INSTITUTO SUPERIOR
DE ENGENHARIA DE LISBOA

Licenciatura em Engenharia Informática e de Computadores

Sistemas de Informação II

Relatório da 2ª Fase do Trabalho

Semestre de Inverno

2016/2017

Realizado por:
Tiago Santos nº40590
João Correia nº40604
João Lopes nº39120

INTRODUÇÃO

Para esta fase do trabalho foi-nos pedido que criássemos uma aplicação .NET que permitisse aceder às funcionalidades (e) a (k) que implementámos na primeira fase do trabalho e que permitisse exportar a informação dos alugueres entre duas datas de acordo com o XML Schema fornecido. Foi também requerido que a aplicação tivesse duas implementações, uma usando a Entity Framework e outra usando ADO.NET. Foi-nos também pedido para comparar a tecnologia EF e ADO.NET quando à facilidade de programação e desempenho e também para indicar em que situações uma apresenta vantagens/desvantagens em relação á outra no que diz respeito á garantia de consistência dos dados.

Arquitetura e Implementação

A solução criada contém, para além da implementação da base de dados em SQLServer, um aplicação WinForms para fornecer uma interface de utilização mais agradável do que um terminal de consola.

Para suportar a comunicação com a base de dados, em ambos os modos conectado e desconectado, adoptou-se uma abordagem *object oriented*. Foi criada uma interface ICommand que estipula um conjunto de operações de acesso a dados que permitem realizar as funcionalidades pedidas. Ambos os modos de acesso a dados estão implementados em classes que aderem ao contrato estipulado por esta interface(AdoCommand & EfCommand).

Torna-se, portanto, bastante fácil alterar o modo de acesso a dados, e se necessário adicionar novos modos no futuro.

A alteração de modos é feita quando se carrega na checkbox no canto inferior esquerdo da UI da aplicação e baseia-se na alteração de uma flag que irá permitir decidir qual o tipo de instância que o método GetCommand() da classe Program irá retornar.

A aplicação usa também um ficheiro de configuração App.config para permitir que múltiplos utilizadores possam ter acesso à base de dados.

2

Facilidade de programação:

EF (3 linhas de código, 1 única chamada a um método)

```
public object EquipamentosLivres(String inicio, String fim) {
   return ctx.EquipamentosLivres(DateTime.Parse(inicio), DateTime.Parse(fim), null);
}
```

ADO.NET (31 linhas de código, é preciso criar o comando, criar e abrir a conexão á base de dados e tratar dos dados recebidos da execução do comando)

```
public object EquipamentosLivres(String inicio, String fim) {
   return executeFunction((cmd) => {
            GetUnusedEquipmentsFor(cmd, inicio, fim);
   });
}...
private DataTable executeFunction(Action<SqlCommand> action) {
   using (SqlCommand cmd = con.CreateCommand()) {
            action(cmd);
            con.Open();
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.HasRows) {
                     DataTable dt = new DataTable();
                     dt.Load(reader);
                     return dt;
            }
            reader.Close();
            return null;
   }
}...
private void GetUnusedEquipmentsFor(SqlCommand cmd, String inicio, String fim)
{
   SqlParameter paramInicio = new SqlParameter("@inicio", SqlDbType.DateTime);
   SqlParameter paramFim = new SqlParameter("@fim", SqlDbType.DateTime);

   paramInicio.Value = inicio;
   paramFim.Value = fim;
   cmd.Parameters.Add(paramInicio);
   cmd.Parameters.Add(paramFim);

   cmd.CommandText = "SELECT * FROM EquipamentosLivres(@inicio, @fim, NULL)";
}
```

Desempenho:

CompareADO_VS_EF():
Após várias execuções deste teste unitário chegou-se sempre ao seguinte resultado *give or take* (aparece no Output do teste):

```
EF-> Total time: 343ms Best time: 908ticks
ADO.NET-> Total time:152ms Best time: 416ticks
```

Este teste executa MAX_RUNS vezes o método EquipamentosSemAlugueresNaUltimaSemana para o EF e também para o ADO.NET, como é observado o ADO.NET foi mais rápido do que o EF (±2x,±2x) provavelmente por este ser construído por cima do ADO.NET adicionando *overhead* ás operações (neste caso de leitura).

CompareADO_VS_EF_Round2():
Output do teste:

```
EF -> Total time: 250ms Best time: 4108ticks
ADO.NET -> Total time:29ms Best time: 1063ticks
```

Este teste executa MAX_RUNS vezes o método InserirPreco seguido do método RemoverPreco, como se consegue observar o ADO.NET foi muito mais rápido que o EF (±8x,±4x), novamente por causa dos *overheads* do EF.

3

O Entity Framework não usa transacções mas contém uma lista de alterações feitas (Change Tracking) e só quando é feito uma chamada a contexto.SaveChanges() é que é aberta uma conexão à base de dados, iniciada uma transacção com nível de isolamento igual ao nível por omissão do servidor, feitas as alterações e depois essa transacção é *committed* e a conexão é fechada.

Na ocorrência de um erro a transacção é *rolled back* e isso é visível no lado da aplicação através de uma exceção lançada pela infra-estrutura.

Sendo assim, consideramos que o modo conectado ADO.NET é melhor em termos de consistência de dados e segurança dado que existe um maior controlo sobre o âmbito transacional.