Diogo André
Lopes Andrade

Plataforma para rastreio Neonatal com suporte a múltiplos formatos de dados e patologias

A Newborn screening platform with multiple data formats and pathologies

Universidade de Aveiro
2022

Diogo André
Lopes Andrade

Plataforma para rastreio Neonatal com suporte a múltiplos formatos de dados e patologias

A Newborn screening platform with multiple data formats and pathologies

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor José Luis Oliveira, Professor catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Luís Bastião Silva, Professor auxiliar convidado .

Dedico este trabalho à minha família pelo apoio que me deu ao longo deste percurso.

o júri / the jury

presidente / president

Prof. Doutor Paulo Jorge Salvador Serra Ferreira
Professor Associado da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Joel Perdiz Arrais
Professor Auxiliar da Universidade de Coimbra

Prof. Doutor José Luis Guimarães Oliveira
Professor Catedrático da Universidade de Aveiro

agradecimentos / acknowledgements

Queria agradecer primeiramente ao Professor José Luis Oliveira e ao Luís Bastião, pelo apoio na realização deste trabalho.

Agradeço à minha familia pelo grande suporte que tornou possivel toda esta jornada, que foi a vida académica.

E por fim, mas não menos importante queria agradecer aos meus colegas e amigos que partilharam esta parte da minha vida comigo.

Palavras Chave

Sistema de apoio à decisão, software de diagnóstico, rastreio neonatal, RBAC, Rest API.

Resumo

Atualmente é dificil encontrar uma sociedade que não se rodeie com as novas tecnologias, que não param de surgir e evolir. Com esta evolução tecnológica, muitas àreas começaram também a entrar numa nova era, como é o caso da medicina, mais especifiacamente o rastreio de doenças metabólicas em recém nascidos.

À muito tempo, que é uma grande preocupação descobrir os motivos aos quais levavam bebés recém nascidos, aparentemente saudáveis, a vir a falecer. Com o aparecimento de novas técnicas de análise, foi possível começar a desvendar e encontrar soluções para evitar essas fatalidades.

Nos dias de hoje, já existem programas para diagnóstico de doenças metabólicas em recém nascidos e equipamentos capazes de analisar amostras que permitem diagnosticar essas doenças. Tendo em vista dar suporte a técnicos que examinam os dados vindos desses equipamentos, e prevenir erros, houve a necessidade de criar sistemas capazes de dar esse apoio.

Keywords Decision support system, diagnostic software, newborn screening, RBAC, Rest API.

Abstract

It is currently difficult to find a society that is not surrounded by new technologies, which are constantly emerging and evolving. With this technological evolution, many areas also began to enter a new era, as is the case of medicine, more specifically the screening of metabolic diseases in newborns.

For a long time, it has been a great concern to discover the reasons why newborn babies, apparently healthy, die. With the emergence of new analysis techniques, it was possible to begin to unravel and find solutions to avoid these fatalities.

Nowadays, there are already programs for diagnosing metabolic diseases in newborns and equipment capable of analyzing samples that allow diagnosing these diseases. In order to support technicians who examine the data coming from these equipment, and to prevent errors, there was a need to create systems capable of providing this support.

Contents

List of Figures

List of Tables

Glossary

PKU	Phenylketonuria	**UI**	User Interface
CF	Cystic Fibrosis	**DAO**	Data Access Object
CH	Congenital Hypothyroidism	**JPA**	Java Persistence API
T4	Thyroxine	**HTTP**	Hypertext Transfer Protocol
TSH	Thyroid-stimulating hormone	**REST**	Representational State Transfer
T3	Triiodothyronine	**API**	Application Programming Interface
IQ	Intelligence Quotient	**JSON**	JavaScript Object Notation
IRT	Immunoreactive Trypsinogen	**SQL**	Structured Query Language
CFTR	Transmembrane Conductance Regulator	**NoSQL**	Not Only SQL
ACBT	Active Cycle of Breathing Techniques	**AJAX**	Asynchronous JavaScript And XML
PDF	Portable Document Format	**HTML**	HyperText Markup Language
CSV	Comma-separated values	**CSS**	Cascading Style Sheets
DBMS	Database Management System	**JS**	JavaScript
MVC	Model View Controller	**RBAC**	Role-Based Access Control

CHAPTER 1

Introduction

In this introductory chapter, the motivation for the elaboration of this work is described, as well as its objectives.

1.1 Motivation

Newborn screening is the name given to the program to identify baby borns with some health conditions. This program is essential, because it leads to better medical care and gives better chances of cure, principally to newborns.

Newborn screening was created so that it is possible to identify newborns who suffer from diseases, usually genetic, such as Phenylketonuria (PKU) or Congenital Hypothyroidism (CH). This has as main objective, to detect the existence of some diseases so that the treatment of these same diseases can be provided as soon as possible.

To give assistance and support in newborn screening, BMD Software has developed two applications, one for analyzing mass spectrometry data, for detection of metabolic diseases, NeoScreen, and another for Cystic Fibrosis (CF) and Congenital Hypothyroidism, NeoScreen FQ and HC.

1.2 Objectives

This work aims to extend the NeoScreen FQ and HC software to support multiple data formats, such as TXT and CSV/XLSX. Currently, the software can only import files with samples in TXT format, and that follow a single structure, so one of the goals is to improve this import system so that the files do not need to follow such a rigid structure.

Another big goal is to add an authorization system to the application. In the current state of the software, only authentication is required and guarantees access to all its resources.

1.3 Outline

This document is structured in six more chapters, briefly described below.

In second chapter, will be initially presented the Portuguese program of early diagnosis and its importance, some diseases that can be analyzed by this program. Also will expose similar newborn screen applications, their features and differences from our own application.

The Chapter 3, some technologies to be used in the development of the solution were discussed.

The following chapter, Chapter 4, contains the system requirements analysis, as well the description of system use cases and actors.

In Chapter 5, we will present what was developed taking into account the requirements already presented in Chapter 4, for the design of the solution.

The chapter 6, contains the importance of tests to ensure the quality of the software, as well as the tests carried out in the development of this project.

In the last chapter, the conclusions about the development of this work will be presented, as well as the difficulties felt and some points for future work.

CHAPTER 2

State-of-Art

In this chapter, will be initially presented the Portuguese program of early diagnosis and its importance, some diseases that can be analyzed by this program. Also will expose similar newborn screen applications, their features and differences from our own application.

2.1 Newborn diagnosis program in Portugal

The public Portuguese health program, known as the "heel prick test", at the initiative of Doutor Jacinto de Magalhães, has the main purpose of neonatal screening for rare diseases. Its objective is to control and prevent the evolution of the tracked pathology, giving the appropriate treatment as soon as possible [1]–[3].

The National Newborn Diagnosis Program, current National Neonatal Screening Program, began in 1979, exclusively for the purpose of screening for PKU, proposed by the Institute of Medical Genetics. Two years later, the screening for CH began, with the program reaching 70% of newborns, and in 1986, 85% [4], [5].

In 1992, the mark of 1 million babies screened was reached and the Portuguese Association of Phenylketonuria and other metabolic diseases (APOFEN) was created. In the same year and until 1995, an experimental screening for CF was carried out, not continuing due to the lack of effective treatment [6], [7].

In the end of the year 2002, two mass spectrometers were purchased. This equipment allows the diagnosis of several hereditary metabolism diseases with a single sample. In the following year, the equipment was assembled and calibrated, the staff was trained and protocols optimized, in order to obtain better test results [8].

In 2004, in collaboration with the University of Aveiro, a new software specialized in screening was created. That year, Phenylketonuria was also tracked by mass spectrometry, in addition to the classical method [8].

In the year of 2016, there was a reform in the organization of the National Newborn Diagnosis Program, originating the Neonatal Screening Unit. With the new equipment and a

restructuring of the program, new pathologies of Hereditary Metabolism Diseases began to be tracked, now numbering 24 pathologies, in addition to CH [8].

The reference limits for CH, which are the values to be taken into account so that there is a new harvest if necessary, were changed/adjusted in 2008, this change led to a better quality of screening. And only in December 2018, the Ministry of Health approved the inclusion of CF screening in the screening plan [8].

Until today, more than 3,800,000 newborns have been screened and around 2,130 positive cases of metabolic diseases have been detected. Nowaday, the program reaches 99.9% of newborns in Portugal, with these results this program is considered efficient and important in the prevention of metabolic diseases [9]–[11].

2.2 Diseases

2.2.1 Hypothyroidism

Hypothyroidism is a very common endocrine disease. Endocrine diseases are caused by excess or deficiency in hormone production, these hormones are essential chemical substances responsible for controlling and regulating the body's activities and are produced by glands and organs of the endocrine system [12].

In case of Hypothyroidism, it is caused by the low activity of the thyroid gland, which results in less Thyroxine (T4) hormones produced. In some cases, symptoms do not exist or are not presented and the levels of T4 hormones are normal, but exist a increase of levels of Thyroid-stimulating hormone (TSH) and the disease already can be detectable through laboratory tests, this is called Subclinical hypothyroidism and is a mild form or early stage of hypothyroidism [13], [14].

The most common signs and symptoms, for this disease, are weight gain, pain in muscles, joints and head, loss of hair, difficulty concentrating and poor memory, excessive tiredness, irregular menstruation, which leads to difficulties in getting pregnant and decreased libido. In extreme cases, where the levels of hormones Triiodothyronine (T3) and T4 are too low, can lead to depression, personality change and even dementia; in childrens can affect their development, which leads to short stature and delayed puberty [13], [15].

When exists hypothyroidism in pregnancy, and is not identified and treated, it can result in miscarriage, mental retardation and decreased Intelligence Quotient (IQ), because the baby needs the thyroid hormones produced by the mother to develop properly. Others main consequences for the baby, in addition to those already mentioned are cardiac changes, low weight at birth and speech problems [13], [16],

Hypothyroidism can also reduce the chances of getting pregnant, because it can alter women's reproductive hormones. The woman may also develop anemia, have hemorrhage after childbirth, a premature birth and pre-eclampsia, a condition that causes high blood pressure [13], [16].

Hashimoto's Thyroiditis is the most common cause of hypothyroidism, this is an autoimmune disease in which antibodies attack the thyroid gland. Hypothyroidism can also happen

due to iodine deficiency, a condition known as goiter, which causes the thyroid to become enlarged. Another way to develop hypothyroidism is during pregnancy or in the postpartum period, but after these periods it tends to return to normal [13], [15].

Hypothyroidism is tested through a blood test, which serves to check the amount of hormones, T3, T4 and TSH. Who should be aware and get tested for hypothyroidism are people with type 1 diabetes, Down's syndrome, Turner Syndrome, autoimmune disease, goiter, in case of thyroid disease in the family, in case of heart failure, who had radiotherapy to the head or neck and who had thyroid surgery. In the case of womens, these should also be examined during the pregnancy, when they have milk production outside of pregnancy or without breastfeeding and they are over 60 years old, because is more likely them to develop this disease after menopause [13].

The treatment for hypothyroidism is hormone replacement, through the ingestion of synthetic hormones, which must be done fasting 30 minutes before the first meal. In addition to the use of medications, a diet should be made, to lower cholesterol levels and have a good functioning of the liver, and stress should also be avoided [13], [17].

Also exists congenital hypothyroidism, that is a metabolic disorder, which consists of a low capacity of the baby's thyroid to produce hormones. This disease has no cure, however when treated and diagnosed early, the baby can develop normally. The diagnosis must be made in the maternity ward, as only 10% of babies with the disease have symptoms, this is done during neonatal screening, where blood is collected from the baby's heel. There are also imaging tests, such as ultrasound and magnetic resonance imaging, used in diagnosis [13], [18].

2.2.2 Cystic fibrosis

Cystic fibrosis is an inherited disorder, this provokes principally harm to the lungs and digestive system, through the accumulation of mucus in these areas. This disease disturbs the cells that produce sweat, mucus, and digestive juices because people with this disorder have a defective gene that makes these fluids sticky [19], [20].

Cystic fibrosis has a variety of signals and symptoms and before newborn screening was in place it was very difficult to make a diagnosis before the symptoms appeared. Symptoms may not appear until adolescence or adulthood, and in the case of a mild form of the disease it can induce atypical symptoms such as inflammation of the pancreas, infertility and pneumonia. In addition to the symptoms already presented, others are cough, lung infections, sinusitis, wheezing, shortness of breath, poor weight gain and growth, and intestinal blockage, particularly in newborns [19], [20].

Diagnose cystic fibrosis, in newborns, is made by a blood test that verifies the level of the Immunoreactive Trypsinogen (IRT), which is produced by the pancreas. In case the previous test came positive, one of the two extra tests needs to be done because the level of the IRT could be high by other motives, like premature birth or a stressful delivery. One test is a sweat test, this disease provokes a high value of salt in sweat and another is a genetic test, proving a defect in the Transmembrane Conductance Regulator (CFTR) gene. The genetic test also can be done in adults, in case of suspicion of cystic fibrosis in the family and want to

have a child [20], [21].

The cure does not exist, but treatment can be done to smooth and control the symptoms. Managing this disease is complex and requires adequate nutrition and a lot of medicine, to prevent lung infections and intestinal blockage, and remove mucus from the lungs. Also, it is recommended to do physical activities and airway clearance techniques, like the Active Cycle of Breathing Techniques (ACBT), that involves deep breathing, huffing, coughing, and relaxed breathing. Another option is a lung transplant, which cares a high risk but can give some extra years [21], [22].

2.3 Screening test analysis methods

2.3.1 Guthrie test

The Guthrie test or heel prick test, developed by Robert Guthrie in 1962, with the purpose of detecting PKU. This test should be done between the 3rd and 6th day, because the values of the markers in the newborn's blood are only valid from the 3rd day onwards and after the 6th day, these same markers lose sensitivity [23].

2.3.2 Mass Spectrometry

Tandem mass spectrometry is used as a newborn screening test, explored firstly by Neogen Screening, Duke University and the California Dept of Health Services, with the aim of improving the accuracy on detection of the metabolic disorder PKU [24].

2.4 Newborn Screening Applications

2.4.1 NeoScreen

NeoScreen is a Portuguese web application and multi-language, made by BMD Software, a company focused on building technological solutions for health and life science industries. Its purpose is to help analyze data samples from mass spectrometry and detect metabolic diseases [25].

This is a complete application, built in Java Play, with a simple workflow, that works by importing and filtering the samples data of MS/MS, after that the data is stored on SQL Server database, and can be analyzed, in a fast and precise way. For that analysis quality controls and thresholds can be defined, and in the end, it is possible to generate a report of the detected abnormalities [25].

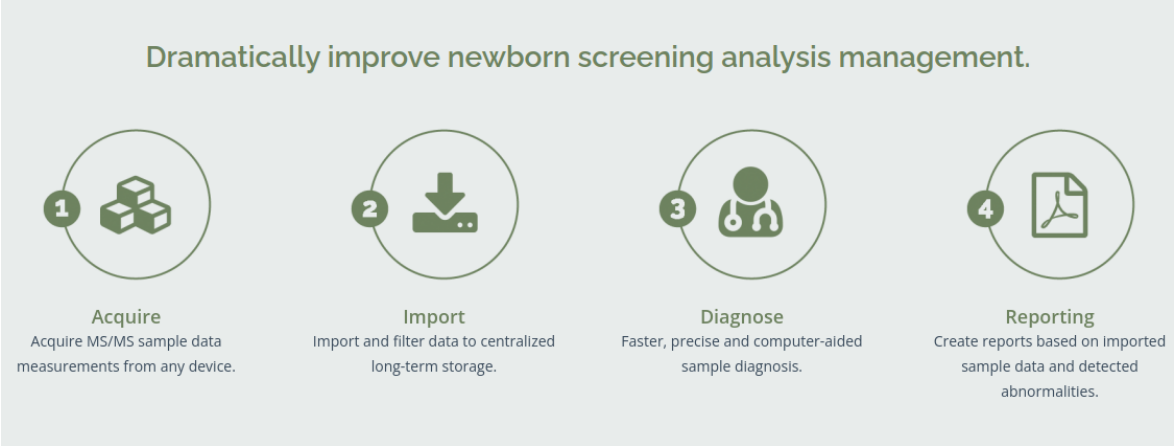


Figure 2.1: NeoScreen application workflow [25].

2.4.2 Specimen Gate

The Specimen Gate is a software which belongs to PerkinElmer, leader in newborn screening, this application is one of the firsts being developed in this area allowing the management of laboratory information. This software has been developed since 1994 and is composed of three configurable applications/modules. [26], [27]

The first module concerns the laboratory, this allows tracking punch samples to results, firstly this receives the specimens and creates a worklist, after that the data pass for a quality control monitor. The result of the last step will be flagged in case of disorders or if needed a re-tests, and could be watched by a cutoff analysis tool [26], [27].

The second one is a multi-language platform, Specimen Gate Screening Center, with this module it is possible to generate patient reports. This application is what makes the connection between the other two, so it is made for tracking of patient contact information and demography, and sending the follow-up cases to the next application [26], [27].

The PatientCare, is the last module, has as primary feature follow the patients, principally cases of positive tests. That is possible because this has a record of the patient's specimens, and a worksheets, that allows to watch the type and status of the disease, and the information about the case [26], [27].

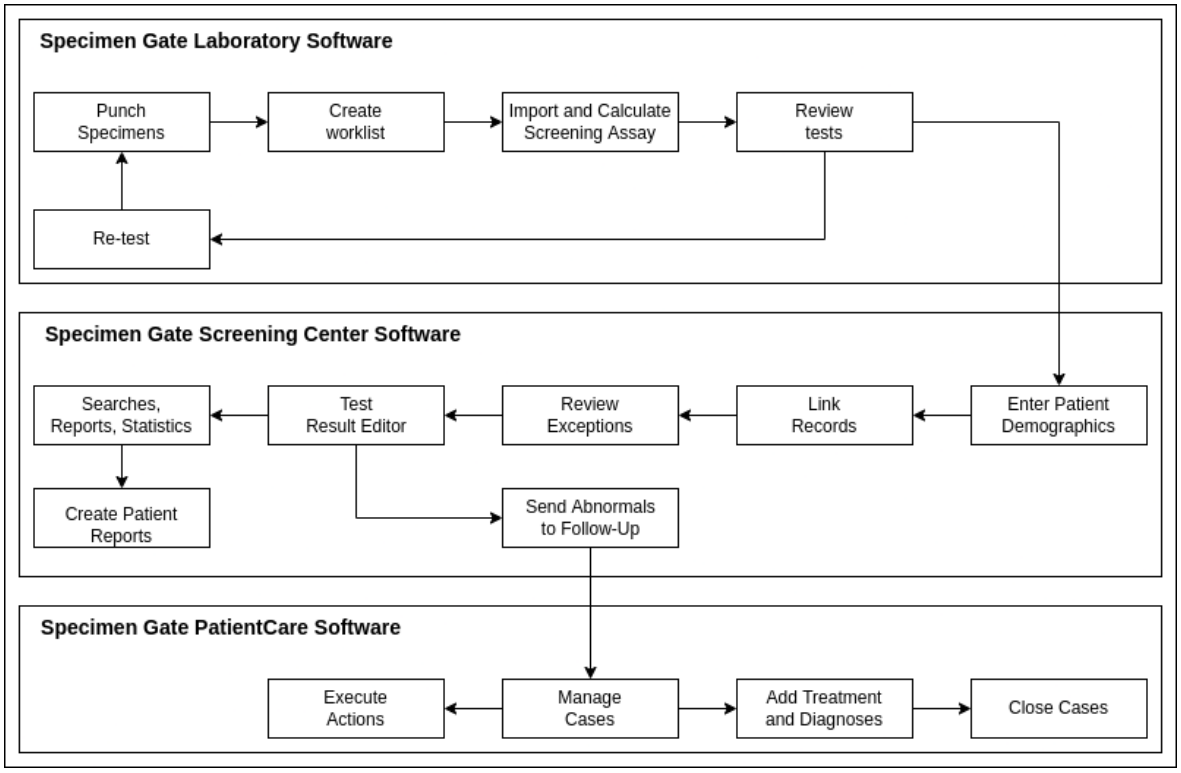


Figure 2.2: Specimen Gate apps workflow [27].

2.4.3 Neometrics Data Management

Neometrics Data Management is software from Natus Medical Incorporated, known as Natus, for managing and reporting metabolic and hearing screening, and environmental testing. Natus is a company that offers beyond diagnostic services, medical equipment, being in the United State most of their clients [28]–[30].

This software like Specimen Gate is also divided into three applications, all fully integrated. A Metabolic Screening Database System (MSDS), that uses the Oracle as its database platform, and allows processing test results and demographic data, for that this application has a powerful search capacity and produces automatic periodic reports. Its objective is to manage data collection and newborn screening reports, easily, quickly and safely [29], [30].

The next application is Neometrics Case Management System (CMS), the goal of this application is to automates the follow-up of patients data creating a workflow process to the responsables. It keeps a complete history of patients, their diagnosis and treatment, and for a better workflow it saves call logs and case history files [29], [30].

The third module, Internet-based Case Management System (iCMS), a web-based software system that extends the functionality of the second application. The goal of this application is not for diagnosing metabolic diseases, but for hearing newborn screening, and works with hearing equipment of the Natus company, the Natus ALGO hearing screeners [29], [30].

2.4.4 Others

OZ Systems

OZ Systems is a company that offers and builds technology solutions for screening. For newborn screening they created a group of applications that work together in order to promote better disease tracking efficiency [31].

That suite of apps is composed of four softwares, the OZ NANI used, on maternity, to save the demographic information of the newborn, the OZ Telepathy NBS, that allows it to manage the status of the sample collection. Another is OZ Connect, which links the first two apps, and aims to reduce duplicate entries and errors caused by manual data entry. The last, is Track Kit from STACS DNA, its aim is to follow the specimens from the maternity to the lab, and give alert in case of delay on sample delivery [31].

StarLims

StarLims Advanced Analytics, is a lab manager application that helps to analyze and visualize the laboratory data. Its purpose is to improve the productivity and efficiency of laboratories, this is not a newborn screen application, but it is also used for that [32].

NéoNat

NéoNat is a French software, used on newborn screen diagnosis, managed by the french, national coordination center for neonatal screening, and is funded by the Assurance Maladie. It receives data from the heel prick test, being capable of processing that data and detecting many diseases, like those explained before [33].

The main features are management of administrative data, duplicate entries and loss to follow-up, monitoring of positive cases and storing them to study, and control markers to check integrity of the samples. This is also capable of making publications of dashboards for the French Association for the Screening and Prevention of Childhood Disabilities and Health Insurance [33].

2.5 Summary

In this chapter, some metabolic diseases are presented in more detail, which the application will support in diagnosing. Some applications similar to this project are also introduced, explaining the differences between them and ours.

CHAPTER 3

Technologies

In this section, some technologies to be used in the development of the solution were discussed. Bearing in mind that there is a long list of possibilities, only the most notable technologies, including those to be used, have been included.

3.1 Database

A database is a set of information/data, related or not to each other, stored in an electronic system. There are two main groups of databases, relational and non-relational, both of which are normally controlled by the Database Management System (DBMS). This controller allow us to manipulate the database, that is, add, update or remove the data stored [34], [35].

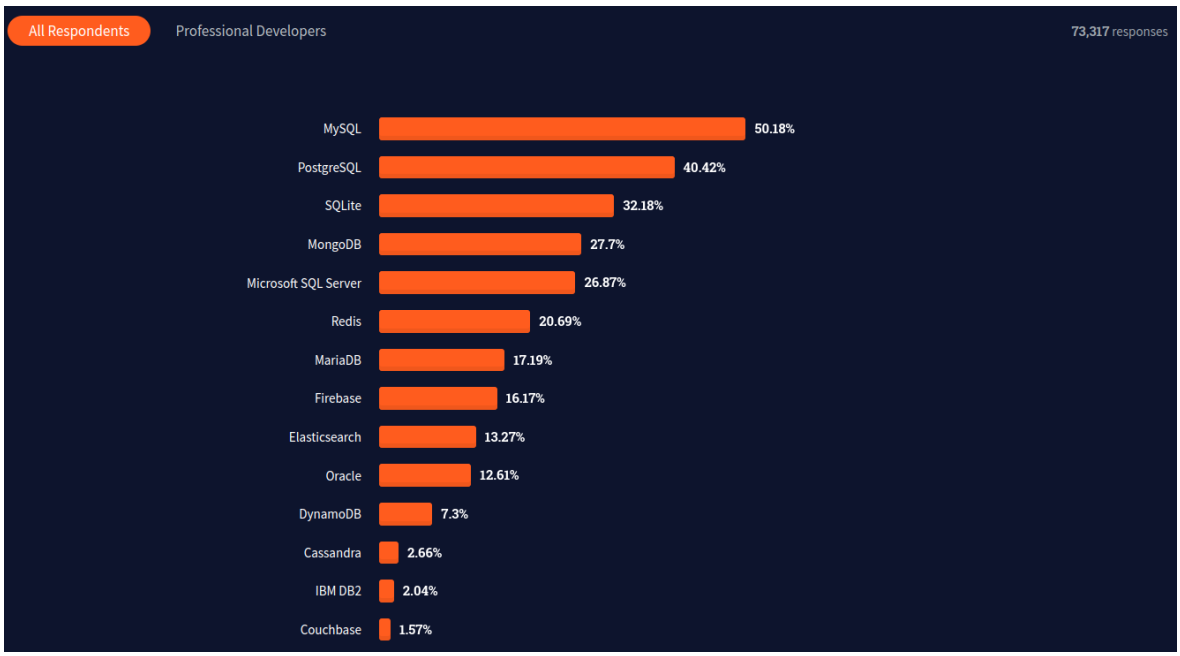


Figure 3.1: Top-Rated Databases for 2022, in a study made by Stack Overflow [36].

3.1.1 Relational

In a relational database, data collections have predefined relationships with each other. The information is stored in tables, and each table containing a unique key, known as "Primary Key", when this key is referenced in another table it is called "Foreign Key" [35].

This type of database has four fundamental types of properties, referred to as ACID, Atomicity (a transaction happens at all or not happens), Consistency (a transaction creates a new valid state of the data), Isolation (several transactions can occur simultaneously, as long as they do not interfere with each other, so as not to cause inconsistencies) and Durability (after the transaction the data is stored persistently on disk) [37].

3.1.2 Non-Relational

Non-Relational databases, also called NoSQL databases, does not use any tabular system, making this type of databases more flexible, scalable and with high performance.

There are four main types of non-relational database, these being graph database, column-oriented database, document data storage and key-value storage [35], [38].

3.1.3 Notable Databases

- **MySQL** is a relational open source database, developed by Oracle, been highly scalable and base on Structured Query Language (SQL). This is available in multiple platforms, like Linux and Windows. Some of there main features are the easily deployment and management, comprehensive transactional support, security and high performance. This is used by many of the most accessed platforms, such Facebook, Uber, Twitter and Netflix [35], [39].
- **SQL Server** is one of the leaders in database technology, this is also a relational database, created by Microsoft, aiming to support a high load of transaction processing. Today, Microsoft SQL Server, is also available in other platforms beyond Windows, and have been receive huge improvements over the years. Its main features are high security, performance, multiple supported editions and excellent data restoration and recovery mechanism. In addition to Microsoft, this database is used by large companies such as Accenture and Stack Overflow [35].
- **MongoDB**, unlike the other databases presented, this one is non-relational, using JSON-like documents. It is open-source and was prepared to deal with large volumes of data, scaling horizontally, with the principles of creating a high performance, scalable and flexible database [35], [40].

3.2 Web Frameworks

Nowadays, web framework have take a huge place on development of web applications, been this a software platform that brings a hole of pre-written components, code snippets and templates. This have as goal to simplify the web development process and make it easier to build web services or web Application Programming Interfaces (APIs). A web application has a back-end and a front-end, with appropriate frameworks to use for each one [41].

3.2.1 Back-end Frameworks

In the back-end, all the manipulation of the database, the authentication and authorization of users and the server handling are carried out, which is why it is called server-side. Back-end frameworks are important to improve the building of the background functioning of the web services, using languages such as Python, Go and Java [41].

3.2.2 Front-end Frameworks

The front-end, also known as the client-side, is the visible part of the website where users can interact. Front-end frameworks bring reusable templates and manage user interaction, using languages such as HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript (JS) [41].

3.2.3 MVC Architecture

The Model View Controller (MVC) is an architectural software pattern, that consists of three components [41], [42]:

- **Model**: representation of the object, which contains the data.
- **View**: visual representation of the data that the model contains.
- **Controller**: sits between the model and the view, controlling the flow of data within the model object and updating the view when the data changes.

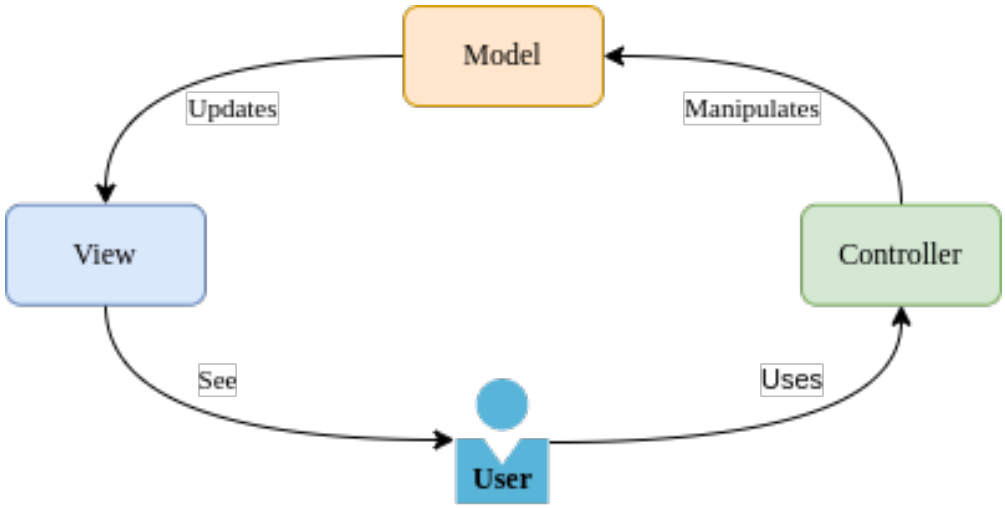


Figure 3.2: MVC pattern architecture diagram [42].

3.2.4 Notable Web Frameworks

- **Play Framework** is a specialized framework for creating modern and scalable web applications, based on a friendly and lightweight web architecture, using Java and Scala languages. It follows the MVC architecture, with the purpose of split the data models from the User Interface (UI) [43].
- **Backbone JS** is a 2010 lightweight, router-concept JS library used to build single-page applications and is designed for client-side structuring. This is also considered an MVC/MV* framework, where the View also does the Controller's job, so it's not a complete MVC architecture [44].
- **React JS** is an open source JS library created by Facebook in 2013. Today, it is one of the most used front-end frameworks for building single-page and mobile apps, and it also adopts MVC architecture. This library is used by large companies like Netflix and Instagram and is based on promoting the reuse of UI components, for improvement of apps speed [45].

3.3 Summary

An important part of software development is the choice of technologies to be used in its construction. Nowadays, there are a vast number of possibilities, which makes the choice of technologies not so easy to do.

Bearing in mind that, the chosen technologies can define the rest of the project, as they all have advantages and restrictions, these choices must be made wisely. To this end in this chapter, some technologies to be used in the development of this project were presented.

CHAPTER 4

Solution Analysis

This chapter contains the system requirements analysis, as well the description of system use cases and actors.

4.1 Requirements

In the development of a software it is necessary, for the good development of the work, to make a simple, but critical analysis of requirements. This analysis must contain the main objectives that the system must fulfill, specifying the functional and non-functional requirements of the application. It is understood as functional requirements of a system, the functionalities that it has, that is, any action that the application allows. Non-functional requirements, on the other hand, serve to impose restrictions on how functional requirements can be implemented.

4.1.1 Functional Requirements

1. **Import sample files**

   This feature of the application is one of the main functions of the system. In this functionality, it should be possible to select the file or files to be imported, validate if each file was correctly parsed, and if is not the case, change the parser and re-validate again. After that should be possible to inspect for a list of control samples, a list of samples and make a upload of an extra PDF file for each sample that complement the sample information. In case, of any failure all the file information can be finally saved.

2. **Sample analysis**

   As a decision support application, in the detection of diseases, all displayed data (samples) must be done in a clear and perceptive way, to avoid mistakes. To facilitate data analysis, it should have a dashboard, where it is possible to filter the samples, by different clinical information, such as samples from a specific file or by diagnosis, among others, or by the date the data was entered into the system.

3. **Flag and publish samples**

   This resource is directly linked to communication with another application, NeoBox. Its objective is to signal samples that need to be evaluated in a special way, as they belong to premature newborns.

4. **Control samples statistics**

   Must be possible, through the uploaded control samples, to view a graphic that allow us to understand if the control samples are between the thresholds or not. That way, is possible to know if the extracted samples are valid to make a diagnose.

5. **Manage configurations**

   In order to be able to carry out any operation in the application, some configurations must first be entered, such as the units, without these the system will not be able to recognize any sample and it will not be possible to save them into the database, or without the sample controls thresholds, the application will be able to identify whether a sample is a control or not. To do so, the system should allow adding, removing and editing configurations, such as, in addition to those already mentioned, ages, diagnosis and sample diagnoses thresholds.

6. **Manage app usage**

   There must be a system of profiles that allow controlling and monitoring the behavior of users, in order to make the platform more robust and secure. Firstly, through the implementation of these profiles it would be possible to give permissions for the user to perform a certain task or not, and secondly, through audit logs to monitor users.

7. **Integration with external applications**

   The goal of this functionality is to have REST API where other authorized applications can through web connection search for samples in our system.

4.1.2 Non-Functional Requirements

1. **Performance**

   The application will have to be able to handle the high number of samples, when validating, importing and searching through the files. It should also be able to retain the information from the validated files, such as their name and samples, for some period of time, before they are definitely saved in the database.

2. **Compatibility**

   As it is a web platform it should be compatible with as many browsers as possible.

3. **Security**

   The application must guarantee data security, for which only authenticated users with access permissions can perform operations in application.

4. **Usability**

   The use of the platform must be learned quickly, for this, it must contain a simple and intuitive workflow, with alert messages in case of error. Another factor that should facilitate its use is multi-language.

5. **Interoperability**

   The software must be able to interact with other products in the NeoScreen family, such as the NeoBox.

6. **Reusability**

   Some application modules are adaptations of the existing NeoScreen platform for diagnosing metabolic diseases.

4.2 Use Cases

The use cases are based on describing the intended features of the software, in a specific way and relating them to the actors of the system.

4.2.1 Actors

The application just has two main actors, being these:

- **Technicians**: these have the most important role in the application, as they are responsible for making the diagnoses. There may be different types of technicians and they will have access to different resources.
- **Administrators**: they have full access to the application's resources, so they can perform all the actions presented below. However, they need to be careful as they have the responsibility to manage all other entities.

Considering that the actors will have different roles in the system, giving each one different privileges, it is mandatory to describe the responsibilities of each one. In figure 4.1, it's showed the use case diagram, where is possible to see the actions that can be made for each entity on the system. The system is compose for two main areas: the administration and the samples diagnosis.

The use cases associated to this areas are:

1. **Manage roles/permissions**

   US 1. Administrators, should be able of create, remove and edit roles.

   US 2. Administrators, should be able of assign the roles to users.

   US 3. Administrators, should be able of make the management of each role privileges.

2. **Manage users accounts**

   US 4. Administrators, should be able of create, remove and edit user accounts.

3. **View System Logs**

   US 5. Administrators, should be able of see users historic logs.

4. **Manage configurations**

   US 6. As a user, should be able to add, remove and edit units of measure.

   US 7. As a user, should be able to add, remove and edit newborn ages.

   US 8. As a user, should be able to add, remove and edit diagnosis names.

   US 9. As a user, should be able to add, remove and edit sample thresholds.

   US 10. As a user, should be able to add, remove and edit control sample thresholds.

   US 11. As a user, should be able to add, remove and edit messages, that tell why is need to repeat the sample.

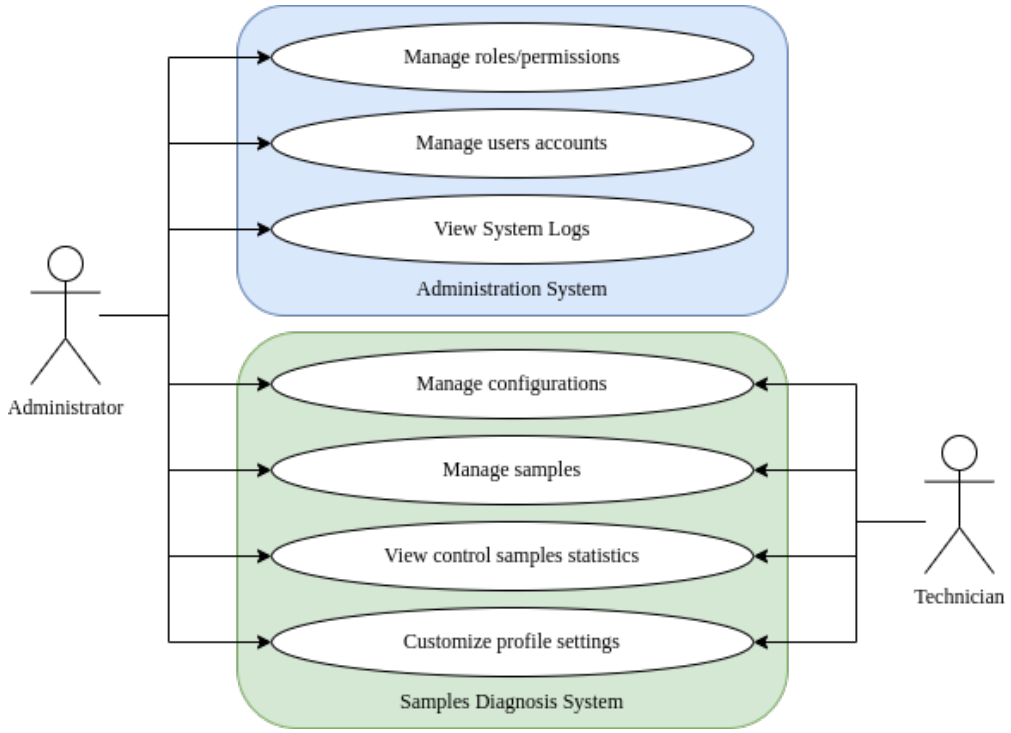


Figure 4.1: Use case diagram.

5. **Manage samples**

   US 12. As a user, should be able of import files with samples.

   US 13. As a user, should be able to add and remove samples.

   US 14. As a user, should be able of edit the sample newborn age.

   US 15. As a user, should be able of edit the sample threshold.

   US 16. As a user, should be able of search of samples, filtered by properties.

   US 17. As a user, should be able of publish samples, into NeoBox.

   US 18. As a user, should be able of export samples, in CSV file.

   US 19. As a user, should be able of view flagged samples.

6. **View control samples statistics**

   US 20. As a user, should be able of view control samples statistics.

7. **Customize profile settings**

   US 21. As a user, should be able of customize profile settings.

4.3 Summary

Another great aspect of software development is to describe the use cases and system requirements before starting to develop the application itself. So that in the future, there are no inconsistencies between what is developed and what is intended, the requirements analysis must be simple, but cover all the objectives that are desired.

In this chapter, the project requirements, as well as their use cases, for each actor in the system are discussed.

CHAPTER 5

Implementation

In this chapter, we will present what was developed taking into account the requirements already presented in Chapter 4, for the design of the solution.

5.1 Technologies

For the development of the solution, it was proposed as an approach to the reuse of some existing modules of the NeoScreen application, such as the sample analysis dashboard and the configurations system.

Bearing in mind that modules from another application are being used, this forces the use of the same technologies as those modules in the development of the new ones. The technologies used in these modules were Java Play as a framework for the web application back-end, Backbone JS in front-end application and Microsoft SQL Server as a database.

Although some of these technologies have been on the market for some time, they have never stopped being worked on and evolved, thus remaining one of the best for the development of this type of applications. So even though we are in a way reusing the old modules, the quality of the application has not been compromised or restricted by the technologies we are using.

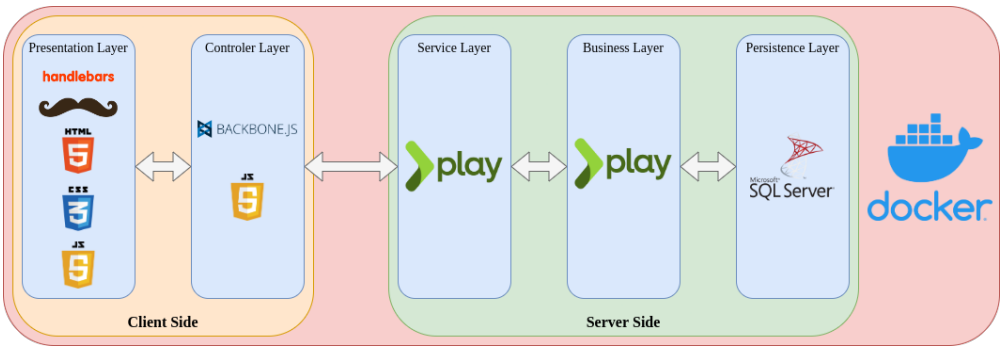


Figure 5.1: System technologies.

5.2 Architecture

The application has the following architecture, as seen in the Figure 5.2, and is composed by four components:

- **Database**: developed in SQL Server, and it is where almost all the information used in the solution;
- **File System**: this is where some extra files with sample information are kept;
- **Rest API**: developed in Java Play, used to respond to web requests, acting as an intermediary between the database and the front-end [49];
- **User Interface**: developed in Backbone JS, so that it is possible to have an easy and perceptible interaction of users with the Rest API. The interaction between the interface and the API is made by HTTP.

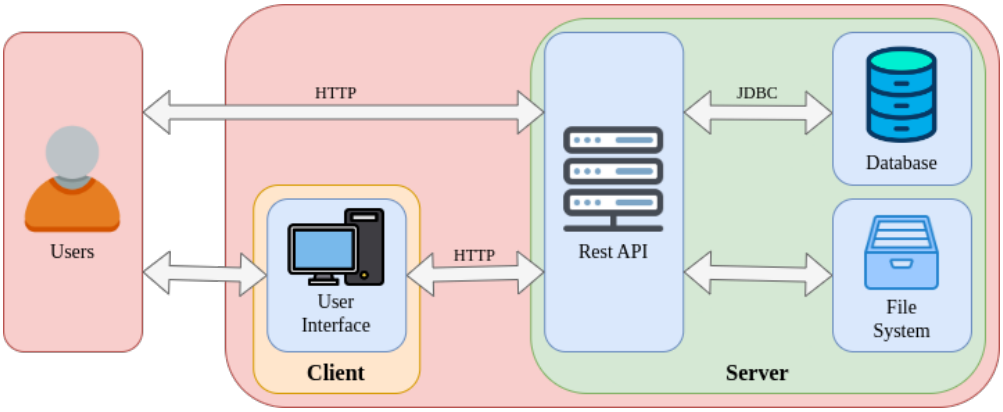


Figure 5.2: Application architecture.

5.2.1 Project Structure

The project is structured in three main modules, as can be seen in the Figure 5.3, each branching off into sub-modules with specific functionalities:

- **app**: this module corresponds to the application's back-end, and is where all the server-side logic is; and can be divided into three distinct groups:
  - The first group of sub-modules make up the MVC architecture pattern:
    - **models**: keep the representations of database tables (JPA entities);
    - **dao**: in this module is made all the accesses to the database, using the Data Access Object (DAO) pattern;
    - **services**: contains the service layer of the application;
    - **controllers**: contains the public methods, which each method represents a endpoint;
    - **views**: where the base view of the interface is kept.

  - The second group of sub-modules allows to control and monitor users:
    * **audit**: records some user actions in the application;
    * **authentication**: makes sure that only authenticated users can access app resources;
    * **authorization**: ensures that only users with permission can access the resource.
  - The last group of sub-module is composed only for one module, with **handler** functions, like requests handling, and a special one, which allows switching between application modes.

- **public**: this module keeps all front-end logic, like public images, external JS libraries, site style and templates. Also, is it where the client-side logic is maintained;

- **conf**: this module includes database migrations, application settings and routes, and files with text message translation, which allows the application to be multi-language.

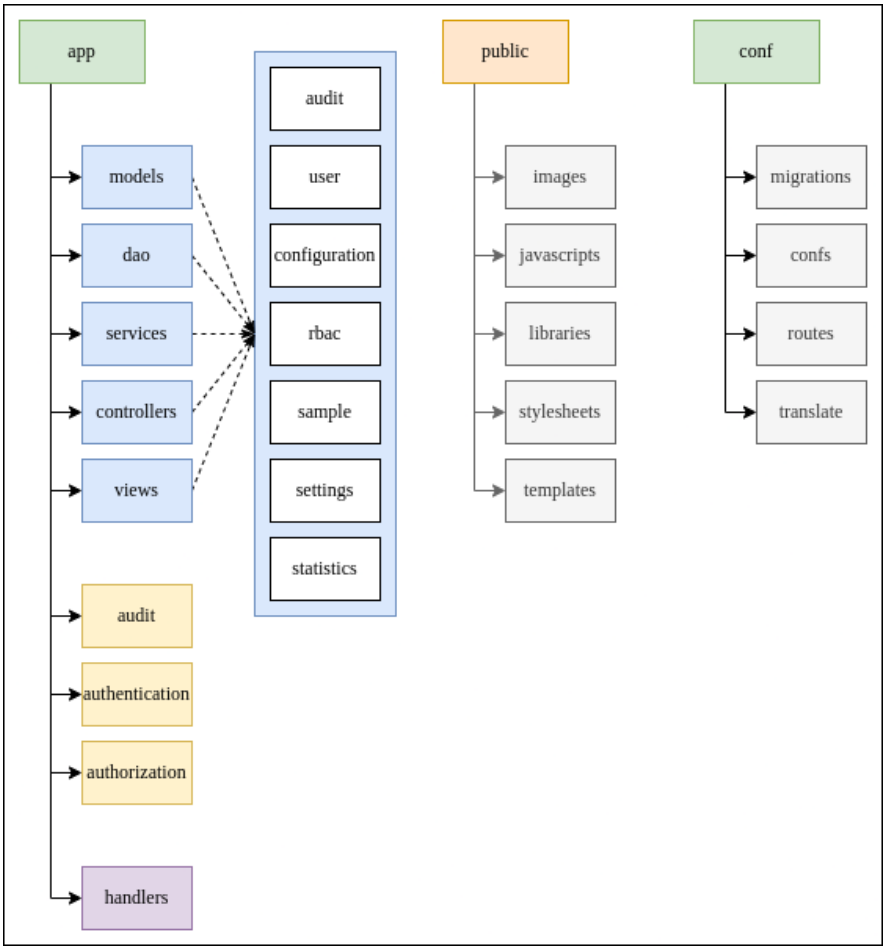


Figure 5.3: Project structure diagram.

5.3 User Interface

The application interface was designed to be responsive, and consists of four main areas Header, Content, Sidebar and Footer. The Sidebar is only used on the configuration/management page and is hidden on other pages.

The Header contains the links that allow browsing on application, the Footer only displays information about the app's rights, the Sidebar, when used, has sub-links that allow browsing through the configurations and management and finally the Content that displays the main content of the page.

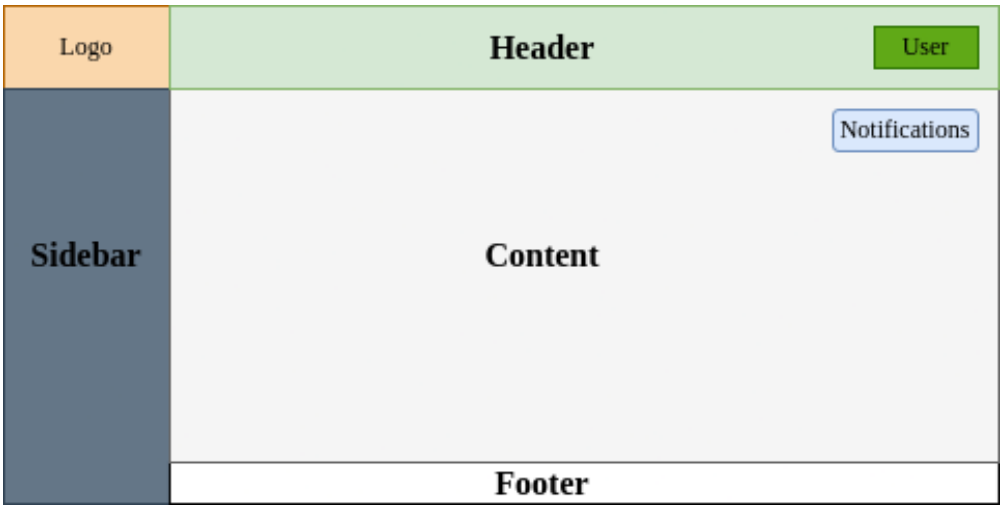


Figure 5.4: Application interface.

The application was designed to provide the necessary feedback to the user, for which a notification system was implemented, which alert the user if the action taken was successful. These notifications appear at the top right of the content, Figure 5.5.

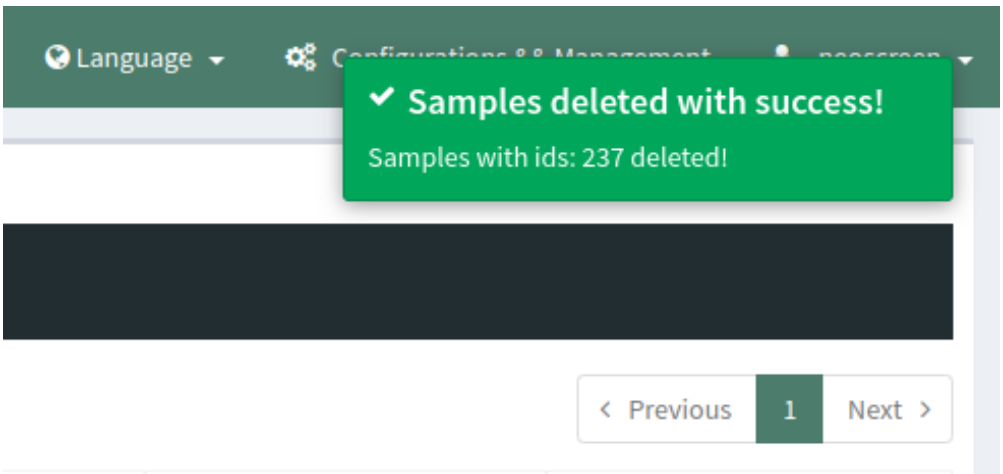


Figure 5.5: Application notification example.

5.4 Role-Based Access Control

In order to be able to control the access to the application's resources, as implemented an authorization system, based on roles, known as Role-Based Access Control (RBAC). This type of authorization system works in the following way, a user can have none or several roles, where each role is an aggregation of permissions, been a permission a definition of what kind of operation (Read or Write) can the user do on a specific resource (endpoint). In case, if a user has a role where has access to a resource and another does not, the system is inclusive, that is, it is enough only one role to have access to the resource for the user to be able to access it.

The management of roles and their assignment to users, can be done by a user with permission to perform these operations, known as a superuser.

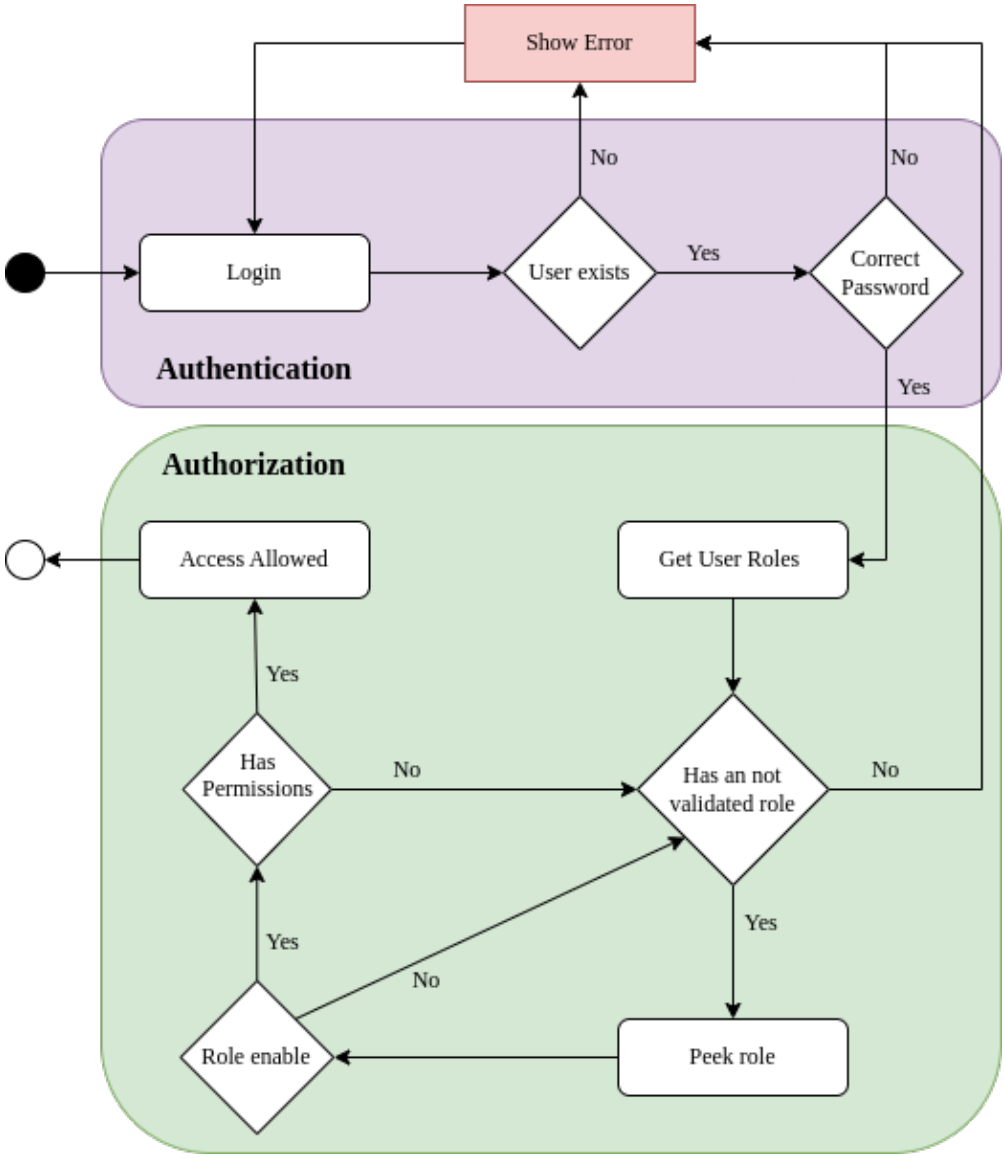


Figure 5.6: Authentication and authorization system state diagram.

5.4.1 Data model

The authorization system database is composed of three main tables, namely:

- **User**: table that stores registered users, that is, who has access to the platform;
- **Permission**: table that stores permissions, that is, the aggregation of an operation (Read or Write) with a category (samples, configurations, ...);
- **Role**: table that stores the roles, that is, the permission groups.

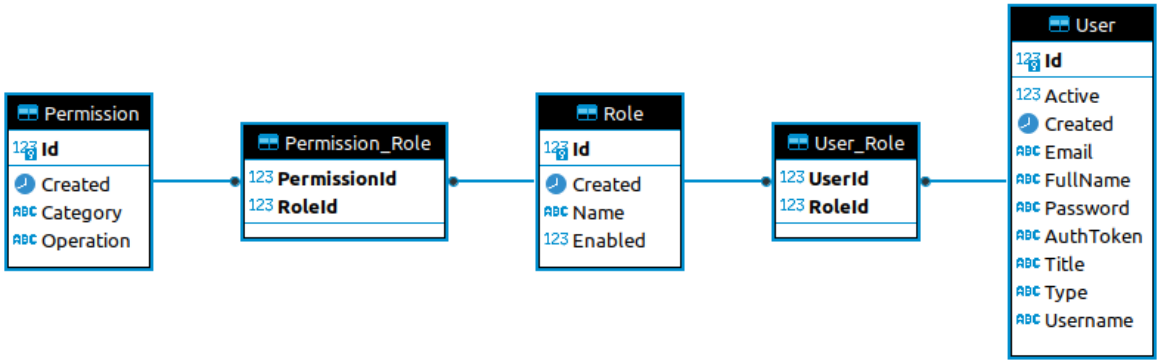


Figure 5.7: RBAC data model entities diagram.

5.4.2 User interface

To make possible to easily manage the roles/permissions of the users, as build a page with a table, as is possible to see in Figure 5.8. In order to have a good usability, three symbols with three different colors were chosen, so that there is a better perception if a role has a certain permission.

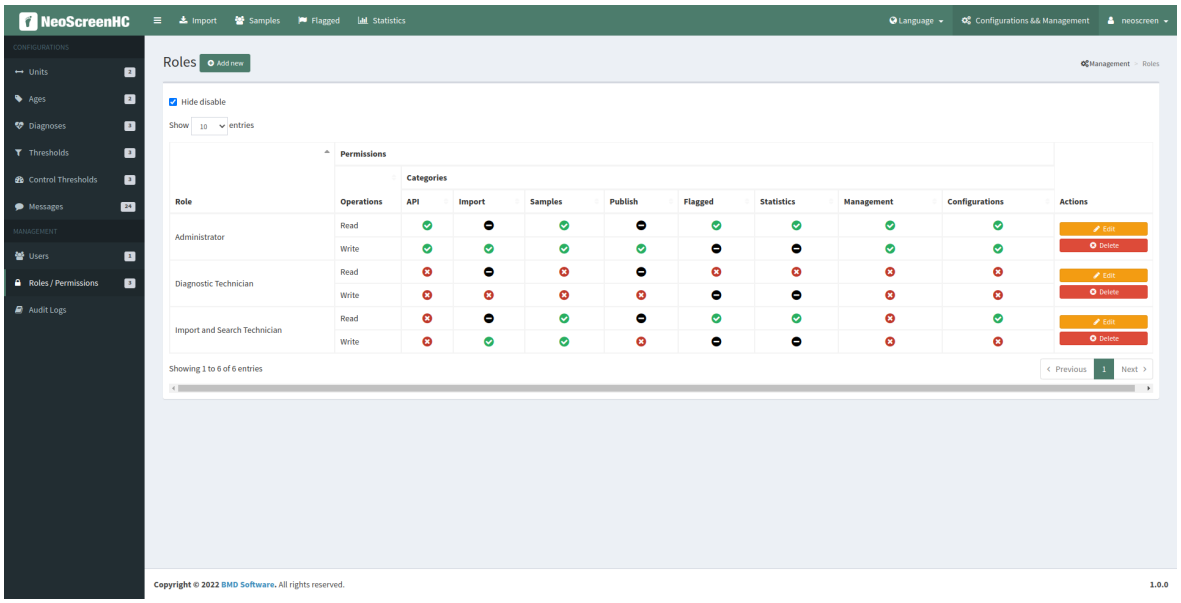


Figure 5.8: Roles/permissions page.

5.5 Import samples

One of the main functions of the application is the upload of text files, with sample information. When the files are imported, they are sent to the server, where they will be validated. Both the validation request and the saving of the samples are done through AJAX.

The system was designed to store the data of the imported files in local storage, and the contents of the files, in cache on the server for a period of time, to make possible to browse in the application or even logout without losing information.

The import of samples is done in four stages, as you can see in the Figure 5.9, which are, importing the file with the samples, parsing that file, validating the imported data and storing them.

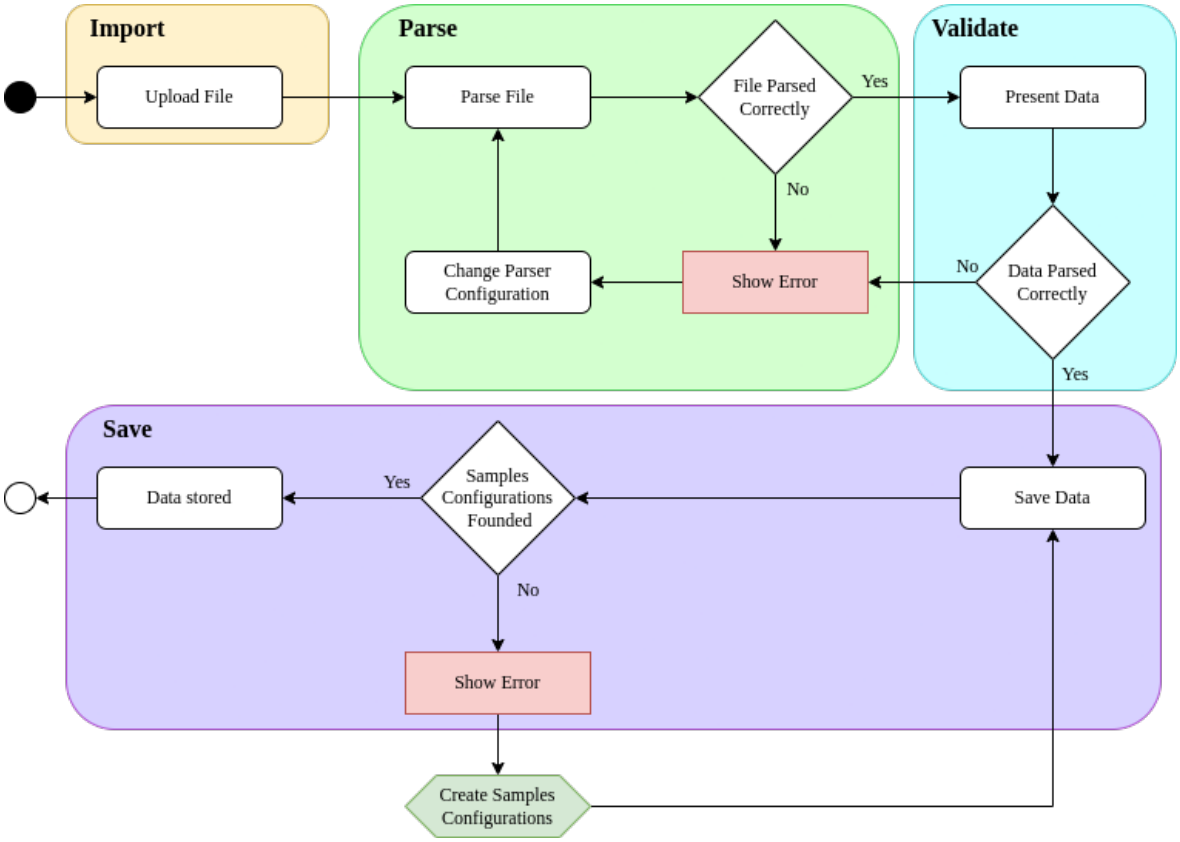


Figure 5.9: Import samples state diagram.

5.5.1 Data model

The import system database is composed of three main tables, namely:

- **File(MODE)**: table that stores the records of imported files;
- **Sample(MODE)**: table that stores the samples data;
- **ControlSample(MODE)**: table that stores the control samples data.

The import system is supported by sample configurations system, and its database is composed of five tables, namely:

- **Age**: table that stores the sample ages;
- **Unit**: table that stores the sample units;
- **Diagnosis**: table that stores the sample diagnosis;
- **Threshold(MODE)**: table that stores the samples thresholds;
- **ControlThreshold(MODE)**: table that stores the control samples thresholds.

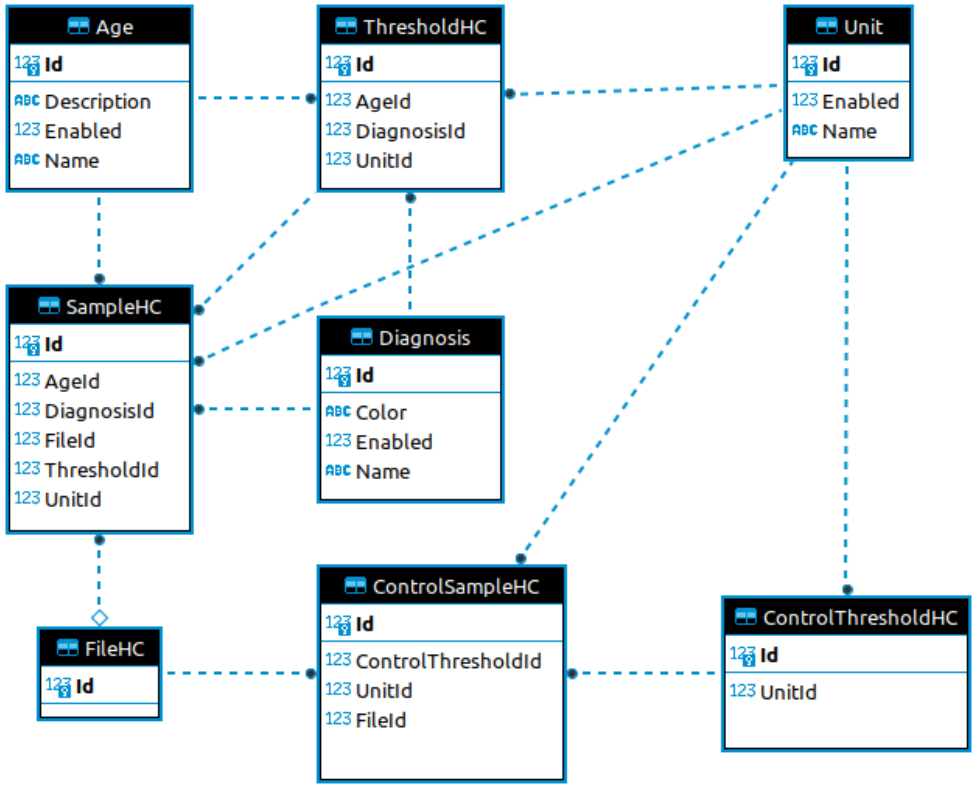
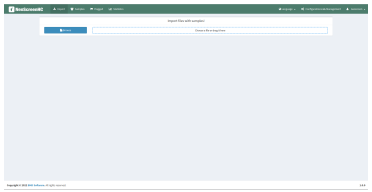


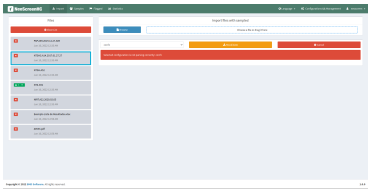
Figure 5.10: Import samples data model entities diagram.

5.5.2 User interface

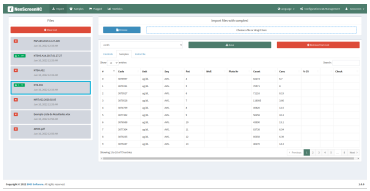
The interface of this page was designed to make it easy to understand the import stage. As you can see in the Figure 5.11, the page can be divided in two, on the left side a list of all imported files, in cache, and some information about it, such as the name, the import date and its import status; on the right side, the content of the selected file appears, such as control samples, samples and description of errors, if any.



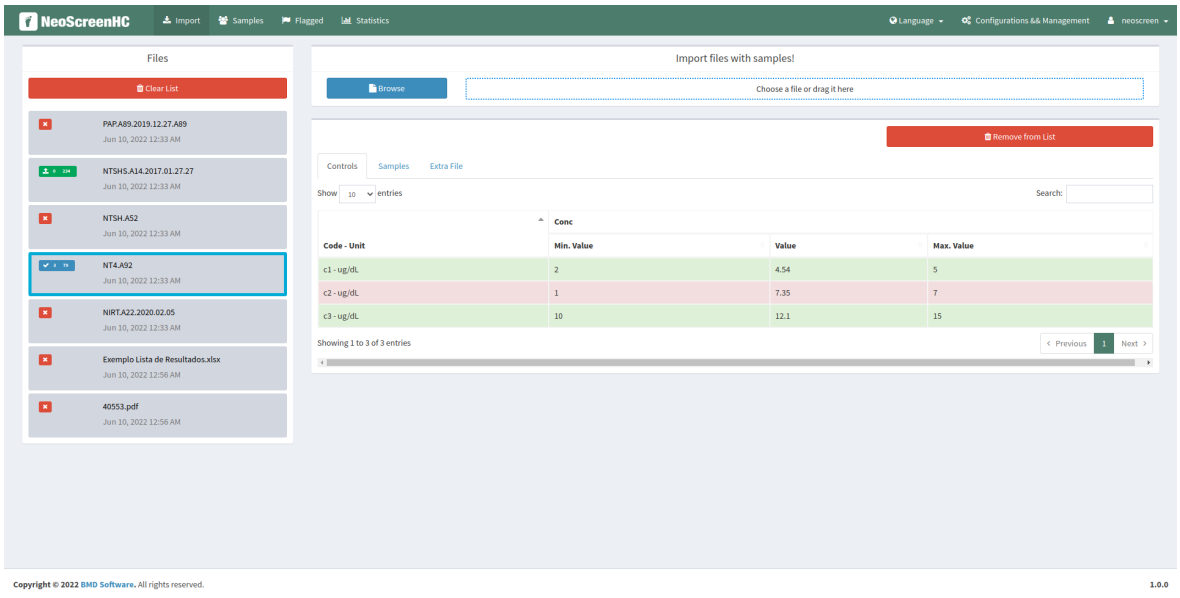
(a) Import page before import any file (Import State).



(b) Imported file with error after parser (Parse State).



(c) Imported file successfully parsed (Validate State).



(d) Imported file successfully saved (Save State).

Figure 5.11: Import page.

5.5.3 Files Parser

TXT Files

The files with sample data, in txt format, come in a tabular form, as you can see in the Figure 5.12a. To parse these files, it was necessary to take into account several factors, such as for each marker type the table has a different header, and that the table alignment was not 100% accurate.

There was already a parser for this type of files, however it was very rigid, that is, each maker type had its own parser and did not use the multiple configuration system, an example is the Figure 5.12b. These settings are saved in the application's settings file and may vary from client to client.

```
65 NT4     neo-T4        92  20-FEB-12.01



 SEQ  PAT     CODE   COUNTS     CONC   UNIT   %CV

AVG.    1       c1   106250     4.54  ug/dL   0.2
AVG.    2       c2    75918     7.35  ug/dL   0.8
AVG.    3       c3    47445    12.1   ug/dL   6.7

AVG.    4  3878957    82274     6.70  ug/dL   9.0
AVG.    5  3879191    70571     8.00  ug/dL  10.1
AVG.    6  3879527    71116     8.19  ug/dL  31.1
AVG.    7  3879528   116095     3.66  ug/dL  13.7
AVG.    8  3874759    45620    12.5   ug/dL   5.0
AVG.    9  3877362    56258    10.3   ug/dL  18.3
AVG.   10  3876588    43990    13.1   ug/dL   8.0
AVG.   11  3877304    83738     6.54  ug/dL   5.5
AVG.   12  3879235    85358     6.39  ug/dL   6.9
AVG.   13  3879247    40276    14.3   ug/dL   1.2
AVG.   14  3879514    45288    12.6   ug/dL   0.8
AVG.   15  3880667    44558    12.8   ug/dL   2.6
AVG.   16  3880842    46776    12.3   ug/dL  12.8
AVG.   17  3880858   137526     1.77  ug/dL  44.6 < STD

 47   18  3879536    75207     7.43  ug/dL
 48   19  3879655    79253     6.99  ug/dL
```

(a) Sample data file for marker type T4.

```
T4 {
  list = [conf1]
  conf1 {
      header = ["SEQ", "PAT", "CODE", "COUNTS", "CONC", "UNIT", "%CV", "NAMELESS"]
      SEQ = 4
      PAT = 4
      CODE = 8
      COUNTS = 8
      CONC = 8
      UNIT = 6
      %CV = 5
      NAMELESS = -1
  }
}
```

(b) Parser configuration for marker type T4.

Figure 5.12: Sample data file and your parser configuration.

CSV/XLSX Files

Files with CSV/XLSX format are text files, in which, as the name of the extension indicates, the data is delimited by a comma. With that in mind, when the file is loaded, the data is parsed and transformed into a JSON format.

To transform this JSON data into a sample object, a JSON Parser was created, which basically converts a map-like object into a sample object.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Linha	No.	ID	Patológico	Padrão Hb	Hb Bart suspected	Hb A	Hb F	Hb D/G/K zone	Hb S zone	Hb E zone	Hb A2	Hb C zone	Outra Hb	A/F	A/X	Observações
2	1	1	AF CTL				30,6	68,8				0,6					
3	2	2	AF CTL				31,3	68,1				0,6					
4	3	3	AF CTL				30,7	68,8				0,5					
5	4	4	AF CTL				31,2	68,2				0,6					
6	5	5	AF CTL				31,3	68,2				0,5					
7	6	6	AF CTL				31,4	68,0				0,6					
8	7	7	AF CTL				31,4	68,1				0,5					
9	8	8	AF CTL				31,5	68,0				0,5					
10	9	9	AF CTL				31,7	67,7				0,6					
11	10	10	AF CTL				31,3	68,2				0,5					
12	11	11	AF CTL				30,9	68,4				0,7					
13	12	12	AF CTL				31,3	68,1				0,6					
14	13	14	4055510		FA		30,4	69,6							0,44		
15	14	15	4055511		FA		22,5	77,5							0,29		
16	15	16	4055512		FA		11,0	89,0							0,12		
17	16	17	4055513		FA		15,1	84,9							0,18		
18	17	18	4055514		FA		21,4	78,6							0,27		
19	18	19	4055515		FA		13,3	86,7							0,15		
20	19	20	4055516		FA		21,7	78,3							0,28		
21	20	21	4055517		FA		23,0	77,0							0,30		
22	21	22	4055518		FA		13,6	86,4							0,16		
23	22	23	4055519		FA		20,0	80,0							0,25		
24	23	24	4055520		FA		23,6	76,4							0,31		
25	24	25	4055521		FA		17,2	82,8							0,21		

Figure 5.13: CSV/XLSX sample data file.

5.6 Chromatography

Chromatography is a method for screening for metabolic diseases, related to genetic abnormalities. This method allows the detection of these anomalies, having the advantage of not needing any extra test, when the results of the tests from this method are confirmed positive for the detection of anomalies [46].

With the need to analyze data from this method, a new application mode, NeoScreenCM, was created. This mode, unlike the other two, NeoScreenHC and NeoScreenFQ, in which the samples had the same properties, this is not the case.

So from there comes the need to make major changes from the database, to the views. For this, it was necessary to develop a special module, previously mentioned, called "ModeHandler" (Code 1), which reads the application's mode in its settings, and uses the classes referring to the correct mode.

```
1  package handlers;
2
3  @Singleton
4  public class ModeHandler {
5
6      private final String mode;
7      private final ISampleDao<? extends Sample> sampleDao;
8
9      @Inject
10     public ModeHandler(Config config, Provider<EntityManager> em) {
11         this.mode = config.getString("mode");
12         this.sampleDao = new SampleDao<>(em, getSampleClass());
13     }
14
15     public Class<? extends Sample> getSampleClass() {
16         switch (mode) {
17             case "HC":
18                 return SampleHC.class;
19             case "FQ":
20                 return SampleFQ.class;
21             case "CM":
22                 return SampleCM.class;
23         }
24         return null;
25     }
26 }
```

Code 1: Piece of code of ModeHandler class.

5.6.1 Data model

As can be seen from the description of some tables in Section 5.5.1, the tables end with the abbreviation of the mode. Previously, these tables were shared by the two existing modes, with the emergence of this new mode this was not possible, so these tables were segmented by the appropriate modes.

In addition to the tables already mentioned, one table in the user settings system was also affected.

- **Options**: table that stores the user options, like the default age, and other that affect directly the samples page;
- **Cols(MODE)**: table that stores the columns of the samples page table.

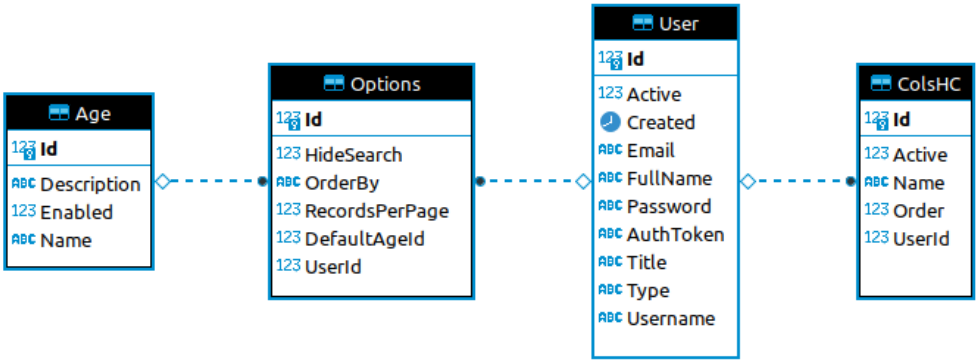
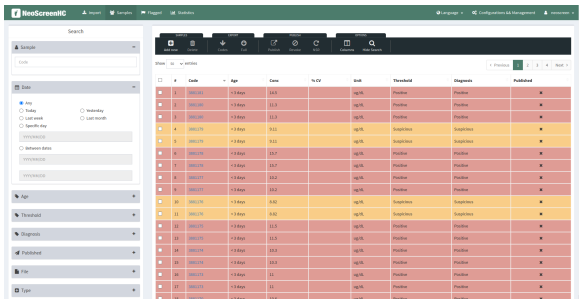


Figure 5.14: User settings data models entities diagram.

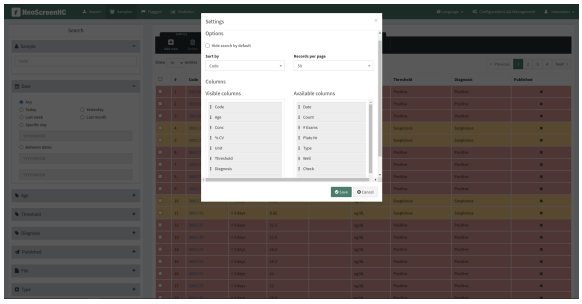
5.6.2 User Interface

As already mentioned, not only the database but also the views needed to be changed, and these changes were made so that the view change was dynamic. The affected views were the Import page (Figure 5.11), Samples page (Figure 5.15) and the sample configurations pages, Thresholds and Control Thresholds.

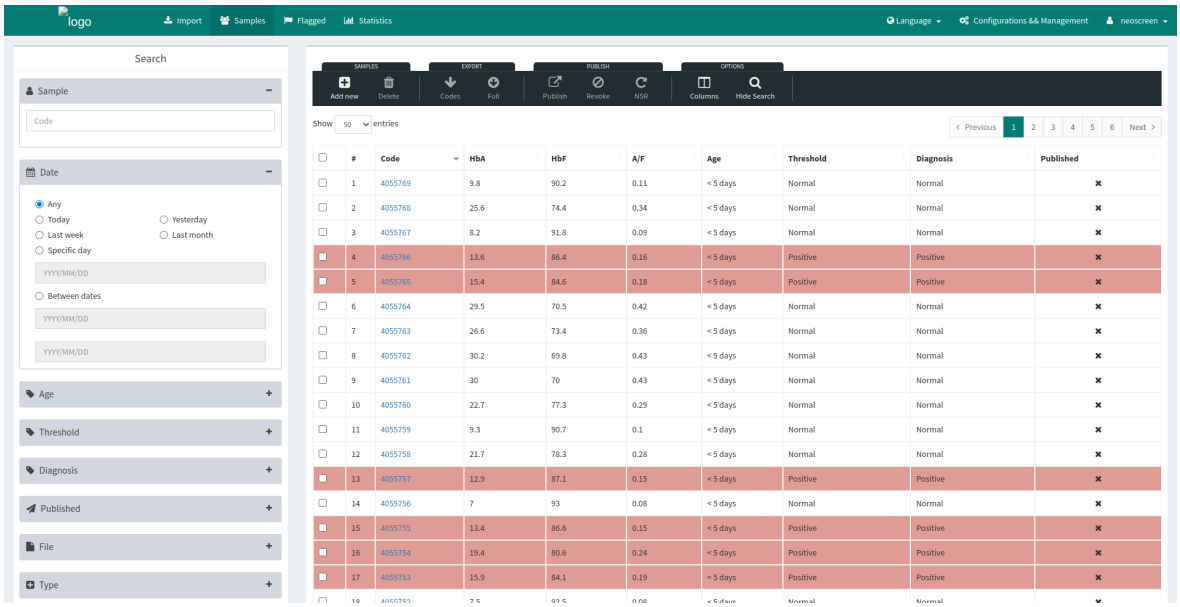
As can be seen in Figure 5.15, the Sample page, each user has their own settings. These settings are shared between the different modes of the application, with the exception of the settings corresponding to the columns of the table, Figures 5.15b and 5.15d, these settings allow the user to choose for each mode which columns he wants to see and in which order.



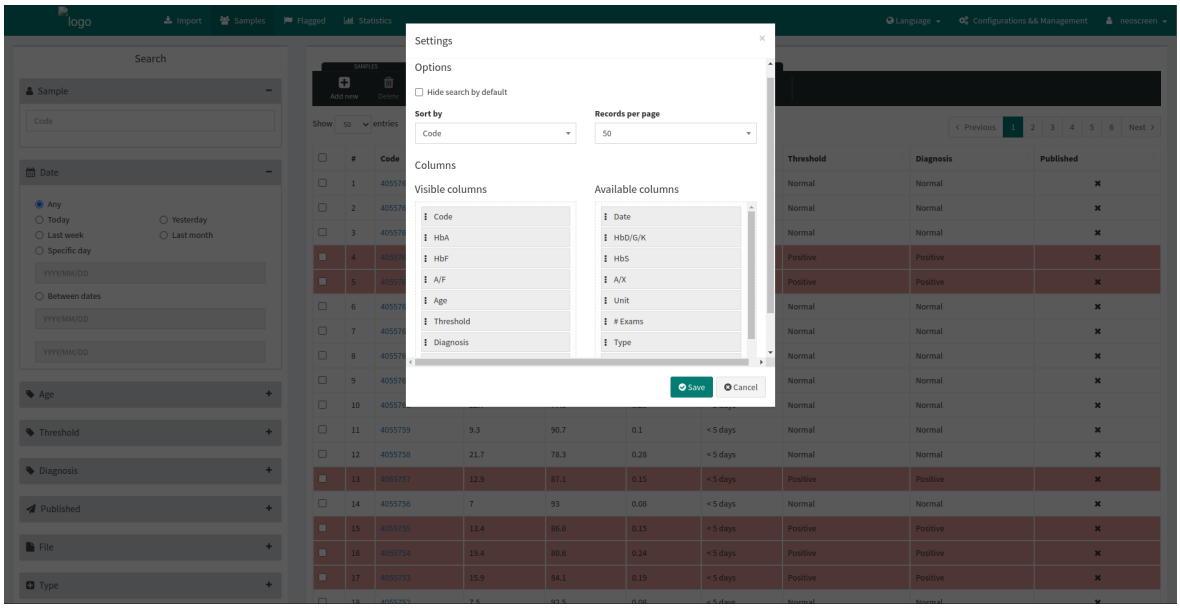
(a) Sample page in HC mode.



(b) Settings of sample page in HC mode.



(c) Sample page in CM mode.



(d) Settings of sample page in CM mode.

Figure 5.15: Samples page.

5.6.3 Import Files

It was from this new mode that the need to import files in the CSV/XLSX format emerged, described in Section 5.5.3.

This method also produces PDF files [47], with extra information regarding each sample, so it was necessary to develop a new functionality capable of importing and storing these files. For this purpose, a new tab was added on the import page for the upload of these files, as can be seen in Figure 5.16, and these files are then saved in the file system.

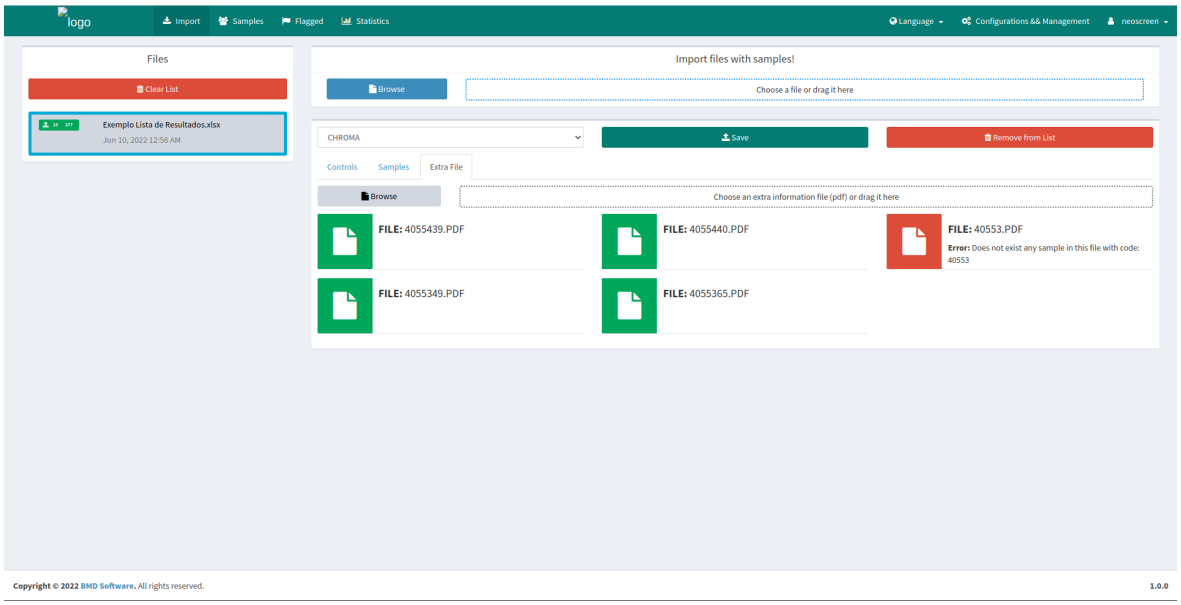


Figure 5.16: Import page showing the extra files imported.

5.7 REST API for external services

The purpose of this API is to facilitate access, to other applications beyond NeoBox, to the information on the samples that are in the NeoScreen database. This API only allows access to samples data, and just for authentified users.

5.7.1 Endpoints

The application has 4 endpoints which allows it to search for samples by different properties. The response is limited to 100 samples, so to access the next group of samples, you need to pass the offset parameter.

URL	Method	Description
/api/search/code/{code:Long}	GET	Search samples by code.
/api/search/diagnosis/{diagnosis:String}	GET	Search samples by diagnosis.
/api/search/date/{startDate:Long}/{endDate:Long}	GET	Search samples between dates.
/api/search/day/{day:Long}	GET	Search samples uploaded by day.

Table 5.1: REST API endpoints.

5.7.2 Authentication

To access API data is needed to be authenticated, for that all requests need to be made with an authentication token or with the user credentials. Each user has their own token and can simply generate a new one, through the settings page or through the PUT request, to the endpoint: /users/token, this will return the new token.

5.7.3 Response Structure

The data response of the API is on JavaScript Object Notation (JSON), and is composed of two attributes: the first is a boolean and represents if a request was successful or not, and second is the body, which contends all the request information. The request body brings the next data, the actual offset of the request query, the count of samples of the query without limit and the list of samples, given by the query.

```
{
    "successful": Boolean,
    "body": {
        "offset": Integer,
        "count": Integer,
        "data": []
    }
}
```

Code 2: API body response.

5.7.4 Queries

Variable	Query
Code	SELECT S FROM Sample S WHERE S.code LIKE :code
Diagnosis	SELECT S FROM Sample S WHERE S.diagnosis.name LIKE :diagnosis
Date	SELECT S FROM Sample S WHERE S.date BETWEEN :startDate AND :endDate
Count	SELECT COUNT(S) FROM Sample S WHERE S.**VARIABLE** LIKE **VALUE**

Table 5.2: REST API SQL queries.

5.8 Summary

In this chapter, all the development steps taken in this project were presented, from the choice of technologies, architecture planning, to the addition of new features.

CHAPTER 6

Software Testing and Quality Assurance

In chapter 6, the importance of tests to ensure the quality of the software is described, as well as the tests carried out in the development of this project.

6.1 Context

In a good software development it is necessary to test and validate that all requirements have been fulfilled.

Software testing is the method of verifying that the software in question meets all the requirements, as well as ensuring that it does not contain any type of errors or bugs. This method involves tests of various types in all phases of software development, which can be automatic or manual.

The use of tests is important, as it avoids future problems, such as having to fix errors after delivery of the product, thus ensuring the quality of the software.

6.2 Tools

Two tools were used in the development of the tests:

- **Junit**: this is an open-source framework for Java programming language, which allows the development of automatic unit tests, in order to improve the quality of the software [47];
- **Selenium**: is also an open-source testing framework, which is used for the development of functional tests for web applications [48].

6.3 Tests

Bearing in mind what was described in the previous chapter, some new modules were created. So there was a need to test them in order to guarantee their quality.

Types of tests, used in this project, to verify the quality of the software:

- **Unit Tests**: serve to evaluate a piece of code, which can be isolated from the rest of the system. These tests are used during the development phase to avoid solving errors at a later stage, which lead to lower productivity and more cost [49];

- **Functional tests**: these are used to validate that, when taking action, the expected reaction is obtained, and that this reaction complies with the functional requirements of the application [50].

6.3.1 Files Parser

As previously explained, one of the main features of the application is to import files, with sample data (Section 5.5). Since these files must be parsed, two parsers were developed (Section 5.5.3).

In order to test these new components, unit tests were written to validate their proper functioning. These tests aimed to cover as many cases as possible of these new classes, in order to avoid future problems. Bearing in mind that these modules are used in the initial phase of using the application, an error can compromise the rest of the application's operation.

The tests carried out made it possible to validate both parsers, even in different application modes, as well as in case of exception, for example when the configuration used is not correct.

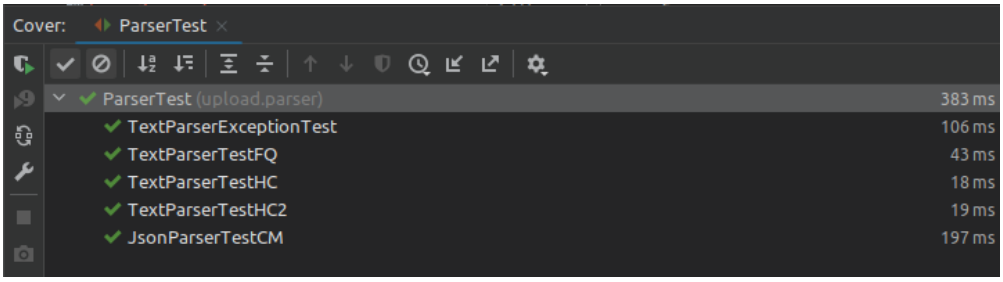


Figure 6.1: Unit tests for parsers.

Code coverage is important in unit tests, as it allows you to see how much the tests analyze the code, and if there is a need to do more tests.

The tests above were executed in IntelliJ, a tool that allows, when running the tests, to automatically generate a coverage report. As we can see in Figure 6.2, the tests performed were enough to cover at least 75% of the parser module.

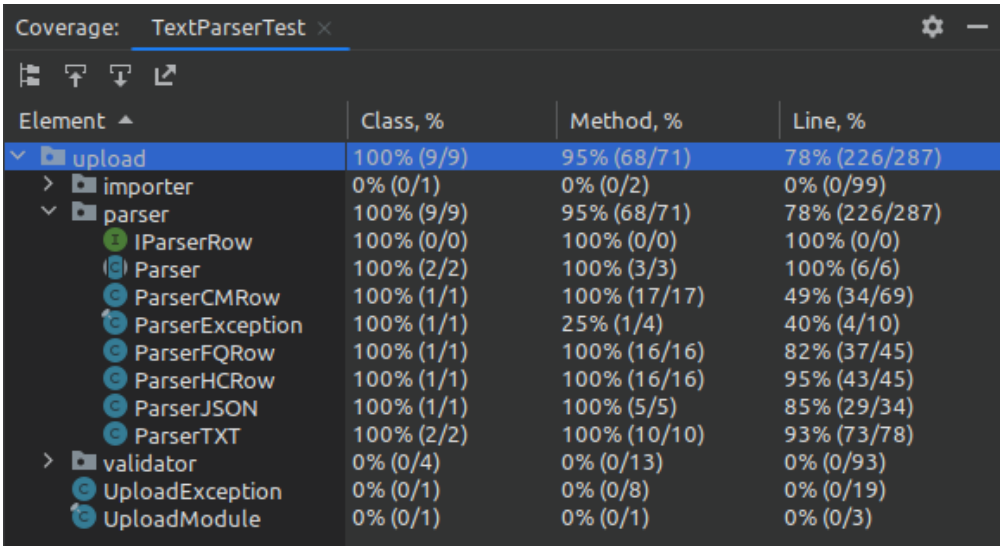


Figure 6.2: Cover of unit tests for parsers.

6.3.2 User Interface

With the unit tests above it is possible to validate a small part of the import process. With the functional tests of the interface, it is intended to validate whether a certain action triggers the intended result.

Tests were carried out on some pages, namely Import and Role/Permission, which were the changed/new pages. Initially, an auxiliary test was made, the Login test, as it is only possible to run the other tests, after logging in.

The functional tests performed on the two pages were:

- **Import Page**:
  - Test that validates if the current mode of the application the file can be validated and stored without problems;
  - Test that validates if the current mode of the application the file cannot be validated, because the mode does not allow the marker type of the file;
  - Test that validates if the current application mode the file cannot be validated, because the parser configuration is not correct;
  - Test that validates if the current mode of the application the file cannot be stored, because sample configurations are missing.
- **Role/Permission Page**:
  - Test that validates whether it is possible to add a new role, edit its permissions and delete it.

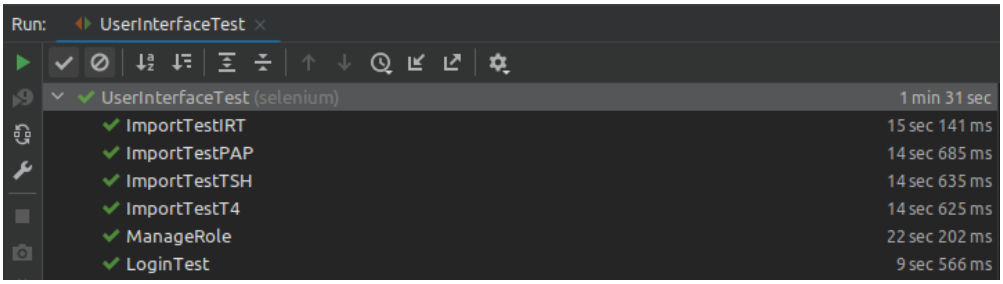


Figure 6.3: Functional User Interface tests.

6.4 Version Control

For version control of this project, the GitHub platform was used, which is one of the many platforms for storing and managing software. This allows several people to work in parallel, without compromising what has already been done and that both are doing separately, using the branching feature.

Normally, a branch is created for the development or repair of some feature and then this branch is merged into the main branch, after going through a code review and testing process so as not to eventually compromise code already in production.

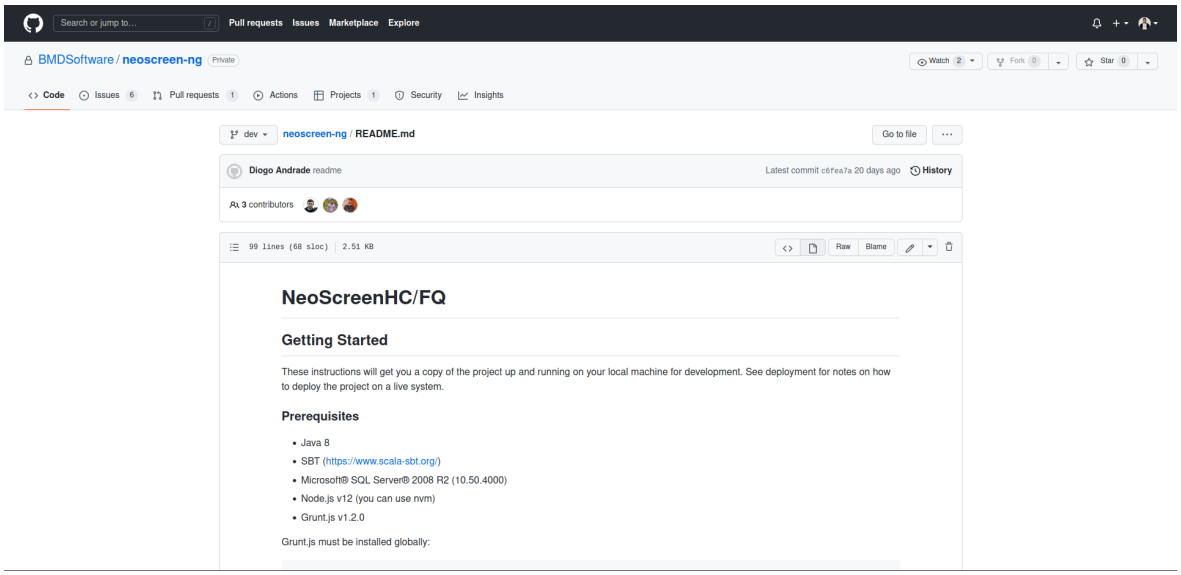


Figure 6.4: Project repository.

6.5 Summary

Throughout this project, several important aspects of software development were mentioned, with software testing being another of these aspects.

Given the new features implemented, explained in the previous chapter, it was necessary to develop tests to validate these same features, and this chapter serves to explain the tests performed.

CHAPTER 7

Conclusion

In this last chapter, the conclusions about the development of this work will be presented, as well as the difficulties felt and some points for future work.

As mentioned, the purpose of this work was to extend an existing software, which allows the screening of metabolic diseases in newborns. This application aims to assist in the diagnosis of these diseases, which when not detected in time can lead to death.

There are few applications of this type on the market, so there is a clear need and reason for the development of this project. As this is a multi-language application, it can be used without any language restrictions, and it also contains several modes, which makes it suitable for diagnosing various diseases.

In conclusion, and taking into account the proposed objectives for the development of this project, all were fulfilled. Having as a final result, an application capable of fulfilling its objective, to support the diagnosis of metabolic diseases in newborns.

7.1 Contributions

My contribution to this application was initially, and taking into account that it already existed, to restructure the project, so that it has an equal logical structure between modules, in order to correctly use the software standards. Regarding the front-end, it was also necessary to make the transition of some modules, from an old technology, to the one used in the new pages, the Backbone JS. With the implementation of this technology on all pages, a system of notifications common to all was easily added, thus improving the usability of users.

Later, I moved on to the development phase of new modules, starting with two sample file parsers, then I create new REST API controllers for external services access and added the authorization system to the entire platform, RBAC. This last module also led to the need to create a view dedicated to managing this system. With the implementation of RBAC it was possible to adjust the views, so that depending on the user's permissions, some buttons and

links, which would lead the user to try to perform an action for which he is not authorized to do so, no longer appear.

Finally, I added a new mode to the application, which led to changes at all levels from the database to the views, as the two existing modes shared the same base, that is, the samples properties, which was not verified in this last mode.

7.2 Difficulties

The difficulties experienced in the course of this work were mostly felt at an early stage. These are due to the fact that this is an ongoing project with some dimension, having spent some time understanding the various modules.

As mentioned above, the starting point was to restructure the project, which led to the loss of some significant time, which did not add anything new to the application, but which plays a fundamental role in the good development of any software.

7.3 Future Work

Some points that were pending in relation to the development of this work, and that should be developed in a future iteration are:

- Improve the authentication system, to also allow authentication by OAuth2, that is, using external services, such as google; and also authentication by API Keys, which facilitated access to the REST API for external services;

- The application has a module, Statistics, which allows you to validate, taking into account the imported control samples, if the quality of the samples is within expectations. This module only works for the HC and FQ application modes, so it is necessary to improve it so that it is possible to use it in other modes.

References

[1] INSA, *Programa nacional de rastreio neonatal categoria*, 2021. [Online]. Available: https://www.insa.min-saude.pt/category/areas-de-atuacao/genetica-humana/programa-nacional-de-diagnostico-precoce/.

[2] ——, *Programa nacional de rastreio neonatal - relatório 2020*, 2021. [Online]. Available: https://www.insa.min-saude.pt/programa-nacional-de-rastreio-neonatal-%5C%E2%5C%94%5C%80-relatorio-2020/.

[3] ——, *Diagnóstico precoce*. [Online]. Available: http://www2.insa.pt/sites/%20INSA/Portugues/DiagnosticoPrecoce/Paginas/diagnosticoprecoce.aspx.

[4] *Programa nacional de diagnóstico precoce - história*. [Online]. Available: http://www.diagnosticoprecoce.org/historia.htm.

[5] *Programa nacional de diagnóstico precoce - programa de 1981 a 1986*. [Online]. Available: http://www.diagnosticoprecoce.org/crono81%5C_86.htm.

[6] *Programa nacional de diagnóstico precoce - programa de 1987 a 1992*. [Online]. Available: http://www.diagnosticoprecoce.org/crono87%5C_92.htm.

[7] *Programa nacional de diagnóstico precoce - programa de 1992 a 1995*. [Online]. Available: http://www.diagnosticoprecoce.org/crono92%5C_95.htm.

[8] *Programa nacional de diagnóstico precoce - programa de 2002 a 2006*. [Online]. Available: http://www.diagnosticoprecoce.org/crono02.htm.

[9] ——, *Programa nacional de rastreio neonatal - relatório 2019*, 2021. [Online]. Available: https://www.insa.min-saude.pt/programa-nacional-de-rastreio-neonatal-%5C%E2%5C%94%5C%80-relatorio-2019/.

[10] H. Rocha, A. Marcão, C. Sousa, *et al.*, "Portuguese newborn screening program," *Revista espanola de salud publica*, vol. 95, Jan. 2021.

[11] A. Marcão, C. Barreto, L. Pereira, *et al.*, "Cystic fibrosis newborn screening in portugal: Pap value in populations with stringent rules for genetic studies," *International Journal of Neonatal Screening*, vol. 4, p. 22, Jun. 2018. DOI: 10.3390/ijns4030022.

[12] J. E. Morley, *Doenças endócrinas*, 2019. [Online]. Available: https://www.msdmanuals.com/pt-pt/casa/dist%5C%C3%5C%BArbios-hormonais-e-metab%5C%C3%5C%B3licos/biologia-do-sistema-end%5C%C3%5C%B3crino/doen%5C%C3%5C%A7as-end%5C%C3%5C%B3crinas.

[13] C. Bezerra, *Hipotireoidismo: O que é, sintomas, causas e tratamento*, 2021. [Online]. Available: https://www.tuasaude.com/hipotireoidismo/.

[14] P. Pinheiro, *Hipotireoidismo subclínico: Sintomas e tratamento*, 2021. [Online]. Available: https://www.mdsaude.com/endocrinologia/hipotireoidismo-subclinico/.

[15] Mayo Clinic, *Hypothyroidism (underactive thyroid)*, 2020. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/hypothyroidism/symptoms-causes/syc-20350284.

[16] S. Sedicias, *Hipotireoidismo na gravidez: Riscos, como identificar e como é o tratamento*, 2020. [Online]. Available: https://www.tuasaude.com/hipotireoidismo-na-gravidez/.

[17] NHS, *Underactive thyroid (hypothyroidism)*, 2021. [Online]. Available: https://www.nhs.uk/conditions/underactive-thyroid-hypothyroidism/.

[18] B. Beltrame, *O que é hipotireoidismo congênito, sintomas e como tratar*, 2020. [Online]. Available: https://www.tuasaude.com/hipotireoidismo-congenito/.

[19] Mayo Clinic, *Cystic fibrosis - symptoms and causes*, 2021. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/cystic-fibrosis/symptoms-causes/syc-20353700.

[20] NHS, *Cystic fibrosis*, 2021. [Online]. Available: https://www.nhs.uk/conditions/cystic-fibrosis/.

[21] Mayo Clinic, *Cystic fibrosis - diagnosis and treatment*, 2021. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/cystic-fibrosis/diagnosis-treatment/drc-20353706.

[22] NHS, *Cystic fibrosis - treatment*, 2021. [Online]. Available: https://www.nhs.uk/conditions/cystic-fibrosis/treatment/.

[23] M. Zhu, "The guthrie test for early diagnosis of phenylketonuria," *Embryo Project Encyclopedia*, 2017. [Online]. Available: http://embryo.asu.edu/handle/10776/11460.

[24] C. Sansom, "Tandem mass spectrometry: The tool of choice for diagnosing inborn errors of metabolism," *Molecular Medicine Today*, 1999. [Online]. Available: https://www.cell.com/molecular-medicine/pdf/S1357-4310(99)01437-9.pdf.

[25] BMD Software, *Neoscreen*, 2021. [Online]. Available: https://www.bmd-software.com/solutions/neoscreen/.

[26] PerkinElmer, *Specimen gate newborn screening software - perkinelme*, 2021. [Online]. Available: https://rh.perkinelmer.com/products/specimen-gate-newborn-screening-software/.

[27] ——, *Total process management for your newborn screening laboratory*, 2021. [Online]. Available: https://rh.perkinelmer.com/wp-content/uploads/2021/04/109971-BRO-Specimen-Gate%5C_FINAL-1.pdf.

[28] NATUS, *Company profile*, 2021. [Online]. Available: https://newborncare.natus.com/company-profile.

[29] ——, *Neometrics data management for newborn care*, 2021. [Online]. Available: https://newborncare.natus.com/products-services/newborn-care-products/tracking-follow/neometrics-data-management.

[30] ——, *Neometrics data management solutions*, 2021. [Online]. Available: https://newborncare.natus.com/products-services/newborn-care-products/tracking-follow/neometrics-data-management.

[31] OZ Systems, *Blood spot screening (nbs)*, 2021. [Online]. Available: https://www.ozsystems.com/hospitals-providers/oz-newborn-screening-tools/blood-spot-screening-nbs.

[32] StarLims, *Starlims advanced analytics*, 2021. [Online]. Available: https://www.starlims.com/wp-content/uploads/advanced-analytics-brochure.pdf.

[33] Epiconcept, *Neonatal screening software - néonat - epiconcept*, 2021. [Online]. Available: https://www.epiconcept.fr/en/produit/neonat/.

[34] Oracle, *What is a database?* 2021. [Online]. Available: https://www.oracle.com/database/what-is-database/.

[35] F. Andrade, *Top databases to use in 2022: What is the right database for your use case?* 2022. [Online]. Available: https://towardsdatascience.com/top-databases-to-use-in-2022-what-is-the-right-database-for-your-use-case-bb8d3f183b21.

[36] S. Overflow, *Stack overflow developer survey 2021*, 2021. [Online]. Available: https://insights.stackoverflow.com/survey/2021#database.

[37] P. Barros, *O que é acid?* 2016. [Online]. Available: https://medium.com/opensanca/o-que-%C3%A9-acid-59b11a81e2c6.

[38] EdPrice-MSFT, *Non-relational data and nosql*, 202. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data.

[39] MySOracleQL, *What is mysql?* 2021. [Online]. Available: https://www.oracle.com/mysql/what-is-mysql/.

[40] H. Jain, *Best 7 real-world mongodb use cases*, 2022. [Online]. Available: https://hevodata.com/learn/mongodb-use-case/.

[41] J. Patel, *List of 10 best web frameworks for web app development in 2022*, 2022. [Online]. Available: https://www.monocubed.com/blog/most-popular-web-frameworks/.

[42] MDN, *Mvc*, 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/MVC.

[43] P. Framework, *Play framework - build modern and scalable web apps with java and scala*, 2021. [Online]. Available: https://www.playframework.com/.

[44] Ricardo, *Introdução ao backbone.js*, 2013. [Online]. Available: https://www.devmedia.com.br/introducao-ao-backbone-js/29303.

[45] React, *React - a javascript library for building user interfaces*, 2022. [Online]. Available: https://reactjs.org/.

[46] O. Coskun, "Separation techniques: Chromatography," en, *North Clin Istanb*, vol. 3, no. 2, pp. 156–160, Nov. 2016.

[47] D. Chacon, *Junit tutorial: Setting up, writing, and running java unit tests*, 2022. [Online]. Available: https://www.parasoft.com/blog/junit-tutorial-setting-up-writing-and-running-java-unit-tests/.

[48] K. Rungta, *What is selenium? introduction to selenium automation testing*, 2022. [Online]. Available: https://www.guru99.com/introduction-to-selenium.html.

[49] O. Moradov, *Unit testing: Definition, examples, and critical best practices*, 2022. [Online]. Available: https://brightsec.com/blog/unit-testing/.

[50] ——, *Unit testing vs. functional testing: In-depth comparison*, 2022. [Online]. Available: https://brightsec.com/blog/unit-testing-vs-functional-testing.

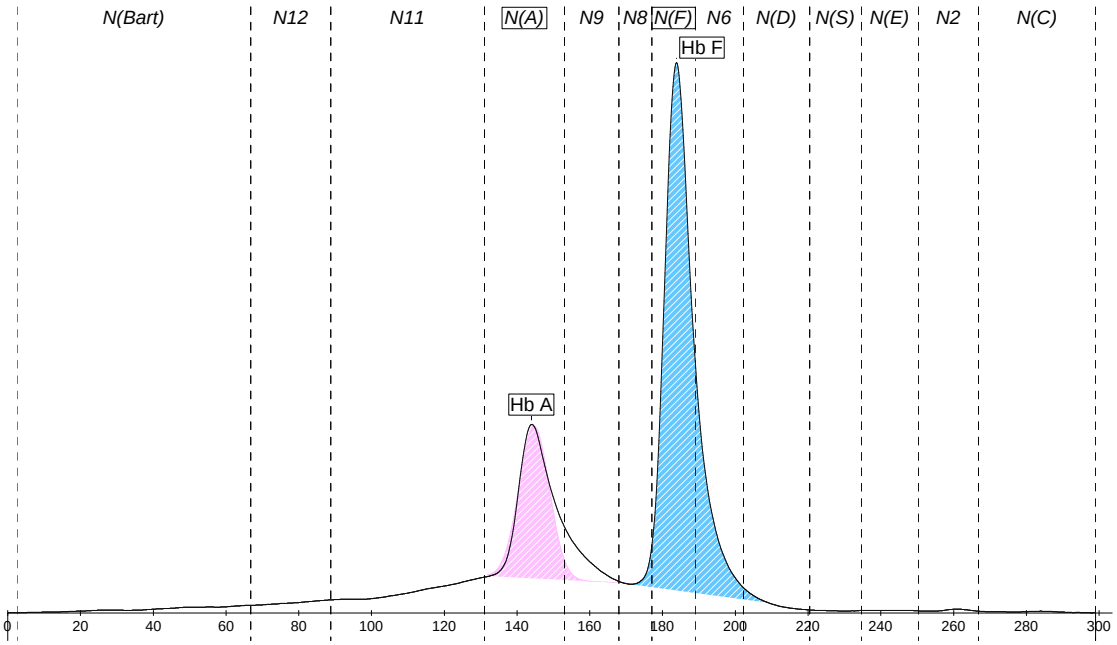
File with extra sample information

Date : **09/03/2022**

Microplate line : M0009151-G

ID : **4055440**

Sample date :



Neonatal Haemoglobin Electrophoresis

Name	%
Hb A	23,7
Hb F	76,3

Hb pattern : FA

Signature

REST API Endpoints

- **Authentication**
  - /login
  - /logout
  - /users
  - /user
  - /user/:id
  - /user/:id/token
  - /audit-logs
  - /settings
- **RBAC**
  - /roles
  - /role
  - /role/:id
  - /permissions
- **Sample Configurations**
  - /units
  - /unit
  - /unit/:id
  - /ages
  - /age
  - /age/:id
  - /age/:id/default
  - /diagnosises
  - /diagnosis
  - /diagnosis/:id
  - /thresholds
  - /threshold
  - /threshold/:id

  - /control-thresholds
  - /control-threshold
  - /control-threshold/:id
  - /messages
  - /message
  - /message/:id

- **Samples**
  - /types
  - /samples
  - /sample
  - /sample/:id
  - /sample/:sampleId/age/:ageId
  - /sample/:sampleId/diagnosis/:diagnosisId
  - /sample/:id/extra-file
  - /files
  - /file/:id
  - /flagged
  - /publish/publish
  - /publish/revoke
  - /publish/repeat
  - /import/validate
  - /import/extra
  - /import/finish
  - /import/parsers
  - /search
  - /stats