Physics Department
FACULTY OF SCIENCES
AND TECHNOLOGY
UNIVERSITY OF COIMBRA

BW-Eye

Ophthalmologic Decision Support System based on Clinical Workflow and Data Mining Techniques – Exams Importation and Visualization

Sistema de Apoio à Decisão para a especialidade de Oftalmologia baseado em técnicas de Workflow e Data Mining aplicadas em dados obtidos em ambiente clínico – Importação e Visualização de Exames

Project Report
Version 1.0

Rui Miguel Filipe Maricato – Nb. 2003125445

September 1, 2008

Centro Cirúrgico de Coimbra

BlueWorks- Medical Expert Diagnosis, Lda

Document Revisions

Date	Version	Description	Author
2008-08-18	v0.1	Document creation	Rui Maricato
2008-08-28	v0.2	Document update after the Project supervisor revision	Rui Maricato
2008-08-29	V1.0	Document conclusion	Rui Maricato

Acknowledgments

Now that this journey has reached its end, I need to show my gratitude to everyone that made this project possible.

To Dr. António Travassos and Eng. José Basílio, for the opportunity to work in this project.

To Eng. Armanda Santos, Eng. Edgar Ferreira and Eng. Paulo Barbeiro for all their help, guidance and availability, and for the pleasant working environment that they provided.

To my colleagues, Liliana Ferraz and Ricardo Martins, for all the friendship that they have shown.

To my friends, for their friendship.

To my family, for helping me to become who I am today.

To Mónica, for all her love.

To God, for everything.

Abstract

Medicine practice is a subjective issue; however, by saving, structuring and analysing the patient therapeutic data in the most complete possible way, it is possible to reduce that subjectivity and aid the doctor on medical decision.

The project analyzed in this report was developed within the Project class of the Biomedical Engineering course of the Faculty of Sciences and Technology of the University of Coimbra. The project was developed by three students, however the present report describes the tasks performed by the student Rui Maricato.

The project objective is to develop a software application to improve the ophthalmologic clinical practice process, by creating a decision support system and consequently reduce the subjectivity associated with medical practice. In order for this objective to be achieved, there is the need to integrate examination equipments, organize exams in a database, and visualize and interact with them, such that image enhancement algorithms and data-mining techniques could be used. This application is known as BlueWorks – Medical Examination Suite (BW-MES), and its development started last year.

Two components of the BW-MES were developed in this project: the module to visualize and interact with the database exams, as well as the tool to import exam files from unintegrated examination equipment.

The visualization module support features and the exams importation tool were fully developed and tested; hence, a first functional version of these tools is available. The visualization module advanced features (that allow the user to apply image enhancement algorithms) are still in development.

Keywords:

C -Sharp, Decision-Support, Exams Importation, Ophthalmology, Visualization Tools

Resumo

A prática de medicina é inerentemente subjectiva; porém, ao guardar, estruturar e analisar os dados terapêuticos da forma mais completa possível, poderá diminuir-se essa subjectividade e assim auxiliar a decisão médica.

O projecto analisado neste relatório foi desenvolvido na disciplina de Projecto do curso de Engenharia Biomédica da Faculdade de Ciências e Tecnologia da Universidade de Coimbra. O projecto foi desenvolvido por três alunos, porém este relatório descreve apenas o trabalho realizado pelo aluno Rui Maricato.

O objectivo do projecto consiste no desenvolvimento de uma aplicação para melhorar o processo da prática clínica da oftalmologia, criando um sistema de apoio à decisão e assim diminuir a subjectividade inerente à prática da medicina. O desenvolvimento desta aplicação passa por integrar equipamentos de exame, organizar exames numa base de dados, e visualizar e interagir com os exames, de modo a que algoritmos de processamento de imagem e técnicas de data-mining possam ser aplicadas. Esta aplicação denomina-se BlueWorks – Medical Examination Suite (BW-MES), e o seu desenvolvimento iniciou-se no ano passado.

Neste projecto foram desenvolvidas duas componentes da aplicação BW-MES: o módulo para visualizar e interagir com os exames registados na base de dados, e a ferramenta para importar ficheiros de exame de equipamento não-integrado.

As funcionalidades de suporte do módulo de visualização e a ferramenta de importação de exames foram completamente desenvolvidas e testadas, pelo que uma primeira versão funcional destas ferramentas está disponível. As funcionalidades avançadas do módulo de visualização (que permitem ao utilizador aplicar algoritmos de processamento de imagem) ainda se encontram em desenvolvimento.

Palavras-chave:
Apoio à decisão, C -Sharp, Ferramentas de visualização, Importação de exames, Oftalmologia

Contents

Figures Index

Diagrams Index

Tables Index

Definitions and Acronyms

Acronym	Description
BW-MES	BlueWorks – Medical Examination Suite
CCC	Centro Cirúrgico Coimbra
C#	C-Sharp
DB	Database
DLL	Dynamic Link Library
ICG	Indocyanine green
ISA	Intelligent Sensing Anywhere, Lda
JPEG	Joint Photographic Experts Group
MI	Main Interface
M-VIS	Visualization Module
M-VIS:AFC	Visualization Module: Advanced Feature Component
M-VIS:SF	Visualization Module: Support Features
OCT	Optical Coherence Tomography
OCT-SLO	Optical Coherence Tomography - Scanning Laser Ophthalmoscope
OD	Ocular Dextra
OS	Ocular Sinistra
PDF	Portable Document Format
PNG	Portable Network Graphics
SQL	Structured Query Language
XML	Extensible Markup Language
XSD	XML Schema Definition

1. Introduction

The medical practice objective is to take care of every human being in the best possible way. Consequently, it is in constant evolution, always trying to use the most advanced technologies available. As a result of this search for constant improvement, the next phase on medicine improvement will be to take advantage of all the computing power that is available nowadays.

Some steps have already been taken towards this objective. As an example, one can mention the implementation of Electronic Patient Records (EPR), which is a tool to save and structure therapeutic data (e.g. images, videos, ECG and EEG data, laboratory results), making it available to advanced processing.

However, the BW-Eye project intends to take one step further: to create a decision support system, by joining human knowledge and computing power, and thus reducing the great fraction of subjectivity that stills surrounds the medical practice nowadays (that could be caused by such factors as the doctor knowledge or experience).

The following tasks have to be performed, in order for this step to be taken:

- Optimize the clinical practice workflow (the collection of steps and data that define element's paths that can be taken to complete a task, where these elements might be goods, persons or information; workflows may contain activities such as displaying content to users, collecting information from users or computer systems, performing calculations, and sending messages to external computer systems [1]);

- Integrate examination equipments in order to save exams in a database;

- Manage the exams and apply data-mining techniques (an analytic process designed to explore large amounts of data in search of

consistent patterns and systematic relationships between variables [2]) to the data saved, thus giving to the doctor more information to rely on.

Some work has already been done last year (on the "Ophthalmologic Decision Support System based on Clinical Workflow and Data Mining Techniques" project) and a software application (**BW-MES**: BlueWorks - Medical Examination Suite) to improve ophthalmologic clinical practice was conceptualized and its development began (more information relating to BW-MES can be found in section *2.2. Last year's project achievements*); in this project two more software components were developed, specifically:

- Exams visualization support features, namely patients search, exams browsing and visualization. This application objective is to allow the user to visualize the data saved in the database.

- Exams importer, which is a tool that registers in the database the exams made by examination equipments that cannot be integrated with the main application.

1.1. Domain

This project is integrated in the Biomedical Engineering Course of the Faculty of Sciences and Technology of the University of Coimbra. It started on September 14th 2007 and its ending matches this document's delivery date: September 1th 2008.

The corporate entities involved in this project are C.C.C. (Centro Cirúrgico de Coimbra), BlueWorks (BlueWorks –Medical Expert Diagnosis) and I.S.A. (Intelligent Sensing Anywhere).

1.2. Report Overview

This report is divided in four components.

First, a background on the subject is given; then, the work done in this project is explained. Next, the most important challenges that were met are enumerated, and lastly, the conclusions and future objectives are explained.

1.3. Project Team

This project trainee students were Liliana Ferraz, Ricardo Martins and Rui Maricato, with the help and support of a large team, listed on Table 1.

Table 1 - Project team members

Name	Designation	Contact
Liliana Ferraz	Trainee Student	lilianajsferraz@gmail.com
Ricardo Martins	Trainee Student	ricardo.f.a.martins@gmail.com
Rui Maricato	Trainee Student	rui.maricato@gmail.com
Prof. José Basílio Simões	Project Coordinator	jbasilio@isa.pt
Dr. António Travassos	Supervisor	info@ccci.pt
Eng. Lara Osório	Supervisor	losorio@isa.pt
Eng. Jorge Saraiva	Supervisor	jsaraiva@isa.pt
Eng. Armanda Santos	Engineering Collaborator	asantos@blueworks.pt
Eng. Edgar Ferreira	Engineering Collaborator	eferreira@blueworks.pt
Eng. Paulo Barbeiro	Engineering Collaborator	pbarbeiro@blueworks.pt

1.4. Planning

The Table 2 shows the main tasks description and duration. A detailed schedule of the develop work stages is shown in Appendix 1.

Table 2 - Project Main Tasks Schedule

Index	Task	Duration	Start	Finish
1	M-VIS:AF – Architecture and requisites definitions	38 days	01-10-2007	21-11-2007
2	M-VIS – Creation of a functional prototype	32 days	01-10-2007	13-11-2007
3	M-VIS:AF – Creation of the module structure	35 days	14-11-2007	01-01-2007
4	M-VIS:AF – Implementation of methods for exams visualization	48 days	02-01-2008	07-03-2008
5	M-VIS:AF – Graphical design update	5 days	10-03-2008	14-03-2008
6	M-VIS:AF – Video support implementation	5 days	17-03-2008	21-03-2008
7	M:VIS:Importer – Importation of all the CCC images	5 days	24-03-2008	28-03-2008
8	M-VIS – Tests and implementation of the module	9 days	31-03-2008	10-04-2008

	first functional version			
9	M-VIS – User training	1 day	11-04-2008	11-04-2008
10	M-VIS:AFC – Parameters definition	5 days	14-04-2008	08-04-2008
11	M-VIS:AFC – Control creation	15 days	21-04-2008	09-05-2008
12	M-VIS:AFC – Implementation of the AFC in the M-VIS	5 days	12-05-2008	16-05-2008
13	M-VIS:AFC – Developments of interactivity tools	5 days	19-05-2008	23-05-2008
14	M-VIS:AFC – Tests	10 days	26-05-2008	06-06-2008
15	M-VIS:AFC – Implementation and user training	5 days	09-06-2008	13-06-2008

2. Background

Since this project is a continuity project (relating to the "Ophthalmologic Decision Support System based on Clinical Workflow and Data Mining Techniques" project that started last year), to fully understand it there is the need to explain what was done before, in order to contextualize it.

Therefore, one will explain the CCC situation before the start of the project, what was done during last year and its conclusions, which form the starting point of this year's work. More information on this topic can be found in [3], [4] and [5].

2.1. Last year's project starting point

In the beginning of the last year's project, the CCC clinical process was studied and evaluated, in order to optimize it.

From that study, some conclusions about the clinical process were achieved, such as:

- The CCC clinical process proved to be highly dependent on paper, with numerous forms, which consumed a lot of time and resources. For example, there was not an automatic process for requesting examinations (it was done through a paper form), and the nurses filled several paper forms for each patient that underwent surgery, which would then be copied to the clinical software system.

- There were two distinct and independent software applications, SilverInfor (administrative system - patients that went to a medical consultation would be registered on this system) and HIGIA (a clinical system closed solution - patients that were to undergo a surgery would be registered on this system); hence, the absence of a common administrative and medical computer system made the clinical entities management a complex task, since each entity on the clinical process

generated different information fluxes that were seldom connected to each other.

- Most of the examination equipments cannot share resources on a network, and are controlled by closed solutions.

- The results of the examinations from equipments that were connected to the network could be viewed on consultation rooms using SilverInfor; however, this software application featured limited functionalities to manage patient's clinical data.

2.2. Last year's project achievements

The CCC clinical process is represented in the following diagram, which represents the clinical practice current status, and served as the problem analysis starting point:

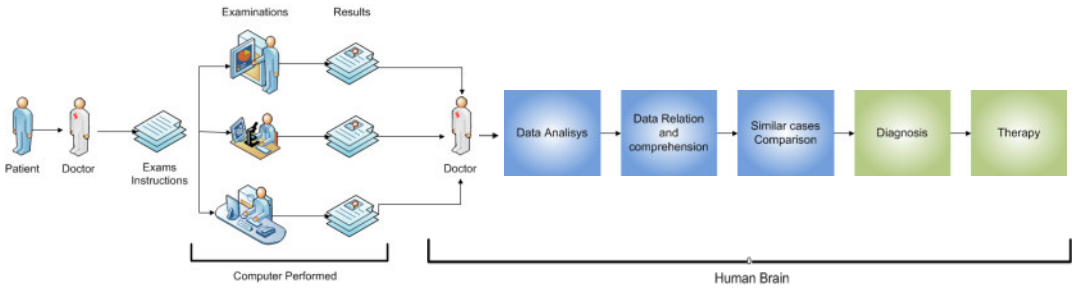


Diagram 1 – Clinical practice current status

As it can be understood from the diagram analysis, presently computational tools are underused, which leads to some amount of subjectiveness.

In order to improve clinical practice, the following model of an ideal methodology was conceptualized:

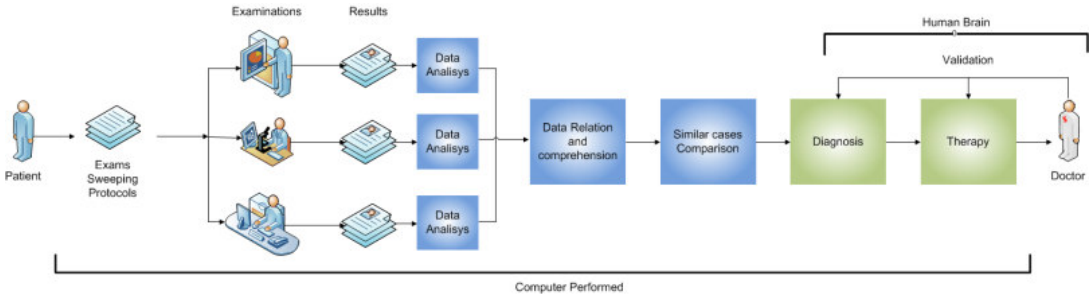


Diagram 2 – Future clinical practice model

As the diagnosis process starting point, the doctor's initially prescribed exams may be replaced by an exhaustive examination of the patient with existing equipments (in order to provide the diagnosis system with the maximum amount of information).

Next, the clinical exams will be automatically acquired and analyzed to extract useful data. This analysis will involve techniques such as image processing and neural-networks classification.

In a second stage, data gathered from different clinical exam sources will be correlated, thus determining a unique health condition status; data-mining techniques will allow to easily and extensively search a database for similar clinical cases, and to review applied therapies and outcomes.

A software tool that is capable of relating different clinical exams' distinguishing features and to match them to existing cases, will be able to provide diagnosis. Since causal relationships between specific treatments and certain diseases are well known and described, the therapeutic indication for a suggested disease is the last logical step of this process. The suggested diagnosis and therapy will then be validated by the doctor.

So, the software application would feature:

- A data management system;
- Signal and image processing, to allow the extraction of useful information from the exams;
- Automatic formulation of diagnosis and therapy.

As a consequence of this methodology and the heavy use of computational tools, it is possible to measure and detect characteristics otherwise ignored; also, the detailed extraction of information allows a better monitoring of patient health stats, disease and therapy evolution, whether computer data-mining also enables the discovery of patterns and relations yet unknown.

In order to implement the conceptual model, a software application was planned. The main objectives of this application are to integrate the diverse examination equipments that exist on CCC, save the examinations results on a database and browse and interact with the saved exams. This application is known as BlueWorks – Medical Examination Suite (**BW-MES**), and was developed on C-Sharp (C#) language and Microsoft Visual Studio environment, using a SQL Server 2005 database.

The BW-MES uses a modular structure (through DLL plugins), that has the great advantage of easily allowing the application improvement and expansion.

The application's architecture is represented on the following diagram:

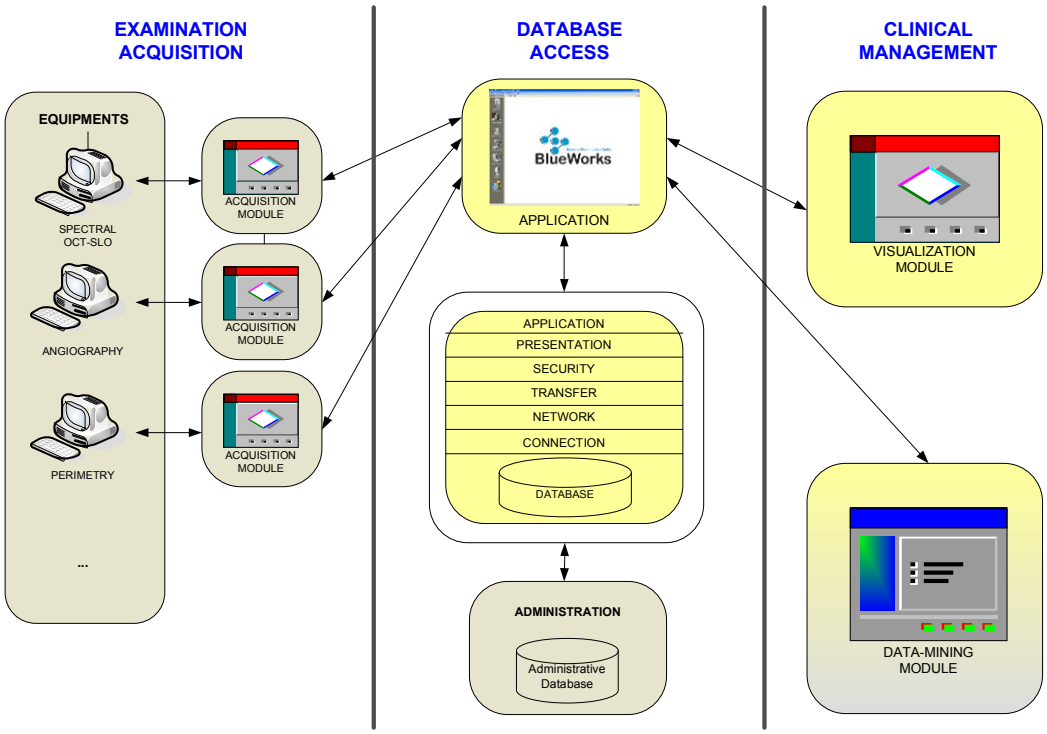


Diagram 3 – BW-MES software architecture

The architecture is divided in three main layers:

- Exams importation: modules that import images (and other parameters) from the equipments software, as well as the creation and saving of the respective exam report.

- Database access: the database access is done through the Main Interface (MI), which supports the modules and connects to DB where all the information relating to doctors, patients and exams needed to the management and intelligent processing of exams data is saved. The importation of health professionals and patients is automatic from the administrative database.

- Exams management: module that allows the exams visualization and images automatic processing. The doctors interact directly with this section, and it is where the application's most advanced features are planned to be hosted.

Besides the development of BW-MES, some work on image processing algorithms was also done (such as algorithms to detect serous detachment, oedema and macular hole) together with some work on neural networks.

3. Problem Analysis

On the beginning of this project, the work that was already done was analyzed, and it was concluded that the best solution would be to divide the work into differentiated tasks, for the BW-MES to become more complete, and consequently more easily commercialized.

The priority work would be to create more integration modules (more specifically the Angiography Module, done by Liliana Ferraz) and to create the Visualization Module (done by Rui Maricato and Ricardo Martins). Later on, Liliana Ferraz worked in some image processing algorithms.

The Visualization Module (M-VIS) is an important part of the BW-MES, not only due to its role as a patients, exams and exams elements browser, but also because it will feature the image processing tools and the connection with the data-mining module.

The M-VIS has three core objectives, specifically:

- Import to the DB the examinations results done with equipment that cannot be integrated in the BW-MES. This tool is known as the **M-VIS: Exams Importer**.

- Provide an interface that allows the user to connect to the DB, search for one specific patient, view the patient's associated exams, choose one and browse through the diverse exams results (such as reports, images and videos). These features compose the **M-VIS: Support Features (M-VIS: SF)**;

- Allow the user to interact with the exams results in an innovative way, using image processing features and neural networks, which hopefully will bring new perspectives to ophthalmology. These features compose the **M-VIS: Advanced Features Component (M-VIS: AFC)**.

Between the two students responsible for the **M-VIS**, the work was divided in the following way:

- Ricardo Martins: development of an image registration algorithm.

- Rui Maricato: develop the M-VIS support features (connections to database, exams organization, graphical aspect, usability) and the exams importer tool.

4. Visualization Module – M-VIS

The Visualization Module (M-VIS) is a BW-MES component that allows the user to browse the database exams and interact with them.

4.1. Dataflow diagram

The M-VIS dataflow diagram is represented in Diagram 4:

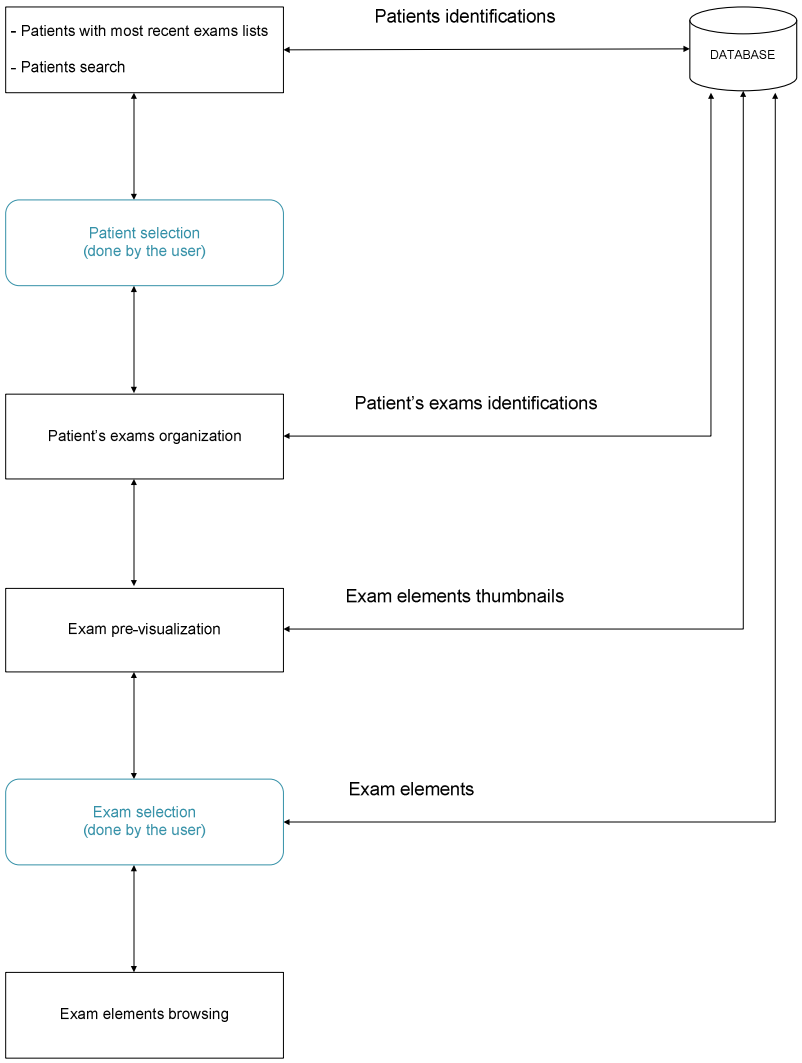


Diagram 4 - M-VIS dataflow diagram

4.2. Software components

The **M-VIS** is divided in three main components:

→ **M-VIS: Exams Importer**

→ **M-VIS: Support Features**, which is divided in:

- **Patients search**, that features:
  - **Patients direct search**
  - **Last exams registered on the database listing**

- **Browser**, that features:
  - **Exams Browser**, that features:
    - **Exams List**
    - **Exam Elements Pre-Visualization**
  - **Exam Elements Browser**, that features:
    - **Exam Elements List**
    - **Display Area**
    - **Exam Textual Information**

→ **M-VIS: Advanced Features Component**

- **Images List**
- **Display Area**
- **Advanced Operation Component**

4.3. Functional Prototype

After the software structure was created, the next step was to create a functional prototype (Figure 1).

The functional prototype purpose was to be shown to the project supervisors and then receive their feedback about it; besides that, the prototype proved to be a good way to get familiarized with Visual Studio and C#, since these subjects are not taught in the Biomedical Engineering Course.

The prototype allowed the user to browse through exams, see the selected exam images thumbnails, open an exam, browse through the opened exam images, and also featured an **AFC** example (Figure 2).

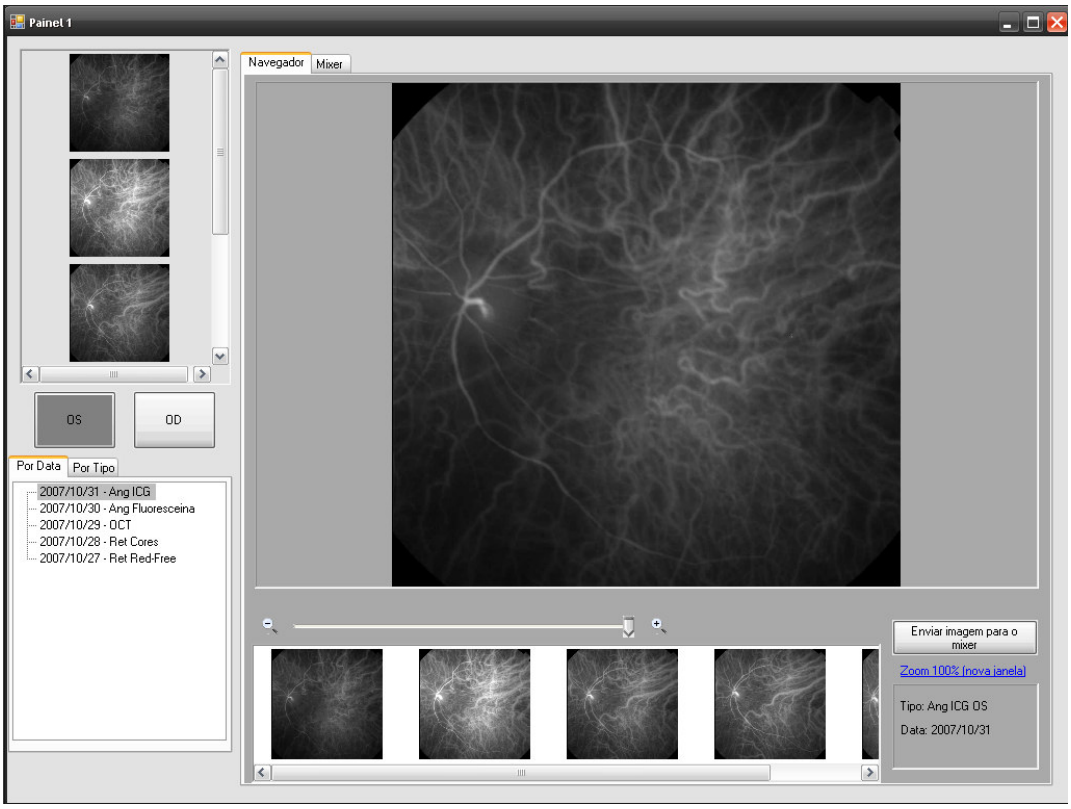


Figure 1 - Functional Prototype: Browser

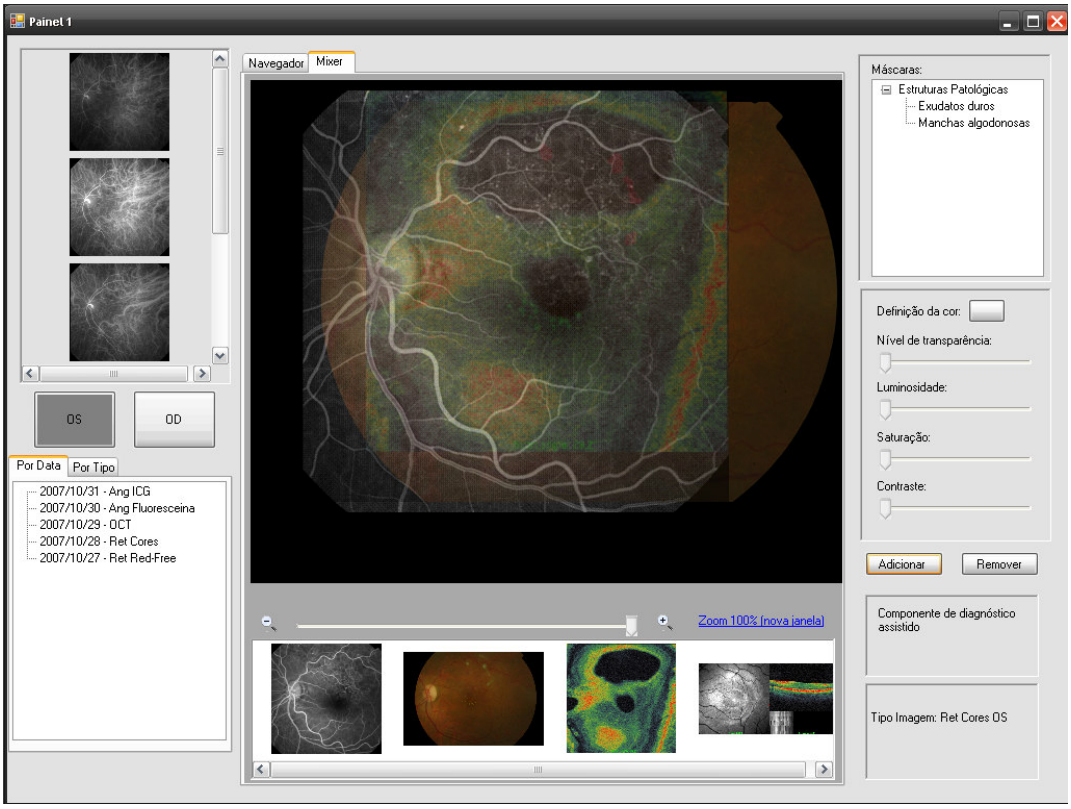


Figure 2 - Functional Prototype: AFC example

4.4. M-VIS: Support Features

4.4.1. Connection with the Database

The connections to the database are made by the **Main Interface**; since the visualization module is connected to the **MI** (through a *plug-in* system), it will use the available database connections.

4.4.2. Patient Selection

The patient selection is the module's first screen (Figure 3), therefore constituting the user's first contact with the module; its purpose is to allow the user to search for the patient whose exams are to be accessed.

There are two main possibilities to find a specific patient:

- Patients' direct search

- Listing of the patients with the most recent exams registered in the DB (this list is splitted in two lists: one list with only the patients that are associated with the doctor, and the other list with all patients)

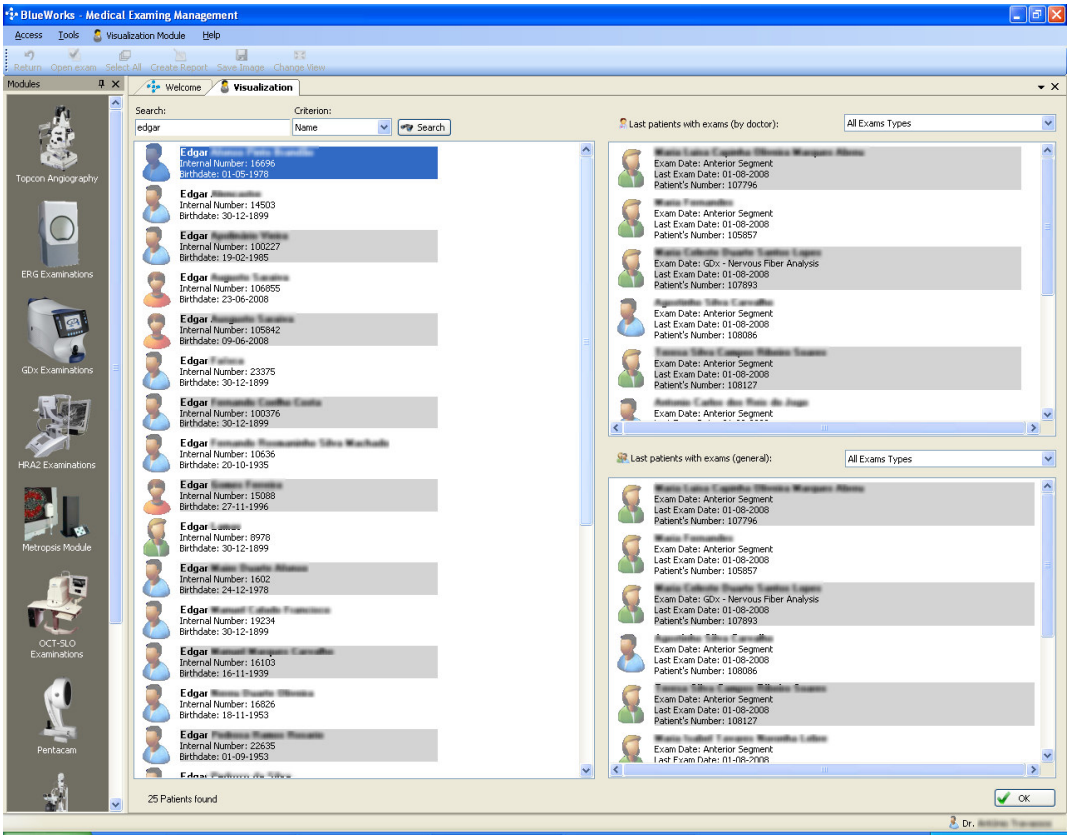


Figure 3 - Patients selection screen

For each patient it is featured an image (related to the patient's sex and age), and the patient's name and internal number. In the listing of the patients with the most recent exams registered in the DB it is also presented the patient's last exam date.

This window is organized in three visual components, so that the user can easily distinguish between the three available features.

For the user commodity, this window features two *split containers*, allowing the user to re-dimension the direct search list width and the last patients lists width (through a vertical *split container* between the direct search list and the last patients lists) and the lists heights (through a horizontal *split container* between the two last patients lists).

Patients' direct search

The patients' direct search allows the user to search for a specific patient. It is directly adapted from code already available on other BW-MES components, and features the possibility of search through diverse criteria, namely:

- Patient's name;
- Patient's internal number;
- Patient's health system number;
- Patient's citizen ID;
- Patient's contributor number;
- Patient's birthdate;
- Patient's registry date;

An important feature that had to be developed (and that had to be maintained in the code adjustment to the M-VIS) is related to the number of results returned by a search; should the number of results be greater than fifty, a warning message appears, asking the user if he/she wants to continue. This feature was implemented because it took a great deal of time to search and list more than fifty results, and because it simplifies the process of choosing a patient.

Last exams registered in the database listing

The list of the last exams registered in the database is a feature that simplifies and speeds the use of the module; there is a great probability that the user wants to see the last exams registered in the database, hence it is of great utility to present them in a simple and immediate way.

This list is splitted in two lists; one features the patients with the most recent exams that were registered in the database, while the other only presents the results that belong to patients associated with the doctor using the software (the association between a patient and a doctor is created when a doctor prescribes an exam to a patient).

Besides that, there is the possibility of filtering the lists results by exam type through the *combobox* on the top of each list.

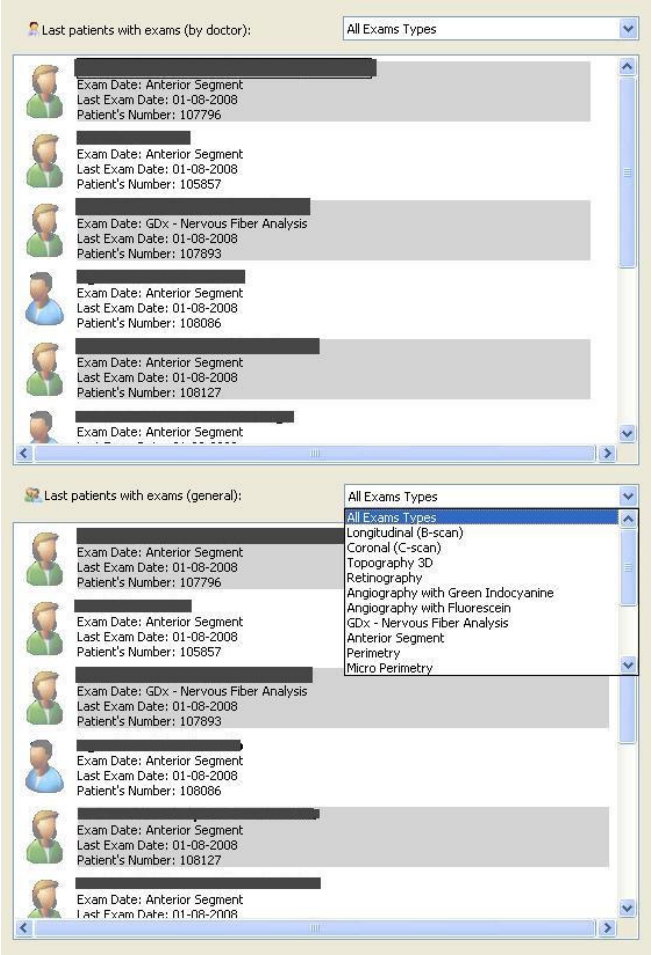


Figure 4 - Most recent exams lists

For the implementation of this feature it was necessary to:

- Create database functions that return the patients with the last exams registered in the DB, accordingly to the exam type and the doctor using the module;

- Create the *listviews*, the *listviews* items and the respective tags, in order to identify the patients that each item refers to;

- Create the ways the user can interact with these elements, namely the selection of an item, the double-click, the confirmation button and the keys interaction.

Patient selection confirmation

The confirmation of a patient selection can be done either through a double-click in the patient's item or (after the patient's item being selected) through the confirmation button.

After a patient selection has been confirmed there is the possibility to return to the patients selection screen (and then there is the possibility to return again to the Browser screen).

For the implementation of this feature it was necessary to:

- Create the usability operations (e.g. warn the user if the selected patient does not have exams);

- Create a database function to load the patient's exams.

4.4.4. Browser

The **Browser** is the M-VIS component where the user can browse the patient's exams and (after an exam being opened) browse through the exam elements.

The Figure 5 presents a *print screen* with an open exam:

Figure 5 – Browser print screen

4.4.4.1. Exams Browser

After a patient being selected, a connection to the database is made, and all the exams related to the selected patient are retrieved; hence the retrieved exams need to be organized and presented in an intuitive way, so that the user can easily browse them and find a specific exam.

The Figure 6 presents a *print screen* of the **Exams Browser**:

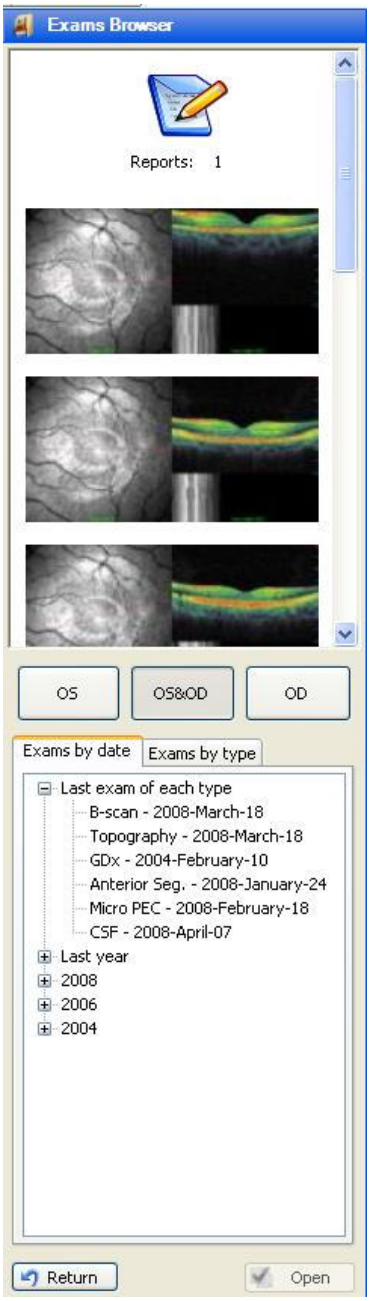


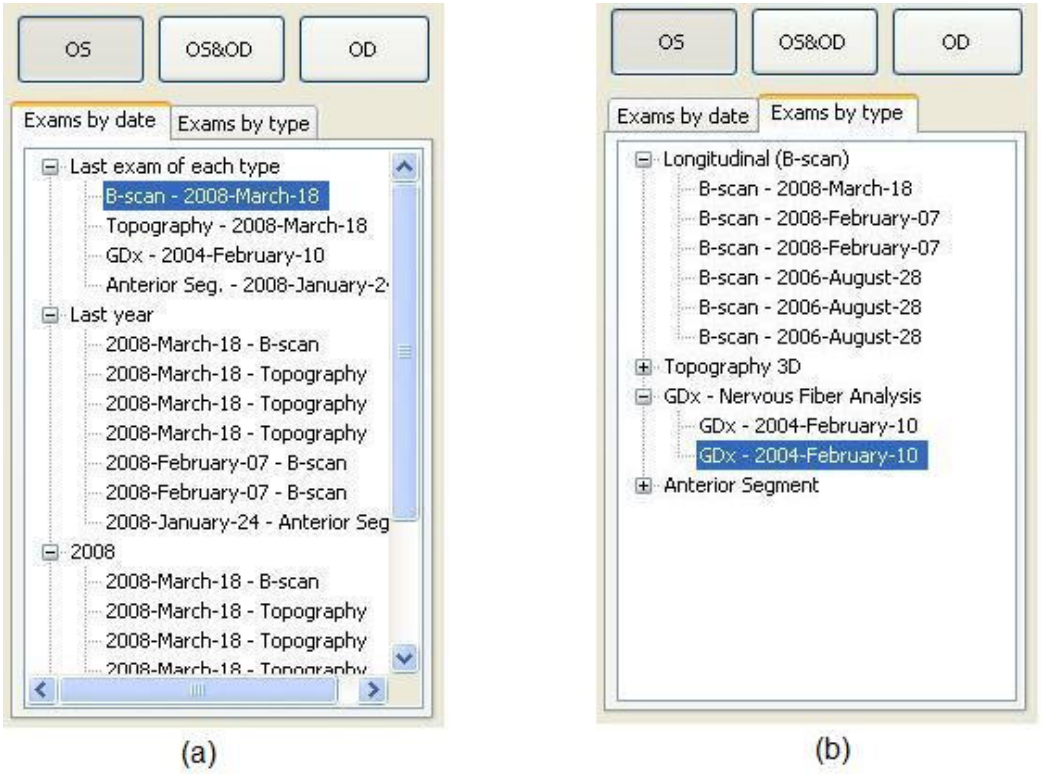
Figure 6 - Exams Browser

The **Exams Browser** is featured on the left side of the screen on a vertical fashion, so that it can be easily hidden to allow more space to the **Exam Elements Browser** or the **AFC**. It features the **Exams List**, where the patient exams are displayed, and the **Exam Elements Pre-visualization**, where the user can see the exam images and videos thumbnails without loading the whole exam.

Exams Organization

The patient's exams can be organized accordingly to the examined eye, and then by date or by type. The visual way to show the organized exams to the user is using *treeviews* which present the data on an intuitive way.

To choose the examined eye the user has to press a button, and the organization by date or by type is changed using tabs, as seen in Figure 7:



(a) (b)

Figure 7 - (a) Exams by date (b) Exams by type

For the implementation of this feature it was necessary to:

- Create algorithms to organize the exams accordingly to the examined eye, and then present them in the *treeviews* by type or by date;

- Create the ways the user can interact with these elements (e.g. present the pre-visualization of an exam when selected, or opening an exam in the **Exam Elements Browser**);

- Create the various settings that the user can define in relation to the exams organization usability.

Organization accordingly to the examined eye

The first step to organize the exams is to filter them accordingly to the examined eye; it could be the left eye (OS), right eye (OD) or both eyes (OS&OD). The "OS&OD" organization features all the patient's exams: the left eye exams, the right eye exams and the both eyes exams.

On the **Settings Window** the user can set the module to (after a patient selection being confirmed) automatically load the selected eye exams.

For the implementation of this feature it was necessary to identify which examined eye did the user selected, and then only select the patient's exams that feature the selected examined eye; if both eyes were selected, all patient's exams are selected.

Organization by exam date and type

After filtering the exams by examined eye, the exams are organized by exam date or by exam type. These two ways of organization are thought to be the most useful ones to the user, allowing the user to quickly find one specific exam, to follow a patient's exams' chronology or to see all the patient's exams of one exam type.

The type of organization can be predefined on the **Settings Windows**.

When the exams are organized by date, the parent nodes to be shown are the following (as seen in Figure 7):

- Last exam of each type: as the name informs, lists the last exam of each type (only the types that have exams)

- Last month's exams: list all the exams that were made on the last 31 days

- Last year's exams: list all the exams that were made on the last 365 days

- Exams by year: creates a parent node for each year that has exams, and groups all exams from that year

Please note that each exam can be presented more than once, appearing in more than one parent node. For instance, the most recent B-Scan exam can be presented on the last B-Scan exam node, on the last month's exams node, on the last year's exams node and in the current year node.

For the implementation of this feature it was necessary to, within the selected examined eye exams:

- Identify the existent exam types and the last exam of each type;

- Verify if there were exams done in the last month and year, and if so, create the parent nodes and create the children nodes, one for each exam;

- Organize the exams chronologically, divide them accordingly to the year, create parent nodes for each year with exams, and create the children nodes, one for each exam.

When the exams are organized by type there is a parent node for each exam type that has exams (as seen in Figure 7).

For the implementation of this feature it was necessary to identify, within the selected examined eye exams, the existent exam types, create a parent node for each one, and then create the children nodes, one for each exam of the parent node's exam type.

Exams Pre-Visualization

Before opening an exam, the user has access to a pre-visualization of it; this pre-visualization allows the user to see the thumbnails of the exam images and videos and know if there are reports (and if so, how many there are) without needing to load the whole exam data, providing a quick way to search for a specific exam (since only the thumbnails are loaded, the loading process is faster than if the whole exam data was to be loaded).

The pre-visualization consists in:

- A tooltip shown when the mouse hovers over an exam node, featuring the hour when the exam was taken and the number of images and videos on the exam

- The visualization of the exam's elements' thumbnails on a *flow layout panel* using *user controls* specifically developed for this purpose (Figure 8). Each image (or video) features the matching thumbnail, while the report existence will be shown through an *user control* with a generic image and the number of reports in the exam.

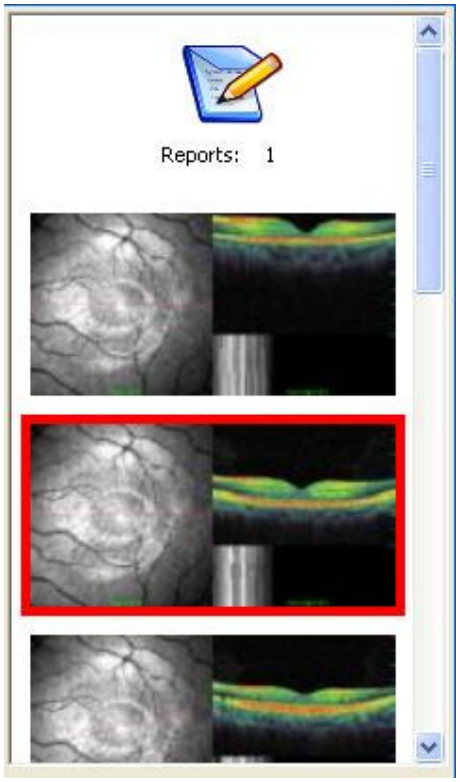


Figure 8 - Exams Pre-Visualization

A *drag&drop* feature for image previews was developed; it will simplify and speed the use of the module: once the specific image is found, the user just needs to drag the image preview to the **Display Area**, and consequently the matching image will be loaded to the **AFC** workspace. However, this feature is currently disabled (consult the section *5.6. M-VIS: Advanced Features Component* for more information).

For the implementation of this feature it was necessary to:

- Create algorithms to count the total number of exam elements, and count the total number of each element type (the total number of videos, images and reports);

- Write code to present a tooltip for each *treeview* node, showing the number of images and videos of the exam;

- Write code to retrieve the images and videos thumbnails from the DB;

- Develop user controls to the reports, images and videos;

- Write code to present the thumbnails in the pre-visualization *flowlayout panel*, and organize the thumbnails there (on the top are the reports, then the videos and then the images, chronologically);

- Develop the graphical behaviour of the user controls (e.g. the thumbnails should occupy all the *flowlayout panel* available area, develop the interaction with the *split containers*);

- Develop the *drag&drop* feature.

4.4.4.2. Exam Elements Browser

The objective of the **Exam Elements Browser** is to display the various exam elements in an intuitive and accessible way, allowing the user to browse through them and perform some operations (listed on section *4.4.4.3. Features*) with them.

The Figure 9 presents a *print screen* of the **Exam Elements Browser**:

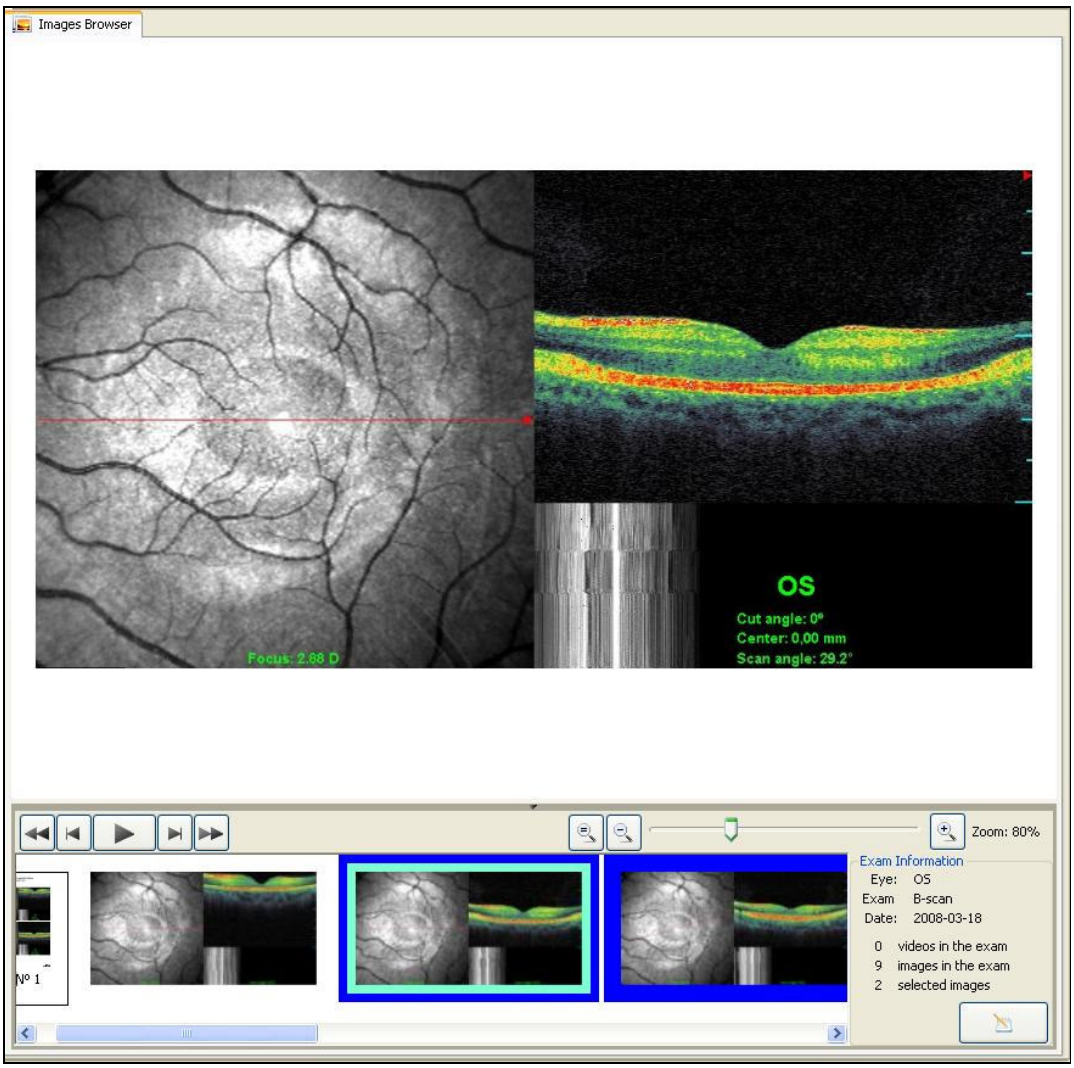


Figure 9 - Exams Elements Browser *print screen*

The **Exam Elements Browser** features the **Exam Elements List**, where the user can browse through the diverse exam elements, the **Display Area**,

where the exam elements are opened and displayed, and the **Exam Textual Information**, which displays the exam details.

Similarly to the patients selection window, this window also features *split containers*, namely:

- A vertical *split container* between the **Exams Browser** and the **Exam Elements Browser**;

- A horizontal *split container* inside the exams browser between the thumbails and the exams *treeviews*;

- A horizontal *split container* between the display area and the **Element Operations Bar**.

These *split containers* allow the user to organize the screen in a more personal way (for example, if an exam has many elements, the user can increase the **Exam Elements List** area, in order to browse through them more easily).

For the implementation of this feature it was necessary to work with the minimum and maximum sizes and with the *anchor* and *dock* properties of every object, in order to avoid visual overlaps. It was also necessary to define the *splitters* minimum and maximum distances and create code to deal with the window minimization.

Exam Elements List

The diverse exam elements are listed in this area using specific user controls for each exam result type.

The **Exam Elements List** is featured at the bottom of the **Exam Elements Browser**, alongside with the **Exam Textual Information**. This allows the user to view the diverse elements, browse through them and open them on the **Display Area**; this way of browsing is quite intuitive and similar to files

browsing on the Windows Operative System, hence it will be simple for the user to learn how to use it.

The exam elements are separated accordingly to their result type, and chronologically organized (the exam's first image is the first element, the exam's second image is the second element, and so on).

For each result type there is a different *user control*, as listed bellow:

ImageThumbnail.cs – This *user control* displays the images' thumbnails, and its dimensions changes accordingly to the thumbnails dimensions. It features two kinds of interaction:

- Visualization: it can be done by clicking once on the control. It will show the image on the **Display Area** and the inner selection frame will change its colour (Figure10), indicating that the thumbnail is being visualized.

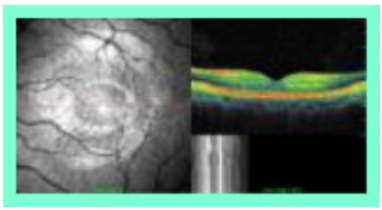
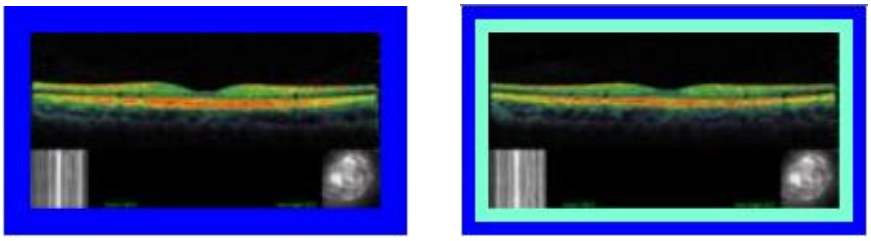


Figure 10 – Visualized image thumbnail

- Selection: it can be done by double-clicking the control. Visually it will only change the outer selection frame colour (Figure 11), but will also list this image as selected to be sent to the **AFC** workspace or to the report creation, should the user decide to do so.



(a) (b)

Figure 11 – (a) Selected image thumbnail (b) Image thumbnail visualized and selected simultaneously

VideoThumbnail.cs – This *user control* displays the video miniature and its graphical aspect resembles that of a film cell (so that the user can easily associate this *user control* with a video element).

It features only the one-click interaction, which will make the *user control* become more bright (indicating to the user that the video is selected) and open the video with the video player; if the video is already playing, it will replay it from the beginning.

Figure 12 – Video thumbnail

ReportThumbnail.cs - This *user control* features the exam's reports (unlike the exams pre-visualization, where one report preview represented all the exam's reports, in the **Exams Elements List** each report has its ReportThumbnail). For some report types (namely B-Scan, Topography, Retinography, Angiography, GDX, Micro-Perimetry, HRA2, CCT and CSF) the report features a specific report thumbnail, while for the remaining exam types it only features a generic image; both situations are represented in Figure 13.

(a) (b)

Figure 13 – (a) – Generic report thumbnail (b) B-Scan report thumbnail

For the implementation of the **Exam Elements List** it was necessary to:

- Create the interactions with the DB;

- Develop user controls to the reports, images and videos;

- Create the keys interaction with the user controls;

- Write code to present the thumbnails in the *flowlayout panel*, and organize the thumbnails there (first there are the reports, then the videos and then the images, chronologically);

- Develop the graphical behaviour of the user controls (e.g. the thumbnails should occupy all the *flowlayout panel* available area, develop the interaction with the *split containers*);

- Create the interaction between each exam element and the **Display Area**.

Display Area

The **Display Area** is where the exam elements are opened and seen by the user. The way it works is different accordingly to the result type, and will be explained for each result type:

Image elements – The image elements are opened using a *picture box*, and the images are resized (proportionally to their original size) in order to fill all the available space. By double-clicking the image, the **Exams Browser** and **Exam Elements List** will be hidden, and the display area will use their space (consequently the image's size is increased to fill the **Display Area** new dimensions), which is exemplified in Figure 14. Also, there is the possibility to zoom the images and to run a slide show with all the image elements.

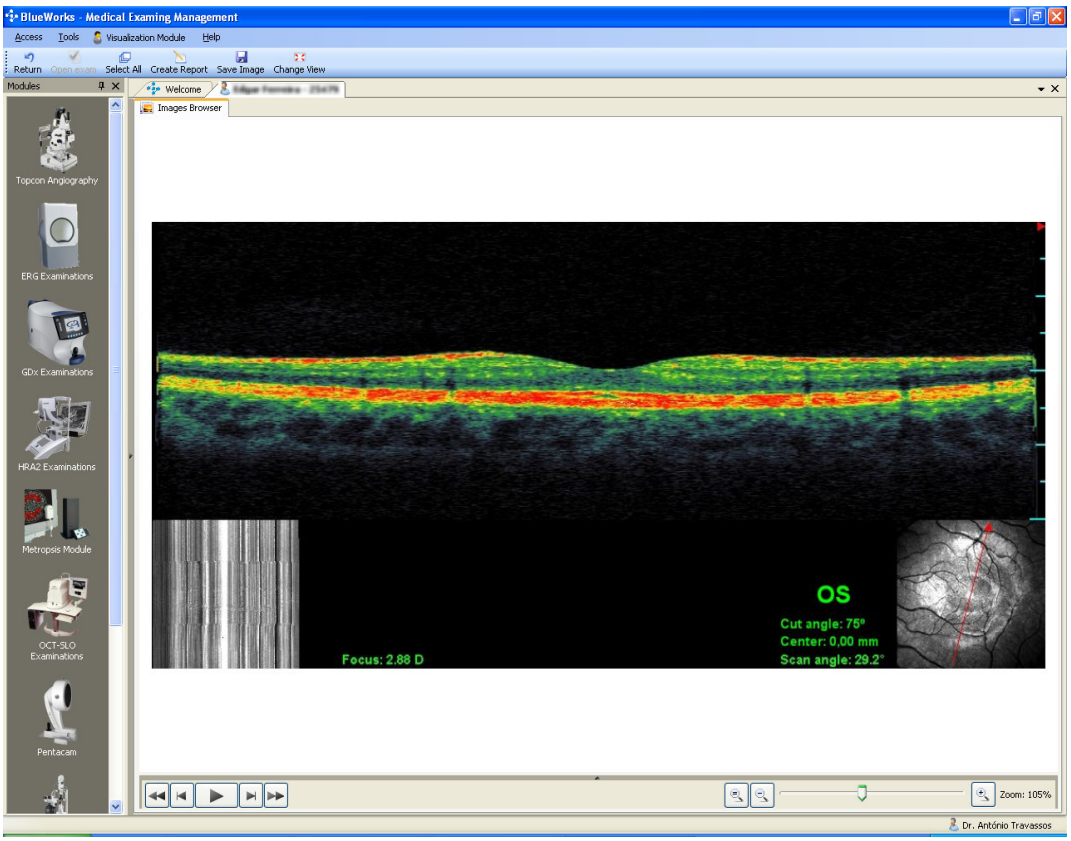


Figure 14 - Image visualized, with the Exams Browser and the thumbnails hidden

Report elements – The report elements are opened using a *web browser*, which will in turn run a PDF reader; consequently, to use the developed application there is the requisite of having a PDF reader installed on the computer. An example can be seen in Figure 15.

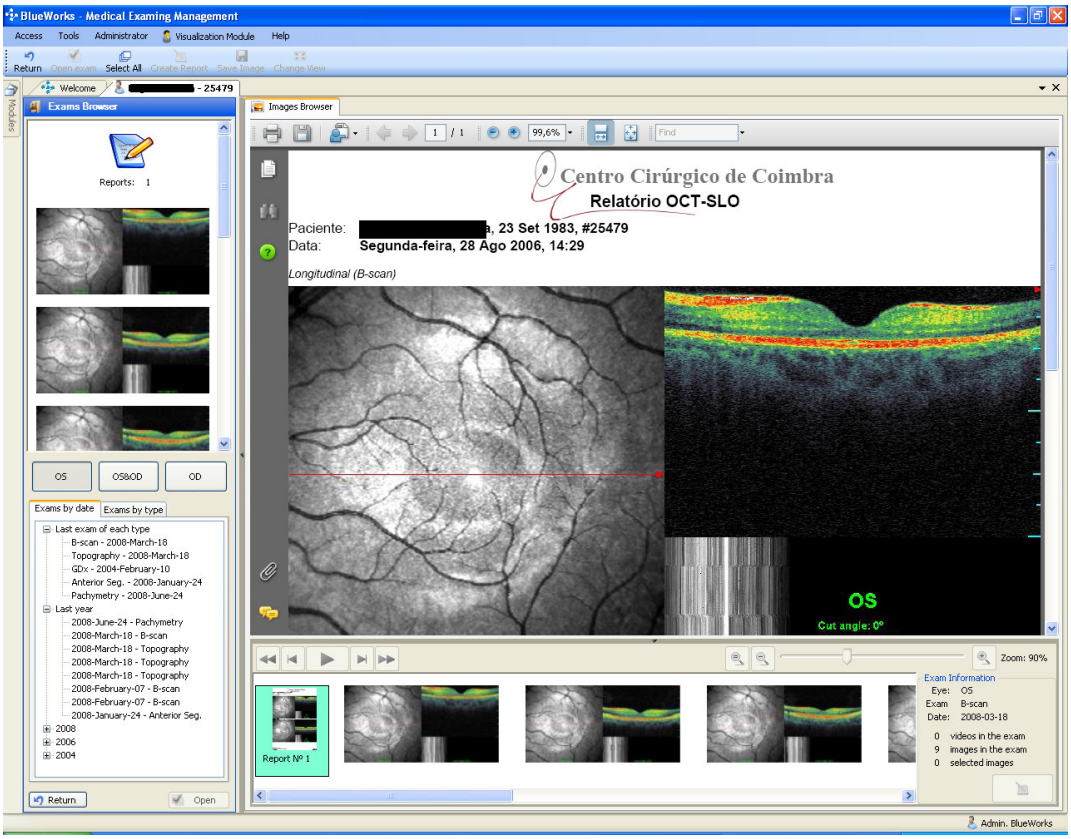


Figure 15 – A report opened on the M-VIS

Video elements – The video are opened using a *panel*. As it happens with the images, the videos are resized (proportionally to their original size) in order to fill all the available space. A small panel also appears on the **Display Area**, with the video playing controls. The code to play videos was developed using as basis the code available on [7]. An example can be seen in Figure 16.

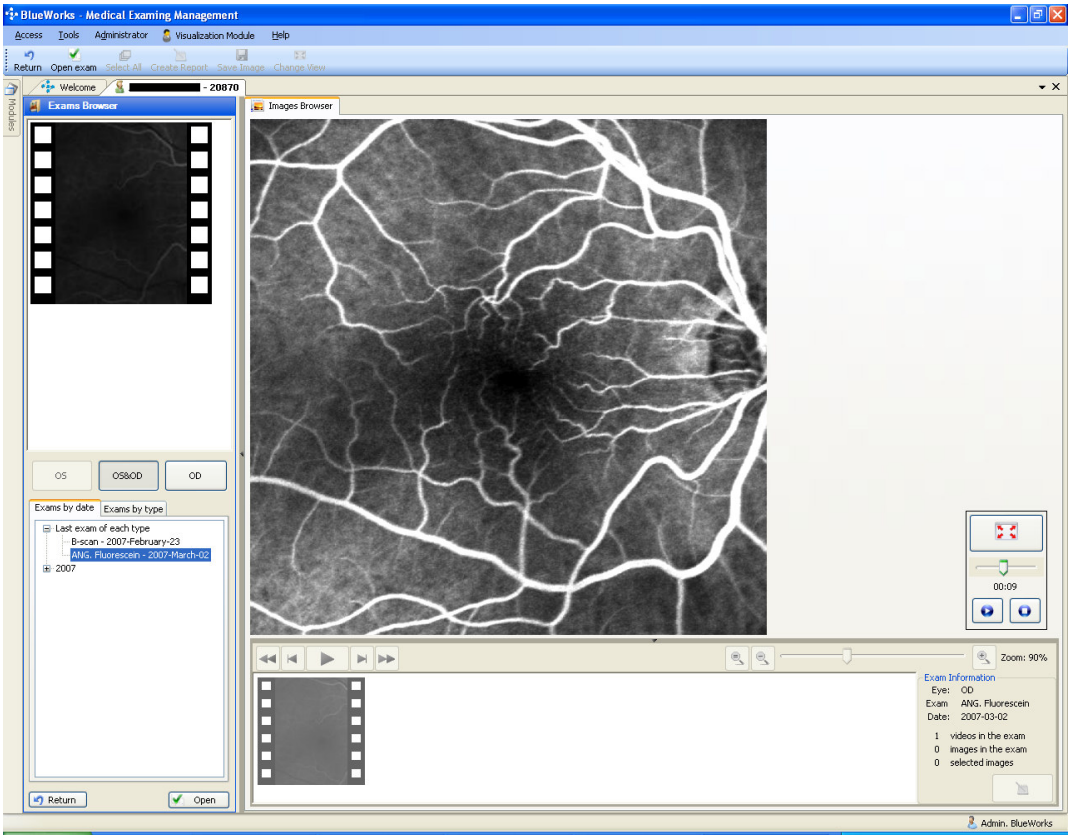


Figure 16 – A video playing on the M-VIS

The following controls are available to control the video playing:

Figure 17 – Video Controls

Play/Pause – This command plays the video (from a specific video frame if the video is paused, or from the beginning if the video is stopped) or pauses the video on the current frame.

Stop – This command stops the video

Change view – This will hide the **Exams Browser** and the **Exam Elements Browser**, so the **Display Area** will occupy more space (and the video will be resized accordingly), or do the reverse operation in case they are already hidden. (Similar to the double-click operation on the image)

Note: This three commands are also available as a context menu by clicking with the mouse right-button on the video.

Video trackbar – This trackbar features a number of ticks equal to the video's seconds. If scrolled to a specific second, the video will also be scrolled to that second, therefore allowing the forward and rewind operations.

Video timer – Shows the video playing time.

For the implementation of the **Display Area** it was necessary to:

- Create various *panels* to host the *picture box*, the *web browser* and the video, and the interactions between them;

- Create the interaction between the various objects of the **Display Area** and the exams elements user controls.

Textual Information Area

On the right side of the element thumbnails there is a *group box* which features textual information related to the selected exam, specifically:

- Exam's examined eye;
- Exam type;
- Exam date;

- Number of videos in the exam;
- Number of images in the exam;
- Number of selected images in the exam;

An example can be seen in Figure 18:

Figure 18 - Textual Information Area

For the implementation of this feature it was necessary to write code to identify the exam element eye, the exam type and count the total number of videos, images and selected images.

4.4.4.3. Features

Besides browsing through an exam elements, the **M-VIS** allows the user to create PDF reports, save the visualized image, apply zoom to an image, do a slideshow with all the exam's images, see information related to the patient, and configure the module settings.

Report Creation

The **M-VIS** allows the user to create a PDF report with the selected images from the exam; hence, the user can browse through an exam, select the most relevant images, and create a report with them, ready to be saved (for example, on an USB pen) or printed. Once again, this features needs the user to have a PDF reader on the computer (although the reader is not involved on the report creation, it is needed to read the report immediately after its creation)

After choosing to create a report, a **Report Options Window** appears (Figure 19), where the user can see the images that are going to be inserted on the report, add a comment to each one, and select the number of images present in each page.

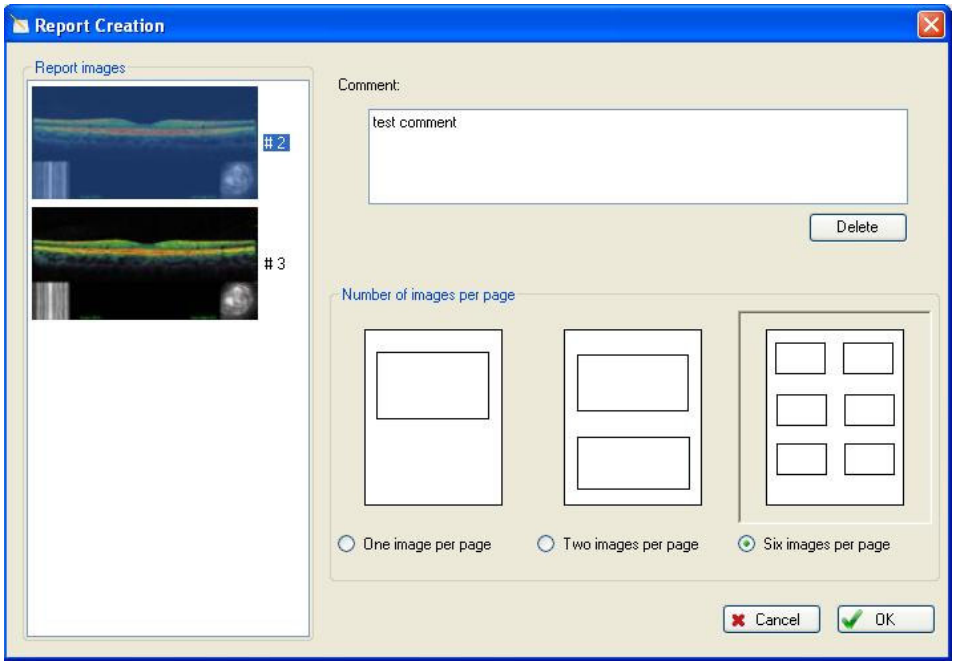
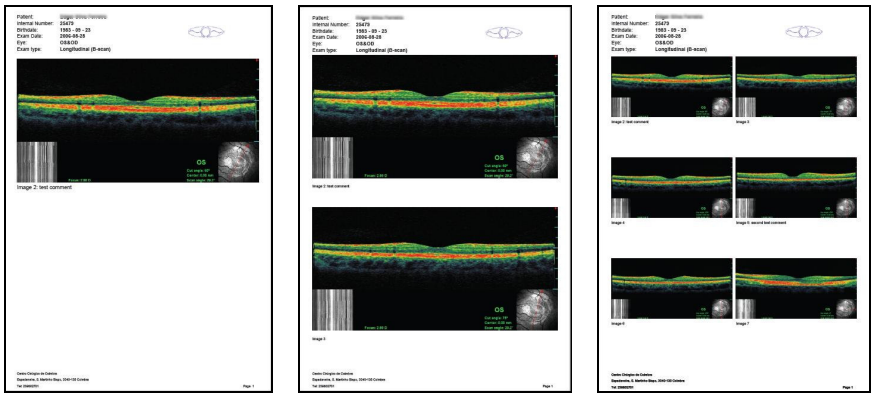


Figure 19 - Report Options Window

After the user pressed the confirmation button, the report is created and presented to the user on a new window (using a *web browser*, which will call the operative system default PDF reader).



(a) (b) (c)

Figure 20 – (a) Report with one image per page (b) Report with two images per page (c) Report with six images per page

For the implementation of this feature it was necessary to study the *iTextSharp* library [6], and develop an algorithm to:

- Place the document header and footer;

- Verify if a report logotype exists, and if so, place it;

- Identify how many images are to be placed by page (one, two or six), resize the images to fit the available area, and place the images;

- Identify if comments were written, and if so, place them on the right place (under the respective image).

Save Image

The **M-VIS** has the option to save the visualized image on a folder specified by the user (for example, an USB pen).

After clicking the **Save Image Button**, a new "Choose Folder" operative system window will appear, where the user can select the image extension (PNG, JPEG, BMP, GIF, TIFF) and select the folder where the image will be saved; after confirmation, a copy of the visualized image is created on the specified folder.

For the implementation of this feature, it was necessary to develop an algorithm to create a copy of the image on the specified folder in the selected file format.

Image Zoom

The zoom feature allows the user to zoom the image between 30% and 200%; besides that, it automatically resizes the image (accordingly to its original dimensions) to fit all the available display area. This feature was developed using as basis the code already available on other BW-MES components.

The following controls are available to control the image zoom:

Figure 21 - Image zoom controls

Original size button: sets the zoom equal to 100%

Increase zoom button: increases the zoom by 5%

Decrease zoom button: decreases the zoom by 5%

Zoom trackbar: by scrolling this trackbar the zoom can be changed between 30% and 200%

Although this feature was developed using as basis a functional code, in order to implement it in the **Display Area** it was necessary to only enable it when an image was being visualized, and develop the interaction between this feature and the report and video's *panels*.

Slideshow

This feature allows the user to play a slideshow with all the exam's images. Similarly to the zoom feature, it was developed using as basis code already available on other BW-MES components.

The following controls (Figure 22) are available to control the slideshow:

Figure 22 - Slideshow controls

Play/Pause: Starts the slideshow (if it is stopped) or pauses it on the currently visualized image (if it is playing)

Next image: Jumps to next image

Previous image: Returns to the previous image

First image: Visualizes the exam's first image

Last image: Visualizes the exam's last image

Besides this controls, the user can specify on the **Settings Window** the time that each image is shown (slideshow interval).

Although this feature was developed using as basis a functional code, in order to implement it, it was necessary to develop the interaction between this feature and the reports and videos.

Patient Information Window

This windows presents to the user information relating to the selected patient, namely:

- The patient's name;
- The patient's birthdate;
- The patient's internal number;
- An image related to the patient's age and sex;
- How many exams the patient has of each exam type;
- The total number of the patient's exams;

The Figure 23 presents a *print screen* of the patient information window:

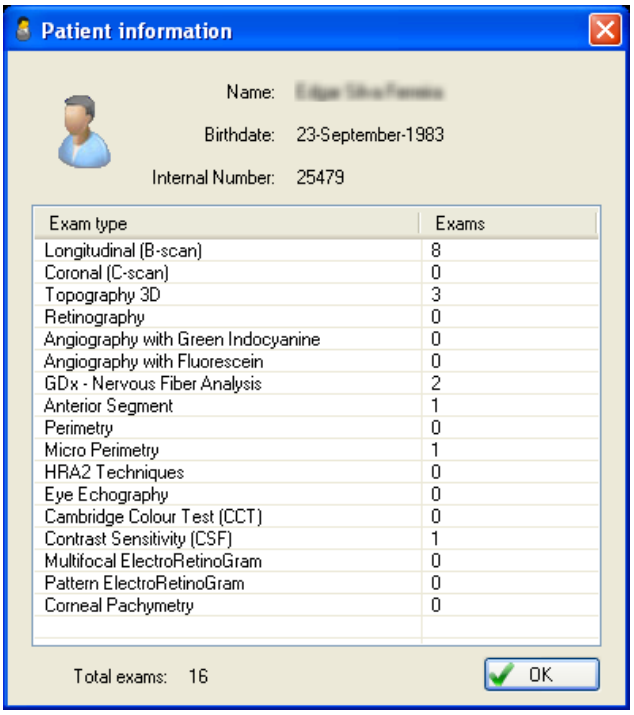


Figure 23- Patient information window

For the implementation of this feature, it was necessary to create algorithms to identify the exam type of each exam and count how many exams there are of each exam type.

Settings Window

The **Settings Window** (Figure 24) allows the user to define the **M-VIS** settings, namely:

- Define if the **Exam Elements List** and the **Textual Information Area** are visible or not on startup;

- Define if the **Main Interface** modules list is visible or not on start-up;

- Define the exams organization start-up tab (by date or by type);

- Define if there should be a selected examined eye on start-up, and if so, which one;

- Define the number of elements on the last patients with exams lists (by doctor and general);

- Define the previews and thumbnails frame colours;

- Change and delete the report logotype;

- Change the slideshow interval;

- Change the toolbar style;

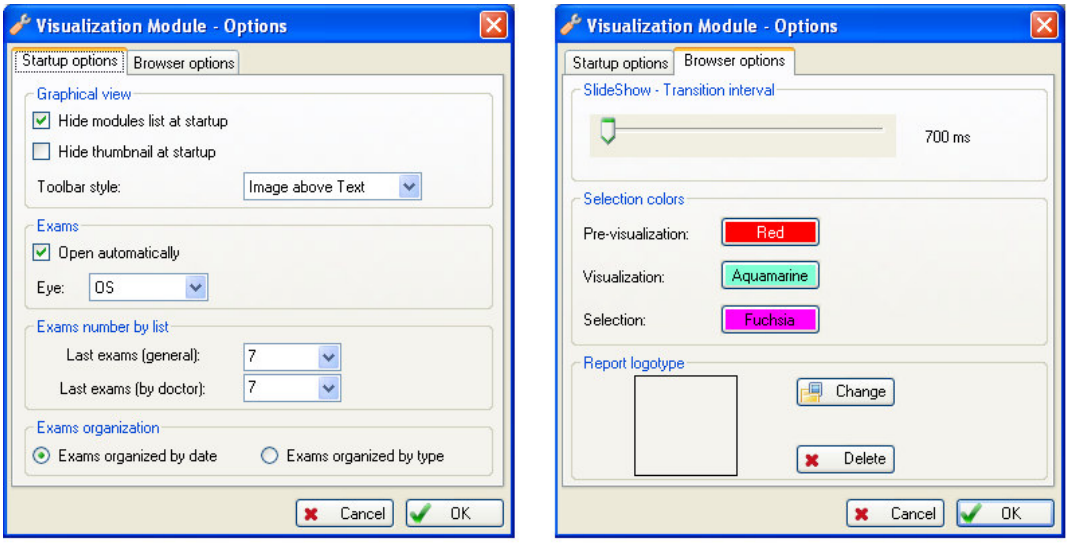


Figure 24 - Settings window

For the implementation of this feature, it was necessary to (when the need for the setting was only perceived after the code had been written) rewrite the code in order to implement the setting, and to detect and solve the effect that the setting could have on other features.

Controls

Menu

The menu is a part of the **Main Interface**'s menu; however, it is only visible when the **M-VIS** is opened. It features shortcuts to the following operations:

- Save image;
- Change view;
- Create a report:
- Patient information;
- Module options;
- "About the Visualization Module" window;
- Exams import (this option is only visible when the user is an administrator);

Toolbar

The toolbar (Figure 25) appears on the top of the screen (except on the patient selection window) and features shortcuts to the following operations:

- Return (either to the patients selection screen, or to the Browser screen);
- Open the selected exam;
- Select all images;
- Create a report;
- Save image;
- Change view.

Figure 25 – Toolbar, with the "image above the text" style

The toolbar style can be changed in the **Settings Window** between the following options:

- Text only;
- Image only;
- Image above the text;
- Image on the left of the text.

4.5. M-VIS: Advanced Features Component

Besides the **M-VIS** support functions (that allow the user to search patients, open exams and browse through the exam elements), this module will also feature a component **(AFC – Advanced Features Component**) to allow the user to perform more advanced operations with the exam images (namely spatial alignment with simultaneous transparent visualization, features extraction, and image enhancement), hence digitally obtaining more information from the exam than the one that could be obtained physically.

Currently the **AFC** is still in development, and consequently it was decided to keep it disabled on the **M-VIS** last version. Although some support features are already developed (namely the operation to send an image to the **AFC** or to delete it from the **AFC** workspace, and the *drag&drop* feature), the bulk of the advanced operations is still in development.

The **AFC** has the following components:

- **Images List**: the images that are on the AFC workspace are listed on the Images List

- **Display Area**: display the images and the advanced operations result

- **Advanced Operation Component**: features the diverse controls for the operations that can be done with the images

In Figure 26 is represented the proposed schematic that is currently being used; accordingly to the development process it may be subjected to changes.

To switch between the **Exam Elements Browser** and the **AFC** it was decided to use *tab pages*.

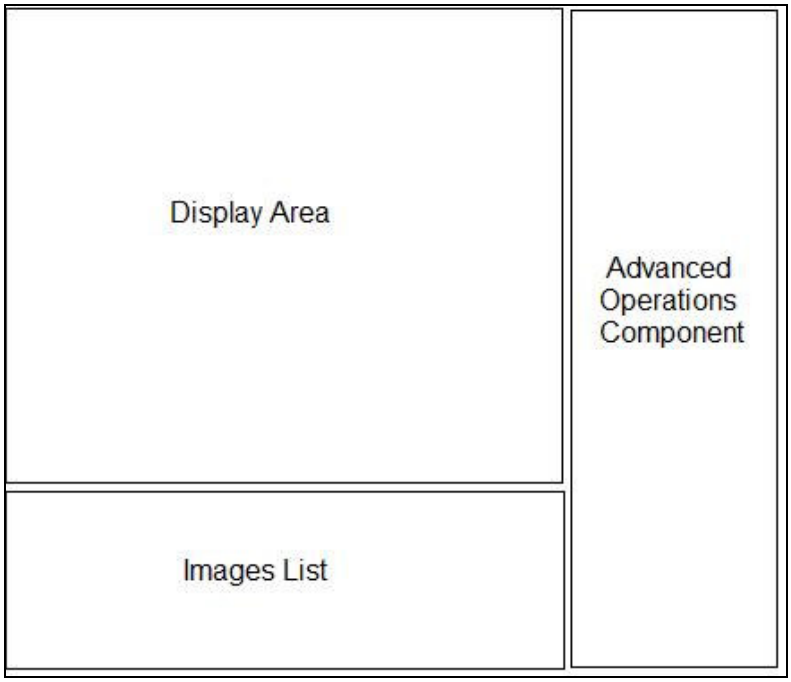


Figure 26 - AFC schematic

Currently, the **AFC** features this graphical aspect:

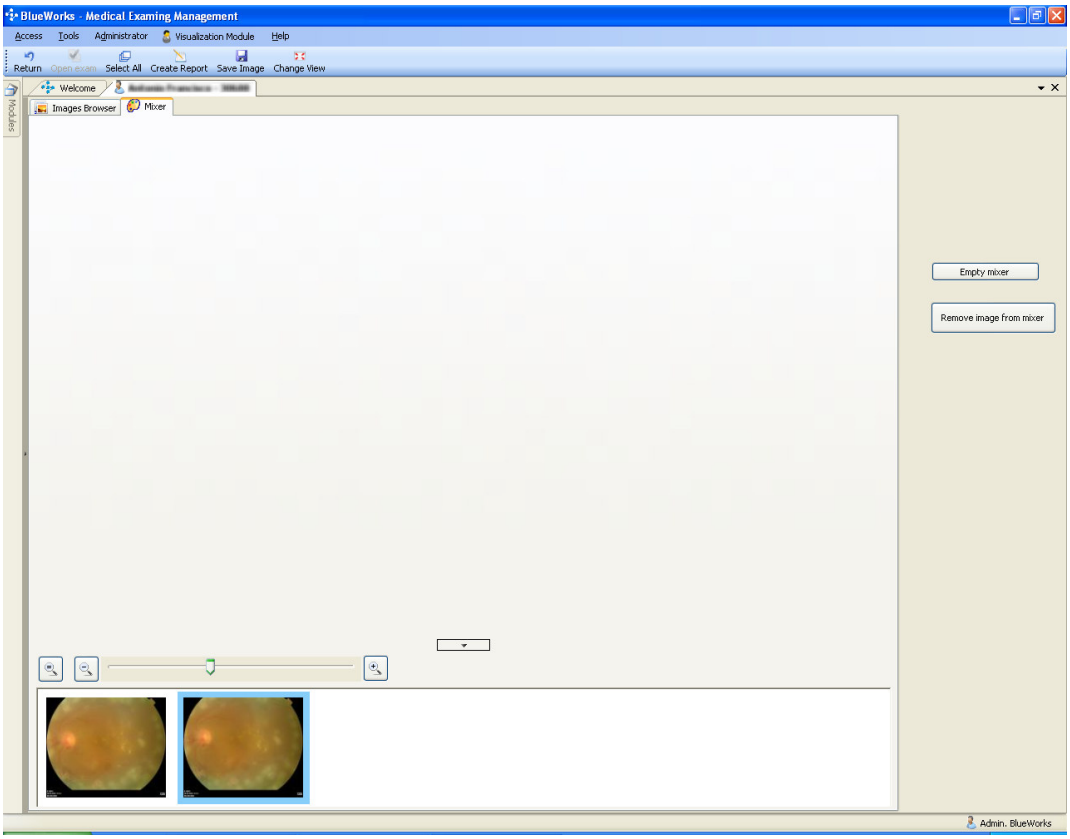


Figure 27 - AFC *print screen*

4.6. M-VIS: Exams Importer

The **Exams Importer** is a software tool that registers exam files in the database. This functionality is important due to the two following situations:

- There are many exam files done before equipment integration that are not registered in the database; thus, there is the need to register all those files.

- There still is some equipment that for some reason (e.g. proprietary software) can not be integrated; hence, every time an exam file from such an equipment is created, the **Exams Importer** will automatically register that file in the database.

For these reasons, the **Exams Importer** main requisites are to allow the user to:

- Select a folder, whose files are to be imported, and for that folder define an associated examination equipment;

- Configure for that folder the exam and image type for each file index (see the section 4*.7.3. Usability* for further information)

- Set the folder to be watched or not (if a folder is watched, every new file that is saved on that folder will be analyzed, and if the result is positive, the file will be imported to the database);

- Manually import the folder's files.

4.6.1. Dataflow

The dataflow is represented in the following diagram:

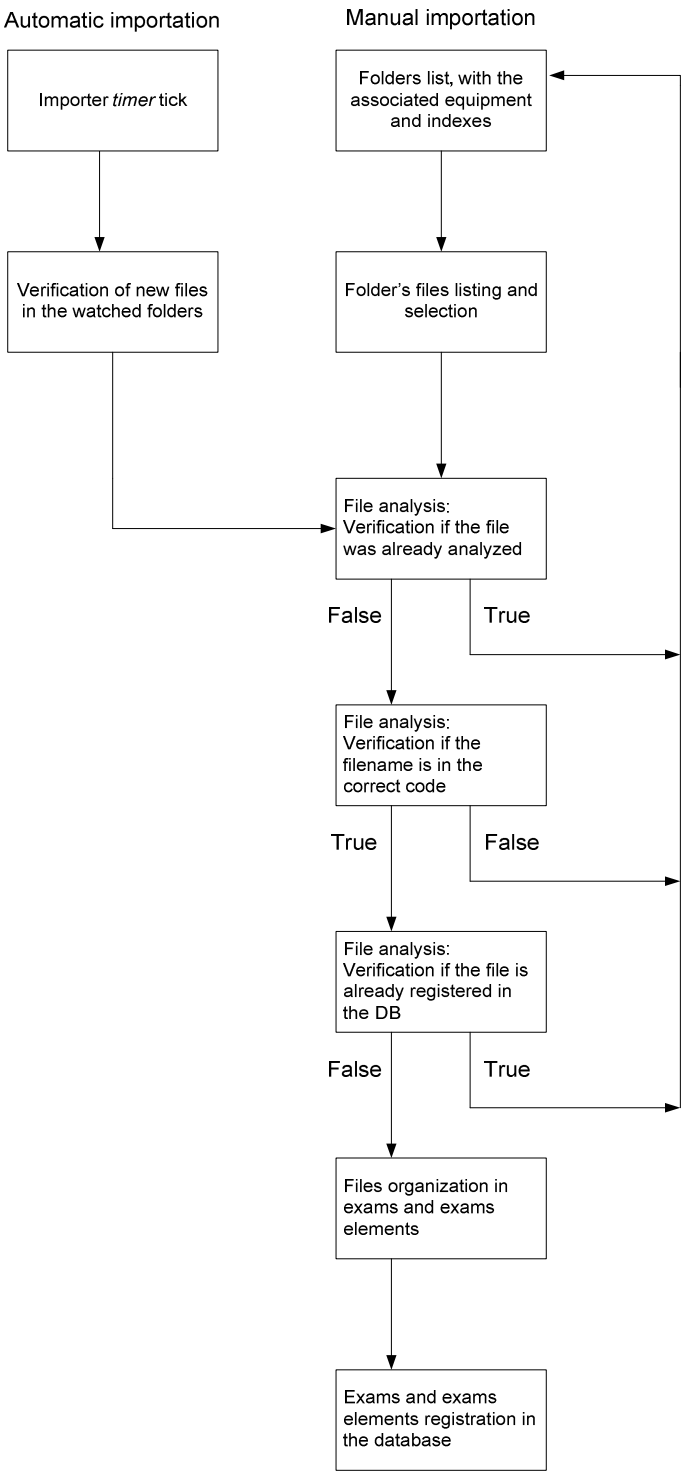


Diagram 5 - Importer dataflow

4.6.2. Implementation

For the implementation of this component it was necessary to:

- Study the way the exam files from unintegrated equipment are saved;
- Learn how to work with XSD (XML Schema Definition);
- Learn how to work with XML (Extensible Markup Language) files;
- Learn how to work with *dictionaries*;
- Learn how to work with *datagrids*;
- Develop an algorithm to work with the XSD, XML file, folders' list and *datagrid* simultaneously;
- Develop an algorithm to analyze the file name;
- Develop an algorithm to verify if the file had already been registered in the database;
- Create a list of assurances and implement it:
- Develop an algorithm to analyze exam files, and aggregate them in exams;
- Develop an algorithm to register exams and exam elements in the DB;
- Implement *threads* to work with the manual importation.

4.6.3. Usability

The way that the **Exams Importer** works was designed to match the methodology used in CCC to save the exams from non-integrated equipment.

Currently, for each examination equipment there is a specific folder in a server where its result files are saved. In the images case, they are saved through *SnagIt*, a software from TechSmith Corporation© that creates a screen capture and allows the user to define the file name, format and path.

The name of each file is set manually accordingly to the following code, in order to save some information related to the exam:

Equipment id – patient id – exam date (yyyymmdd) – index

´

So, as an example, the first image for patient 12345 observed by a machine with id 02 on the first of November of 2007 would be saved with this name:

02-12345-20071101-1

The file index can have two meanings:

- Can just indicate the exam element index on the whole exam (ex: the first image has index 1, the second image has index 2, and so on);

- Can indicate the exam type (ex: on an retinal angiography equipment, the index 1 can mean that the image is a retinography, the index 2 can mean that the image is a fluorescein angiography, and the index 3 can mean that the image is a ICG angiography)

The meaning of the file index has to be specified by the user on the application, since it is different in every folder.

Due to this methodology, the **Exams Importer** allows the user to define a folder and to define the examination equipment that is associated with the folder. Next, the user can define the files' indexes and their meaning; that is, the user defines a number for the index, and for that number the exam type (only between the exam types that are associated with the chosen equipment) and the image type are defined. An example can be seen in Figure 28:

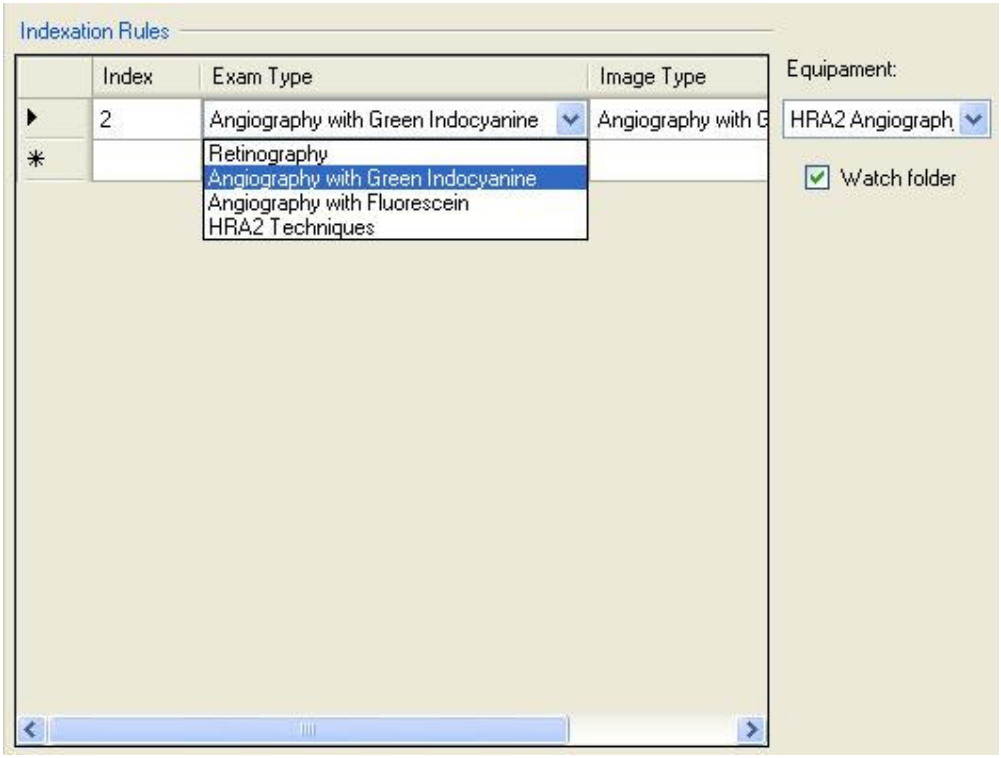


Figure 28- Indexation rules example

If there are files in the correct code, but with an undefined index, the software will consider their exam and image type to be equal to the ones defined for the index with the greatest numerical value.

After these definitions have been made and saved, the software is able to analyze the name of a file, check if it is in the correct code, and register it in the database, accordingly to the file index.

The process of analyzing a file and register it in the database can be triggered manually or automatically:

- Manual registration: the user chooses a specific folder, chooses the files within that folder's files that are to be registered, and tries to register them

- Automatic registration: there is a timer (in the M-VIS), and each time the timer ticks the software runs the list of defined folders, checks which ones are being watched, verifies if there are new files, and if so, tries to register them.

A *print screen* of the M-VIS:Importer is featured in Figure 29:

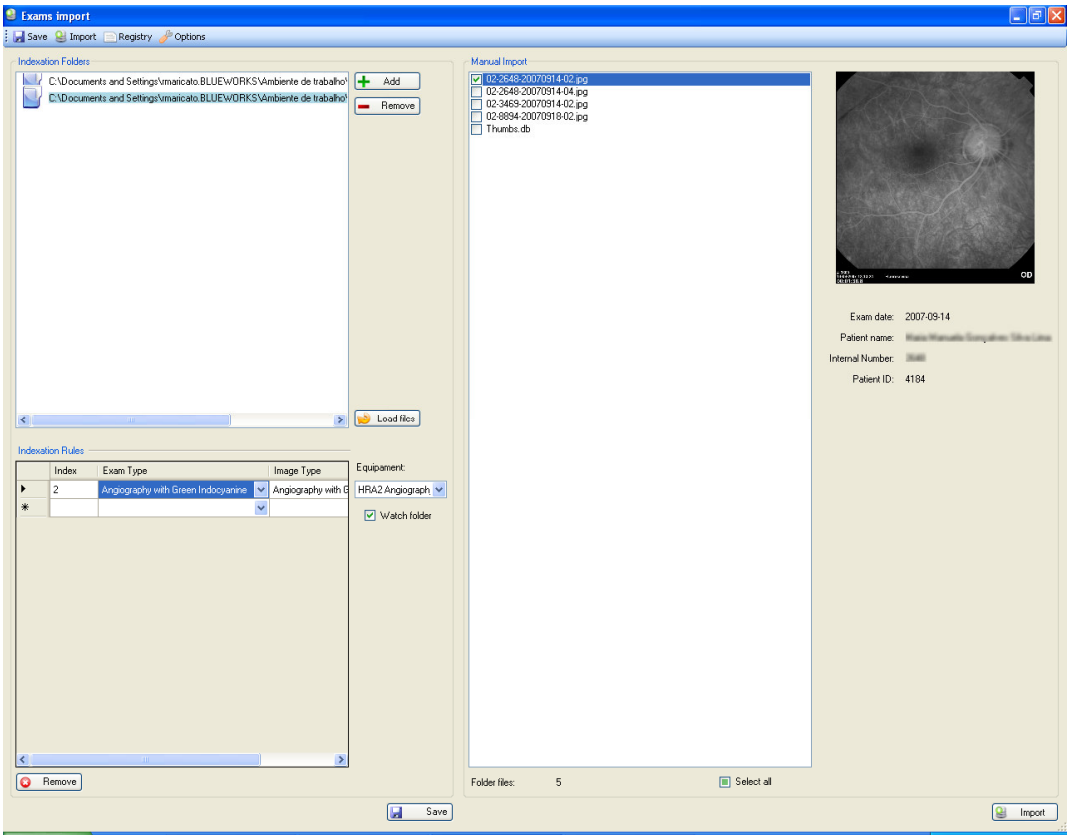


Figure 29 - Importer *print screen*

4.6.4. XSD structure and XML file

In order for the **Exams Importer** to work properly, there is the need to save some data some permanently, particularly:

- The list of folders
- The relations between indexes, exam types and images types for every folder
- A log of analyzed files (either if they end up being registered or not)

To save and manage this information, it was decided to create a XSD (XML Schema Definition), with the following structure:

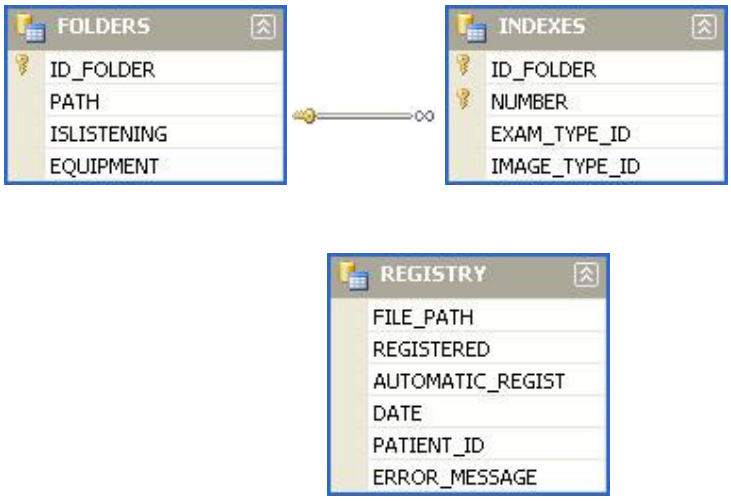


Diagram 6 – XSD structure

The created XSD is saved in a XML (Extensible Markup Language) file which is updated every time the user saves the changes done and every time there is the automatic registration of files in the database (either successfully or not – a log of the analyzed files has to be maintained).

Besides that, if the application is called and there is not any defined XML file, the application will not start until the user selects a folder (where the XML file will be created); this folder can later be changed in the options.

4.6.5. Validations

The validations that have to be done are of two kinds:

- Usability validations: validations done while using the **Exams Importer**, namely in the definitions made by the user;

- Data validations: files validations, to check if the files are suitable to be analyzed and registered in the database;

Usability validations

The usability validations are done on the added folders and on the defined indexes.

Folders

- Verifies that the folder exists

- Verifies that there are no repeated folders

- Verifies that the user defined an associated equipment for the newly added folder

Indexes

- Verifies that there is at least a defined index

- Verifies that there are not incomplete rows in the indexes' *datagrid*

- Verifies that the index is a number

File validations

The file validations are done on the filename and format.

- Verifies if the file was already analyzed (that is, if it is present in the analyzed files log – if so, its name or presence in the database will not be checked, hence saving time)

- Verifies the file format (only images (PNG, JPEG, BMP, GIF, TIFF) and pdf files are allowed)

- Verifies if the name can be splitted in 4 segments

- Verifies if all segments are numbers

- Verifies if the third segment is a date

- Verifies if the second segment is a valid patient id

- Verifies if the file index is defined (if not, it will be considered equal to the defined index of greatest numerical value)

- Verifies if the file is already registered in the database

4.6.6. Registering files in the database

The process of registering files in the DB consists in two steps:

1 – Analyzing the exam files and aggregating the exam files that belong to the same exam; by other words, organizing the exam files in exams.

2 – Registering the exams and the respective exam files in the database.

Organizing the exam files in exams

Before registering the exam files in the database, they have to be analyzed, in order to aggregate the exam files that belong to the same exam.

The criteria used to perform this organization are the equipment id, patient id and exam date; hence, the exam files that feature the same equipment id, patient id and exam date are considered to belong to the same exam.

Registering exams and exam files in the database

Once the exam files are organized in exams, there is the need to create an exam id in the database for each exam, and then register the respective exam files in the database using the created exam id.

To perform this operation, it was used as a starting point the code used in the integration modules to save the exams results. However, the code had to be adapted, because the original code saved the exams in the server and then registered them.

In relation to the automatic importation, a separate thread is created to perform it, so not to affect the **M-VIS** performance.

5. Challenges

During the course of this project there where certain tasks that proved to be more difficult than expected, therefore taking more time than what was expected to be done. Bellow is a list of the main difficulties found:

5.1. Visual Studio and C#

Neither Visual Studio nor C# is taught in the Biomedical Engineering course; hence, the work developed was not only the creation and implementation of algorithms and features, but also a process of constant learning and research on the Visual Studio and C# specificities (using resources such as [8], [9] and [10]).

5.2. Report Creation

The algorithm to create a PDF report used as a starting point an algorithm already developed on other BW-MES components. (which work using the *iTextSharp* library [6]). The objective was to create a generic algorithm that would allow the user to put one, two or six images on each page, with or without an individual comment for each image; this algorithm would have to be generic, able to process different image sizes (since the images sizes vary for each exam type). Such an algorithm was achieved through a lot of “generate and test” until the specificities of the *iTextSharp* library class were understood and a generic algorithm could be written.

5.3. Module Settings

The module settings were specified after the module requisites were specified; however, as the work went by, the need for some new settings was perceived, which led to a problem: since the code was not thought from the beginning to implement these settings, their implementation implicated the rewrite of some of the code.

5.4. Video Player

The video player challenge was similar to the module settings; the need for a video player was perceived only in an advanced stage of the project. Hence its implementation needed demanded the redesign of a lot of code, because at that point the code had been written expecting only report and image files.

So, to implement the video player, research was done and a first solution was developed; this solution at first was successful, until a problem related to the form resizing was detected, which led to the research of a second solution, which has proven successful.

5.5. Visual Designing

The **M-VIS** visual design is of great importance, since it has to be simple and intuitive, in order for the module to be easily used by everyone without the need for a detailed specific formation. However, during the Biomedical Engineering course, software visual designing is barely teached (being the course focus' algorithm creation), so operations such as working with *split containers*, understand the **Display Area** behaviour while leading with the diverse exam file types, or keeping a *tree* node selected while changing *tabs* were an unexpected difficulty, which took a deal of time longer than initially expected to be solved, due to the lack of experience in working with visual designing.

5.7. Module usability

The usability of the module isn't straightforward: the user can select a patient, browse his/her exams, return to the patients search, select another patient, open an exam, do a slideshow, then open a video, *et cetera*.

As a consequence of the usability flexibility, a great attention had to be paid to the various (and sometimes unexpected) interactions that happen between each feature (example: automatically stop the slideshow and disable the slideshow buttons when a video is opened), in order to avoid *bugs*.

5.7. Writing the project report in English

Writing the project report in English proved to be a challenge; but with time and persistence it was possible to overcome it.

6. Conclusions and future objectives

After analysing the developed work, some conclusions can be withdrawn. The project core objective is to improve the ophthalmologic clinical practice by developing a decision support system. In order for this objective to be achieved, intermediary objectives have to be achieved, namely:

- Optimize the clinical practice workflow;
- Integrate examination equipments (through integration modules), in order to save exams in a database;
- Design a database suitable for the requisites faced;
- Manage and visualize the exams (through the visualization module);
- Apply data-mining techniques to the saved data(through the data-mining module).

Since the **BW-MES** is an application that saves exams in a database, a priority task was to create a tool to browse and visualize those exams; beyond that, there was also the objective to implement advanced visualization features, such as the overlapped visualization of different examination of the same retinal area image. These features are very important in order for the **BW-MES** to work as a decision support system, because they allow the user to browse the available data and withdraw conclusions from it that couldn't be withdrawn before.

The objectives and requisites defined on the beginning of the project related to the **Support Features** and **Exams Importer** were met and the current version of the **M-VIS** features these two components. This current version is fully functional and tested, being ready to be used.

However, the **Advanced Features Component** objectives and requisites are yet to be met; these objectives are rather ambitious, and more time and research are needed in order for a fully functional version to be implemented. In despite of that, the foundations for the implementation of more advanced visualization features where layered.

On a personal level, the developed work allowed me to improve my skills, namely on code writing and graphical aspect design. Also, I would point out that the greatest difficulty met during the project was its dynamic nature, and the consequent dynamic change of requisites. This was counterproductive, because the implementation of new unplanned features implied the rewriting of already functional code, hence wasting time.

Although the **M-VIS** support features' objectives were met, the module could still be improved further. Suggested improvements are:

- Search exams using as a criterion the exam type;
- Search exams using as a criterion the pathology (possible once the pathology catalogue (manual and automatic) is implemented);
- Implement exam tags, that could also be used as a search criterion.
- Since the M-VIS is intended to be used by doctors, it would be constructive to gather doctors' opinions on the module usability, in order to improve it.

Focusing in the future objectives, the development of the visualization module was just one objective among all the work that there is yet to be done in order for the **BW-MES** to become complete and work as a decision support system. Proposed future objectives are the development of more integration modules, the development of more advanced visualization features (and their consequent integration with the **M-VIS**), and the data-mining module development.

References

[1] – Workflow: Adapted from www.scbos.com/Info/SCBOS-Site-Glossary.htm (retrieved: August 2008)

[2] – Data mining: www.statsoft.com/textbook/stdatmin.html (retrieved: August 2008)

[3] - Barbeiro, Paulo. "WIA-DM Ophtalmologic Decision Support System based on Clinical Workflow and Data Mining Techniques." Coimbra, 2007.

[4] - Ferreira, Edgar. "WIA-DM Ophtalmologic Decision Support System based on Clinical Workflow and Data Mining Techniques." Coimbra, 2007.

[5] - Santos, Armanda. "WIA-DM Ophtalmologic Decision Support System based on Clinical Workflow and Data Mining Techniques." Coimbra, 2007.

[6] – iTextSharp: http://sourceforge.net/projects/itextsharp/ (retrieved: August 2008)

[7] – Video player example: http://www.codeproject.com/KB/directx/directshowmediaplayer.aspx?df=100&forumid=4399&select=820191 (retrieved: August 2008)

[8] – The Code Project: http://www.codeproject.com/ (retrieved: August 2008)

[9] – Visual C# Developer Center: http://msdn.microsoft.com/en-us/vcsharp/default.aspx (retrieved: August 2008)

[10] – C# Examples: http://www.java2s.com/Code/CSharp/CatalogCSharp.htm (retrieved: August 2008)

Appendices

Appendice 1 – Project classes and features

On this appendix is presented a description of each Project class and a table with its features (and respective methods)

→ *FormVisualization* (**DockContent**): the *FormVisualization* is the module main window, and it is loaded as a plugin. It allows the user to select a patient, browse the patients exams, open an exam and interact with its exams elements

Table 1 - FormVisualization.cs features

Features	Description	Methods
Patient selection		
Patient search	The patient search can be done through name, internal number, health system number, citizen ID, contributor number, birthdate or registry date.	Database connection GetAllPatientsFromDatabase() textBoxSearch_KeyDown() textBoxSearch_TextChanged() comboBoxCriterium_SelectedIndexChanged() buttonFind_Click() buttonRefind_Click() Find_Patients()
Creates the lists that feature the patients with the most recent exams (general and by doctor) with or without exam type filter	Database connection and last patients with exams lists creation	GetPatientsWithLastExamsByDoctor() GetPatientsWithLastExams() GetPatientsWithLastExamsByExamType() GetPatientsWithLastExamsByDoctorByExamType()
Shows the lists that feature the patients with the most recent exams (general and by doctor) with or without exam type filter	Shows the last patients with exams lists created previously	ShowLastPatientsWithExamsByDoctor() ShowPatientsWithLastExams() ShowPatientsWithExamsByDoctorByExamtype() ShowPatientsWithLastExamsByExamType()
Exam type filter	Checks if the user applied any	comboBoxLP_SelectedIndexC

Features	Description	Methods
application	exam filter to the last patients with exams lists	hanged() comboBoxLPD_SelectedIndex Changed()
Resize of the patient selection visual elements	Resizes the last patients with exams *listviews,* the patient search *listview* and the *tableLayoutPanel* that notifies the user that there are more than 50 search results	listViewLPD_Resize() listViewLP_Resize() listViewSearchResult_Resize() tableLayoutPanelOver50results _Resize()
Patient selection	Selects a patient (among the patients represented through *listview* items)	Selected index change listViewResultVis_SelectedInde xChanged() listViewLPD_SelectedIndexCha nged() listViewLP_SelectedIndexChan ged()
Exams browser		
Patient selection confirmation	After the patient selection, the patient's exams can be accessed through the *buttonAcceptVis*, or by pressing the "enter" key. Besides that, the user can double-click the *listview* item that represents the patient, and directly access the patient's exams	Double-click listViewSearchResult_DoubleCl ick() listViewLPD_DoubleClick() listViewLP_DoubleClick()
		"Enter" key listViewResultVis_KeyDown() listViewLPD_KeyDown() listViewLP_KeyDown()
		Button buttonAcceptVis_Click()
		Method to access the exams AcceptSelection()
Database connections	Methods to connect to the database and retrieve the patient's exams	GetExamsByPatient()
Exams organization	Organizes the patient's exams (by type or by date) accordingly to the selected eye	UpdateExamsNodes()
Exam elements pre-visualization	Shows the exam elements preview (from the exam selected on the *tree view*)	ExamSendToPreviews()

Features	Description	Methods
Exam visualization	Opens the selected exam, and shows its elements	Method to load the exam elements buttonExamOpen_Click() ExamSendToImagesBrowser()
		Method to open an exam through a double-click on an exam image preview PreviewDoubleClick() PreviewTothumbnailMatcher()
		Method to open an exam through a double-click on an exam video preview VideoPreviewDoubleClick() VideoPreviewToVideoThumbnail()
Patient information	Shows the PatientInfo *form*, presenting to the user information about the patient	ShowPatientInfo()
Examined eye selection	Selects the examined eye through the *radio buttons*	radioButtonOD_CheckedChanged() radioButtonOS_CheckedChanged() radioButtonBoth_CheckedChanged()
Exams reorganization after examined eye change	This method is called when the examined eye is changed. It receives the selected eye as a parameter, and updates the exams *tree views* accordingly	SelectedEyeChanger(string eyeRef)
User interaction with the exams *trees*	Methods that allow the user to interact with the exams *treeviews* nodes	TreeDate treeDate_KeyDown() treeDate_AfterSelect() treeDate_MouseDoubleClick() treeDate_MouseMove()

Features	Description	Methods
		TreeType treeType_Keydown() treeType_AfterSelect() treeType_MoudeDoubleClick() treeType_MouseMove()
Keep the exam node selected after changing the examined eye or the exams organization	Method that saves the selected exam node (if the user changes the examined eye or the exams organization), so if the user returns to the same conditions (that is, selects again the previously examined eye or type of exams organization), the node will be selected	tabControlExams_SelectedIndexChanged(
Hide (or show) the Exams Browser	Method that allows the user to hide (or show, if it is hidden) the Exams Browser	buttonExamsBrowser_Click() ShowOrHideExamsBrowser()
Return to the patient selection screen	Methods that, after the Browser being opened, allow the user to return to the patients selection screen	buttonBackToPatients_Click() BackToPatients()
Return to the Browser (after the user confirmed a patient selection, and then returned to the patients selection screen, the user can then return to the previously selected patient)	Methods that, after the user confirmed a patient selection (and browsed the patient's exams), and then returned to the patients selection screen, allow the user to return again to the previously selected patient	buttonBackToNav_Click() BackToNav()
Unselect all previews	Method that allows the user to unselect all the PreviewImage *user controls* on the pre-visualization *flowlayout panel*	UncheckAllPreviews()
Exam Elements Browser		
Select a thumbnail	Method to select an exam element *user control* in the exam elements *flowlayout panel*, and consequently unselect all the remaining *user controls*	SelectThumb()

Features	Description	Methods
Unselect all thumbnails	Method to unselect all the *user controls* in the exam elements *flowlayout panel*	UncheckAllThumbs()
Browse the exam images using keys	Method that receives the key pressed by the user as a parameter, and accordingly to the pressed key, allows the user to select the current image, or visualize the next (or previous) image	NextThumbnail()
Show the exam textual information	Method to show the exam textual information on a *groupbox*, and resizes it	SetTextualInfo() groupBoxExamInfo_Resize()
Select (or unselect) all the exam images	Método que permite seleccionar todas as imagens presentes no *flowlayout panel* do browser de exame (ou, caso todas as imagens estejam seleccionadas, permite desseleccionar todas) Methods to select all the image thumbnails present on the exam elements *flowlayout panel* (if all images thumbnails are selected, unselects them)	toolStripButtonSelectAll_Click() SelectOrUnselectAllThumbnails()
Count the total of exam images	Method that counts how many images the exam has	ImagesInExamCounter()
Count how many exam images are selected	Method that counts how many exam images thumbnails are selected	SelectedImagesToMixerCounter()
Show an image	Methods to visualize an image, and resize it accordingly to the Display Area size	ShowMainPicture() ResizeMainPicture()
Images slideshow	Methods to perform a slideshow with the exam images	buttonSlideShowPlay_Click() timerSlideShowPlay_Tick() enum ImagesBrowserSlideShow SlideShowPlay()
Browse the exam images using the buttons	Methods that allow the user to browse the exam images (visualize the next image, the previous image, the first image, or the last image from the exam)	buttonSlideShowLast_Click() buttonSlideShowFirst_Click() buttonSlideShowNext_Click() buttonSlideShowPrevious_Click()
Enable (or	Method to enable the zoom and	panelNavBig_VisibleChanged()

Features	Description	Methods
disable) the zoom and the slideshow, if an image is being visualized (or not)	the slideshow if an image is being visualized (disables them if a image is not being visualized)	
Resize the image (hiding the Exams Browser, the Exam Elements List and the Textual Information Area)	Methods that hide the Exams Browser, the Exam Elements List and the Textual Information Area, and resize the image in order to fill the Display Area new dimensions (also perform the inverse operation)	pictureBoxNavMain_DoubleClick() toolStripButtonFullScreen_Click() MainPictureBoxChangeScreenSize()
Hide the Exam Elements List and the Textual Information Area	Methods that hide the Exam Elements List and the Textual Information Area, and resize the image in order to fill the Display Area new dimensions (also perform the inverse operation)	buttonHideThumbnails_Click() HideThumbnails()
Save the visualized image	Methods that allow the user to save the visualized image on a specific location	toolStripButtonSaveSelectedImage_Click() SaveSelectedImage()
Perform an image zoom	Methods that allow the user to perform an image zoom, and drag the image if its dimensions are bigger that the Display Area dimensions	Zoom SetZoom() trackBarZoom_Scroll() buttonZoomLess_Click() buttonZoomEqual_Click() buttonZoomMore_Click()
		Image dragging pictureBoxNavBig_MouseDown() pictureBoxNavBig_MouseMove() pictureBoxNavBig_MouseUp()
Show an exam report	Method that shows the report on a *web browser* in the Display Area	ShowReportAtImagesBrowser()
Create a PDF report with the exam selected images	Methods that call the report options *form*, and create the report	buttonPrint_Click() CallCreateReport() CreateReport() UpdateSelectedImagesList() InsertImagesInDocument()

Features	Description	Methods
Show an exam video	Methods to play, pause or stop a video, and browse through the video frames using a *trackbar*. Besides that, allow the user to resize the video	StartVideo() buttonVideoControlPause_Click() toolStripMenuItemPause_Click() PausesOrPlaysTheVideo() buttonVideoPlayerStop_Click() toolStripMenuItemStop_Click() StopsTheVideo() timerVideoPlay_Tick() trackBarVideoPlay_Scroll() ()
		Video resize buttonVideoChangeView_Click() toolStripMenuItemChangeView_Click() pictureBoxVideo_Resize() VideoChangeView() VideoResize()
AFC (currently disabled)		
Drag&drop images to the AFC	Methods that check if the *user controls* are ThumbnailImage or PreviewImage, and if so, send the *user controls* to the AFC	tabControlNavMix_DragEnter() tabControlNavMix_DragDrop()
AFC operations	Methods to send images to the AFC, remove an image or all images from the AFC, and hide the images thumbnails on the AFC	tabControlNavMix_SelectedIndexChanged() buttonSendToMixer_Click() SendToMixertoolStripMenuItem_Click() SendSelectedImagesToMixer() buttonEmptyMixer_Click() EmptyMixertoolStripMenuItem_Click() EmptyMixer()

Features	Description	Methods
		buttonRemoveMiniMixer_Click() RemoveImageFromMixer() buttonHideMixerThumbnails_Click() HideMixerThumbnails()
Unselect all the AFC images	Method to unselect all the *user controls* (MixerThumbnail) present on the AFC *flowlayout panel*	UncheckAllMixerThumbnails()
Module		
Show the module options	Method that shows the module options *form*	toolStripOptions_Click()
Show information about the module	Method that shows the *form* with information about the module	toolStripAbout_Click()
Toolbar return button	Button that allows the user to return to the patients selection screen if a patient's exams are being browsed. After this operation, the button allows the user to return to the previously selected patient	toolStripButtonBack_Click()
Split containers definitions	Methods that define the *split containers* minimum and maximum sizes	Horizontal split container between the last patients with exams *listviews* splitContainerLPLPDHorizontal_SizeChanged()
		Split Container vertical entre a lista de pacientes encontrados na pesquisa e as listas de últimos pacientes com exame Vertical split container between the patient search *listview* and the last patients with exams *listviews* splitContainerSearchVertical_SizeChanged()

Features	Description	Methods
		Horizontal split container between the pre-visualization *flowlayout panel* and examined eye *radio buttons* splitContainerExamsNav_Size Changed()
		Vertical split container betnween the Exams Browser and the Exam Elements Browser splitContainerExamsTree_Size Changed()
		Horizontal split container between the Display Area and the Exam Elements List splitContainerNav_SizeChange d() splitContainerNav_SplitterMove d()
		Horizontal split container in the AFC, between its display area and its images thumbnails *flowlayout panel* splitContainerMixerHorizontal_ SizeChanged() splitContainerMixerHorizontal_ SplitterMoved()
If the user defines a different toolbat style, apply it immediately	Method to apply immediately the toolbar style defined on the module options	ApplyToolStyle()

Features	Description	Methods
If an exception occurs, add an entry to the event log	Method that, if an exception occurs, adds an entry to operative system *event log*	AddLogEntry()
Exams import		
Open the Exams Importer	Method to open the Exams Importer	toolStripMenuItemImporter_Click()
Define the folder where the XML file will be saved	Method that allow the user to define the folder where the XML file will be saved	CreateXmlFolderPath()
Folders watch	Methods to perform the folders watch, in order to automatically import new exam files	GetAllElementsPaths() FolderWatcher() UpdatePrevExam timerFoldersWatcher_Tick() RestartFoldersWatcher()

→ *FormVisualizationOptions* (**Form**): form that allows the user to configure the module settings

Table 2 – FormVisualizationOptions.cs features

Features	Description	Methods
Check the saved settings, and present them to the user	At startup, the various settings are read, and presented (the slideshow interval *trackbar* value is set to the defined value, the various *check boxes* are selected or not)	FormOptions_Load()
Change the slideshow interval	Method that, by scrolling the *trackbar*, updates the slideshow interval	trackBarSlideShowTickInterval_Scroll()
If the automatic load of exams	Method that, if the *check box* to automatically load exams at	checkBoxExamsAutoLoad_CheckedChanged()

Features	Description	Methods
at startup is enabled, select the examined eye	startup is selected, enables the *combobox* that allows the user to select the examined eye whose exams will be automatically loaded at startup	
Change the PDF report logotype	Method that allows the user to changes the PDF report logotype	buttonChangeReportLogo_Click()
Delete the PDF report logotype	Method that allows the user to delete the PDF report logotype	buttonDeleteLogo_Click()
Cancel the changes done, and close the settings window	Method that closes the *form* without saving the changes (consequently the settings will keep their original values)	buttonCancel_Click()
Save the changes done, and close the settings window	Method that saves the settings' new values and closes the form (and immediately applies the toolbar style defined by the user)	buttonOK_Click()

→ *FormVisualizationReportConfiguration* (**Form**): form that allows the user to create a report with the selected images, add a comment to each image, and select the number of images per page

Table 3 – FormVisualizationReportconfiguration.cs features

Features	Description	Methods
Load the list of selected images, and present it to the user	Method to show on the *listview* the selected images and their index	FormVisualizationReportConfigurarion_Load()
Change the number of images per page	Methods that allow the user to change the number of images per page (either by selecting a *radio button* or by clicking on	radioButtonOnePerPage_CheckedChanged() pictureBoxOnePerPage_Click()

Features	Description	Methods
	picture box)	radioButtonTwoPerPage_CheckedChanged() pictureBoxTwoPerPage_Click() radioButtonSixPerPage_CheckedChanged() pictureBoxSixPerPage_Click()
Confirm the report creation	Method that confirms the report creation, and calls the method to create the report	buttonOK_Click()
Close the report options window, without creating a report	Method that closes the *form*, without creating a report	buttonCancel_Click()
Delete the comment associated with an image	Method that deletes the comment associated with the selected image on the *listview*	buttonDeleteComment_Click()
By selecting an image, shows the associated comment (if there is one)	Method that, if an image is selected on the *listview*, shows in the *textbox* the associated comment	listViewPictures_SelectedIndexChanged()
Automatically save the comment associated with an image	Method that, if the *focus* leaves the *textbox*, saves the comment	textBoxComment_Leave()
Limitation of the number of lines in the comment	Method that limits the number of lines in the comment to 3	textBoxComment_TextChanged()

→ *PatientInfo* (**Form**): form that presents information about the patient and the patient's exams

Table 4 – PatientInfo features

Features	Description	Methods
Show the patient information	Method that shows the patient information, as well as how many exams there are of each type and in the total	PatientInfo_Load()
Reorganize the listview	Method that allows the user to reorganize the *listview* lines in an ascendant or descendant way, accordingly to the selected column	listViewExams_ColumnClick()

→ *Importer* (**Form**): form that allows the user to define the importation folders and indexes, and perform manual import operations

Table 5 – Importer features

Features	Description	Methods
Listing all the files of a folder	Methods that list in the *checkedlistbox* the selected folder files	buttonLoadFiles_Click() LoadFolderFiles()
Pre-visualize a file	Methods that show the image (if the file is an image file), as well as show the filename code information (if possible). If a error occurs (e.g. the filename is not in the correct code) a error message appears.	checkedListBoxFilesInFolder_SelectedIndexChanged() SetTextualInfoToError()
Select all files	Methods that allow the user to select (or unselect) all the files present in the *checkedlistbox*	checkedListBoxFilesInFolder_ItemCheck() checkBoxSelectAll_CheckedChanged() CheckAllFiles()
Register files in the database	Methods that register in the database the *checkedlistbox* selected files	toolStripButtonImport_Click() buttonInsert_Click() ManualRegistryInTheDB() GetAllImagesPaths() GetAllPdfsPaths() AnalyzeFile() SaveExamination() SetPreviousExamID()
Keep a registry of all the analyzed files	Method that, for each analyzed file, adds an entry on the XSD, regardless of the file successful database registry	AddRegistryEntry()
Save in a TXT file the names of the files that could not be registered in the database (only in manual importation)	Method that creates a TXT file with the names of the files that could not be registered in the database through manual importation, as well as the failure reason (only in manual importation)	AddFileToErrorLog()
Add a folder	Methods that add a folder for files importation (in the *listview* and in the XSD)	buttonFolderAdd_Click() AddFolder()

Features	Description	Methods
Remove a folder	Methods that a remove a folder (from the *listview* and from the XSD), as well as its associated indexes	buttonFolderRemove_Click() RemoveFolder()
Update the equipment and the indexes when a folder is selected	Method that when a folder is selected, updates the equipment *combobox* and the indexes *datagrid*	listViewFolders_SelectedIndex Changed()
When an equipment is selectec, update the list of possible exam types	Method that updates in the indexes *datagrid* the list of exam types associated with selected equipment	comboBoxEquipment_Selected IndexChanged()
Associate an equipment to every folder	Method that, if the equipment *combobox* is closed without selecting any equipment, shows a *message box* asking the user to select an equipment, and drops the equipment *combobox* again	comboBoxEquipment_DropDo wnClosed()
Save the changes done	Methods that check the changes done and save them in the XML file	toolStripButtonSave_Click() buttonSaveXSD_Click() UpdateImporterFolders()
Change a folder status (being watched or not)	Method that changes the folder status (if i tis being watched or not)	checkBoxIsListening_Checked Changed()
Verify if the indexes are valid	Methods that verify if the data inserted by the user in the *datagrid* is valid (e.g. verify if there are repeated indexes)	dataGridViewIndexExamType_ Leave() IndexesDataGridVerification() dataGridViewRelations_DataEr ror()
Remove an index	Method to remove an index (from the *datagrid* and from the XSD)	buttonIndexRowRemove_Click( )
Change the folder where the XML file is saved	Method that allows the user to change the folder where the XML file is saved. The XML file is copied to the new folder, and deleted from the previous folder.	ChangeImporterXmlFolderPath ()
Show the analyzed files registry	Method that shows the ImporterRegistryViewer *form*, where the analyzed files registry is shown in a *listview*	toolStripButtonLog_Click()

Features	Description	Methods
Change the Exams Importer settings	Method that shows the ImporterSettings *form*, where the user can change the Exams Importer settings	toolStripButtonOptions_Click()

→ *FormImporterWait* (**Form**): form that presents the exams importation progress during a manual importation

Table 6 – FormImporterWait features

Features	Description	Methods
Update the analyzed file information and the number of analyzed files	Method that updates the *labels* with the name of the file being analyzed and the number of already analyzed files. Also updates the *progress bar* and the *label* with the percentage of analyzed files	UpdateCounter()
Calculate the percentage of analyzed files	Method to calculate the percentage of analyzed files (in relation to the total of selected files)	SetPercentage()

→ *ImporterRegistryViewer* (**Form**): form that shows the registry of all the analyzed files (to be imported)

Table 7 – ImporterRegistryViewer features

Features	Description	Methods
Present the analyzed files registry	Method that, when the *form* is loaded, shows in the *listview* all the analyzed files, the importation result (registered in the database or not), the importation method (manual or automatic), importation date, patient ID and, if the file was not registered, the error message	ImporterRegistryViewer_Load()
Reorganize the listview	Method that allows the user to reorganize the *listview* lines in an ascendant or descendant way, accordingly to the selected column	listViewImporterRegistry_ColumnClick()

Features	Description	Methods
Keep a regular visual aspect while resizing	Method that calculates the *listview* columns width so that a regular visual aspect is achieved regardless of the *form* dimensions	FormImporterRegistryViewer_Resize()

→ *ImporterSettings* (**Form**): form that allows the user to configure the Exams Importer settings

Table 8 – ImporterSettings features

Features	Description	Methods
Check the saved settings, and present them to the user	Method that checks the settings saved values, and presents them to the user (converts the folders watching interval from milliseconds to minutes, and sets the *trackbar* value to it)	ImporterSettings_Load()
Change the folder where the XML file is saved	Method that calls the method to change the folder where the XML file is saved (ChangeImporterXmlFolderPath())	buttonChangeXMLLocation_Click()
Change the folders watching interval	Method that changes the folders watching interval accordingly to the *trackbar* value	trackBarListener_Scroll()
Exit without saving the changes done	Method that closes the *form* without saving the changes done	buttonCancel_Click()
Save the changes done and exit	Método que guarda nas *settings* as alterações realizadas e fecha a *form* Method that updates the settings and closes the *form*	buttonOK_Click()

→ *SaveExamImporter (SaveExamImporter.cs)*: class to import exams to the database

Table 9 – SaveExamImporter features

Features	Description	Methods
Register exams and exam elements in the database	Method that registers exams and exam files in the database	PerformSave()

→ *PreviewImage* (**UserControl**): *user control* that represents a exam image thumbnail in the exams pre-visualization *flowlayout panel*

Table 10 – PreviewImage features

Features	Description	Methods
Perform drag&drop (currently disabled)	Method that allows the user to perform *drag&drop* with this *user control*	pictureBox_MouseDown()
On double-click, visualize the image and open the exam	Method that visualizes the image and opens the exam by double-clicking the *user control*	pictureBoxtPreviews_DoubleClick()

→ *PreviewReport* (**UserControl**): *user control* that represents the exam reports (and indicates how many reports there are in the exam) in the exams pre-visualization *flowlayout panel*

Table 11 – PreviewReport features

Features	Description	Methods
Show how many reports there are in the exam	Method that presents to the user (using a *label*) how many reports there are in the exam	PreviewReport_Load()

→ *PreviewVideo* (**UserControl**): *user control* that represents a exam video thumbnail in the exams pre-visualization *flowlayout panel*

Table 12 – PreviewVideo features

Features	Description	Methods
On double-click, play the video and open the exam	Method that plays the video and opens the exam by double-clicking the *user control*	pictureBoxTVP_DoubleClick()

→ *ThumbnailImage* (**UserControl**): *user control* that represents an image thumbnail in the Exam Elements List *flowlayout panel*

Table 13 – ThumbnailImage features

Features	Description	Methods
Perform drag&drop	Method that allows the user to perform *drag&drop* with this *user control*	pictureBox_MouseDown()
Select an image	Method that allows the user to select the image represented by the *user control*	pictureBox_DoubleClick()
Visualize an image	Method that allow the user to visualize in the Display Area the image represented by the *user control*	VisualizeThumbnail()
Browse the images thumbnails using keys	Method that allows the user to browse through the images thumbnails using keys (select the current image, or visualize the next or previous image)	pictureBox_PreviewKeyDown()

→ *ThumbnailReport* (**UserControl**): *user control* that represents a report thumbnail in the Exam Elements List *flowlayout panel*

Table 14 – ThumbnailReport features

Features	Description	Methods
Show the report	Method that allows the user to the report associated with the *user control* in a *web browser* in the Display Area	ThumbnailReport_Click() ViewReport()

→ *ThumbnailVideo* (**UserControl**): *user control* that represents a video thumbnail in the Exam Elements List *flowlayout panel*

Table 15 – ThumbnailVideo features

Features	Description	Methods
Play the video	Method that allows the user to play the video associated with the *user control*	pictureBox_Click()

→ *MixerThumbnail* (**UserControl**): *user control* that represents an image thumbnail in the AFC. Currently is not being used.

Table 16 – MixerThumbnail features

Features	Description	Methods
Select a thumbnail in the AFC	Method that unselects all the other thumbnails in the AFC *flowlayout panel*, selects the current thumbnail, and updates the textual information	pictureBoxMixerThumbnail_Click()

Appendice 2 – Project Timetable

Table 1 September Time Table

Tasks	14-Sep	15-Sep	16-Sep	17-Sep	18-Sep	19-Sep	20-Sep	21-Sep	22-Sep	23-Sep	24-Sep	25-Sep	26-Sep	27-Sep	28-Sep	29-Sep	30-Sep
Research about the work done in the previous year																	

Table 2 October Time Table

Tasks	1-Oct	2-Oct	3-Oct	4-Oct	5-Oct	6-Oct	7-Oct	8-Oct	9-Oct	10-Oct	11-Oct	12-Oct	13-Oct	14-Oct	15-Oct	16-Oct	17-Oct	18-Oct	19-Oct	20-Oct	21-Oct	22-Oct	23-Oct	24-Oct	25-Oct	26-Oct	27-Oct	28-Oct	29-Oct	30-Oct	31-Oct
Research about the work done in the previous year																															
Functional prototype																															

Table 3 November Time Table

Tasks	1-Nov	2-Nov	3-Nov	4-Nov	5-Nov	6-Nov	7-Nov	8-Nov	9-Nov	10-Nov	11-Nov	12-Nov	13-Nov	14-Nov	15-Nov	16-Nov	17-Nov	18-Nov	19-Nov	20-Nov	21-Nov	22-Nov	23-Nov	24-Nov	25-Nov	26-Nov	27-Nov	28-Nov	29-Nov	30-Nov
Functional Prototype																														
Exams organization by date																														
Exams organization by type																														
Study and development of *user controls*																														
Interaction between thumbnails and the Display Area																														
Textual information																														
Report viewer																														
Zoom feature																														
Thumbnails selection																														
Counters																														

Table 4 December Time Table

Tasks	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	8-Dec	9-Dec	10-Dec	11-Dec	12-Dec	13-Dec	14-Dec	15-Dec	16-Dec	17-Dec	18-Dec	19-Dec	20-Dec	21-Dec	22-Dec	23-Dec	24-Dec	25-Dec	26-Dec	27-Dec	28-Dec	29-Dec	30-Dec	31-Dec
Interaction between the Exam Elements Browser and the AFC																															
Counters																															
Drag & drop feature																															
Creation of the English resources																															
Exams *treeviews* code update																															
Development of *user controls*																															
Keys navigation on the thumbnails																															
Creation of a generic image structure and consequent code rewriting																															
Slideshow feature																															
Revision of the written code																															
Patients selection feature																															

Table 5 January Time Table

Tasks	1-Jan	2-Jan	3-Jan	4-Jan	5-Jan	6-Jan	7-Jan	8-Jan	9-Jan	10-Jan	11-Jan	12-Jan	13-Jan	14-Jan	15-Jan	16-Jan	17-Jan	18-Jan	19-Jan	20-Jan	21-Jan	22-Jan	23-Jan	24-Jan	25-Jan	26-Jan	27-Jan	28-Jan	29-Jan	30-Jan	31-Jan
Patients selection feature																															
Module menu																															
Options window																															
Patient's data window																															
Code update in order to work with polarimetry exams																															
Image saving feature																															
Return to the patients selection screen feature																															
Report creation feature																															
Implementation of the changes suggested after the module analysis																															

Table 6 February Time Table

Tasks	1-Feb	2-Feb	3-Feb	4-Feb	5-Feb	6-Feb	7-Feb	8-Feb	9-Feb	10-Feb	11-Feb	12-Feb	13-Feb	14-Feb	15-Feb	16-Feb	17-Feb	18-Feb	19-Feb	20-Feb	21-Feb	22-Feb	23-Feb	24-Feb	25-Feb	26-Feb	27-Feb	28-Feb	29-Feb
Report creation feature																													
Work on the 1st Intercalary Presentation																													
Revision of the written code																													
1st Intercalary Presentation																													
Bugs solving																													
Redesign of the textual information graphical aspect																													
Redesign of the previews graphical aspect																													
Implementation of the exam type filters on the last exams registered in the DB lists																													

Table 7 March Time Table

Tasks	1-Mar	2-Mar	3-Mar	4-Mar	5-Mar	6-JMar	7-Mar	8-Mar	9-Mar	10-Mar	11-Mar	12-Mar	13-Mar	14-Mar	15-Mar	16-Mar	17-Mar	18-Mar	19-Mar	20-Mar	21-Mar	22-Mar	23-Mar	24-Mar	25-Mar	26-Mar	27-Mar	28-Mar	29-Mar	30-Mar	31-Mar
Redesign of the thumbnails graphical aspect																															
Bugs solving (graphical design – *split containers)*																															
Work on the module menu																															
Implementation of the “OS&OD” exams organization																															
Study of a video player example																															
Implementation of a video player on the module																															
Update of the module to work with exam elements in video																															
Automatic image resize to the available display area																															
Video resize																															

Table 8 April Time Table

Tasks	1-Apr	2-Apr	3-Apr	4-Apr	5-Apr	6-Apr	7-Apr	8-Apr	9-Apr	10-Apr	11-Apr	12-Apr	13-Apr	14-Apr	15-Apr	16-Apr	17-Apr	18-Apr	19-Apr	20-Apr	21-Apr	22-Apr	23-Apr	24-Apr	25-Apr	26-Apr	27-Apr	28-Apr	29-Apr	30-Apr
Work on the module graphical design																														
Work on the module options																														
Revision of the report creation code																														
Implementation of the report header and footer, and consequent code changes																														
Implementation of the option to save a report logotype																														
Bugs solving																														
Implementation of the changes suggested after the module analysis																														
Research, study and implementation of another video player																														
Revision of the written code																														
Exams importer: folder system implementation																														

Table 9 May Time Table

Tasks	1-May	2-May	3-May	4-May	5-May	6-May	7-May	8-May	9-May	10-May	11-May	12-May	13-May	14-May	15-May	16-May	17-May	18-May	19-May	20-May	21-May	22-May	23-May	24-May	25-May	26-May	27-May	28-May	29-May	30-May	31-May
Exams importer: folder system implementation																															
Exams importer: folders and indexes saving in a XML file																															
Exams importer: reading of the XML file and *datagrid* filling																															
Exams importer: folder's files listing																															
Exams importer: creation of an exam structure																															
Exams importer: filename interpretation																															
Exams importer: verification if the exam element is already registered in the DB																															
Exams importer: organization of the exam elements into exams																															
Exams importer: study and adaptation of the algorithms to register exams in the DB																															
Work on the 2st Intercalary Presentation																															

Table 10 June Time Table

Tasks	1-Jun	2-Jun	3-Jun	4-Jun	5-Jun	6-Jun	7-Jun	8-Jun	9-Jun	10-Jun	11-Jun	12-Jun	13-Jun	14-Jun	15-Jun	16-Jun	17-Jun	18-Jun	19-Jun	20-Jun	21-Jun	22-Jun	23-Jun	24-Jun	25-Jun	26-Jun	27-Jun	28-Jun	29-Jun	30-Jun
Exams importer: study and adaptation of the algorithms to register exams in the DB																														
2st Intercalary Presentation																														
Exams importer: *Bugs* solving																														
Exams importer: implementation of the exam element image type parameter																														
Module update to work with several reports by exams																														
Exams importer: TXT file to save the errors that may occur in the manual importation																														
Exams importer: implementation of the analyzed files log and respective viewer																														
Exams importer: wait window while manually importing exams																														
Exams importer: automatic importation and respective code update																														
Exams importer: redesign of the graphical aspect																														
Exams importer: tests and usability improvements																														

Table 11 July Time Table

Tasks	1-Jul	2-Jul	3-Jul	4-Jul	5-Jul	6-Jul	7-Jul	8-Jul	9-Jul	10-Jul	11-Jul	12-Jul	13-Jul	14-Jul	15-Jul	16-Jul	17-Jul	18-Jul	19-Jul	20-Jul	21-Jul	22-Jul	23-Jul	24-Jul	25-Jul	26-Jul	27-Jul	28-Jul	29-Jul	30-Jul	31-Jul
Tests, revision of the code written and *bugs* solving																															
Project report																															

Table 12 August Time Table

Tasks	1-Aug	2-Aug	3-Aug	4-Aug	5-Aug	6-Aug	7-Aug	8-Aug	9-Aug	10-Aug	11-Aug	12-Aug	13-Aug	14-Aug	15-Aug	16-Aug	17-Aug	18-Aug	19-Aug	20-Aug	21-Aug	22-Aug	23-Aug	24-Aug	25-Aug	26-Aug	27-Aug	28-Aug	29-Aug	30-Aug	31-Aug
Project report																															

BW-Eye Project Weekend/Holiday Classes