Licenciatura em Engenharia Informática e Computadores

Sistemas de Informação II

Semestre de Inverno 2017/2018

Segunda Fase

Docente:

Walten Vieira

Discentes:

Tejal Amratlal, n.º40605

Eduardo António, n.º 40686

Ana Souto, n.º41112

Grupo 8

Introdução

Nesta segunda fase do trabalho tem como objetivo desenvolver uma aplicação em C#, em que utilize diferentes implementações de acesso a dados; tem também como objetivo a utilização correta de ADO.NET e (ADO.NET) Entity Framework; e também tem como garantir a correta libertação de ligações e recursos, quando estes não estão sendo utilizados e por fim garantir a correta implementação das restrições de integridade.

Nesta etapa é-nos pedido que sejam implementadas as seguintes funcionalidades c) a l) da fase anterior:

c) inserir, remover e atualizar informação de um hóspede;

d) inserir, remover e atualizar informação de um alojamento;

e) inserir, remover e atualizar informação de um extra de alojamento;

f) inserir, remover e atualizar informação de um extra pessoal;

g) inserir, remover e atualizar informação de uma atividade;

h) criar uma estada para um dado período de tempo. Este processamento deve ser dividido nos seguintes sub-processamentos:

1. Criar uma estada dado o NIF do hóspede responsável e o período temporal pretendido;

2. Adicionar um alojamento de um dado tipo com uma determinada lotação a uma estada;

3. Adicionar um hóspede a uma estada;

4. Adicionar um extra a um alojamento de uma estada;

5. Adicionar um extra pessoal a uma estada;

i) inscrever um hóspede numa atividade;

j) proceder ao pagamento devido por uma estada, com emissão da respetiva fatura;

k) enviar emails a todos os hóspedes responsáveis por estadas que se irão iniciar dentro de um dado período temporal. Os emails dever ser enviados usando o procedimento armazenado SendMail que recebe o NIF do cliente e o texto da mensagem a enviar.

l) listar todas as atividades com lugares disponíveis para um intervalo de datas especificado;

É nos pedido também para implementar as seguintes funcionalidades: obter o total pago por hóspede relativo a estadas num dado parque num intervalo de datas especificado; eliminar um dos parques (e respetivos alojamentos e estadas). Os dados sobre os hóspedes que não tenham estadas em qualquer outro parque também devem ser eliminados. Estas funcionalidades adicionais não podem utilizar procedimentos armazenados.

Desenvolvimento

SQL

Para esta parte do trabalho foram necessárias algumas modificações no código SQL entregue na fase anterior pois não cumpriam os requisitos pretendidos. Essas alterações foram executadas pois iremos utilizar esses procedimentos utilizados em ADO.NET e em Entity Framework.

As alterações foram feitas nas alíneas j), k) e m) pois estas alíneas estavam mal-executadas.

Modelo Relacional

ParqueCampismo (nome, morada, estrelas, mail)

Telefone (telefone, parque[FK])

FK : {parque references ParqueCampismo(nome)}

Alojamento (nome, parque[FK], localização, descrição, precoBase, nMaxPessoas, tipo)

FK : {parque references ParqueCampismo(nome)}

Tendas (nomeAlojamento[FK], area, tipo)

FK : {nomeAlojamento references Alojamento(nome)}

Bungalows (nomeAlojamento[FK], tipologia)

FK : {nomeAlojamento references Alojamento(nome)}

Hospede (nIdentificacao, nif, nome, morada, mail, exist)

Estada (id, dataInicio, dataFim, nIdentificacao[FK])

FK : {nIdentificacao references Hospede}

Extra (id, descrição, precoDia, tipo)

EstAlojExtra (id[FK], alojamento[FK], extra[FK])

FK : {id references Extra; alojamento references Alojamento(nome); extra references Extra(id)}

ExtraPessoa (id[FK], tipo)

FK : {id references Extra}

ExtraAloj(id[FK], tipo)

FK : {id references Extra}

Fatura(id, ano, idEstada[FK], descrição, nome, nif)

FK : {idEstada references Estada}

Atividade (num, ano, parque[FK], nome, descrição, lotacaoMaxima, dataRealizacao, precoParticipante)

FK : {parque references ParqueCampismo(parque)}

Email (nif, email, texto)

HospEst (id[FK], nIdentificacao[FK])

FK : {id references Estada; nIdentificacao references Hospede}

HospEstAti (num[FK], ano[FK], id[FK], nIdentificacao[FK])

FK : {id references Estada; nIdentificacao references Hospede; ano references Atividade; num references Atividade}

HistoricoAloj (alojamento[FK], preco, dataInicial)

FK : {alojamento references Alojamento(nome)}

HistoricoExtra (extra[FK], preco, dataInicial)

FK : {extra references Extra(id)}

Restrições de Integridade

R.I.1 - email é outra chave candidata

R.I.2 - nrEstrelas é de 0 a 5

R.I.3 - localizacao é caracterizada por uma sequência de caracteres alfanuméricos únicos

R.I.4 - precoBase é do tipo Money

R.I.5 - área é medida em m^2

R.I.6 - tipo pode ser safari, yurt ou tipi

R.I.7 - tipologia pode ser:'T0','T1','T2','T3'

R.I.8 - precoDia é do tipo Money

R.I.9 - tipo pode ser 'ExPessoa' ou 'ExAloj'

R.I.10 - tipo = ='pe' ou tipo = 'ac' ou tipo = 'ea'

R.I.11 - tipo ='pa' ou tipo = 'mp' ou tipo = 'pc'

R.I.12 - email é outra chave candidata

R.I.13 - nif é outra chave candidata

R.I.14 - precoParticipante é do tipo money

ADO.NET

Nesta vertente do trabalho, é pedida uma ligação à base de dados implementada no trabalho, e que, segundo certas funcionalidades apresentadas ao utilizador, possam ser feitas operações ou pesquisas sobre essa base de dados.

Para a parte da ligação à base de dados, nós utilizámos uma instância de SqlConnection. Através dessa instância iniciamos a transação e depois utilizámos essa mesma instância para a criação de uma instância do tipo SqlCommand e aí irmos criar o comando e através desse comando que iremos passar o nome do procedimento ou da função pretendida e passar os respetivos parâmetros. Após isto , é feito commit e fecha-se a ligação.

Entity Framework

Para esta vertente do trabalho, na implementação das funcionalidades pretendias, o processo foi parecido com o processo desenvolvido em ADO.NET.

Enquanto que em ADO.NET se criava a conexão através da instância SqlConnection, aqui em Entity Framework cria-se uma instância do contexto, no nosso caso é uma instância de GlampinhoEntities. Aqui também se obtém os parâmetros necessários através da Consola e depois utiliza-se o contexto para chamar o procedimento pretendido.

Entity Framework vs ADO.NET

Facilidade de Programação

Após a execução do trabalho, percebemos que a vertente ADO.NET é mais trabalhosa que a vertente Entity Framework, pois nesta última vertente referida para a sua utilização só é necessário a criação de uma instância do contexto, no nosso caso, GlampinhoEntities e chamar o procedimento ou função e basicamente utiliza-se uma linha de código, enquanto que na vertente ADO.NET é necessário a criação da conexão e do comando, em que depois sobre o comando , diz-se o nome do procedimento ou função e depois passa-se os parâmetros em que estes ocupados várias linhas de código.

Desempenho

No entanto, em termos de desempenho, o ADO.NET é mais rápido. Isto deve-se ao facto de estarmos a libertar as conexões e os comandos quando estes deixem de ser utilizados. Um motivo que leva ao Entity Framework ser mais é lento é devido ao acesso ao contexto por isso vai ter mais representações de memória de todos os tipos criados mesmo quando não estão sendo utilizados.

Consistência de dados

A Entity Framework pode verificar automaticamente qualquer alteração nos dados antes de se comprometer e lançar exceção se os dados mudarem para fora do seu contexto atual. Isto é a Entity Framework está relacionada com a concorrência otimista. A concorrência otimista consiste não bloqueia os registos quando estes são lidos. Caso algum utilizador desejar atualizar uma linha, a aplicação irá decidir se o outro utilizador alterou a linha desde que esse foi lido. Normalmente é utilizada em ambientes com contenção de dados baixa. Mas também existe a concorrência pessimista está relacionada com o conceito de bloqueio do registo na base de dados, impedindo assim que os utilizadores modifiquem os dados de que alguma forma afeta os outros utilizadores. Isto é, quando um utilizador executa uma ação em que provocará um bloqueio, os restantes utilizadores não podem realizar ações em que entrem em conflito com o bloqueio até o responsável pelo bloqueio o finalize. Este modelo é principalmente utilizado onde há contenção pesada de dados, onde o custo de proteção de dados com bloqueios é menor que o custo de reverter as transações caso ocorram conflitos ao mesmo tempo.