**Enterprise Application Integration**

09

**Application Integration with Message Queues**

**Assignment #4**

**Carlos Simões  
Miguel Graça Oliveira  
Pedro Saraiva**

# Introduction

In the scope of the project “Application Integration with Message Queues”, from the assignment #4 of the Enterprise Application Integration course, a set of design and implementation decisions had to be made. The goal of this report is to describe these decisions.

In addition, the report provides the instructions for deploying and executing the application.

## Time Spent

|  |  |  |
| --- | --- | --- |
| Student | Mostly involved in | Time spent |
| Carlos Simões | * User Interface application | Reading 0:00  Coding 0:00  Report 0:15 |
| Miguel Graça Oliveira |  | Reading 0:00  Coding 0:00  Report 0:00 |
| Pedro Saraiva |  | Reading 0:00  Coding 0:00  Report 0:00 |

# Installation Instructions

1. User Interface
   1. Run the program Setup.exe

# Implementation and Design decisions

## Architecture

This project is an evolution of the first project. The project goal was to integrate the three applications done in the first project using message queues.

This is the dynamic view of the imposed architecture:



The first decision was to use the Microsoft Message Queue Server (MSMQ) as the Message Queuing middleware and to use .NET as development framework. This decision was made

Decisions:

1. Give GenericRead and GenericWrite permissions to the “Everyone” user (?)
2. Create the MyOrchestrator application as a normal Windows Forms Application, instead of a Windows Service, which would be correct. This decision was taken for the sake of simplicity. In an industrial environment, this would be the most appropriate decision, since it is important that the orchestrator is always running. [DESENVOLVER] Nevertheless, all the applications [DESENVOLVER]
3. Use an Util library for common functions
4. Not to create a generic Integration Wrapper application, instead to have three different Integration Wrappers, …
5. To use the MSXLS program instead of modifying our CameraListBeautifier because
6. Integration Wrappers search the directories for newly created files to know when the blablabla

Threads:

1. To manage communication from the UserInterface application to the CameraSearchXML application.
2. To manage communication from the CameraSearchXML application to the CameraSummaryXML application and the CameraBeautifier application.
3. To manage communication from the CameraSummaryXML application and the CameraBeautifier application to the UserInterface application.

Configurations

MyOrchestrator

* EAIUserInterfaceInbox -
* EAIUserInterfaceOutbox -
* iwCameraSearchXmlInboxQueue -
* iwCameraSearchXmlOutboxQueue -
* iwCameraSummaryXmlInboxQueue -
* iwCameraSummaryXmlOutboxQueue -
* iwCameraListBeautifierInboxQueue -
* iwCameraListBeautifierOutboxQueue –

IntegratioWrapperCameraListBeautifier

* iwCameraListBeautifierInboxQueue
* iwCameraListBeautifierOutboxQueue

IntegratioWrapperCameraSummaryXML

* iwCameraSummaryXmlInboxQueue
* iwCameraSummaryXmlOutboxQueue
* JavaCameraSummaryXMLArguments
* JavaPath
* JavaCameraSummaryXMLPath

IntegratioWrapperCameraSearchXML

* iwCameraSearchXmlInboxQueue
* iwCameraSearchXmlOutboxQueue
* JavaCameraSearchXMLArguments
* JavaPath
* JavaCameraSearchXMLPath

UserInterface

* userInterfaceInboxQueue
* userInterfaceOutboxQueue