Faculdade de Engenharia da Universidade do Porto

Universidade do Porto

FEUP Faculdade de Engenharia

Relatório de trabalho

Diginote Exchange System

Daniel Pereira - ei11132
Pedro Silva - ei11061

Relatório do projecto #1 de *.Net Remoting* realizada no âmbito da unidade curricular Tecnologias de Distribuição e Integração do Mestrado Integrado em Engenharia Informática e Computação.

20/04/2015

Contéudo

1. Introdução
2. Arquitectura
3. Funcionamento
4. Testes
5. Conclusão
6. Bibliografia
7. Software

1. Introdução

Serve o presente documento para descrever o trabalho *Diginote Exchange System* realizado no âmbito da unidade curricular Tecnologias de Distribuição e Integração, a arquitectura definida, as funcionalidades implementadas o modo de funcionamento do projecto e testes executados usados para validar o mesmo. Incluímos também um guia de instruções para a construção do projeto a partir do código fonte.

O projeto tem como objetivo o desenvolvimento de um sistema de venda e compra de um produto único, aqui designado de *diginote*. Este sistema deverá ser capaz de gerir compras e vendas assim como a cotacão de valores digitais (diginotes) perante uma rede de utilizadores distribuída.

O sistema centraliza toda a informacão relevante, ou seja, as diginotes existentes, quem é o seu dono atual, todas as ofertas por cumprir e as transações realizadas com sucesso até ao momento. De notar que cada diginote existente é unica e representada pelo seu numero de identificação numa base de dados *SQLite 3* de forma a persistir a informação quando o sistema é desligado .

Em cada momento existe uma cotacão do mercado, iniciada com o valor de 1€. Qualquer utilizador que pretenda comprar ou vender diginotes à cotação atual pode emitir uma ordem de compra ou venda respetivamente, encarregando-se o sistema do emparelhamento entre compradores e vendedores.

O projecto foi desenvolvido usando a linguagem orientada a objectos *C#,* assim como a framework de comunicação .NET *Remoting*, nomeadamente para a criação da API (*application programming interface*) e da comunicação entre processos usada por cada cliente do sistema. Foi também usada uma tecnologia de base de dados *SQLite 3* para persistência de dados em *back-end*. O front-end foi desenvolvido usando as bibliotecas *Windows Forms*, fornecidas por defeito pela IDE *Visual Studio*. Finalmente, para gestão do projecto foi usado o sistema de controlo de versões *git*, usando como repositório para o código a plataforma *GitHub*.

2. Arquitectura

No sentido de concretizar os objectivos do projecto, a aplicação foi dividida em duas partes: a aplicação do cliente e a aplicação do servidor. A aplicação servidor permite comunicar directamente com uma base de dados SQLite3, cujo objectivo passa também por forma garantir que os dados da aplicação persistem no caso de uma falha.

Aplicação Cliente

A aplicação cliente define-se como a interface entre o utilizador e a aplicação. Faz uso de uma interface gráfica, que permite ao utilizador entrar no sistema, submeter, editar e apagar ordens que tenha na sua conta, assim como verificar valores importantes do sistema como o valor actual da cotação, quantas *diginotes* tem na sua posse, e quantas ordens de cada tipo existem no sistema.

Aplicação Servidor

a aplicação servidor serve de intermediária entre o utilizador e a base de dados. Está encarregue de operações como o registo e edição de ordens, assim como de notificar os utilizadores na eventualidade de serem efectuadas mudanças no sistema. Tem ainda a função de emparelhar ordens no sistema, dando lugar a transações.

3. Funcionamento

Para melhor identificar as funcionalidades assim descritas, mostramos abaixo alguns screenshots da interface que o utilizador dispõe para interagir com o sistema:

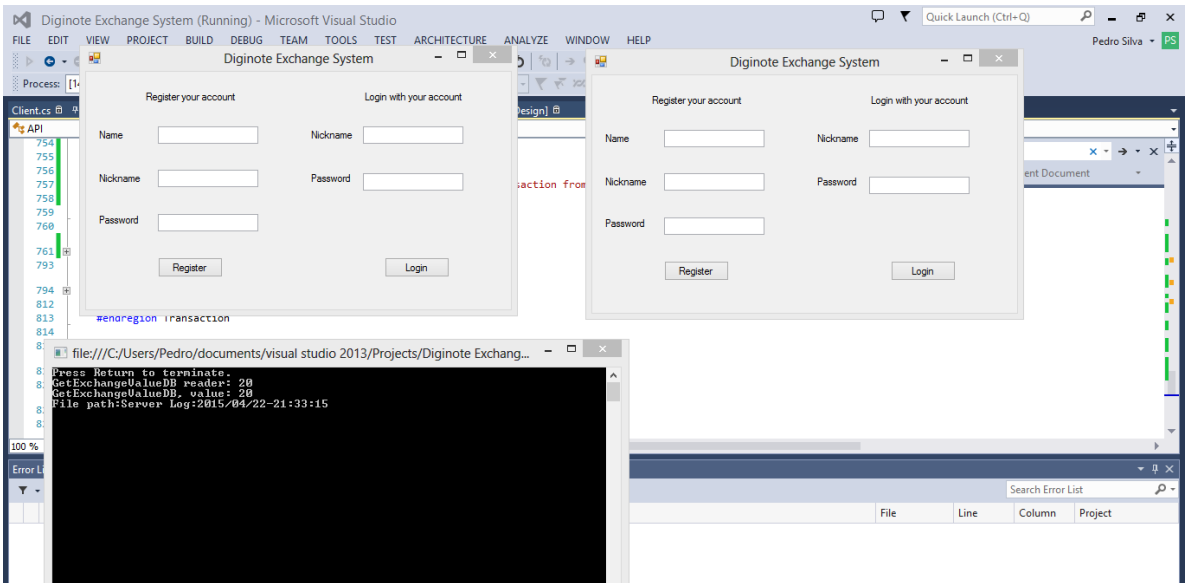

Imagem 1: Login

Este é o primeiro menu mostrado ao utilizador. Aqui, é possível criar uma nova conta fornecendo um nome, um *nickname* é uma palavra-passe, ou entrar inserindo essas credenciais. Cada utilizador recebe 100 *diginotes* quando se regista na aplicação.

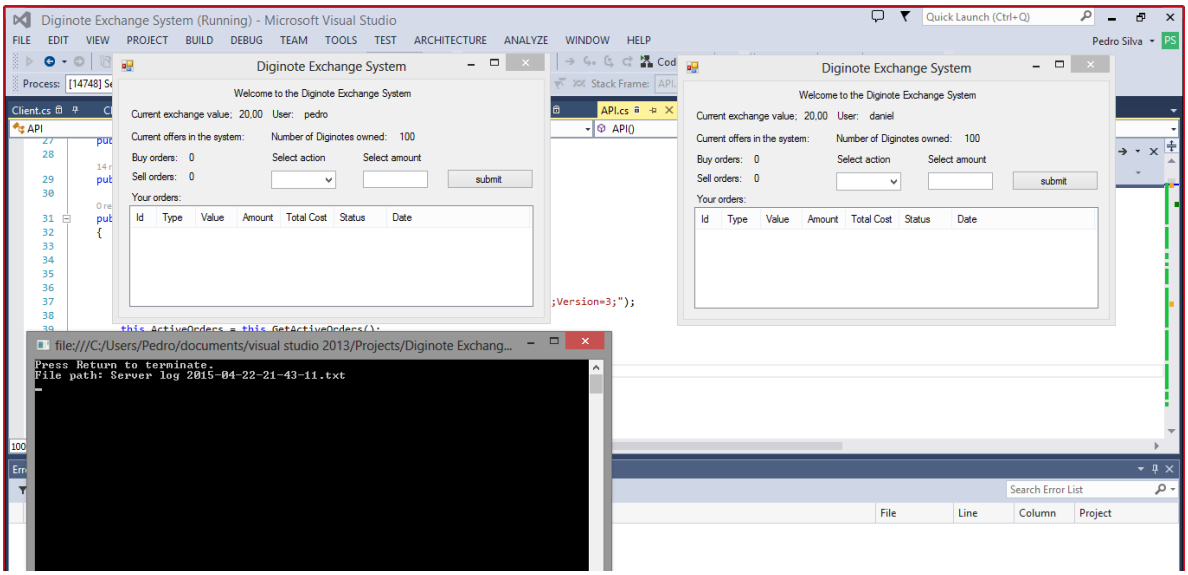

Imagem 2: Página principal

Após o login, o utilizador é apresentado com o ecrã principal da aplicação, onde estão listadas informações importantes do sistema, tais como o valor actual da cotação, quantas ordens existem de cada tipo, quantas diginotes esse utilizador possui e quais as ordens (e respectivos detalhes) tem registadas em seu nome. A partir daqui poderá criar novas ordens facilmente, escolhendo o tipo de ordem e a quantidade de diginotes que pretende vender ou comprar.

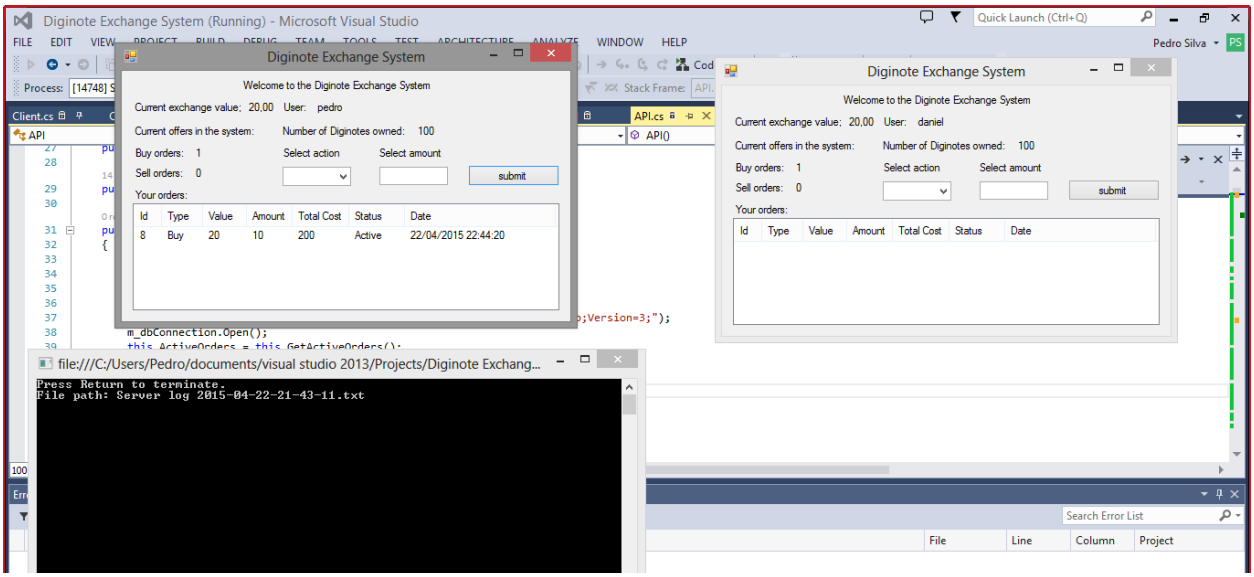

Imagem 3: Visualização de ordens criadas (no cliente do lado direito, que não é o dono da ordem criada só existe uma notificação numérica de quantas ordens existem no sistema).

Após o registo de uma ordem, esta é mostrada na lista de ordens registadas na aplicação cliente do utilizador que a autorou. Para todos os outros utilizadores, é incrementado o número de ordens que existem no sistema.

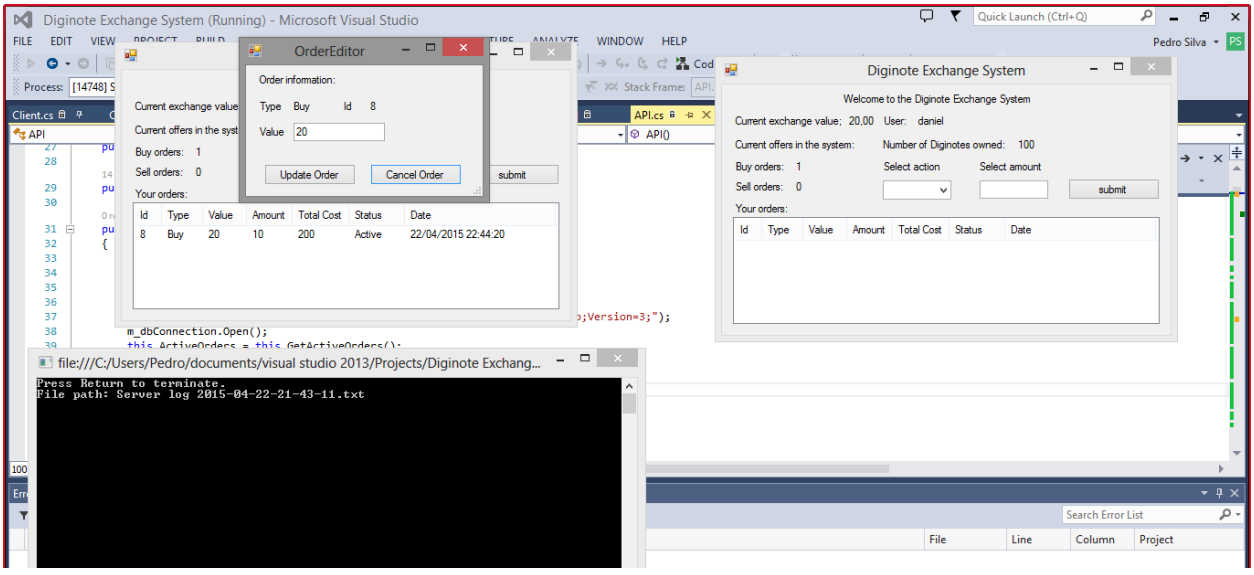

Imagem 4: Interface de edição da cotação de uma ordem

Fazer duplo clique numa ordem permite ao utilizador editar o valor desta. O novo valor deve seguir as regras de negócio estipuladas pela aplicação, sendo necessariamente um valor igual ou mais baixo se for do tipo de ordem de venda, é igual ou mais alto se for do tipo de ordem de compra. O utilizador pode ainda cancelar a ordem seleccionada.

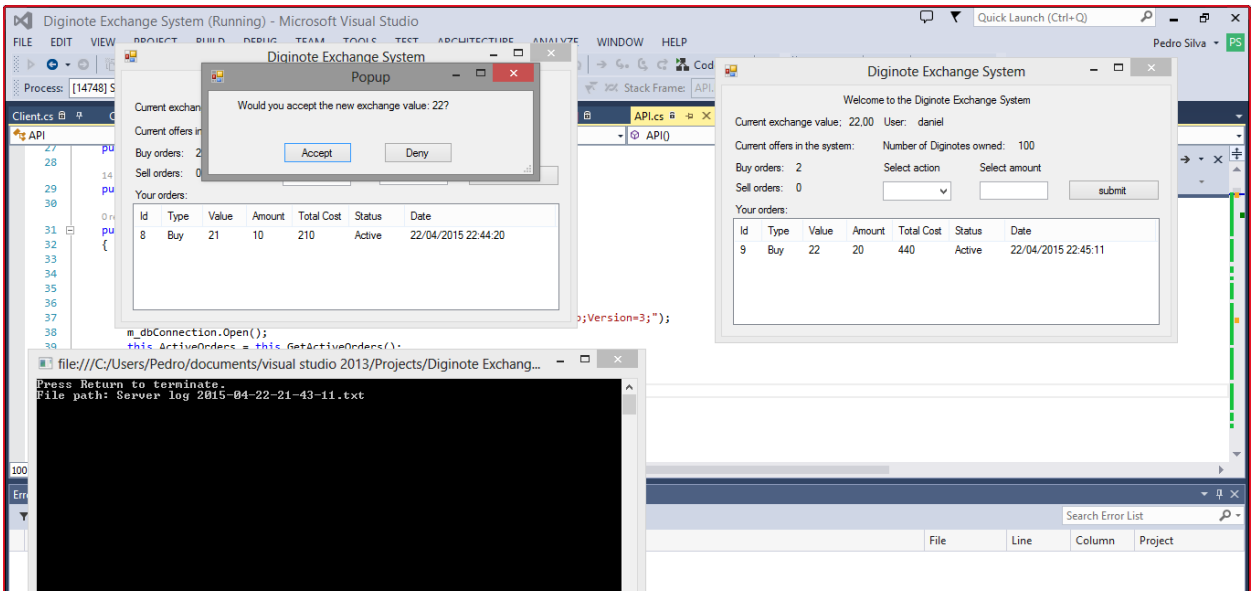

Imagem 5: UI para questionar um utilizador se pretenda ou não alterar as cotações das suas ordens para o novo valor do sistema.

Caso eleja alterar o valor da sua ordem, o valor actual da cotação irá mudar. Quaisquer utilizadores com ordens de venda de valor diferente dessa cotação são notificados, e dados a escolher entre alterar o valor de todas as suas ordens, ou de cancelar todas as ordens de valor diferente da cotação. Têm um minuto para decidir. Findo esse tempo, o sistema modifica o valor das ordens automaticamente.

4. Testes

De forma a testar o trabalho desenvolvido, foram efetuadas, de forma exaustiva, simulações nas secções mais importantes da aplicação, por forma a testar a interação cliente-servidor, analisando em cada atura a persistência e consistência dos dados manipulados. Também se procurou testar múltiplos *inputs* inválidos na interface gráfica de forma a garantir que o servidor trataria apenas de lógica de negócio válida e consistente com a mostrada ao utilizador.

5. Conclusão

O produto final apresentado contempla os objectivos descritos no enunciado do projecto. Mostrou-se particularmente complicado aliar as diversas interacções entre o cliente e o servidor, dando por vezes lugar a casos em que se verificavam chamadas recursivas que impediam o funcionamento correcto da aplicação. Porém, após um planeamento mais adequado, foi possível criar soluções que resolviam esses problemas.

6. Bibliografia

- Distribution and Integration Technologies, Miguel Monteiro, Faculdade de Engenharia da Universidade do Porto, http://paginas.fe.up.pt/~apm/TDIN/.
- .NET Remoting Overview, https://msdn.microsoft.com/library/kwdt6w2k%28v=VS.71%29.aspx
- Stack Overflow (para procura de soluções pontuais).

7. Software

- Visual Studio 2013 Ultimate, Microsoft, http://www.visualstudio.com/.
- SQLite, http://www.sqlite.org/.
- SQLiteClient, Community, http://www.nuget.org/packages/Community.
- NuGet, http://www.nuget.org/.
- GitHub, http://github.com/.