



Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Eletrónica e
Telecomunicações e de Computadores

Mestrado em Engenharia Informática e de Computadores
Arquitetura de Sistemas Distribuídos
2013/2014
Relatório Aula Prática Nº 4
Grupo Nº 5

Nome	Nº de Aluno	E-mail
Rui Miranda	A32342	a32342@alunos.isel.pt
David Coelho	A21359	a21359@alunos.isel.pt
Frederico Ferreira	A7066	a7066@alunos.isel.pt

Exercício 1

1. Considere o serviço implementado no projecto Ex1.

a) Crie um cliente para este serviço que invoque a operação Transferir duas vezes entre as contas 1 e 2 com o valor 500, crie previamente a tabela contas com a seguinte constituição:

```
create table contas (  
  numero int primary key,  
  titular varchar(20),  
  saldo real  
)
```

b) Execute o serviço e o cliente e justifique os valores de n que observa escritos na consola da aplicação hospedeira do serviço.

c) Volte a colocar o saldo das contas com o valor 1000. Altere o cliente de forma a que a segunda transferência de 500 euros seja feita da conta 1 para a conta 1111. Verifique o valor do saldo da conta 1. Justifique a existência deste saldo. Qual a solução?

Implementação

No âmbito das alíneas, foram criados os seguintes ficheiros:

- !Ex1-RunAll-AsAdmin.bat
- Ex1.a.CREATE-DB-TBL.sql
- Ex1.a.Run.CREATE-DB-TBL.bat
- Ex1.b.Run-Ex1-AsAdmin.bat
- Ex1.b.Run-Ex1Cliente.a.bat
- Ex1.c.0.Run-Ex1Cliente.c.0.bat
- Ex1.c.1.Run-Ex1.c-AsAdmin.bat
- Ex1.c.1.Run-Ex1Cliente.c.1.bat
- Ex1.Select.sql
- Ex1.Select-Run.bat

O primeiro *batch* utiliza os restantes ficheiros para executar uma série de tarefas necessárias à execução destas as alíneas deste exercício.

Alínea a)

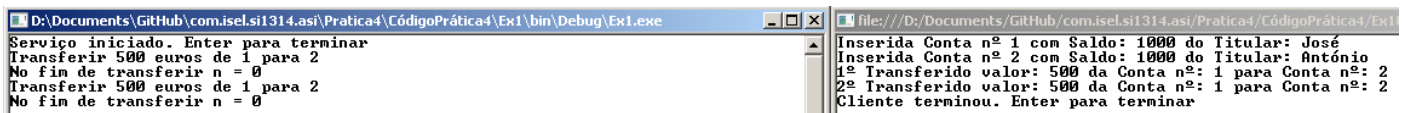
O ficheiro Ex1.a.Run.CREATE-DB-TBL.bat é executado em seguida com o intuito de criar a BD e respetivas tabelas.

O Cliente criado no projecto Ex1Cliente.a, executa as seguintes acções:

- 1) Instância a *proxy* para o serviço (adicionado por *service reference* no projecto)
- 2) Invoca as operações de inserir para as duas contas (1 e 2)
- 3) Invoca as operações de transferir duas vezes entre a conta 1 e 2

Alínea b)

É executado agora o Serviço e o Cliente através do `Ex1.b.Run-Ex1-AsAdmin.bat` e , respectivamente, gerando-se o *ouput* seguinte (Serviço à esq., Cliente à dir.):



Observa-se que a variável n tem sempre o valor inicial, zero.

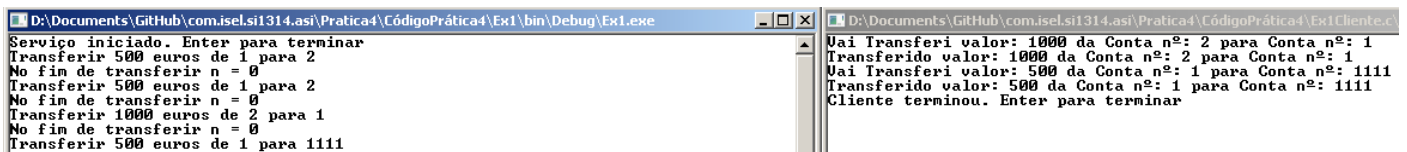
Sendo n uma variável global do Serviço (*Ex1.Servico.n*), iniciada a zero na instanciação do Serviço, e incrementada no final da operação Transferir.

Observando a *ServiceBehavior InstanceContextMode.Single*, no Serviço seria esperado que o valor de n incrementa-se com cada chamada da operação Transferir.

No entanto como a operação Transferir tem o *OperationBehavior TransactionScopeRequired* a verdadeiro, a execução desta operação implica a criação de uma nova transacção, o que por sua vez implica a instanciação de um novo Serviço e sub-consequentemente a iniciação de n a zero.

Alínea c)

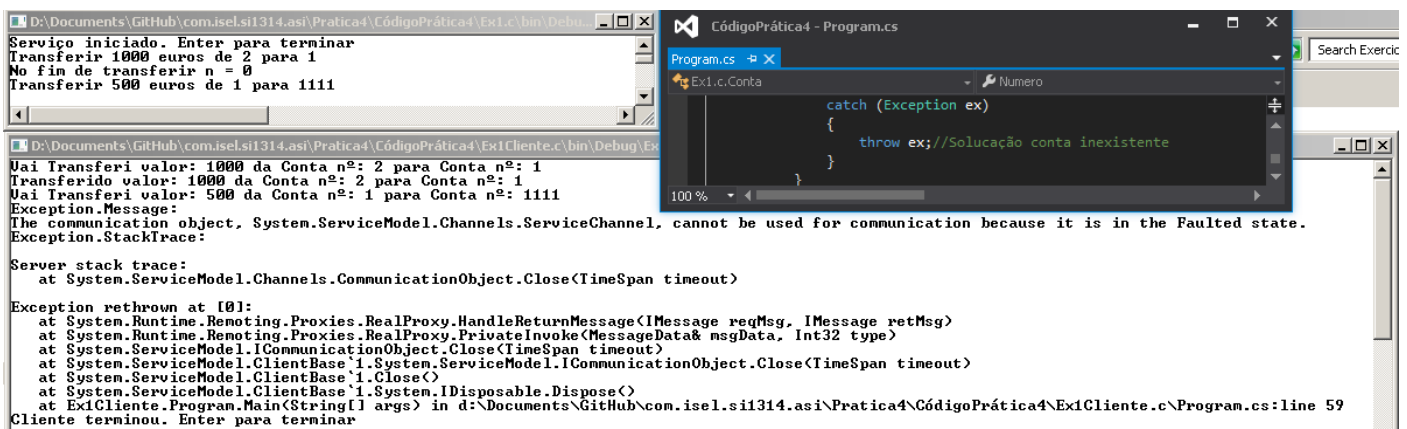
Criou-se um novo Cliente para a alteração desejada, *Ex1Cliente.c.0*, ainda com a instância do Serviço *Ex1* a correr da alínea, executa-se o novo Cliente, e observa-se o seguinte output (Serviço à esq., Cliente à dir.):



O saldo de 500 é debitado apesar de a conta destino não existir, no serviço é gerada excepção, mas no método Transferir não existe código para o bloco de *catch*, o que aparenta estar tudo correcto.

Uma possível solução será nesse bloco de *catch* fazer *throw* da excepção gerada, assim como a operação tem *TransactionScopeRequired* a *true*, será o próprio WCF encarregue de fazer *rollback* de toda a operação. Observamos nesse caso, que saldo já não é debitado, caso a conta destino não exista.

A solução encontra-se implementada num novo Serviço, *Ex1.c*, e respectivo novo cliente, *Ex1Cliente.c.1*, onde-se se observa o seguinte output após a sua execução (Serviço à esq., Cliente à dir.):



Exercício 2

2. Considere o serviço implementado no projecto Ex2. Altere o cliente do ponto anterior de forma a realizar apenas uma transferência de 500 euros da conta 1 para a conta 2. Volte a colocar os saldos das duas contas com o valor 1000.

- Execute passo-a-passo o cliente e verifique com um *select* realizado no management studio que após a transferência e enquanto não se fechar o *proxy*, a transacção continua a existir. Justifique este comportamento.
- Faça *close* do *proxy* e verifique que a transacção não produziu efeito. Justifique porquê.
- Apresente três soluções diferentes para que a transacção produza os efeitos devidos.

Implementação

No âmbito das alíneas, foram criados os seguintes ficheiros:

- !Ex2-RunAll-AsAdmin.bat
- Ex2.Run-Ex2-AsAdmin.bat
- Ex2.Run-Ex2Cliente.bat
- Ex2.Select.sql

O primeiro *batch* utiliza os restantes ficheiros para executar uma série de tarefas necessárias à execução destas as alíneas deste exercício.

Alínea a)

Executando o Serviço, e o Cliente passo-a-passo em *debug* no *visual studio*, observa-se o seguinte:

- 1) Antes da invocação da operação Transferir

The screenshot displays two windows from a development environment. The top-left window shows a console application running, with output indicating a transfer of 500 euros from account 1 to account 2. The top-right window shows the SQL Server Enterprise Manager interface, displaying a query result for the 'contas' table. The bottom-left window shows the C# code for the client application, with the 'Transferir' method being executed. The bottom-right window shows the SQL query and its results in the Enterprise Manager.

Console Output (Top Left):

```
Serviço iniciado. Enter para terminar
```

SQL Query Results (Top Right):

```
Alterada Conta nº 1 com Saldo: 1000 do Titular: José
Alterada Conta nº 2 com Saldo: 1000 do Titular: António
Vai Transferir valor: 500 da Conta nº: 1 para Conta nº: 2
```

C# Code (Bottom Left):

```
Program.cs
Ex2Cliente.a.Program
Main(string[] args)
{
    + " da Conta nº: " + c1.Numero.ToString()
    + " para Conta nº: " + c2.Numero.ToString();
    proxy1.Transferir(c1.Numero, c2.Numero, valor);
    Console.WriteLine(
        "Transferido valor: " + valor
        + " da Conta nº: " + c1.Numero.ToString()
        + " para Conta nº: " + c2.Numero.ToString());
}
Console.WriteLine("Cliente terminou. Enter para terminar");
Console.ReadLine();
}
catch (Exception e)
```

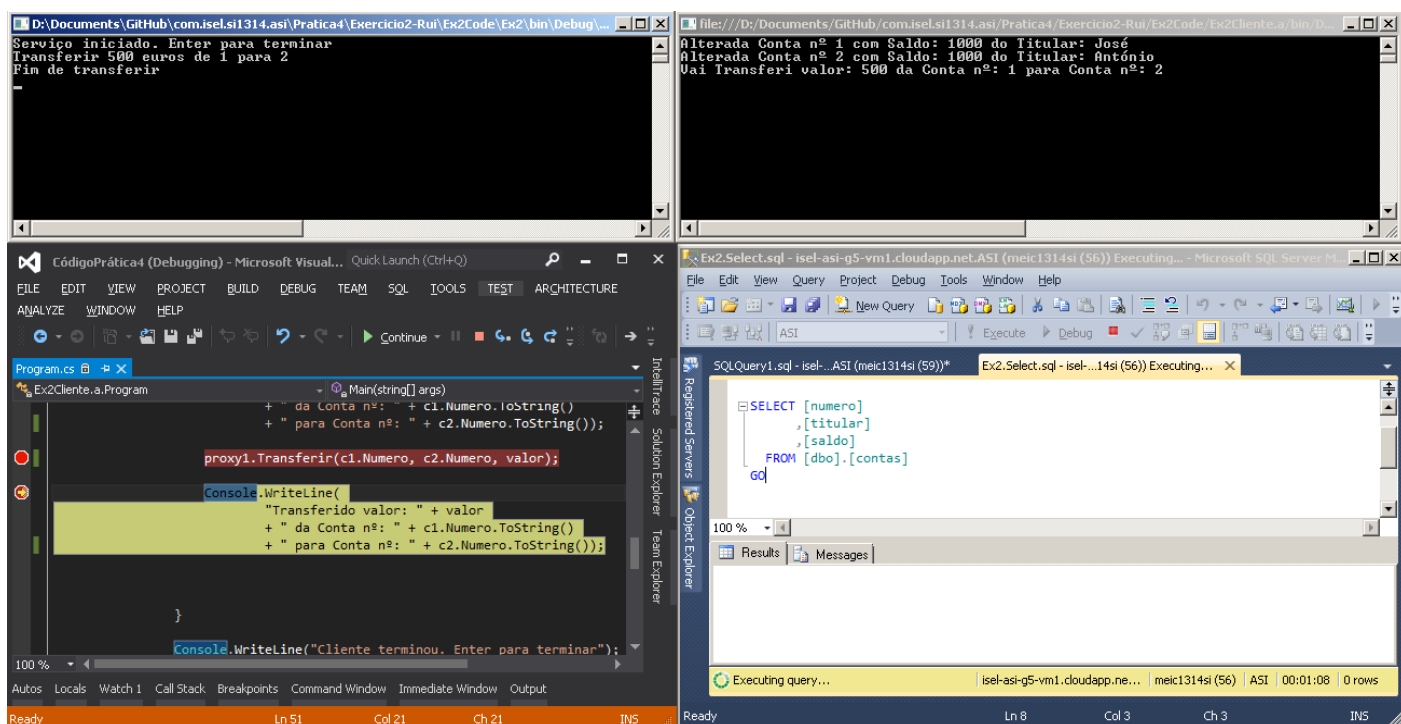
SQL Query (Bottom Right):

```
USE [ASI]
GO
SELECT [numero], [titular], [saldo]
FROM [dbo].[contas]
```

Query Results Table:

numero	titular	saldo
1	José	1000
2	António	1000

2) Após invocação e antes de fechar o proxy

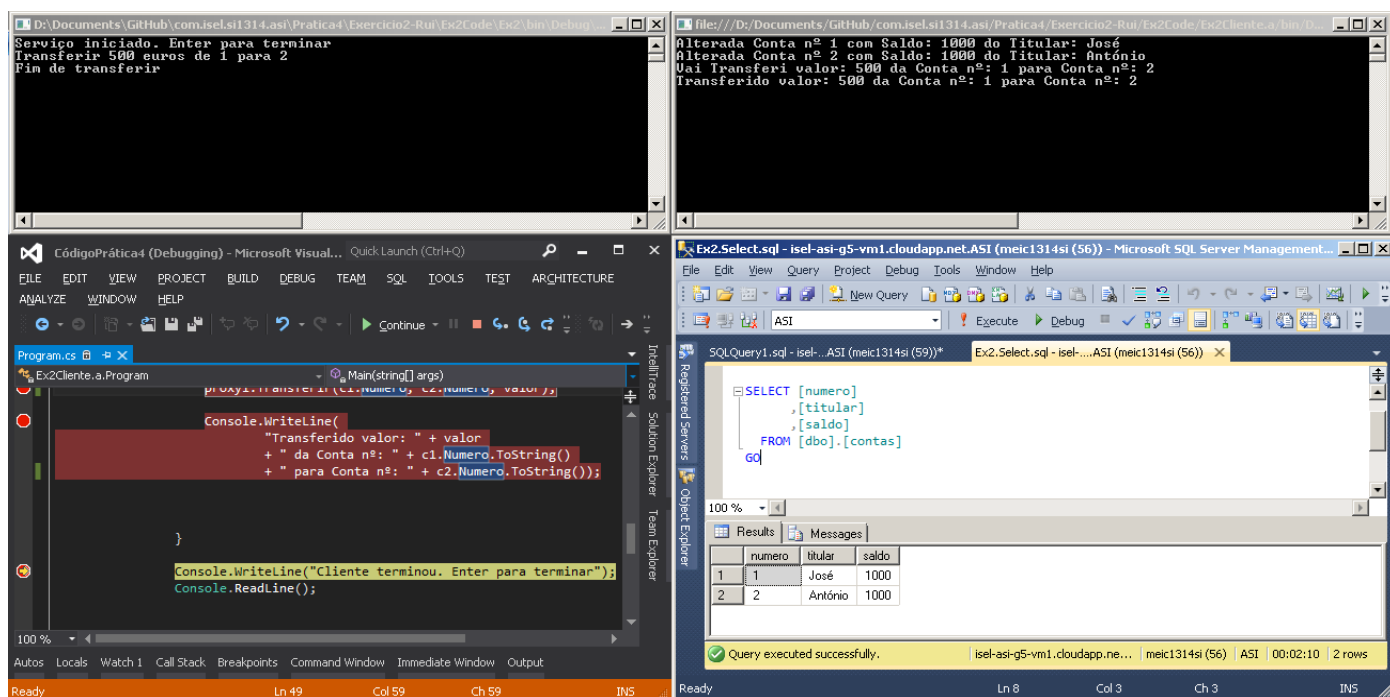


O observado sucede-se porque o *WCF* está a espera da votação da respectiva transacção, alocando assim todos os recursos para a mesma, até que receba votação para conclusão ou aborte.

Alínea b)

Continuando a execução da alínea anterior:

1) Após o fecho do proxy



O observado sucede-se porque o método *Ex2.Servico.Transferir* está configurado com o *OperationBehavior TransactionAutoComplete* a *False*, ou seja, isto faz com o *WCF* não termine a transição quando a execução da operação termina, sendo essa responsabilidade passada para o interior da operação, como a operação encerra sem terminar a transacção o *WCF* faz *rollback* da transacção, fazendo com que os valores não se alterem.

Alínea c)

Três soluções possíveis para situação observada anteriormente:

- a) Na operação *Ex2.Servico.Transferir* colocar o *TransactionAutoComplete* a *True*, fazendo assim a votação implícita para a conclusão da transacção.
- b) Na operação *Ex2.Servico.Transferir* retirar o atributo *TransactionAutoComplete*, tendo assim o valor por defeito (*True*) e assim a votação implícita para a conclusão da transacção.
- c) Antes de terminar a execução da operação incluir a instrução *OperationContext.Current.SetTransactionComplete()*; fazendo assim uma votação explícita para a conclusão da transacção.

Exercício 3

3. Considere o serviço apresentado no projecto Ex3 e o cliente apresentado no projecto Ex3Cliente. Volte a colocar o saldo de ambas as contas com o valor 1000.
- a) Execute o código da aplicação cliente passo-a-passo e vá verificando o estado da fila. Confirme que na fila só aparecem mensagens depois de terminado o bloco using. Verifique que existem duas mensagens. Lance a aplicação hospedeira do serviço e verifique que uma das operações produz efeito sobre a base de dados e outra não. Justifique a razão deste comportamento.
- b) Altere serviço e cliente de forma a que ou ambas as operações produzam efeito, ou nenhuma produza. Volte a realizar os testes da alínea anterior e compare os resultados e comportamentos verificados com os da alínea anterior.

Implementação

No âmbito das alíneas, foram criados os seguintes ficheiros:

- !Ex3-RunAll-AsAdmin.bat
- Ex3.0-Start-CompSrv.bat
- Ex3.1-ConfigDTC.txt
- Ex3.a.Run-Ex3-AsAdmin.bat
- Ex3.a.Run-Ex3Cliente.bat
- Ex3.b.0.Run-Ex3.b.0-AsAdmin.bat
- Ex3.b.0.Run-Ex3Client.bat
- Ex3.b.0-Update-Jose.bat
- Ex3.Select.sql

O primeiro *batch* utiliza os restantes ficheiros para executar uma série de tarefas necessárias à execução destas as alíneas deste exercício.

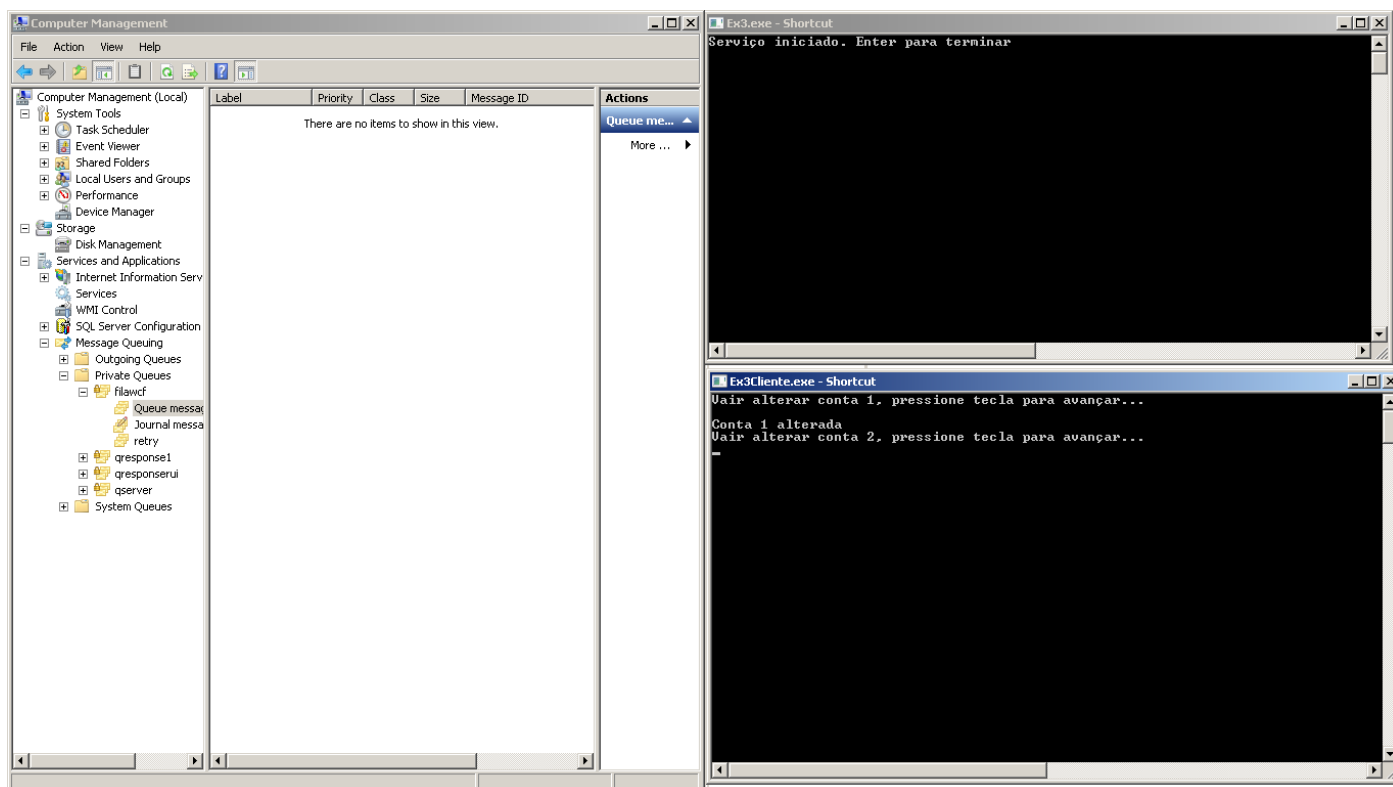
O segundo, Ex3.0-Start-CompSrv.bat, apenas lança a consola de configuração de componentes de serviços do Windows, para que seja correctamente configurado o *DTC*, segundo as instruções em Ex3.1-ConfigDTC.txt.

O *batch* Ex3.b.0-Update-Jose.bat volta a colocar os registos no estado inicial pretendido.

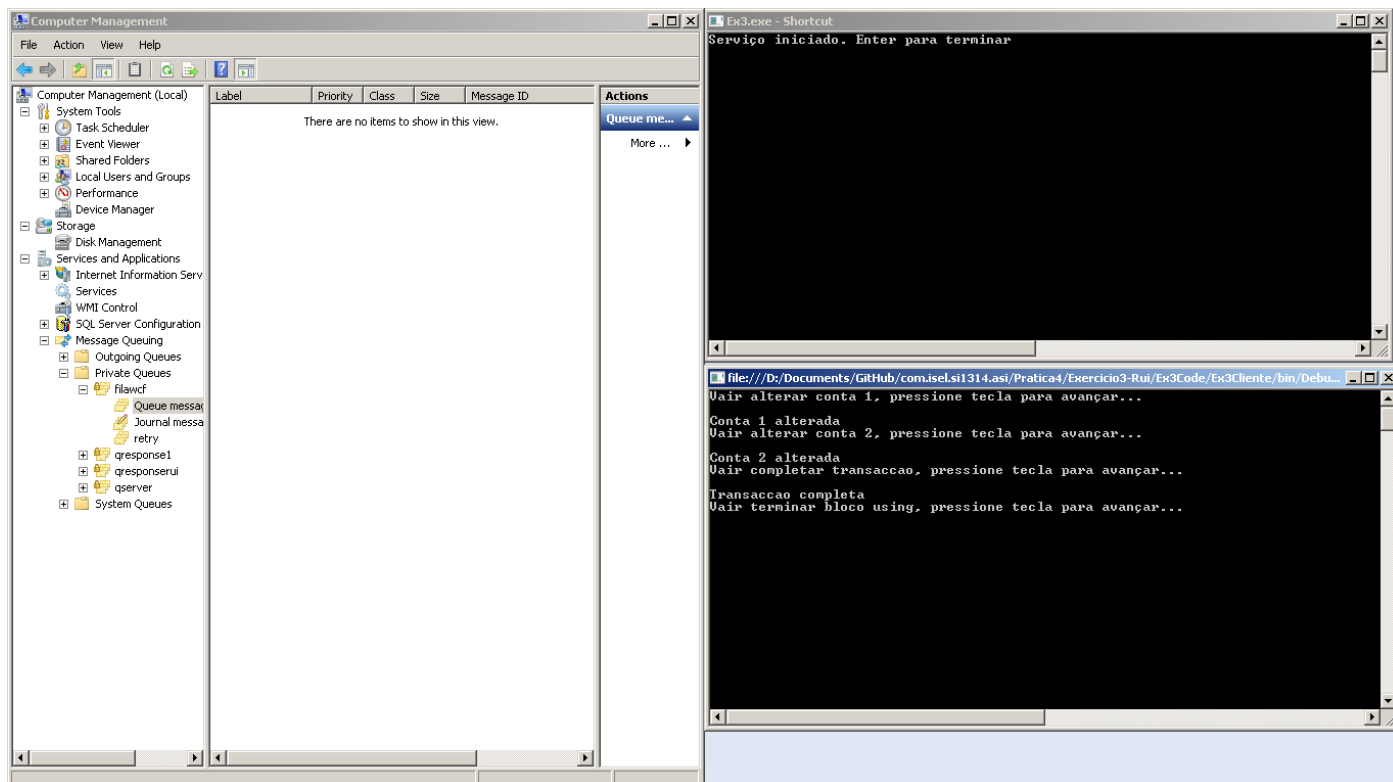
Alínea a)

Executando o Serviço (*Ex3*), e o Cliente (*Ex3Cliente*), observa-se o seguinte:

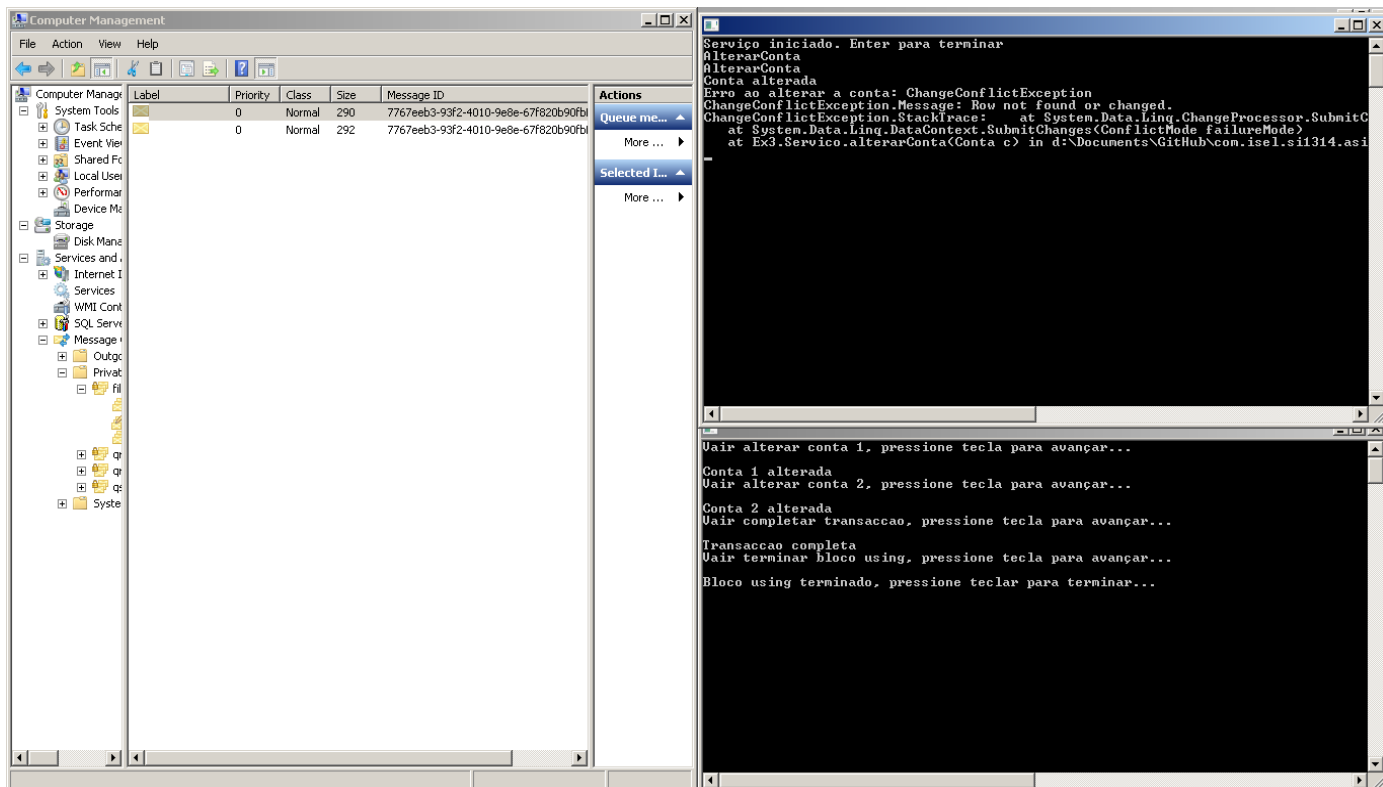
1) Depois de invocado o método de alteração da primeira conta



2) Depois de o Cliente completar a transacção

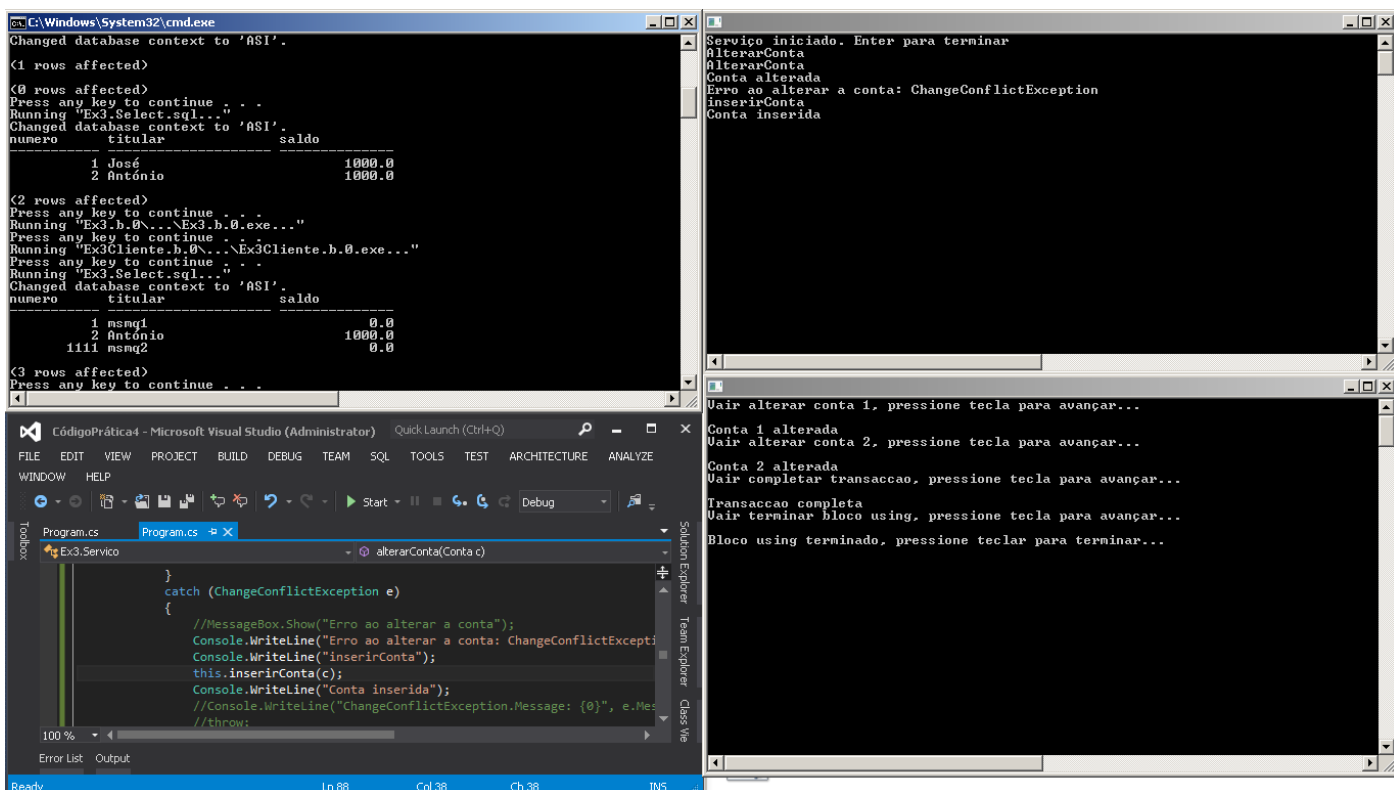


3) Depois de terminar o bloco using



Alínea b)

Foram criados o Serviço, *Ex3.b.0*, e o Cliente, *Ex3Cliente.b.0*, a suas execuções provocam o seguinte output:



Alterou-se o Serviço para que no caso de ocorrer a exceção provocada pela não existência do registo, e chamado o método *Servico.inserirConta*, verificando assim que para além de ser alterada a conta 1, também é inserida a conta 1111.

Exercício 4

4. Considere o serviço apresentado no projecto Ex4 e o cliente apresentado no projecto Ex4Cliente. Volte a colocar o saldo de ambas as contas com o valor 1000.

a) Tire os comentários da zona de código da aplicação cliente comentada com “alínea a)”, execute a aplicação cliente e comente o que acontece.

b) Execute o código marcado com “alínea b)” e compare o resultado com o que observou na alínea anterior.

Implementação

No âmbito das alíneas, foram criados os seguintes ficheiros:

- !Ex4-RunAll-AsAdmin.bat
- Ex4.a.Run-Ex4-AsAdmin.bat
- Ex4.a.Run-Ex4Cliente.bat
- Ex4.b.Run-Ex4Cliente.b.bat
- Ex4.Select.bat
- Ex4.Update-Delete.bat
- Ex4.Update-Delete.sql

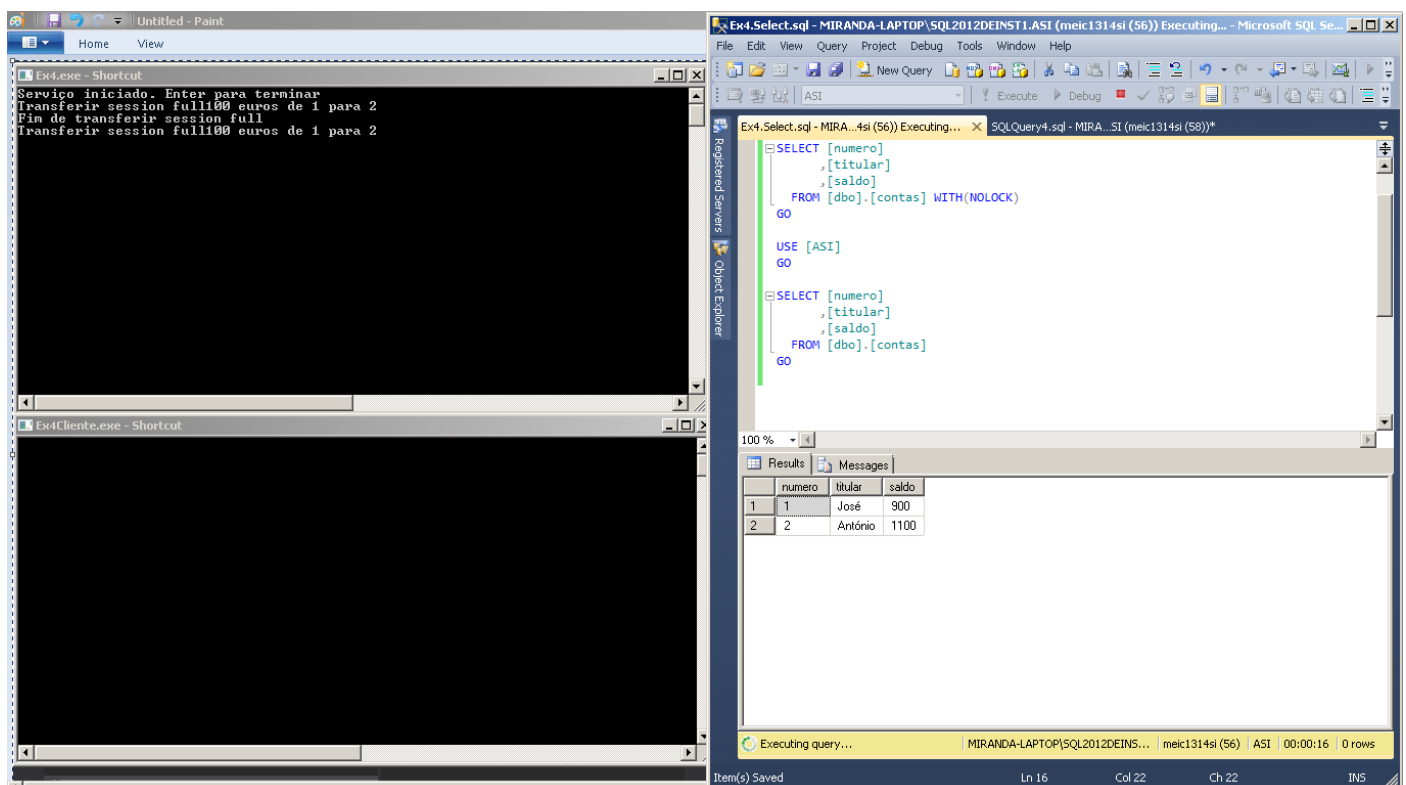
O primeiro *batch* utiliza os restantes ficheiros para executar uma série de tarefas necessárias à execução destas as alíneas deste exercício.

O *batch* Ex4.Update-Delete.bat é utilizado para colocar os registos no estado inicial.

Alínea a)

Após efectuar a alteração indicada ao *Ex4Cliente*, e executando o Serviço e Cliente, observa-se o seguinte output:

- 1) Invocação da segunda operação de transferir, verifica-se que o Serviço fica em espera



The screenshot displays three windows from a Windows operating system:

- Ex4.exe - Shortcut:** A command prompt window showing the following text:

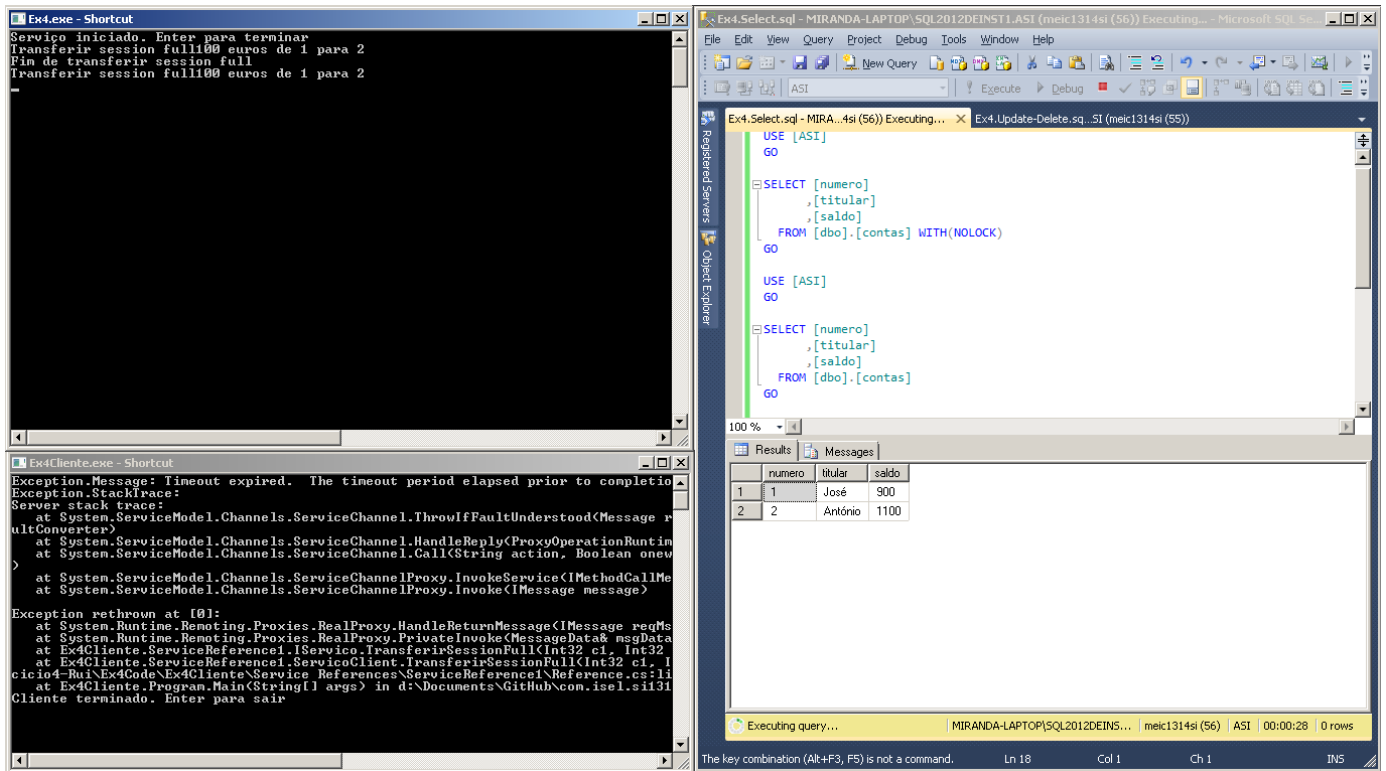
```
Servico iniciado. Enter para terminar
Transferir session full100 euros de 1 para 2
Fin de transferir session full
Transferir session full100 euros de 1 para 2
```
- Ex4Cliente.exe - Shortcut:** An empty application window.
- Microsoft SQL Server Enterprise Manager:** A window showing a query execution plan and a results table. The query is:

```
SELECT [numero]
,[titular]
,[saldo]
FROM [dbo].[contas] WITH(NOLOCK)
GO
USE [ASI]
GO
SELECT [numero]
,[titular]
,[saldo]
FROM [dbo].[contas]
GO
```

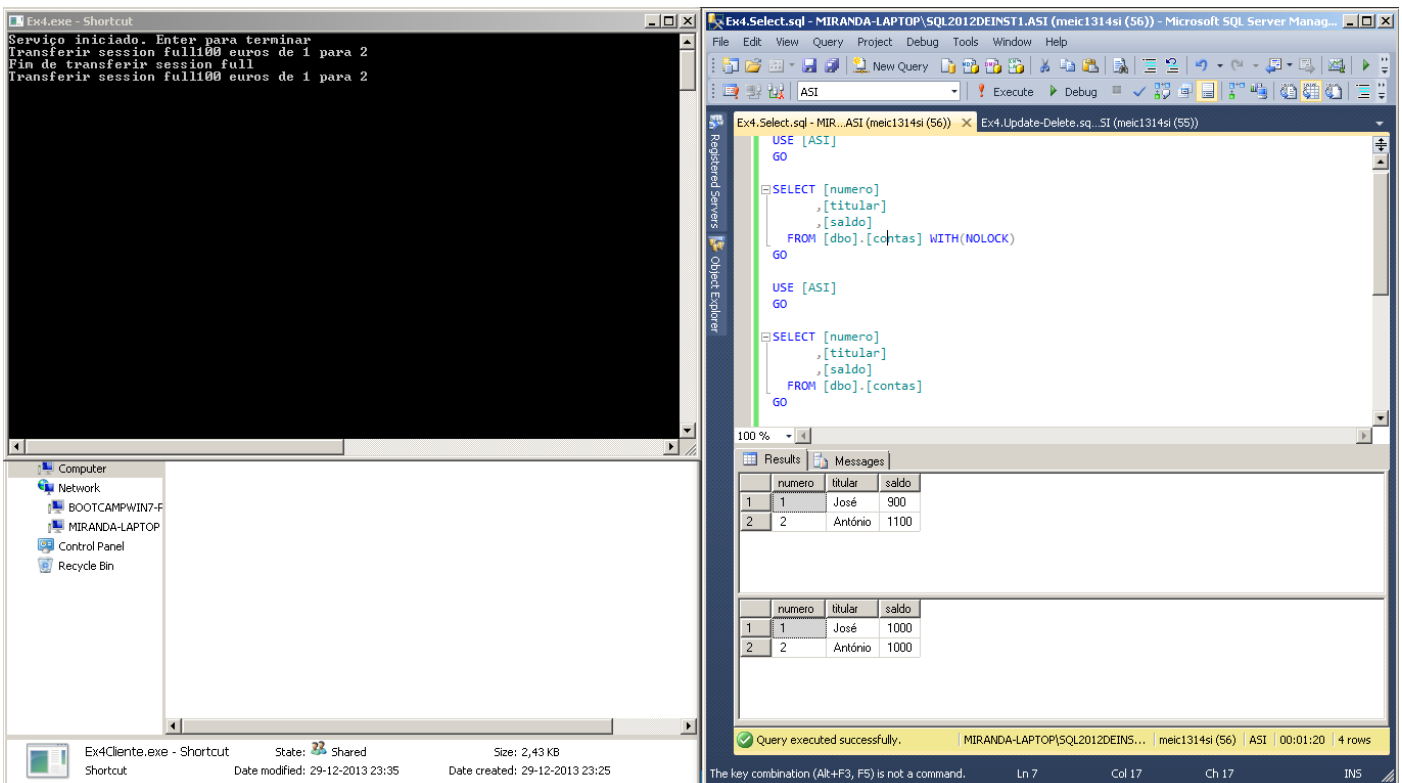
The results table shows two rows:

	numero	titular	saldo
1	1	José	900
2	2	António	1100

2) Após alguns segundos, é lançada para o Cliente uma exceção de *timeout* no acesso aos dados



3) Terminando o Cliente, verifica-se que não foi consistida a alteração nos dados

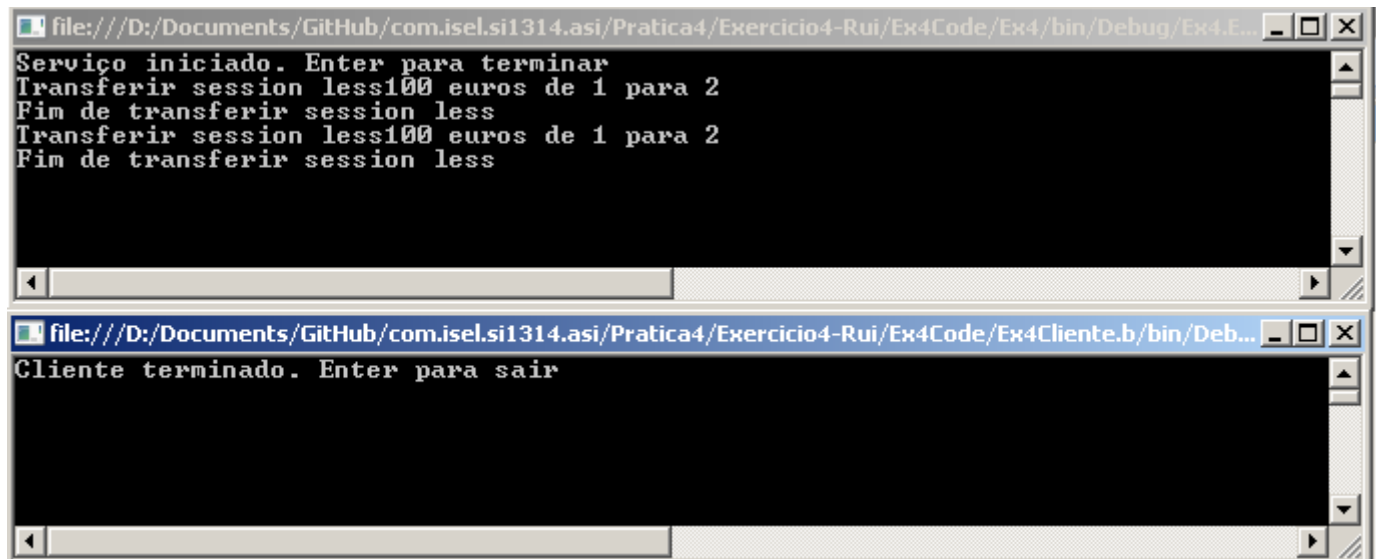


Ao executar a Aplicação Cliente, verifica-se que ao invocar pela segunda vez a operação *Servico.TransferirSessionFull*, na instância do serviço quanto é invocada a operação de leitura da base de dados, a mesma dá timeout, devido ao conflito de *deadlock* gerado pela reserva dos recursos da primeira invocação da operação *Servico.TransferirSessionFull*. Como a exceção é passada pelo *WCF* para o cliente, este não consegue finalizar a primeira invocação da operação, logo não se observa qualquer alteração nos dados.

Alínea b)

Foi criado um novo cliente com a alteração indicada nesta alínea, *Ex4Cliente.b*.

Executando este novo cliente observa-se o seguinte, após terminado:



The image shows two overlapping windows of a Windows command prompt. The top window, titled 'file:///D:/Documents/GitHub/com.isel.si1314.asi/Pratica4/Exercicio4-Rui/Ex4Code/Ex4/bin/Debug/Ex4.E...', displays the following text: 'Serviço iniciado. Enter para terminar', 'Transferir session less100 euros de 1 para 2', 'Fim de transferir session less', 'Transferir session less100 euros de 1 para 2', and 'Fim de transferir session less'. The bottom window, titled 'file:///D:/Documents/GitHub/com.isel.si1314.asi/Pratica4/Exercicio4-Rui/Ex4Code/Ex4Cliente.b/bin/Deb...', displays the text: 'Cliente terminado. Enter para sair'.

Ao efectuar as transacções em modo *session less* observa-se que os efeitos das mesmas são automaticamente repercutidos na base de dados, o que é esperado já que a operação do Serviço está com *TransactionAutoComplete* a *True*, o que faz com que o *WCF* termine a transacção assim que a operação termina. O mesmo não acontecia na alínea anterior, já que a operação *session full* está com *TransactionAutoComplete* a *False*, seria necessário o cliente chamar a operação finalizar (*TransactionAutoComplete* a *True*) para que *WCF* terminasse com sucesso a transacção, o mesmo não acontece porque o Serviço entra em *deadlock*.