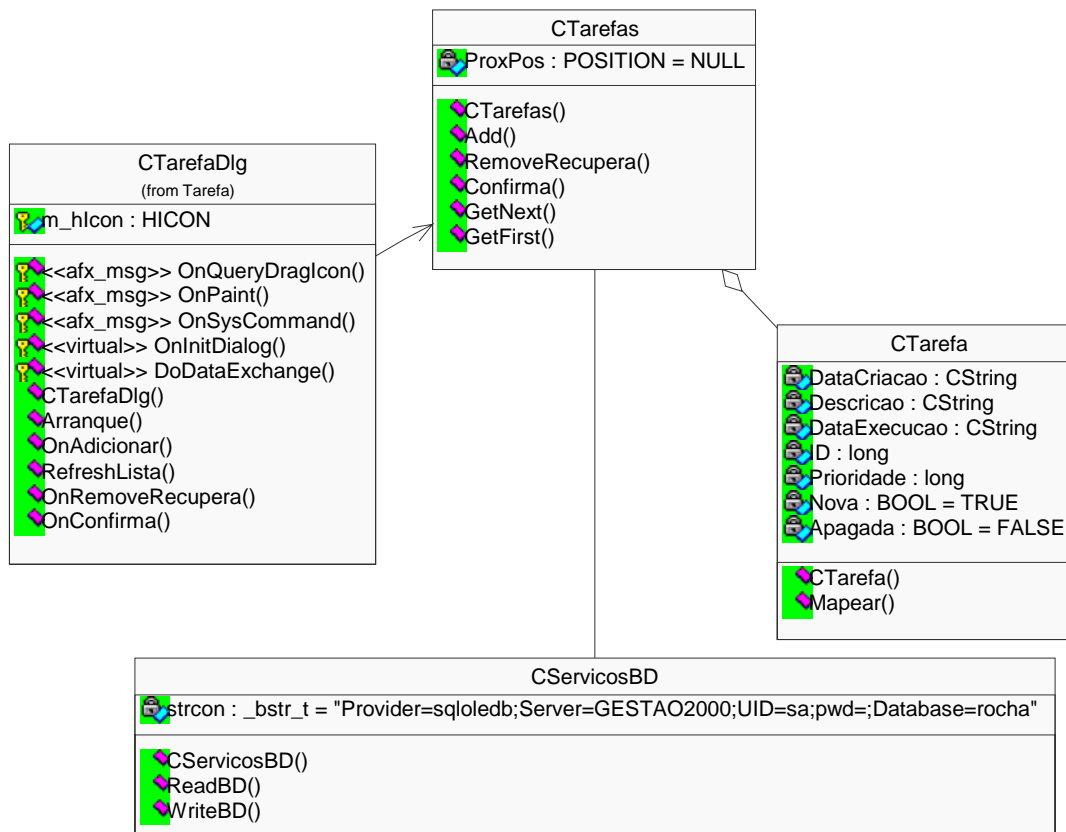
**Classe CTarefas**

Classe	Descrição
ProxPos:POSITION=NULL	Indica o próximo elemento da lista, se não existir é NULL, é útil para ciclos
GetNext():CTarefa GetFirst():CTarefa	Operações de suporte para percorrer a lista, podiam ser dispensadas pois o CList já tem este tipo de operações.
Get_IDLivre():long	Retorna um identificador sequencial para atribuir a cada nova tarefa

**Classe CServicos**

Classe	Descrição
strcon:_bstr_t	Contém a string de conexão ADO à BD

strcon:\_bstr\_t = "Provider=sqloledb;Server=sap;UID=aluno;pwd=;Database=XXX"



## 6.2.8 Definir métodos automáticos no Model Assistant

### Classe CTarefas

Seleccionar método get\_ProxPos

Definir a associação theConjTarefas como CList<CTarefa, CTarefa&>

### Class CTarefa

Seleccionar métodos

get\_Apagada e set\_Apagada

get\_DataCriacao

get\_DataExecucao e set\_DataExecucao

get\_Descricao e set\_Descricao

get\_ID e set\_ID

get\_Nova

get\_Prioridade e set\_Prioridade

## 6.2.9 Gerar código

- a. Retirar os métodos OnAdicionar OnRemoveRecupera e OnConfirmar pois vão ser criados automaticamente no VC++ ;
- b. Update Code.

## 6.2.10 Acertar projecto VC++

Voltar ao VC++

- a. No ficheiro stdafx.h colocar:

- #import "c:\program files\common files\system\ado\msado15.dll"  
rename("EOF", "EndOfFile") no\_namespace

- b. Na classe Ctarefas.GetNext e na Ctarefas.GetFirst colocar o seguinte código apenas para não dar erro de compilação:

- CTarefa AuxTarefa;
- return AuxTarefa;

- c. Compilar : Deverá não dar erros

d. Retirar o texto do ecrã e o botão de OK e Cancel

e. Criar todo ecrã

ID	Tipo	Text
Não interessa	Static text	Prioridade
Não interessa	Static text	Descrição
Não interessa	Static text	Data Execução
IDC_PRIORIDADE	Edit Box	
IDC_DESCRICA0	Edit Box	
IDC_DATA_EXEC	Edit Box	
IDC_LISTA_TAREFAS	List Box	
Não interessa	Static text	ID
Não interessa	Static text	Prioridade
Não interessa	Static text	Descrição
Não interessa	Static text	Data Execução
Não interessa	Static text	Apagada
IDC_ADICIONAR	Button	Adicionar
IDC_REMOVE_RECUPERA	Button	AdicionarRemover
IDC_GRAVAR	Button	Gravar Sessão

f. Criar variáveis

ID	Nome Variável	Categoria	Tipo
IDC_PRIORIDADE	m_Prioridade	Value	Long
IDC_DESCRICA0	m_Descricao	Value	CString
IDC_DATA_EXEC	m_DataExecucao	Value	CString
IDC_LISTA_TAREFAS	m_ListaTarefas	Control	CListBox

g. Criar métodos para o click em cima dos botões:

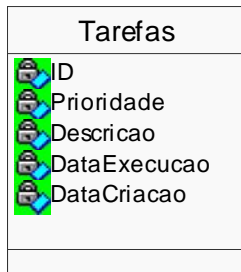
Botão	Método
IDC_ADICIONAR	OnAdicionar
IDC_REMOVE_RECUPERA	OnRemoveRecupera
IDC_GRAVAR	OnConfirma

h. Compilar e executar

i. Voltar ao Rose e actualizar o modelo em Tools/VC++/Update Modelo from Code

## 6.2.11 Criar classe para gerar DDL

- a. Criar classe Tarefas que vai simbolizar a tabela da BD



- b. Definir para cada atributo os dados DDL

Atributo	Tipo de dados	Tamanho
ID	int	
Prioridade	int	
Descricao	Varchar	50
DataExecucao	Varchar	10
DataCriacao	Varchar	10

- c. Colocar em Tools/Options/DDL tab/Project/TableName a branco  
Gerar DDL para SQL Server  
Executar o código gerado no Query Analyzer

```
CREATE TABLE Tarefas(
  ID int,
  Prioridade int,
  Descricao VARCHAR(50),
  DataExecucao VARCHAR(10),
  DataCriacao VARCHAR(10),
  PRIMARY KEY(ID))
```

## 6.2.12 Gerar documentação para WWW

É possível gerar documentação para www em Tools/Web Publisher. Para este exemplo excluir no Logic View tudo o que seja MFC 6.0, para evitar gerar documentação desnecessária.

## 6.2.13 Programação

### a. Diagrama iniciar sessão

I.No final do método CTarefaDlg::OnInitDialog colocar Arranque();

II.No método CTarefaDlg::Arranque() colocar:  
 theCTarefas = new CTarefas;  
 RefreshLista();

III.No método CTarefaDlg:: RefreshLista () colocar:  
 CTarefa AuxTarefa;  
 CString AuxStr;  
 m\_ListaTarefas.ResetContent();

```

if (theCTarefas->theConjTarefas.GetCount() > 0)
{
    AuxTarefa = theCTarefas->GetFirst();

    AuxStr.Format("%6d %10d   %-50s   %-30s   %1d",
                  AuxTarefa.get_ID(),
                  AuxTarefa.get_Prioridade(),
                  AuxTarefa.get_Descricao(),
                  AuxTarefa.get_DataExecucao(),
                  AuxTarefa.get_Apagada());

    m_ListaTarefas.AddString(AuxStr);

    while (theCTarefas->get_ProxPos() != NULL)
    {
        AuxTarefa = theCTarefas->GetNext();

        AuxStr.Format("%6d %10d   %-50s   %-30s   %1d",
                      AuxTarefa.get_ID(),
                      AuxTarefa.get_Prioridade(),
                      AuxTarefa.get_Descricao(),
                      AuxTarefa.get_DataExecucao(),
                      AuxTarefa.get_Apagada());

        m_ListaTarefas.AddString(AuxStr);
    }
    UpdateData(FALSE);
}

```

IV.No ficheiro CTarefas.h colocar #include "afxTempl.h"

V.No construtor CTarefas::CTarefas colocar:  
theCServicosBD = new CServicosBD;  
theCServicosBD->theCTarefas = this;  
theCServicosBD->ReadBD();

VI.Os métodos que trabalham com a base de dados vamos deixar para uma segunda iteração: CServicos::ReadBD(), CServicos::WriteBd() e CTarefa::Mapear()

VII.Compilar : Não deverá dar erros

### **b. Diagrama Adicionar Tarefa**

I. No método CTarefaDlg:: OnAdicionar () colocar:  
CTarefa AuxTarefa;  
UpdateData(TRUE);  
AuxTarefa = theCTarefas->Add(m\_Prioridade, m\_Descricao, m\_DataExecucao);  
RefreshLista();

II. No método CTarefas::Add() colocar:  
CTarefa AuxTarefa;  
AuxTarefa.set\_Prioridade(vPrioridade);  
AuxTarefa.set\_Descricao(vDescricao);  
AuxTarefa.set\_DataExecucao(vDataExecucao);  
AuxTarefa.set\_ID(get\_IDLivre());  
theConjTarefas.AddHead(AuxTarefa);

ProxPos = NULL;

return AuxTarefa;

III. No método CTarefas::get\_IDLivre() colocar:  
if (theConjTarefas.IsEmpty())  
return 1;  
else  
return (theConjTarefas.GetHead().get\_ID()+1);

IV. Compilar : Não deverá dar erros

### c. Definir os métodos de navegação

I. No método CTarefas:: GetFirst() colocar:

```
CTarefa AuxTarefa;
```

```
ProxPos = theConjTarefas.GetHeadPosition();
```

```
if (ProxPos != NULL)
```

```
    AuxTarefa = theConjTarefas.GetNext(ProxPos);
```

```
return AuxTarefa;
```

II. No método CTarefas:: GetNext() colocar:

```
CTarefa AuxTarefa;
```

```
if (ProxPos !=NULL)
```

```
    AuxTarefa = theConjTarefas.GetNext(ProxPos);
```

```
return AuxTarefa;
```

III. Compilar : Não deverá dar erros, e já está a funcionar o adicionar tarefa

### d. Diagrama Remove/Recuperar Tarefa

I. No método CTarefaDlg::OnRemoveRecupera() colocar:

```
CString AuxStr;
```

```
if (m_ListaTarefas.GetCurSel() >= 0)
```

```
{
```

```
    m_ListaTarefas.GetText(m_ListaTarefas.GetCurSel(),AuxStr);
```

```
    theCTarefas->RemoveRecupera(atoi(AuxStr.Mid(1,6)));
```

```
    RefreshLista();
```

```
};
```

II. No método CTarefas::RemoveRecupera() colocar:

```

POSITION PrevPos, Pos;
Pos = theConjTarefas.GetHeadPosition();
if (theConjTarefas.GetHead().get_ID()==vID)

theConjTarefas.GetAt(Pos).set_Apagada(!theConjTarefas.GetAt(Pos).get_Apagada());
else
    while (Pos!=NULL)
    {
        PrevPos = Pos;
        if (theConjTarefas.GetNext(Pos).get_ID()==vID)
        {

theConjTarefas.GetAt(PrevPos).set_Apagada(!theConjTarefas.GetAt(PrevPos).get_Apagada());
            Pos = NULL;
        }
    }

ProxPos = NULL;

```

III. Compilar : Não deverá dar erros, e já está a funcionar o remover/recuperar tarefa

**e. Diagrama Gravar sessão**

I. No método CTarefaDlg::OnConfirma() colocar:

```

theCTarefas->Confirma();
theCTarefas=NULL; //iniciar nova sessão...
Arranque();

```

II. No método CTarefas::Confirma() colocar:

```

theCServicosBD->WriteBD();

```

III. Compilar : Não deverá dar erros

## f. Definir tratamento com a base de dados (2ª iteração)

I. No método CTarefa::Mapear() colocar:

```
ID=Rs->GetFields()->GetItem("ID")->Value;
Prioridade=Rs->GetFields()->GetItem("Prioridade")->Value;
Descricao=(char*)(_bstr_t)Rs->GetFields()->GetItem("Descricao")->Value;
DataExecucao=(char*)(_bstr_t)Rs->GetFields()->GetItem("DataExecucao")->Value;
DataCriacao=(char*)(_bstr_t)Rs->GetFields()->GetItem("DataCriacao")->Value;
Apagada=FALSE;
Nova=FALSE;
```

II. No método CServicos::ReadBD() colocar:

```
::CoInitialize(NULL);

_ConnectionPtr pCon=NULL;
pCon.CreateInstance(__uuidof(Connection));
pCon->Open(strcon, "", "", -1);

_RecordsetPtr Rs1=NULL;
Rs1.CreateInstance(__uuidof(Recordset));
Rs1->Open("select * from Tarefas Order By ID", (_Connection*)pCon,
adOpenForwardOnly, adLockReadOnly, -1);

if (!Rs1->EndOfFile)
    Rs1->MoveFirst();

while (!Rs1->EndOfFile)
{
    CTarefa AuxTarefa;
    AuxTarefa.Mapear(Rs1);
    theCTarefas->theConjTarefas.AddHead(AuxTarefa);
    Rs1->MoveNext();
}
```

### III. No método CServicos::WriteBD() colocar:

```
::CoInitialize(NULL);
```

```
_ConnectionPtr pCon=NULL;
pCon.CreateInstance(__uuidof(Connection));
pCon->Open(strcon, "", "", -1);
```

```
_CommandPtr pCmd=NULL;
pCmd.CreateInstance(__uuidof(Command));
pCmd->ActiveConnection = pCon;
BOOL ok = TRUE;
```

```
CString AuxSQL;
char AuxID[20];
char AuxPrioridade[20];
CTarefa AuxTarefa;
```

```
POSITION pos = theCTarefas->theConjTarefas.GetHeadPosition();
```

```
pCon->BeginTrans();
```

```
try
{
```

```
    for (int i=0;i < theCTarefas->theConjTarefas.GetCount();i++)
    {
```

```
        AuxTarefa = theCTarefas->theConjTarefas.GetNext(pos);
        _itoa( AuxTarefa.get_ID(), AuxID, 10);
        _itoa( AuxTarefa.get_Prioridade(), AuxPrioridade, 10);
```

```
        if (!((AuxTarefa.get_Nova() == (AuxTarefa.get_Apagada()))))
        {
```

```
            if (AuxTarefa.get_Apagada())
```

```
                AuxSQL = "DELETE FROM Tarefas WHERE ID = " +
                    (CString)AuxID;
```

```
            else // É nova
```

```
                AuxSQL = "INSERT INTO Tarefas VALUES ("
                    + (CString)AuxID
                    + "," + (CString)AuxPrioridade
                    + "," + AuxTarefa.get_Descricao() + ""
                    + "," + AuxTarefa.get_DataExecucao() + ""
                    + "," + AuxTarefa.get_DataCriacao() + ")";
```

```
                pCmd->CommandText = (_bstr_t)AuxSQL;
                pCmd->Execute(NULL,NULL,adCmdText);
```

```
        }  
    }  
}  
catch(_com_error &e)  
{  
    AfxMessageBox(e.Description());  
    ok = FALSE;  
}  
if (ok==TRUE)  
    pCon->CommitTrans();  
else  
    pCon->RollbackTrans();
```

#### IV. Compilar : Está pronto