



FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Diginote Exchange System

Mestrado Integrado em Engenharia Informática e
Computação

Tecnologias de Distribuição e Integração
(TDIN)

Grupo:

David Azevedo - up201405846

Eduardo Leite - gei12068

Tiago Filipe - up201610655

Faculdade de Engenharia da Universidade do Porto Rua Roberto Frias,
sn, 4200-465 Porto, Portugal

11 de Abril de 2018

Índice

1. Introdução	2
2. Arquitetura	2
2.1 Descrição	2
2.2 Servidor	3
2.3 Cliente	3
2.4 Base de Dados	3
3. Funcionalidades	4
3.1 Autenticação	4
3.2 Obter a taxa atual	5
3.3 Emitir ordem	5
3.5 Incrementar e decrementar preço da ordem	6
3.6 Transações	7
4. Testes	8
4.1 Login	8
4.2 Registo	8
4.3 Compra e venda de uma ordem	9
4.4 Persistência de dados	9
5. Demonstração	10
6. Conclusão	12

1. Introdução

O projecto foi desenvolvido no âmbito da unidade curricular de Tecnologias de Distribuição e Integração pertencente ao Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

Este trabalho consiste numa aplicação distribuída baseada em .NET Remoting, que permite aos utilizadores comprar e vender uma taxa de valores digitais, denominado de diginotes. A informação relativa às diginotes está centralizada num sistema, como será abordado no próximo capítulo.

2. Arquitetura

2.1 Descrição

A arquitetura do sistema consiste numa estrutura cliente-servidor, onde cada cliente possui uma interface gráfica intuitiva e de fácil utilização, que comunica com o servidor central através de *remote objects* pertencentes ao mesmo. De forma, a prevenir uma possível falha do sistema, o servidor persiste todos os dados.

A figura seguinte representa a arquitetura do sistema:

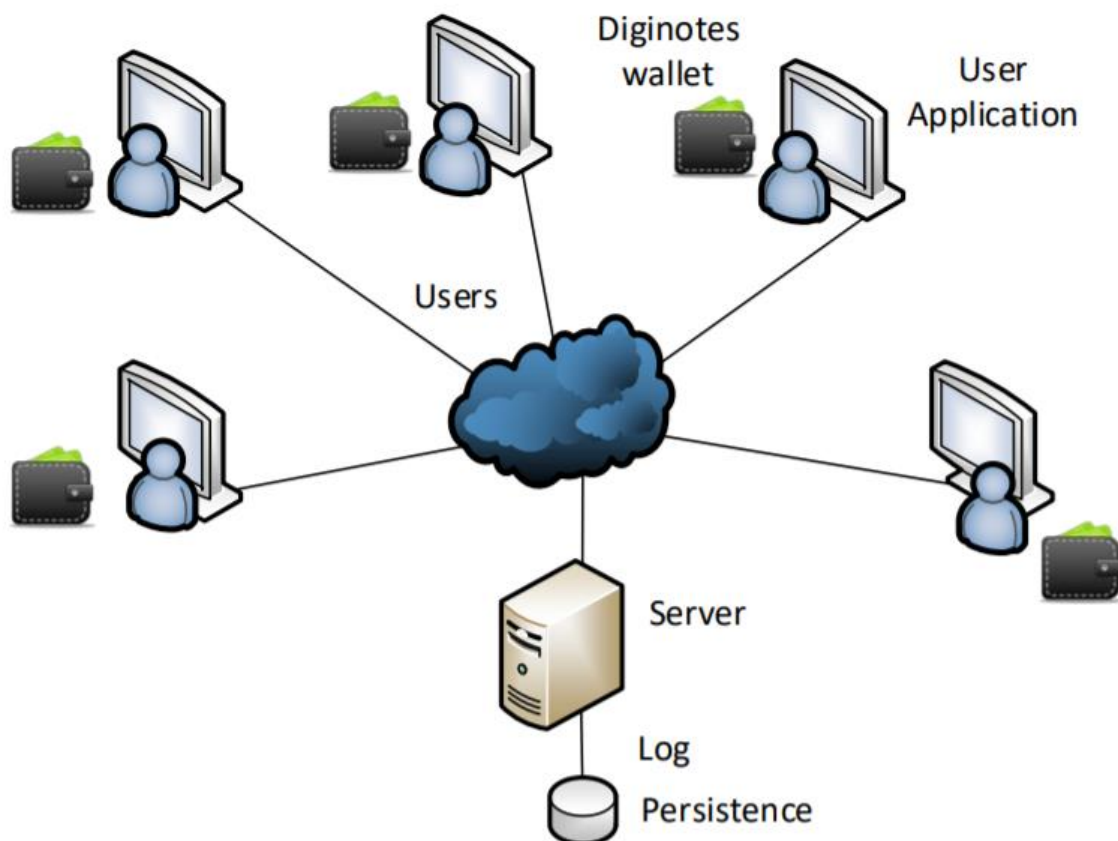


Figura 1 - Arquitetura do sistema

Os próximos tópicos descreverão com mais detalhe cada componente desta arquitetura.

2.2 Servidor

O servidor é responsável por todas as operações de persistência de dados e implementa o objeto remoto utilizado para a comunicação com os clientes. Esse objeto remoto, denominado DiginoteSystem, é que gere todos os eventos da aplicação.

Para além disso, o servidor tem como referência a assembly Shared.dll, que contém a interface do objeto remoto e as seguintes classes da lógica do sistema: Order, PurchaseOrder, SellingOrder, Transaction e Diginote.

2.3 Cliente

O lado do cliente foi implementado com uma GUI utilizando o Windows Forms, de forma a proporcionar um acesso simples e amigável. Sendo o projecto em .NET Remoting, tal como o servidor, o projecto do cliente também referencia a assembly Shared.dll descrita no tópico anterior, tirando assim partido das classes do sistema e do objeto remoto.

2.4 Base de Dados

Sobre a persistência de dados do sistema utilizou-se uma base de dados em SQL com a seguinte estrutura:

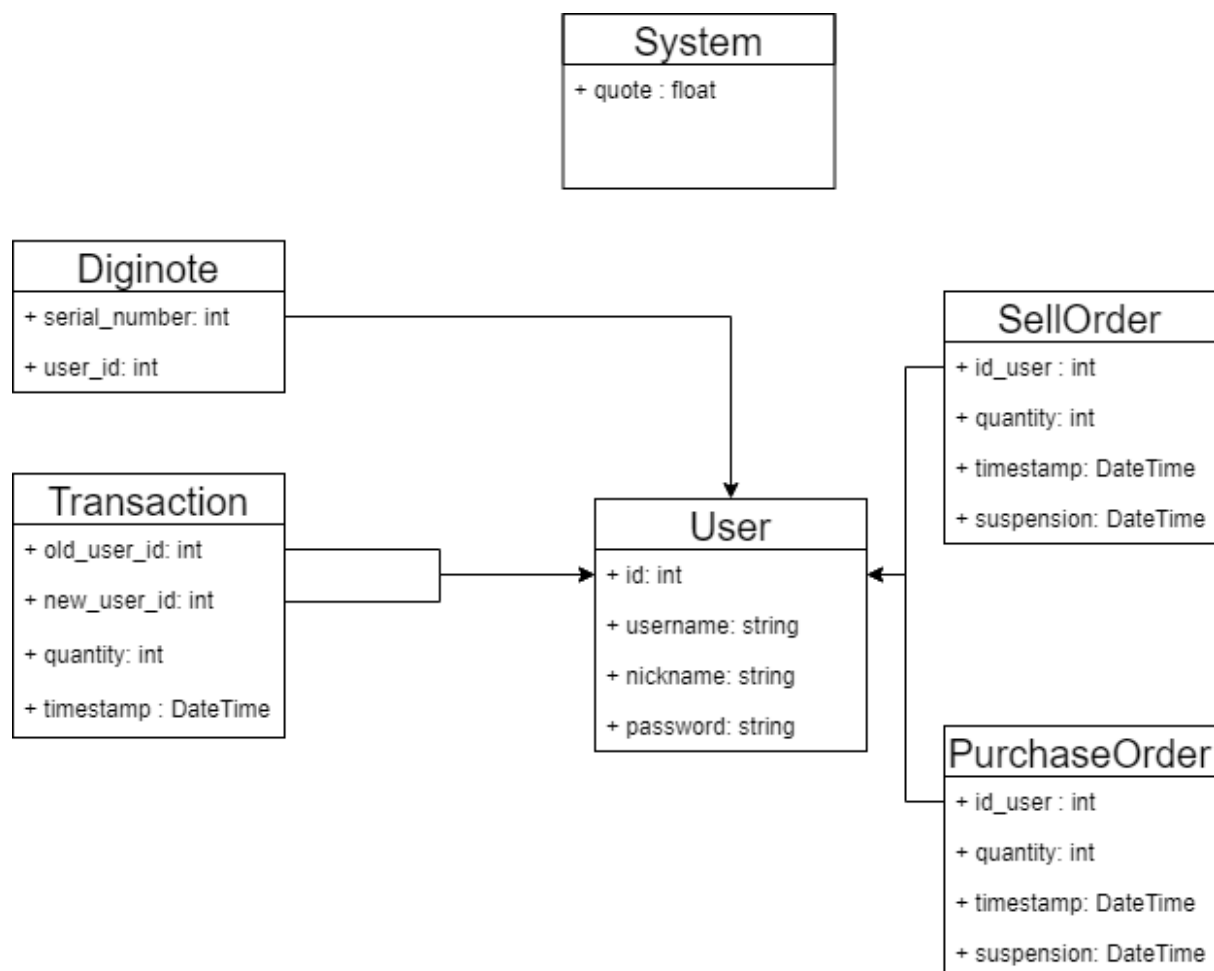


Figura 2 - Modelo Relacional

3. Funcionalidades

3.1 Autenticação

Relativamente ao sistema de autenticação, o utilizador pode registar-se no sistema e fazer o respectivo login e logout. Para o registo é necessário o preenchimento de um formulário com os campos nickname, username e password, como se pode visualizar na figura seguinte:

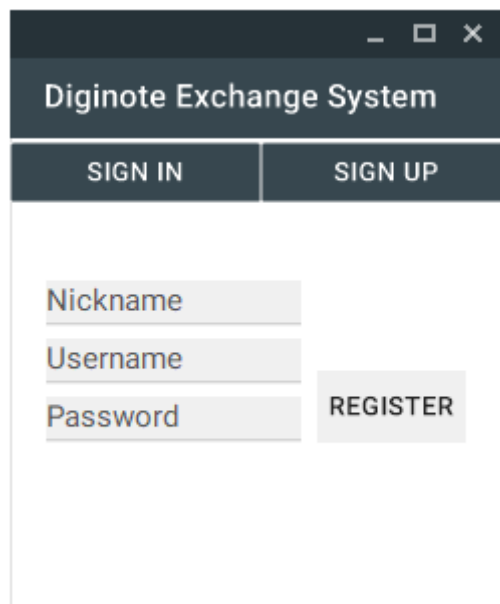
The image shows a web application window titled "Diginote Exchange System". At the top, there are two buttons: "SIGN IN" and "SIGN UP". Below these, there are three input fields labeled "Nickname", "Username", and "Password". To the right of the "Password" field is a button labeled "REGISTER". The interface is clean and modern, with a dark header and light input fields.

Figura 3 – Autenticação do Cliente

Para se efectuar o login é necessário introduzir o username e a password. Após a validação do login por parte do sistema, o utilizador é redireccionado para a seguinte página:

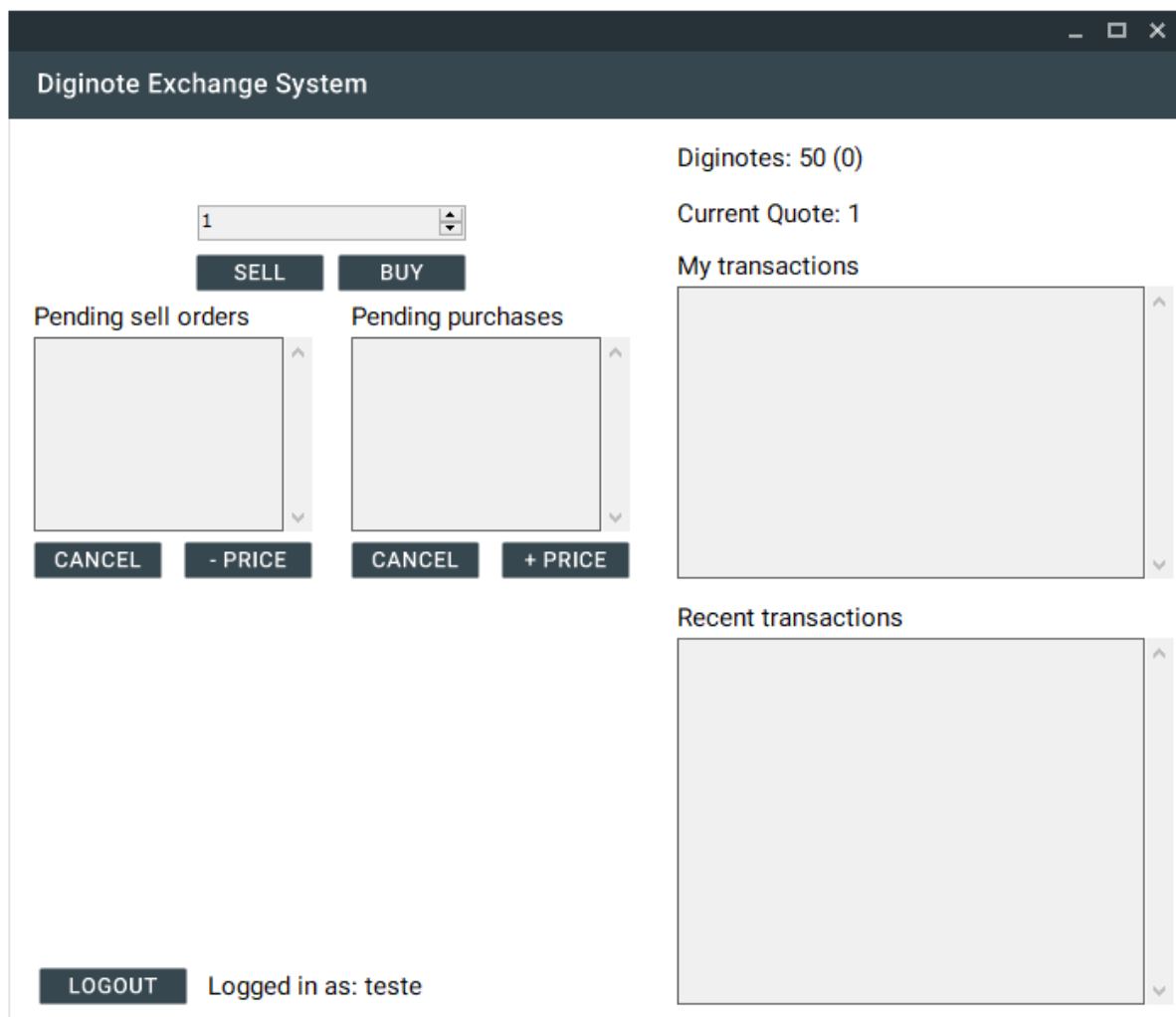


Figura 4 - Dashboard do Cliente

3.2 Obter a taxa atual

O utilizador a qualquer momento pode verificar qual é a taxa atual do sistema. Para isso, basta verificar o valor do campo "Current Quote" da página da aplicação, como se pode ver na Figura 4.

3.3 Emitir ordem

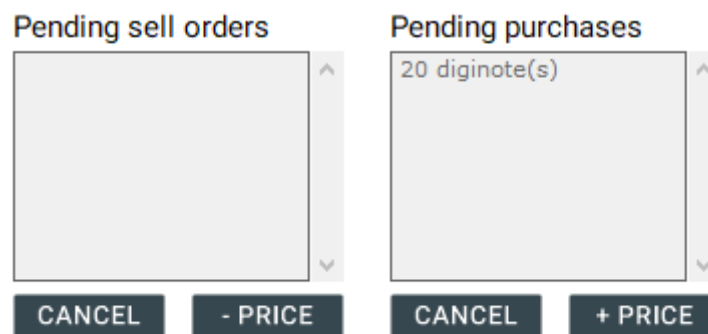
No que se refere à emissão de ordens, podem ser ordens de compra ou venda de diginotes. Em que o utilizador insere a quantidade de diginotes que quer vender ou comprar, como mostra na imagem seguinte:



Figura 5 - Emissão de uma ordem

Caso existam ordens no sistema suficientes para satisfazer o pedido, o sistema devolve uma mensagem a informar do sucesso da ação e o número de diginotes do utilizador é atualizado.

Caso não seja possível satisfazer, o sistema devolve uma mensagem a informar do sucedido e a quantidade não satisfeita passa a ser apresentada nas ordens pendentes do utilizador:



No diginote was bought. Pending...

Figura 6 - Ordens pendentes

3.5 Incrementar e decrementar preço da ordem

Em caso de o emissor de uma ordem conter ordens no estado pendente, pode a qualquer momento alterar o preço da ordem. A única restrição existente é que para as ordens de compra e venda o preço tem de ser, respectivamente, superior e inferior à taxa atual.

Essa alteração é feita através dos botões “Price” que se encontram abaixo da lista de ordens pendentes, que despoletam a seguinte janela:

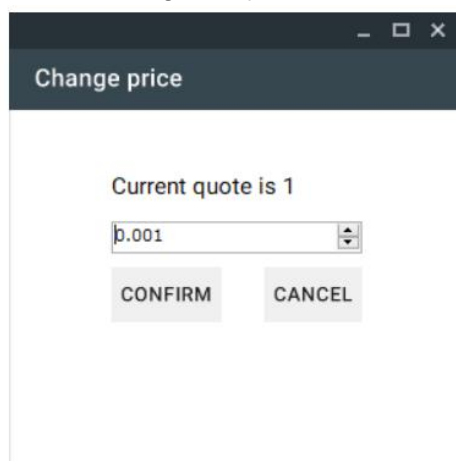


Figura 7 - Alterar preço da ordem

Devido ao facto do novo preço passar a ser a nova taxa do sistema, os utilizadores que tenham ordens pendentes podem confirmar a nova taxa ou cancelar a ordem. Este último caso é efetuado através da janela seguinte:

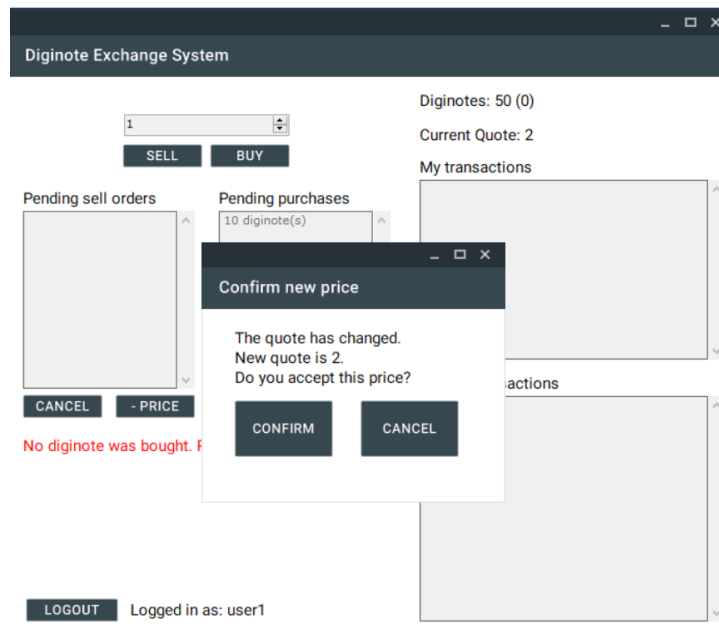


Figura 8 - Confirmação da nova taxa

3.6 Transações

As transações permitem um registo no sistema de todas as trocas de diginotes que ocorreram, sendo que na interface do cliente existem duas listagens de transações: as mais recentes que ocorreram no sistema e todas as que estão relacionadas com o utilizador.

A figura seguinte representa um exemplo de ambas as listas:

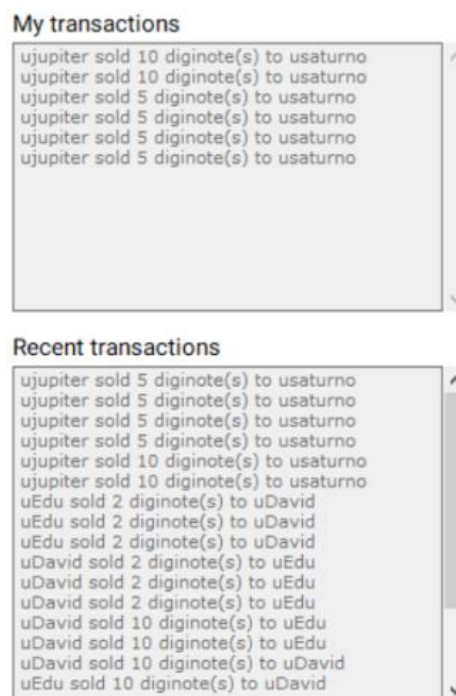


Figura 9 - Listagem das transações

3.7 Boas práticas .NET Remoting

Neste projeto, o grupo teve o cuidado de implementar as boas práticas associadas ao desenvolvimento de aplicações em .NET Remoting. As práticas utilizadas incluem:

- Utilização de um ficheiro de configuração. Com isto, é possível alterar alguns parâmetros do programa (endereço, porta, entre outros) sem que seja necessário recompilar os programas.
- Separação de código partilhado numa assembly diferente. No nosso caso, a assembly "Shared" contém a implementação da interface do DiginoteSystem, as classes relativas aos dados do sistema (user, order e transaction) e ainda a definição dos delegates e do Event Repeater.
- Uso do Event Repeater para respeitar as restrições do compilador. Desta forma, o servidor conhece a função que o seu evento irá chamar (função do repeater).
- Práticas associadas ao uso de interface gráficas com Windows Forms, tal como especificado no exemplo do professor (".NET Remoting - Changing the graphical user interface from a remote event").

4. Testes

Neste capítulo serão descritos os testes de aceitação que foram realizados durante a implementação do projecto. Cada tabela descreve um teste referindo o procedimento e o critério de aceitação do mesmo.

4.1 Login

Passo	Procedimento	Critério de aceitação
1	Inserir um nome de utilizador não existente.	A página mantém-se no login e apresenta uma mensagem de erro a informar o insucesso do login.
2	Inserir um nome de utilizador existente e a password errada.	A página mantém-se no login e apresenta uma mensagem de erro a informar o insucesso do login.
3	Inserir dados de utilizador corretos.	O utilizador é redireccionado para a página principal da aplicação.

4.2 Registo

Passo	Procedimento	Critério de aceitação
1	Não preencher por completo o formulário de registo.	A página mantém-se no registo e apresenta uma mensagem de erro a informar que não podem haver inputs vazios.
2	Inserir um username já existente no sistema.	A página mantém-se no registo e apresenta uma mensagem de erro a informar que o username já existe no sistema.

4.3 Compra e venda de uma ordem

Passo	Procedimento	Critério de aceitação
1	Inserir uma quantidade de venda superior ao número de diginotes que o utilizador contém.	O sistema impede a ação e informa que o utilizador não tem diginotes suficientes.
2	Utilizador 1 insere que pretende comprar 30 diginotes.	Tendo em conta que ainda ninguém tentou vender diginotes, o sistema coloca a ordem com o estado pendente.
3	Utilizador 2 insere que pretende vender 20 diginotes.	O sistema informa o Utilizador 2 do sucesso da operação e atualiza as suas diginotes. Para além disso, informa o Utilizador 1 que conseguiu comprar 20 diginotes, atualizando a sua ordem pendente para 10 diginotes. A lista de transações também é atualizada nos dois utilizadores.

4.4 Persistência de dados

Passo	Procedimento	Critério de aceitação
1	Após o login, não tendo ordens pendentes, verificar as diginotes que contém e fazer logout.	O sistema guardar os dados referentes ao utilizador.
2	Efetuar login novamente e verificar se o número de diginotes se mantém.	O sistema carregar corretamente todos os dados referentes ao utilizador que existiam antes de ele terminar sessão.

5. Demonstração

Para a iniciação do projecto é necessário criar uma base de dados em SQL através do SQL Server Management Studio e executar o script que se encontra na pasta do projecto para todas as tabelas serem devidamente criadas. Após isso, executa-se o Servidor, deste modo fica tudo preparado para qualquer Cliente que se execute.

O ciclo da aplicação do Cliente segue a sequência das funcionalidades que foram descritas anteriormente.

Em primeiro lugar, o utilizador regista-se e entra no sistema, como apresentado na Figura 3.

A partir deste momento, o utilizador possui um conjunto de diginotes e pode comprar outras diginotes ou vender as suas. Para demonstração, o cenário é o Utilizador 1 ter 50 diginotes e querer comprar mais 20 e o Utilizador 2 estar a vender 15 diginotes. As figuras seguintes representam este cenário:

The screenshot displays the 'Diginote Exchange System' interface. At the top, a dark header bar contains the title 'Diginote Exchange System'. Below the header, the interface is divided into several sections. On the left, there is a dropdown menu showing '1', with 'SELL' and 'BUY' buttons below it. To the right of these buttons, the text 'Diginotes: 50 (0)' and 'Current Quote: 1' is displayed. Below the dropdown, there are two main panels: 'Pending sell orders' and 'Pending purchases'. The 'Pending purchases' panel shows a list with '20 diginote(s)' and a 'CANCEL' button. Below this panel, there are buttons for '- PRICE' and '+ PRICE'. To the right of these panels, there is a section titled 'My transactions' with a large empty box. Below this, there is a section titled 'Recent transactions' with another large empty box. At the bottom left, there is a 'LOGOUT' button and the text 'Logged in as: user1'. A red error message 'No diginote was bought. Pending...' is visible in the center of the interface.

Figura 10 - Demonstração de uma ordem de compra

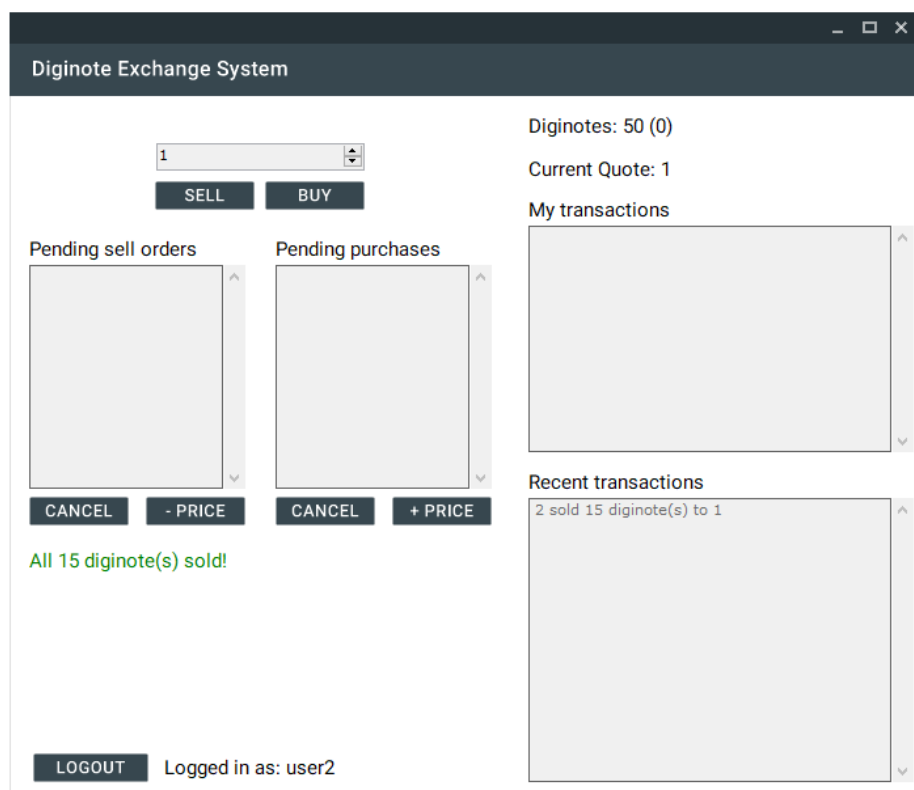


Figura 11 – Demonstração de uma ordem de venda

Como se verifica na Figura 10 a ordem de compra do Utilizador 1 ficou pendente, visto que naquele momento o sistema não tinha ordens de venda pendentes para poder satisfazer o pedido. De seguida (Figura 11), o Utilizador 2 coloca à venda algumas diginotes que fazem com que parte do pedido do Utilizador 1 e o seu pedido sejam satisfeitos. Como a ordem do Utilizador 1 não foi completamente satisfeita a ordem continua pendente, mas a quantidade é atualizada, como mostra a imagem seguinte:

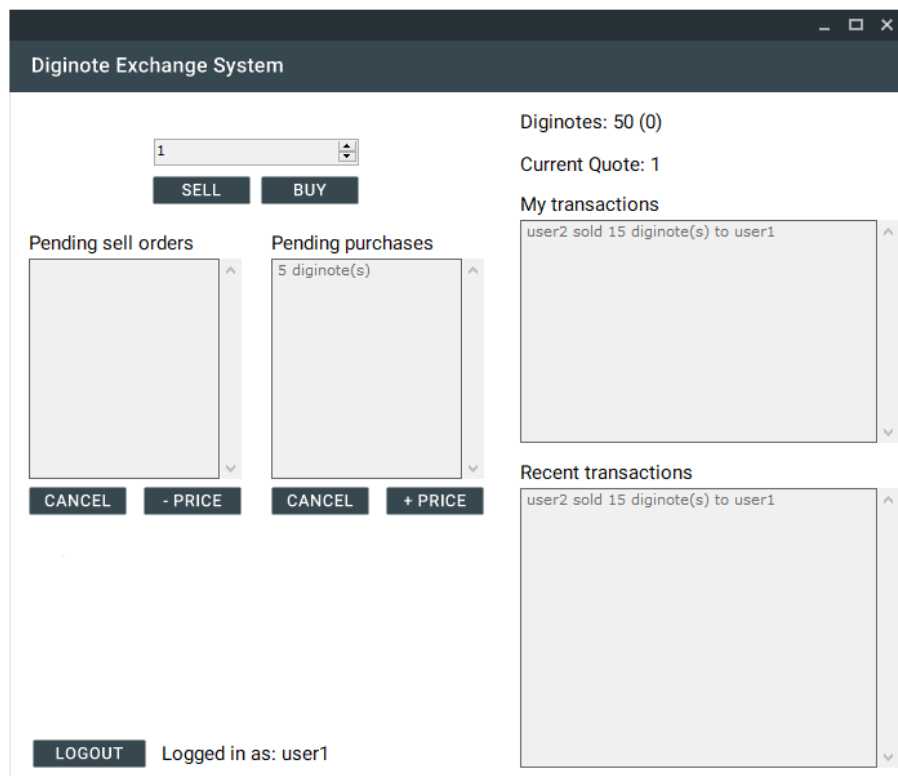


Figura 12 – Ordem pendente atualizada

Quando um utilizador tem uma ordem pendente pode incrementar ou decrementar a taxa do sistema, quando isso sucede, se mais utilizadores tiverem ordens pendentes ficam com a ordem suspensa durante 1 minuto para terem a oportunidade de a cancelar, visto que a taxa do sistema sofreu alterações. Neste caso é despoletada a janela representada na Figura 8.

Em suma, o ciclo da aplicação é o descrito acima, sendo que vários utilizadores podem estar ligados ao sistema e a emitir várias ordens ao mesmo tempo.

6. Conclusão

Este projecto permitiu colocar em prática os conhecimentos da linguagem de programação C# e ficar com uma noção mais clara da utilidade e do mecanismo do .NET Remoting. Inicialmente, houve uma pequena dificuldade com a ligação do cliente com o servidor através dos objectos remotos, mas após a primeira conexão com sucesso o trabalho desenvolveu a um bom ritmo.

Conclui-se que o projecto teve um balanço muito positivo, visto que houve uma aprendizagem crescente, uma boa comunicação entre o grupo e todas as funcionalidades do trabalho foram implementadas.