Distribution and Integration Technologies

ASSIGNMENT 1: DIGINOTE EXCHANGE SYSTEM

Professor António Miguel Pontes Pimenta Monteiro

Ruben Fernando Pinto Cordeiro – ei11097
Duarte Nuno Pereira Duarte – ei11101
April 21, 2015

Contents

Introduction

The following report describes the implementation of a *Diginote Exchange System*. The system allows the end user to purchase and sell digital assets known as *diginotes* in a centralized marketplace.

Architecture

PHYSICAL ARQUITECTURE

The system is based on a client-server model. The client is a desktop WPF[1] application that communicates with the server via .NET remoting[2]. The server's data persistence is ensured by a data log. Each operation in the server is stored in this log. Upon startup, the server loads the system state from the log file.

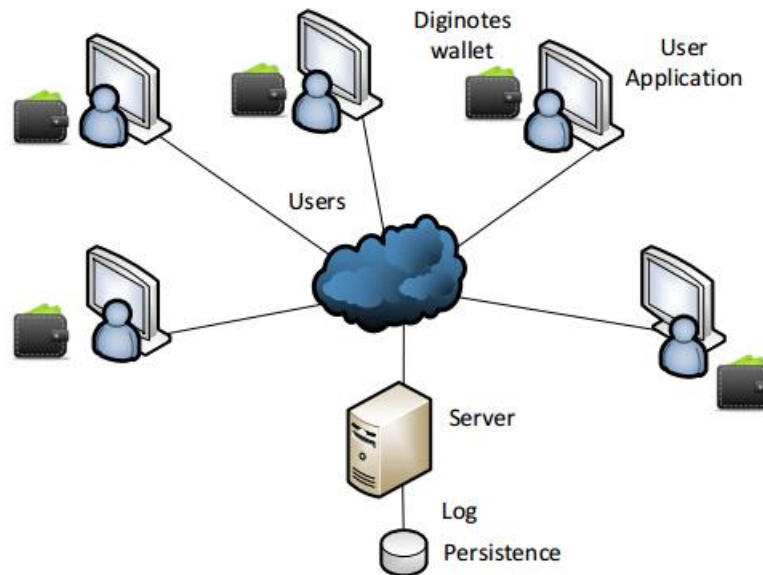A diagram with the physical architecture can be seen below:

Figure 1: Physical Architectures Diagram

---

[1] Windows Presentation Foundation: presentation system for building Windows client applications.
[2] Microsoft application programming interface (API) for interprocess communication.

LOGICAL ARCHITECTURE

Client

The client's implementation follows the Model-View-ViewModel (MVVM) pattern. This approach allows the separation of the business and presentation logic of the application from its user interface (UI). Maintaining a clean separation between application logic and UI has helped to address numerous development and design issues and made the application much easier to test, maintain, and evolve. It also greatly improved code re-use opportunities.

Using the MVVM pattern, the UI of the application and the underlying presentation and business logic is separated into three separate classes: the view, which encapsulates the UI and UI logic; the view model, which encapsulates presentation logic and state; and the model, which encapsulates the application's business logic and data, as seen in the diagram below:
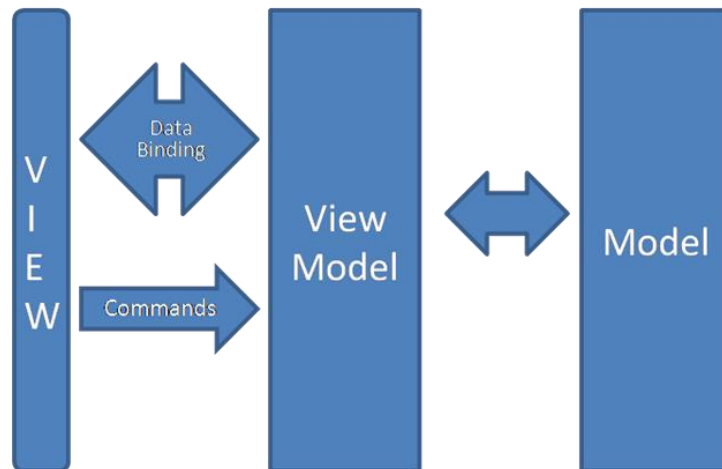

Figure 2: MVVM pattern diagram

Server

The server is implemented in .NET with .NET remoting, providing an abstract approach to interprocess communication that separates the remotable object from a specific client or server application domain and from a specific mechanism of communication.

The application is structured in three main modules: Client, Common and Server.

The Common module aggregates all the entities shared by the client and server. These entities are marshalled and sent back and forth in the network.

The diagram below describes the main remote entities:
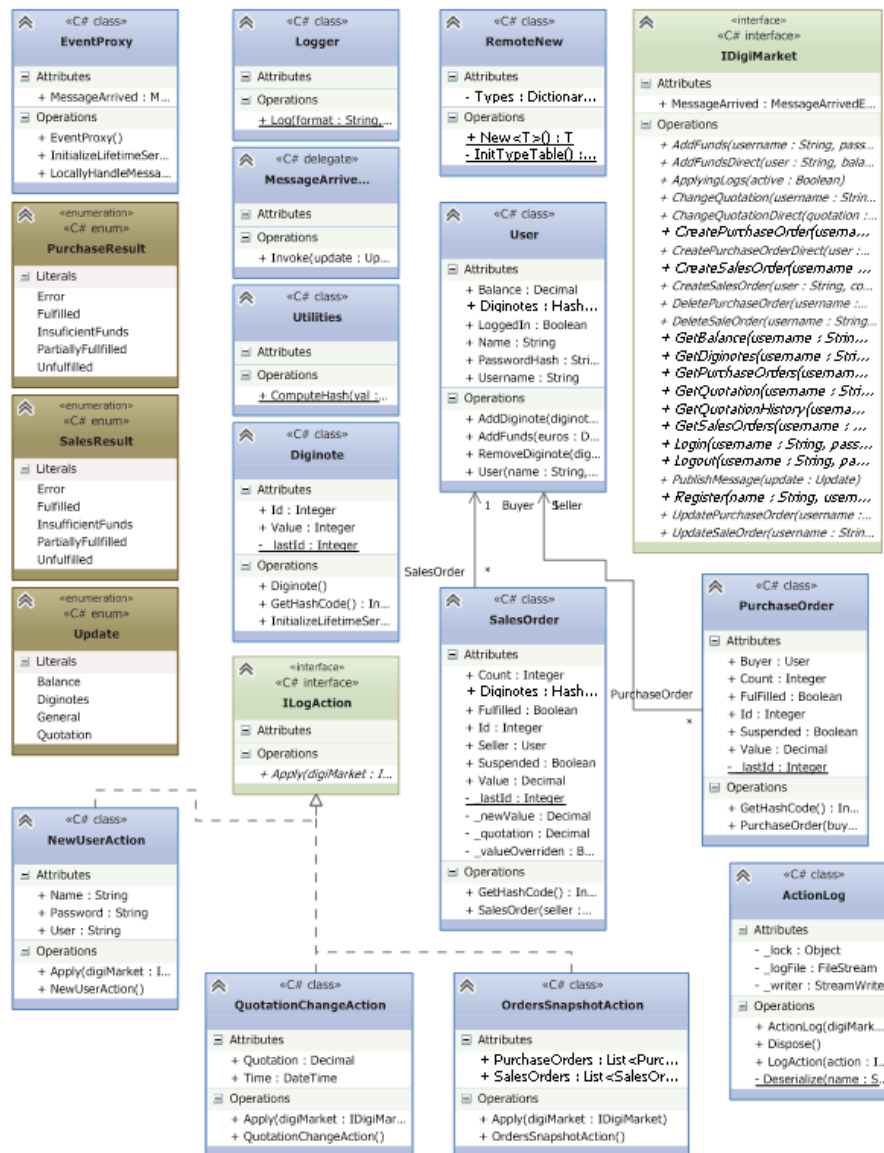
Figure 3: Remote entities class diagram

Functionalities

The functionalities of the system are detailed in the diagram below:
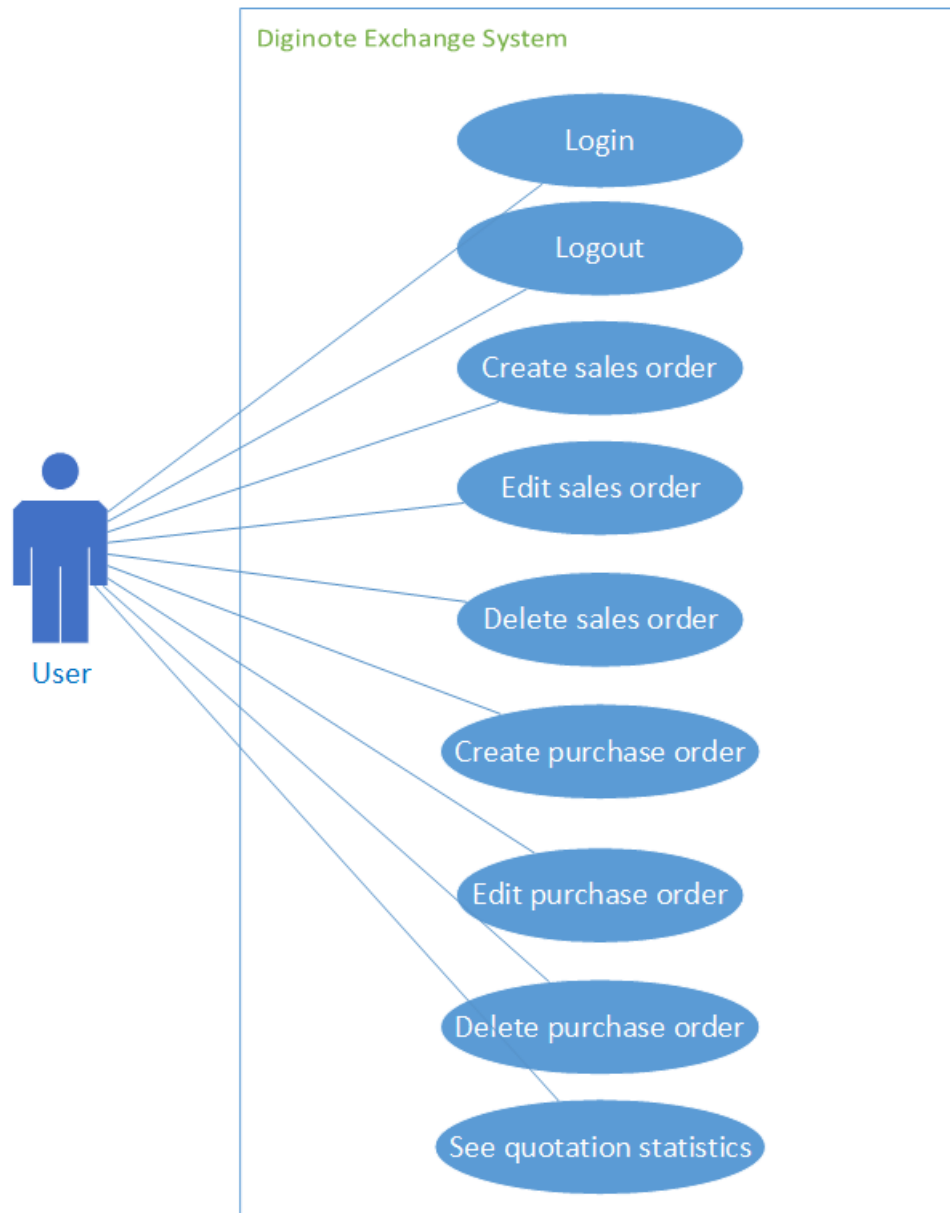

Figure 4: System use case diagram

User interfaces

All the user interfaces in the desktop client are based on Google's Material Design guidelines. Material Design is visual language for our users that synthesizes the classic principles of good design with innovation.

LOGIN/REGISTER

When the system starts, the user is presented with a login or registration screen.

Figure 5: Login/Register screen

DASHBOARD

After the user authenticates in the system, he is redirected to the dashboard screen.

The top header has four tab controls: *Info, Charts, Buy* and *Sell*.

- *Info*: The current dashboard view;
- *Charts*: Statistical information about the quotation fluctuation;
- *Buy*: Creation, edition and erasure of purchase orders;
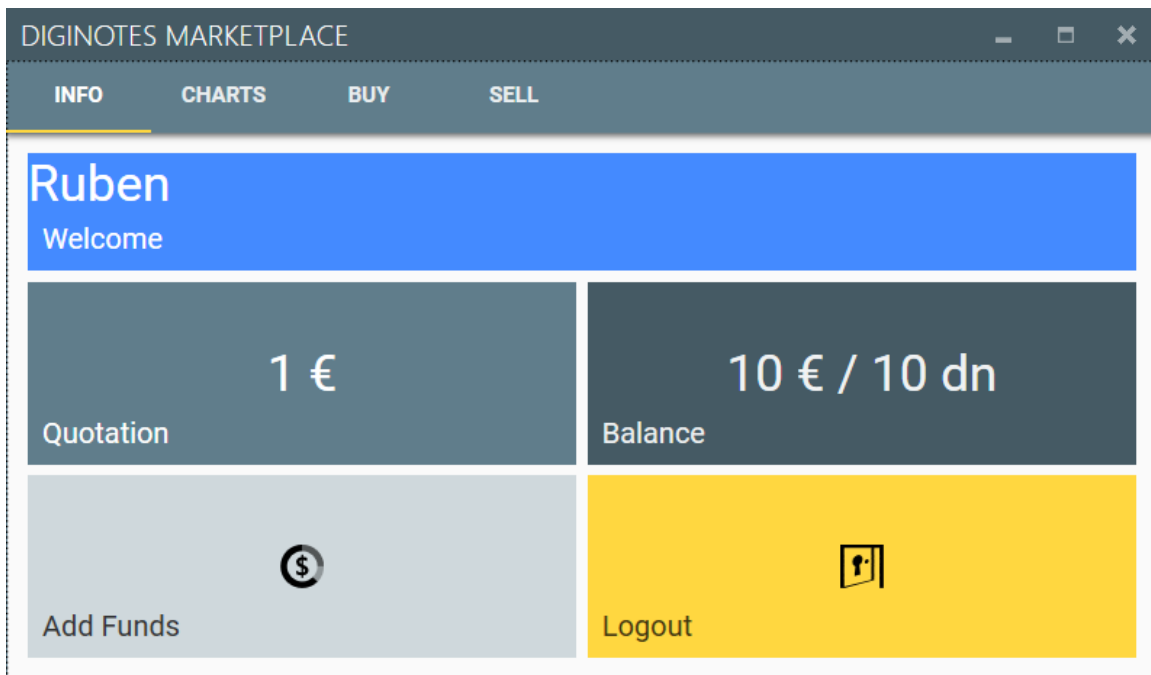- *Sell*: Creation, edition and erasure of sales orders.

Figure 6: Dashboard view

CHARTS

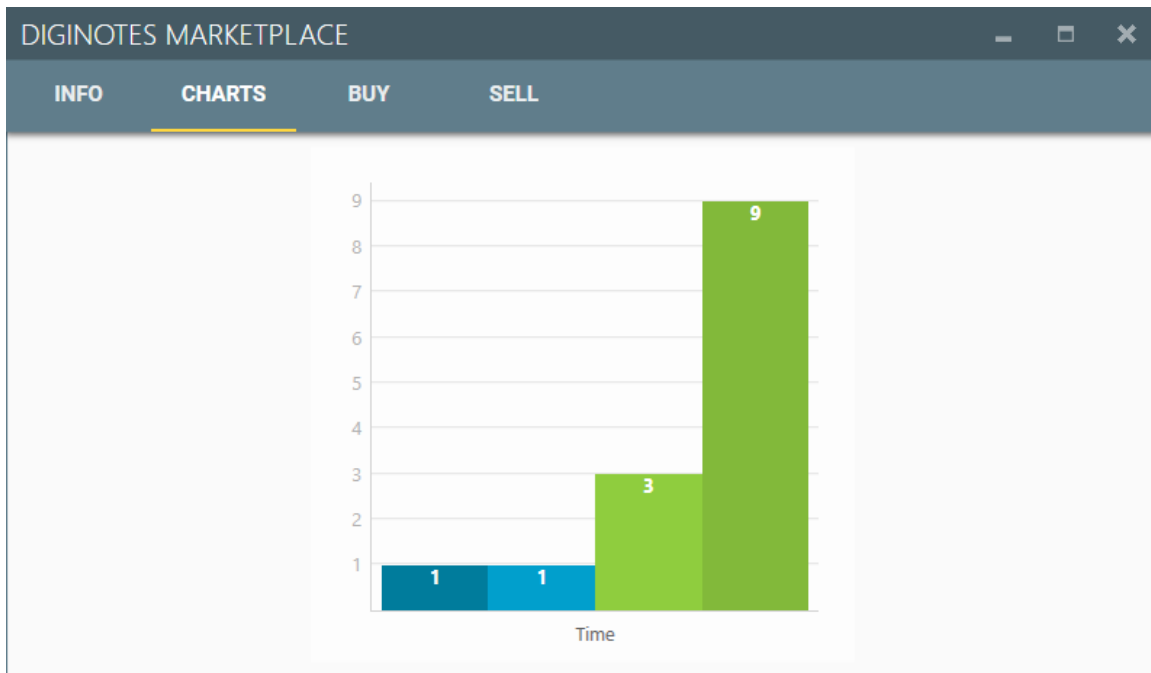The *Charts* view displays statistical information about the quotation fluctuation.

Figure 7: Charts view

BUY

The *Buy* view allows the user to create, edit and erase purchase orders.

Figure 8: Buy view

SELL

The *Sell* view allows the user to create, edit and erase sales orders.

Figure 9: Sales view